AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

OLEH :

ZAQQI YAMANI A

09042611822005

PROGRAM MAGISTER ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJYA

TAHUN 2020

LEMBAR PENGESAHAN
AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

TESIS
Diajukan Untuk Melengkapi Salah Satu Syarat
Memperoleh Gelar Magister

OLEH:

ZAQQI YAMANI
09042611822005

Palembang, Agustus 2020

Pembimbing I,

Prof. Dr. Ir. Siti Nurmaini, M.T.
NIP 196908021994012001

Pembimbing II,

Dian Palupi Rini, M.Kom.,Ph.D
NIP 197802232006042002

Mengetahui,
Ketua Program Studi Magister Ilmu Komputer

Dr. Ir. H. Sukemi, M.T
NIP 196612032006041001

AUTHOR NAMES DISAMBIGUATION IN AUTHOR MATCHING CLASSIFICATION USING MACHINE LEARNING METHOD IN ANOMALY DETECTION APPROACH

THESIS

Submitted to Complete of the Term Obtaining
A Master Degree

BY:

ZAQQI YAMANI
09042611822005

Palembang, Agustus 2020

Supervisor I,

Prof. Dr. Ir. Siti Nurmaini, M.T.
NIP 196908021994012001

Supervisor II,

Dian Palupi Rini, M.Kom.,Ph.D
NIP 197802232006042002

Acknowledged,
Head of Department Magister Computer Science

Dr. Ir. H. Sukemi, M.T
NIP 196612032006041001

HALAMAN PERSETUJUAN

Pada Rabu 05 Agustus 2020 telah dilaksanakan ujian sidang tesis secara daring oleh Magister Ilmu Komputer Ilmu Komputer Universitas Sriwijaya.

Nama : Zaqqi Yamani

NIM : 09042611822005

Judul : Author Names Disambiguation Dalam Klasifikasi Author Matching Dengan Menggunakan Metode Machine Learning Pada Pendekatan Anomaly Detection

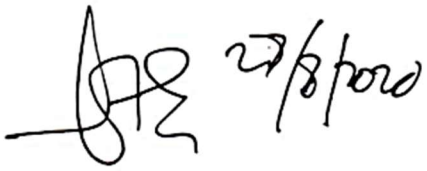
1. Pembimbing I

   Prof. Dr. Ir. Siti Nurmaini, M.T. ..... 
   NIP 196908021994012001

2. Pembimbing II

   Dian PalupiRini, M.Kom., Ph.D .....
   NIP 197802232006042002

3. Penguji I

   Dr. Ir. Bambang Tutuko, M.T ..... 
   NIP 196001121989031002

4. Penguji II

   Dr. Iwan Pahendra, M.T .....
   NIP 197403222002121002

Mengetahui,
Ketua Program Studi Magister IlmuKomputer

Dr. Ir. H. Sukemi, M.T
NIP 196612032006041001

AUTHOR NAMES DISAMBIGUATION IN AUTHOR MATCHING CLASSIFICATION USING MACHINE LEARNING METHOD IN ANOMALY DETECTION APPROACH

Zaqqi Yamani (09042611822005)
Dept of Master Computer Science, Computer Science Faculty, Sriwijaya University
Email : zaqqi.yamani@gmail.com

ABSTRACT

Author Name Disambiguation (AND) is a problem that reduces the quality of information in publication. Quality of service is strongly influenced in the problem of ambiguity of author names in citations and considered to be the most difficult issues in digital library from researchers and has an impact on the information of authors, organizations and other matters presented as part of the publication. Therefore, anomaly detection approach is proposed to solve this problem. The method proposed is using the isolationForest algorithm and local Outlier Factor (LOF) which produces an accuracy score 95%.

Keywords:

Acknownledged,

Palembang, Agustus 2020

Supervisor 1,

Prof. Dr. Ir. Siti Nurmaini, M.T.
NIP. 196908021994012001

Supervisor 2,

Dian Palupi Rini, M.Kom., Ph.D
NIP. 197802232006042002

Head of Department Master Computer Science

Dr. Ir. Sukemi, M.T.
NIP. 196612032006041001

AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

Zaqqi Yamani (09042611822005)
Jurusan Magister Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya
Email : zaqqi.yamani@gmail.com

ABSTRAK

Author Name Disambiguation (AND) merupakan masalah yang menurunkan kualitas informasi dalam publikasi. Kualitas layanan sangat dipengaruhi oleh masalah ambiguitas nama penulis dalam kutipan dan dianggap sebagai masalah tersulit di perpustakaan digital dari peneliti dan berdampak pada informasi penulis, organisasi dan hal-hal lain yang disajikan sebagai bagian dari publikasi. Oleh karena itu, pendekatan deteksi anomali diusulkan untuk mengatasi masalah ini. Metode yang diusulkan menggunakan algoritma isolation Forest dan local Outlier Factor (LOF) yang menghasilkan nilai akurasi 95%.

Kata Kunci:

Mengetahui,

Palembang, Desember 2019

Pembimbing 1,

Prof. Dr. Ir. Sisi Nurmaini, M.T.
NIP. 196908021994012001

Pembimbing 2,

Dian Palupi Rini, M.Kom., Ph.D
NIP. 197802232006042002

Ketua Program Studi Magister Ilmu Komputer

Dr. Ir. Sukemi, M.T.
NIP. 196612032006041001

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab ini menjelaskan tentang latar belakang diangkatnya tema permasalahan tesis ini, yang kemudian dari masalah dilakukan perumusan masalah. Di dalam bab ini pula dibahas mengenai batasan masalah, tujuan penelitian serta metodologi penulisan dalam penelitian ini.

1.1. Latar Belakang

Digital Library (DL) saat ini merupakan salah satu pusat literasi yang paling berkembang pesat, salah satunya pada bidang akademik. Hal ini dipengaruhi berbagai faktor seperti pemotongan anggaran untuk perpustakaan tradisional, ruang penyimpanan yang hampir tak terbatas dengan biaya yang jauh lebih rendah, kemudahan penggunaan dan tidak ada batas fisik dari penelitian (Palfrey, 2016; Weiss, 2016). Beberapa DL, antara lain DBLP1, MEDLINE2, CiteSeer3 arXiv4, MAS5, Google Scholar6, dan BDBComp7 secara luas digunakan oleh para peneliti untuk menemukan literatur ilmiah untuk penelitian dan penemuan mereka (Nicholson and Bennett, 2016). Selain memberikan beberapa informasi dan analisis yang berguna bagi pengambilan keputusan dan menyediakan konten berkualitas tinggi pada penelitian (Mitra *et al.*, 2007), DL juga memiliki beberapa sumber kesalahan antara lain adalah tipografi, pemindaian dan konversi data, menemukan dan mengganti, menyalin dan menempel, meta data, perangkat lunak pengumpulan kutipan yang tidak sempurna, format kutipan yang berbeda, nama-nama penulis yang ambigu, pembuatan konten terdesentralisasi dan singkatan dari judul tempat publikasi yang ambigu, dll (Ferreira, Gonçalves and Laender, 2012).

Ambiguitas Nama Penulis atau yang lebih dikenal dengan *Author Name Disambiguation* (AND) adalah salah satu masalah yang menurunkan kualitas dan keandalan informasi yang diperoleh dari DL (Tran, Huynh and Do, 2014). Konten DL dan kualitas layanan sangat dipengaruhi oleh masalah ambiguitas nama penulis dalam kutipan dan dianggap sebagai salah satu masalah tersulit yang dihadapi oleh para peneliti perpustakaan digital dari penelitian (Hussain and Asghar, 2017). AND menjadi sebuah masalah ketika satu set catatan publikasi berisi nama penulis yang

menimbulkan lebih dari satu interpretasi, yaitu penulis yang sama dapat muncul dengan nama yang berbeda (Ferreira, Gonçalves and Laender, 2012). Hal tersebut menjadi poin yang mengurangi kualitas dari informasi serta mengurangi pula keandalan informasi tersebut karena berdampak pada informasi terhadap penulis, organisasi dan hal lain yang ditampilkan sebagai bagian dari catatan publikasi tersebut (Müller, Reitz and Roy, 2017).

Beberapa penelitian telah dilakukan dalam rangka mengklasifikasikan AND, terutama pada proses klasifikasi *author matching*. Proses klasifikasi tersebut mengedepankan proses *pairwise* dan *distance* dari nama-nama author (Wang *et al.*, 2013). Klasifikasi terhadap data *author* diharapkan memberikan interpretasi yang tepat dan prediksi serta akurasi yang tinggi ketika dijalankan pada setiap dataset AND dan *bibliography*.

Dalam perkembangannya, AND dalam klasifikasi *author matching* menciptakan tantangan yang menakutkan dalam teknik disambiguasi karena sering menarik kesimpulan yang salah pada data publikasi yang tidak lengkap (Song, Kim and Kim, 2015). Ada sejumlah solusi yang dijalankan terhadap hal tersebut, diantaranya dengan *un-supervised* yaitu dilakukan berdasarkan kesamaan catatan bibliografi atau pola penulisan yang bersifat umum (Milojević, 2013), model NDMC atau multi step clustering yang dilakukan dengan menyamaratakan nama penulis atau menggabungkan karakteristik singkat dari informasi data publikasi (Gu *et al.*, 2016). Selain itu, ada teknik lain yang pernah digunakan yaitu LUCID, di mana dilakukan dengan menggunakan algoritma pendeteksian komunitas dan operasi grafik yang di akhir fase teknik ini tetap menggunakan fungsi kemiripan dari data publikasi tersebut (Hussain and Asghar, 2018b). Lalu ada teknik sistem analisis visual yang disebut NameClarifier yang secara interaktif mengelompokkan nama penulis dalam publikasi di dalam lingkaran tertentu, lalu menghitung dan memvisualisasikan kesamaan antara nama ambigu dan yang telah dikonfirmasi di Digital Library (Shen *et al.*, 2017). Namun, keempat metode tersebut tidak mementingkan akurasi dari proses klasifikasi data publikasi. Semua metode tersebut memberikan gambaran yang sama dalam teknik menuju klasifikasi yaitu dengan melakukan *pairwise* lalu menemukan kemiripan dari data yang dipasangkan tersebut.

Di sisi lain, *Machine Learning* juga telah banyak digunakan untuk melakukan proses klasifikasi AND dan menghasilkan kinerja yang memuaskan (Hussain and Asghar, 2017). Diantaranya dengan *supervised AND techniques* dengan menggunakan *boosted tree classification* yang fokus pada proses pemfilteran dan pencocokan nama dan afiliasi dalam sebuah publikasi (Wang *et al.*, 2013). *un supervised AND techniques* dengan pendekatan teori DST (*Dempster-Shafer Theory)* yaitu menghitung kesamaan fitur tingkat tinggi seperti afiliasi, tempat, content, rekan penulis, kutipan, korelasi Web(Wu *et al.*, 2014). Selanjutnya semi *supervised* AND *techniques* dengan menggunakan Algoritma mendeteksi kesamaan di antara objek. Mereka membangun matriks dua dimensi untuk penulisan bersama dan hubungan topik dan menghitung jarak antara dua simpul dengan bantuan jarak *Euclidean* (Zhu and Li, 2013). Selain itu, ada *Graph-Based AND techniques* dengan menggunakan algoritma *multi-level Graph Partinging* (MGP), dan algoritma *Multi-Level Graph Partinging* dan *Merging* (MGPM) (On, Lee and Lee, 2012). dan *Graph Based AND techniques* yang menggunakan kesamaan antara catatan *bibliografi* dan kelompok catatan baru untuk penulis dengan catatan kutipan yang sama di DL, atau untuk penulis baru ketika bukti kesamaan tidak cukup kuat. Beberapa *heuristik* khusus digunakan untuk memeriksa apakah referensi dari catatan kutipan baru milik penulis yang sudah ada di DL atau milik penulis yang baru (yaitu, penulis tanpa catatan kutipan di DL), menghindari menjalankan proses disambiguasi di seluruh DL (de Carvalho *et al.*, 2011). Namun, metode diatas menghasilkan kinerja akurasi, presisi, spesifisitas dan sensitivitas kurang memuaskan. Dan sama seperti sebelumnya, semua metode tersebut dilakukan dengan melakukan *pairwise,* lalu dilanjutkan dengan menghitung jarak untuk menghasilkan *similarity* dari data author yang dipasangkan.

Selain itu, metode lain yang pernah dilakukan untuk melakukan klasifikasi terdapat *author matching* yaitu dengan menggunakan *deep structure*. salah satu metode yang menggunakan struktur tersebut adalah metode *Deep Neural Network.* Arsitektur ini memiliki dua komponen utama. Dalam komponen pertama, data diambil sebagai input dan representasi data dihitung dengan mencari kemiripan dari data. Komponen kedua mengambil set fitur dasar sebagai inputnya dan mempelajari fitur-fitur di dalamnya adalah lapisan tersembunyi untuk menyamarkan nama

pembuatnya (Tran, Huynh and Do, 2014). Metode ini telah menghasilkan akurasi tinggi pada nilai 99,31%. Sama seperti penelitian sebelumnya, proses yang dilakukan adalah dengan *pairwise*. Namun, penelitian ini dilakukan dengan data yang sedikit dimana *pairwise* data menghasilkan hanya sekitar 30.537 data.

Dari berbagai metode yang telah dilakukan, dapat ditarik kesimpulan bahwa pra pemrosesan sebelum masuk pada proses klasifikasi adalah dengan melakukan *pairwise* dan dilanjutkan dengan menghitung jarak (*similarity)* dari data yang telah dipasangkan. Dalam prosesnya, teknik *pairwise* pada data author akan menimbulkan *imbalanced* data yang tinggi, dimana semakin banyak data yang dipasangkan akan menimbulkan tingkat *imbalanced* yang semakin tinggi pula atau dengan kalimat lain dimana data negatif menjadi sangat dominan dibanding data yang positif (Kim and Kim, 2018). Hal itu menjadikan hasil proses klasifikasi menjadi meragukan karena proses pencarian data yang bernilai positif menjadi sulit dan bisa dikatakan seperti mencari data yang langka.

Proses diatas sama halnya dengan dengan proses klasifikasi pada deteksi intrusi pada sistem komputer, dimana deteksi pada gangguan sistem komputer seperti mencari barang langka di dalam proses yang secara dominan berjalan normal (Ferreira, Gonçalves and Laender, 2012). Selain itu, kasus pada deteksi gangguan pada *network* pun memiliki masalah yang sama, dimana hal-hal yang dianggap gangguan atau *instrusion* merupakan hal yang sulit dikenali atau dicari karena persentase keberadaannya sangat kecil dibandingkan proses yang ada dan dianggap bukan merupakan sebuah *intrusion* (Dawoud, Shahristani and Raun, 2019). Adapun teknik yang dilakukan untuk mengklasifikan dua kasus diatas adalah dengan menggunakan metode *machine learning* pada pendekatan *Anomaly Detection* yang dapat menghasilkan nilai akurasi pada data negatif maupun data positif.

Berdasarkan hal tersebut, tingkat *imbalanced* yang tinggi pada proses klasifikasi *Author matching* dapat dikategorikan sama dengan yang terjadi pada proses *intrusi* atau gangguan pada sistem komputer dan *network.* Oleh karena itu, pendekatan *anomaly detection* akan menghasilkan tingkat akurasi yang tinggi pada proses klasifikasi. Sama seperti pada penggunaan *anomaly detection* di intrusi sistem komputer, model yang digunakan untuk proses klasifikasi pada penelitian

ini juga dengan menggunakan model algoritma *isolationForest* dan *local Outlier Factor (LOF).*

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka perumusan masalah yang diambil pada penelitian ini adalah tentang "Bagaimana membuat sistem pengklasifikasi ambiguitas nama penulis (*author name disambiguation)* dalam *author matching* dengan menggunakan *Machine Learning* pada pendekatan *Anomaly Detection?*".

Dalam penelitian ini, perumusan masalah dijabarkan dalam bentuk pertanyaan sebagai berikut :

a. Bagaimana proses pra pengolahan data *Author Matching* dari suatu *digital library?*
b. Bagaimana menemukan model *Anomaly Detection* yang terbaik dengan menggunakan algoritma *IsolationForest* dan *Local Outlier Factor* atau *auto encoder* untuk melakukan klasifikasi pada data yang berdimensi tinggi?
c. Bagaimana mengukur hasil kinerja dari model *Anomaly Detection* berdasarkan parameter akurasi, sensitivitas, spesifisitas, presisi dan F1 score?

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah :

1. System hanya berupa simulasi untuk melakukan proses klasifikasi data penulis (*author matching)* pada data di DL.
2. Dataset yang digunakan adalah data nama-nama penulis lengkap dengan judul jurnal serta tahun terbit yang diambil dari dataset SCAD-zBMATH dengan judul featured-dataset-merged yang terdiri dari 11.924 baris data publikasi (Müller, Reitz and Roy, 2017).
3. Pra pengolahan akan menggunakan teknik *pairwise* dan *similarity*.

1.4. Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Melakukan proses pra pengolahan data publikasi yang memiliki disambiguitas untuk menginterpretasi dan mengelompokkan penulis berdasarkan kemiripan nama, judul, tahun dan nama kecil dari penulis.
2. Menganalisis model klasifikasi dari Anomaly Detection dalam mengklasifikasikan penulis yang sama atau bukan pada judul tulisan dan tahun terbit yang berbeda.
3. Mengukur kinerja klasifikasi Anomaly Detection berdasarkan parameter akurasi, sensitivitas, spesifisitas, presisi dan F1 score.

1.5. Manfaat

Manfaat dari penelitian ini adalah menjadi landasan penggunaan metode anomaly detection dalam klasifikasi Author Names Disambiguation terutama dalam konsep Author Matching. Selain itu, manfaat dari penelitian ini adalah sebagai berikut :

1. Anomaly Detection dapat menjadi metode tambahan baru dalam proses disambiguasi terhadap nama penulis.
2. Model klasifikasi yang dipakai dalam penelitian ini dapat digunakan lebih lanjut untuk meningkatka hasil accuracy dan presisi dalam proses AND terutama pada konsep Author Matching.

1.6. Metodologi Penulisan

Metodologi penulisan pada tesis ini terdiri dari lima bab sebagai berikut:

BAB I : PENDAHULUAN

Bab I berisi pendahuluan berupa latar belakang, perumusan masalah, tujuan dan manfaat dari topik yang dipilih.

BAB II: TINJAUAN PUSTAKA

Bab II berisi kerangka teori dan pustaka yang berhubungan dengan klasifikasi *Author Matching* pada data DL dengan menggunakan metode *Anomaly Detection* yang mengacu pada beberapa penelitian jurnal publikasi.

BAB III : METODOLOGI PENELITIAN

Bab III berisi metodologi yang menjelaskan secara bertahap dan terperinci tentang langkah-langkah yang digunakan untuk mencari, mengumpulkan dan menganalisa yang berkaitan dengan *author Matching*. Metodologi ini menjelaskan pendekatan atau algoritma *author matching*, serta model yang digunakan sehingga tujuan dari penulisan dapat tercapai.

BAB IV: HASIL DAN ANALISA SEMENTARA

Bab IV berisi hasil pengujian yang telah dilakukan, data-data yang diambil dari pengujian tersebut akan dianalisa menggunakan berbagai macam teknik, selain itu di bab ini juga membahas kevalidasian dari sistem yang telah dibuat.

BAB V: KESIMPULAN

BAB V berisi tentang kesimpulan apa yang diperoleh oleh penulis serta merupakan jawaban dari setiap tujuan yang ingin dicapai.

BAB II
TINJAUAN PUSTAKA

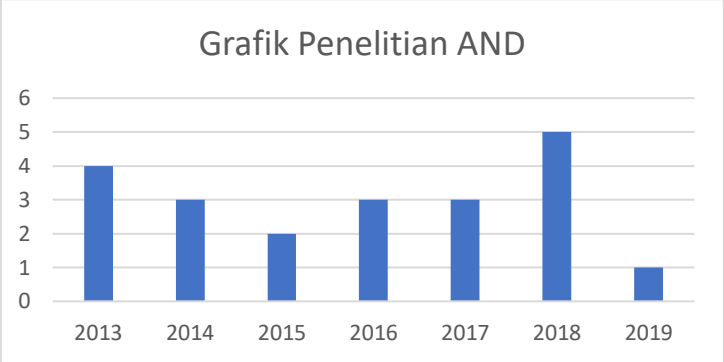
Bab ini berisi mengenai tinjauan pustaka yang berhubungan dan relevan dengan *Author Names Disambiguation* dalam Klasifikasi *Author Matching* dengan Menggunakan Metode *Machine Learning* Pada Pendekatan *Anomaly Detection*. Bab ini juga akan menjabarkan penelitian-penelitian terkait dan membandingkan hasil penelitian tersebut dari sisi kinerja algoritma klasifikasi, pembahasan teori *Author Names Disambiguation*, *Author Matching, Pairwise and Similarity Technique,* dan metode *anomaly detction*.

2.1. Tinjauan Penelitian

Ambiguitas Nama Penulis atau yang lebih dikenal dengan *Author Name Disambiguation* (AND) adalah salah satu masalah yang menurunkan kualitas dan keandalan informasi yang diperoleh dari *Digital Library* (DL) (Tran, Huynh and Do, 2014) dan (Hazra *et al.*, 2016). Konten DL dan kualitas layanan sangat dipengaruhi oleh masalah ambiguitas nama penulis dalam kutipan dan dianggap sebagai salah satu masalah tersulit yang dihadapi oleh para peneliti perpustakaan *digital* (Hussain and Asghar, 2017). AND menjadi sebuah masalah ketika satu set catatan publikasi berisi nama penulis yang menimbulkan lebih dari satu interpretasi, yaitu penulis yang sama bisa muncul dengan nama yang berbeda (Ferreira, Gonçalves and Laender, 2012) dan (Firdaus, 2018), hal itu disebabkan inkonsistensi dalam penyajian nama *author* (Milojević, 2013). Konten DL yang menimbulkan lebih dari satu interpretasi tersebut menjadi tantangan dalam proses klasifikasi terutama pada pengolahan data *author matching* (Song, Kim and Kim, 2015).

Pembahasan mengenai AND belum mendapat perhatian secara khusus dalam beberapa penelitian terutama dalam penentuan akurasi , hal ini dikarenakan kebutuhan akan *resource* yang besar dalam pengolahan data yang memiliki jumlah yang sangat besar pula (Milojević, 2013) dan jumlah penelitian spesifik terhdapa AND dalam 6 tahun terakhir dapat dilihat pada gambar 2.1. Beberapa evaluasi telah dilakukan terhadap penelitian mengenai AND, metode dan teknik yang dilakukan pada sarnya memiliki kesamaan yaitu dengan melakukan pemasangan dari data

authors dan lebih lanjut dengan melakukan pencarian kemiripan dati hasil pemesangan tersebut (Kim, 2018).



Gambar 2.1. Grafik Jumlah Penelitian AND

Penelitian terhadap AND ini selanjutnya dipisahkan menjadi 3 bentuk penelitian yaitu *author matching, author grouping* dan *author assignment* dimana masing-masing tingkatan memiliki karakteristik atau interpretasi yang berbeda-beda dalam hasil akurasi (Ferreira, Gonçalves and Laender, 2012). Masing-masing pola tersebut telah dilakukan penelitian lebih lanjut untuk menghasilkan tingkat akurasi dalam klasifikasi data *authors* (Hussain and Asghar, 2017).

AND atau lebih spesifiknya adalah *author matching* dalam perkembangannya menjadi topik yang menarik dalam dunia klasifikasi terhadap data publikasi dalam DL. Hal itu disebabkan karena AND menjadi poin yang mengurangi kualitas dari informasi serta mengurangi pula keandalan informasi tersebut karena berdampak pada informasi terhadap penulis, organisasi dan hal lain yang ditampilkan sebagai bagian dari catatan publikasi tersebut (Müller, Reitz and Roy, 2017).

2.1.1. Klasifikasi Author Matching

Author matching merupakan langkah awal dalam proses disambiguasi terhadap konten DL (Wang *et al.*, 2013). Beberapa penelitian yang dilakukan terhadap AND, hampir semuanya berpijak atau diawali pada proses *author matching* tersebut. Beberapa penelitian dengan berbagai macam metode telah

dilakukan untuk menemukan cara terbaik dan metode terbaik dalam proses klasifikasi disambiguasi *author matching.*

Beberapa metode yang pernah dilakukan antara lain, *heuristic based hierarchical methode* (Cota *et al.*, 2010), dengan menggunakan *associative classifier* (Ferreira, Gonçalves and Laender, 2012), lalu ada pula metode K-NN, *Random forest*, C4.5 (Ferreira, Gonçalves and Laender, 2012), selanjutnya dengan metode SVM dan *Naive Bayes* (Han *et al.*, 2004). Selain itu, ada teknik *boosted tree classification* (Hussain and Asghar, 2018b) (Berzins *et al.*, 2012). Ada lagi dengan menggunakan teori dan algoritma *dempster shafer* (Wu *et al.*, 2014) dan *Graph structural* dan *hybrid simlarity* (Hussain and Asghar, 2018a). Dan ada pula penelitian yang menggunakan teknik deep learning dengan menggunakan pendekatan deep neural network (DNN) (Tran, Huynh and Do, 2014).

Tabel 2.1

Penelitian Terhadap Author Matching

No.	Peneliti/Jurnal	Metode	Implementasi	Teknik
1	(Cota *et al.*, 2010)	Heuristic Based Hirarchical	Klasifikasi Biner antar author	Pairwise dan Similarity
2	(Ferreira *et al.*, 2010)	Associative classifier	Klasifikasi Biner	Pairwise
3	(Ferreira, Gonçalves and Laender, 2012)	K-NN	Klasifikasi Biner	Pairwise dan Similarity
4	(Ferreira, Gonçalves and Laender, 2012)	Random Forest	Klasifikasi Biner	Pairwise dan Similarity
5	(Ferreira, Gonçalves and Laender, 2012)	C4.5		Pairwise dan Similarity
6	(Han *et al.*, 2004)	SVM	Klasifikasi Biner	Pairwise dan Similarity

7	(Li and Han, 2017)	Naive Bayes	Klasifikasi Biner	Pairwise dan Similarity
8	(Hussain and Asghar, 2018b) (Berzins *et al.*, 2012)	Boosted tree Classification	Klasifikasi Biner	Pairwise dan Similarity
9	(Wu *et al.*, 2014)	Dempster-Shafer Theory		Pairwise
10	(Hussain and Asghar, 2018a)	Graph Structural	Klasifikasi Biner	Pairwise dan Similarity
11	(Tran, Huynh and Do, 2014)	Deep Neural Network	Klasifikasi Biner	Pairwise dan Similarity

2.2. Teknik Pemasangan (*Pairwise*) dan Penghitungan Kemiripan (*Similarity*)

Dalam teknik disambiguasi terhadap data penulis atau *author* terutama pada *author matching*, teknik pengolahan data menjadi penting sebelum masuk ke dalam algoritma atau mesin *classifier.* Proses pengolahan data atau pra processing ini menjadi titik awal pengenalan atau pengelompokkan fitur dan label yang akan menjadi mode pembelajaran dari classifier yang akan digunakan (Wu *et al.*, 2014).

Di beberapa penelitian yang telah dilaksanakan teknik pengolahan terhadap data author di dominasi oleh konsep pemasangan atau *pairwise.* Dan teknik pemasangan tersebut banyak dilakukan dengan teknik matematika sederhana yaitu kombinasi. Lalu setelah semua data memiliki pasangannya, pra pengolahan data author dilaksanakan dengan mencari jarak antar data tersebut atau sering dikenal dengan teknik *similarity*. Teknik tersebut bertujuan memberi nilai pada data author yang telah dipasangkan dengan harapan memberi prediksi kemiripan dari data tersebut (Kim, 2018), (Tran, Huynh and Do, 2014), (Ferreira, Gonçalves and Laender, 2012), (Hussain and Asghar, 2017).

Kedua teknik diatas menjadi sangat penting dalam proses pra pengolahan data *author matching.* Keduanya akan menjadi titik awal untuk meningkatkan nilai akurasi pada proses klasifikasi *author matching.*

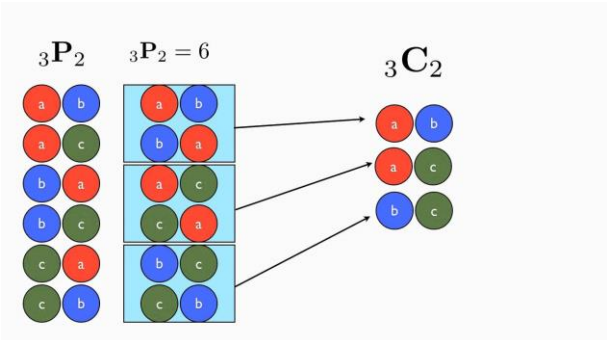
2.2.1. Pemasangan dengan Kombinasi (*Pairwise combination*)

Ada beberapa teknik untuk mendapatkan hasil probabilitas dari perbandingan data atau item yang dikoleksi, antara lain dengan *pariwise combinations.* Metode ini menjadi efektif ketika kita akan mendapat hasil dari suatu perbandingan dengan tujuan probabilitas di masa mendatang (Shah, Balakrishnan and Wainwright, 2016). Teknik ini mengedepankan proses perbandingan berpasangan yang dapat digunakan untuk memperoleh kecenderungan terkait dari setiap kriteria yang dibandingkan.

Dalam *pairwise combination*, teknik yang biasa digunakan untuk meningkatkan hasil dari perbandingan yang dilakukan adalah dengan melakukan kombinasi dari data yang ingin dipasangkan. Tujuan dari dilakukannya kombinasi adalah agar setiap baris data memiliki pasangan atau berdampingan dengan data di baris berikutnya dan seterusnya. Dan di dalam pengolahan dataset untuk *author matching*, proses kombinasi akan sangat membantu dalam menyajikan baris data demi meningkatnya akurasi dalam proses klasifikasi (Zhao, Wang and Huang, 2013). Di dalam rumus matematika sederhana, kombinasi adalah menggabungkan beberapa objek dari suatu grup tanpa memperhatikan urutan (rumus 2.1).

$$\frac{\boldsymbol{n}!}{\boldsymbol{r}!(\boldsymbol{n}-\boldsymbol{r})!} = \binom{\boldsymbol{n}}{\boldsymbol{r}} \qquad \textbf{(2.1)}$$

Dimana ***n*** adalah jumlah objek yang bisa dipilih dan ***r*** adalah jumlah yang harus dipilih. Hasil dari pengolahan data dengan kombinasi akan menyajikan data yang berpasangan secara menyeluruh, dan akan memberikan jumlah data yang lebih banyak dari data awal.



Gambar 2.2. teknis berpasangan menggunakan kombinasi

Proses ini akan menghasilkan baris data yang lebih banyak dari jumlah baris data sebenarnya dengan tujuan dapat dibandingkan antara baris data pertama dengan baris data kedua, baris pertama dengan baris ketiga dan seterusnya sampai setiap baris bertemu. Saat semua baris telah dipasangkan, akan diambil cara untuk menghitung atau menganalisis data tersebut berdasarkan output yang akan dihasilkan dan digunakan.

2.2.2. Penghitungan Kemiripan (*Similarity*)

Similarity merupakan proses dalam *feature extraxtion* dengan melakukan pencarian nilai kemiripan atau kesamaan dari data. Dalam metode *machine learning*, untuk kasus pencarian kemiripan dari data *string*, teknik *similarity* menjadi teknik yang paling dominan dipakai. Teknik ini mampu membangun sebuah informasi melalui nilai atau dengan kata lain menerjemahkan sebuah data *string* menjadi data yang memiliki nilai (Mozafari and Tahayori, 2019) Lu dkk. (2019).

Dalam penerapannya, teknik ini dilakukan setelah terbentuknya data yang berpasangan. Selain untuk data *author matching*, teknik ini dipakai pula pada proses pengenalan emosi berbasis gambar. Teknik ini dilakukan dengan menbandingkan dua set data untuk melihat mana data yang sama dan mana data yang berbeda.

Di banyak penelitian tentang AND, teknik *similarity* yang banyak digunakan adalah dengan menggunakan koefisien jaccard, jaro, euclidean dan levenshtein. Adapun koefisien jaccard yang paling umum digunakan untuk mengukur kesamaan antara dua set data. Hasil yang dimunculkan adalah rentang 0% sampai dengan 100%. Semakin tinggi persentasenya, semakin mirip populasi kedua data tersebut (Ferreira, Gonçalves and Laender, 2012), (Tran, Huynh and Do, 2014), (Song, Kim and Kim, 2015), (Shen *et al.*, 2017), (Zhu and Li, 2013).

Keofisien jaccard didefinisikan dengan rumus matematis untuk menghasilkan pengukuran jarak antar string (rumus 2.2).

$$\boldsymbol{jaccard}\,(\boldsymbol{a},\boldsymbol{b}) = \frac{|\boldsymbol{a}\cap\boldsymbol{b}|}{|\boldsymbol{a}\cup\boldsymbol{b}|} = \frac{|\boldsymbol{a}\cap\boldsymbol{b}|}{|\boldsymbol{a}|+|\boldsymbol{b}|-|\boldsymbol{a}\cap\boldsymbol{b}|} \qquad \textbf{(2.2)}$$

Koefisien jaccard memiliki kelemahan dimana koefisien ini tidak memperhatikan term frequency (berapa kali suatu term terdapat di dalam suatu dokumen). Perlu diketahui, bahwa terms yang jarang muncul dalam suatu koleksi sangat bernilai dari sisi informasi, tetapi koefisien Jaccard tidak mempertimbangkan hal ini. Jadi kita butuh cara lain untuk menormalisasikannya. Namun untuk pengolahan data AND terutama pada poin *author matching,* teknik perhitungan koefisien jaccard dinilai lebih baik untuk digunakan (Ferreira, Gonçalves and Laender, 2012).

2.3. Ketidakseimbangan Data (Imbalance Data)

Ketidakseimbangan data atau biasa disebut *imbalance data* merupakan kondisi dimana ada satu kelas minoritas yang memiliki jumlah data yang sangat sedikit dibandingkan kelas lainnya (Vluymans, 2019). Hal ini merupakan permasalahan umum dalam proses klasifikasi data *machine learning,* dimana terjadi rasio pengamatan yang tidak seimbang pada setiap kelas. Kondisi ini sering ditemukan pada data diagnosis medis, penyaringan spam dan deteksi pada penipuan kartu kredit. Keadaan imbalanced mengakibatkan kinerja algoritma classifier standar menurun secara signifikan karena kebanyakan algoritma classifier standar mengasumsikan distribusi instance dalam kelas adalah balanced sehingga hasil klasifikasi lebih cenderung ke kelas mayoritas (Luque *et al.*, 2019).

Metode utama pada pendekatan level classifier adalah menyesuaikan operasi algoritma yang ada untuk membuat classifier lebih konduktif terhadap klasifikasi dalam kelas minoritas. Metode-metode yang digunakan pada level classifier adalah one-class learning, metode ensemble dan cost sensitive learning. Pendekatan pada level data merujuk pada berbagai teknik resampling (oversampling dan undersampling) dan sintesis data untuk memperbaiki kecondongan distribusi kelas data. Salah satu metode yang digunakan pada level data adalah Synthetic Minority Oversampling Technique atau lebih dikenal dengan nama SMOTE.

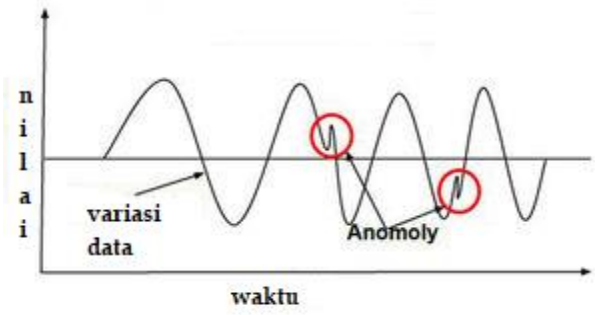
Pada dataset disambiguasi nama penulis atau AND, ketidakseimbangan atau imbalance data tersebut tidak dapat dilihat. Namun, teknik yang dipilih dalam proses pengolahan data sebelum masuk ke classifier, akan menentukan apakah data

menjadi seimbang atau tidak seimbang. Dalam hal ini, pemilihan teknik pairwise menggunakan kombinasi membuat data author yang positif (benar) menjadi lebih sedikit dibandingkan dengan data author yang negatif (bukan sebenarnya).

2.4. Deteksi Data Anomali (*Anomaly Detection*)

Anomaly detection merupakan sebuah teknik yang digunakan untuk mengidentifikasi pola-pola yang tidak biasa atau tidak sesuai perilaku yang diharapkan, dan hal tersebut lebih sering dikenal dengan outlier. Hal ini telah dilakukan dalam banyak kasus, mulai dari deteksi instrusi pada sistem komputer, lalu intrusi pada pada network (dilakukan dengan mendeteksi pola aneh dalam lalu lintas jaringan data untuk mengetahui jaringan terkena hacking atau tidak), pemantauan kesehatan (mendeteksi tumor ganas dalam pemindaian MRI), hingga penipuan dalan transaksi kartu kredit. (Fugate and Gattiker, 2002; Ferreira *et al.*, 2015; Dawoud, Shahristani and Raun, 2019).

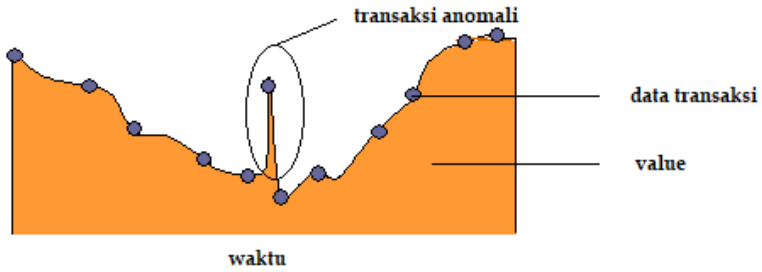
Dalam semua kasus yang pernah dipakai di atas, terdapat satu kesamaan dimana titik data itu berbeda dari item lain di dalam kumpulan data dan tidak dapat diidentifikasi dengan cepat karena tidak berada pada jalur yang diharapkan. Anomali detection dipusatkan pada pengenalan pada data yang langka atau sulit dikenali dalam barisan data.



Gambar 2.3. Pola Data Pada Konsep Anomaly Detection

Dalam proses klasifikasi data, anomali detection mengenali atau mengidentifikasi barang langka, peristiwa atau pengamatan yang menimbulkan kecurigaan dengan berbeda dari signifikan data mayoritas data (Wressnegger *et al.*, 2013). Anomali dapat dikategorikan menjadi :

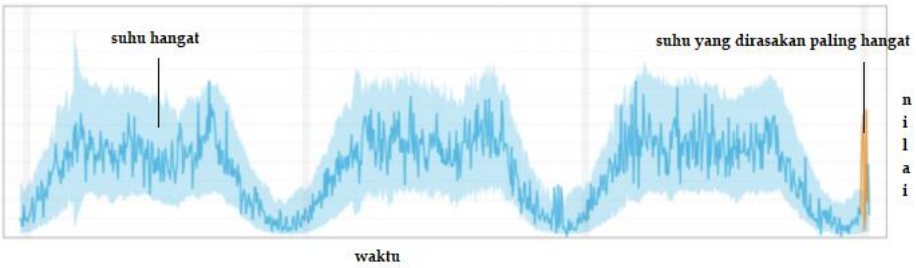
1. Anomali Terpusat (*Point anomalies*) : contoh data yang terlalu jauh atau yang terlalu berbeda dari signifikan data, misalnya pada penipuan kartu kredit.



Gambar 2.4. Plot Point Anomaly Detection

gambar diatas menjelaskan satu titik data yang sangat berbeda dari pola data atau alur data yang ada dalam penggunaan kartu kredit. Titik tersebut berjarak sangat dekat dengan nilai yang sangat signifikan dan hanya sekali dari beberapa kali penggunaan. Dalam ha ini, poin anomali merupakan data yang sangat berbeda dari kebiasaan data yang berjalan secara normal.

2. Anomali Kontekstual (*Contextual anomalies*) : penyimpagan data yang mengarah ke anomali didasarkan pada sesuatu yang kontekstual, misalnya pada kondisi cuaca, dima satu wilayah sedang merasakan puncak cuaca hangat (yang tidak pernah dirasakan sebelumnya) pada musim panas yang sedang berlangsung. Tapi hal tersebut tidak berlaku di wilayah lain pada musim yang sama. Hal tersebut dinamakan anomali karena berdasarkan kontekstual.



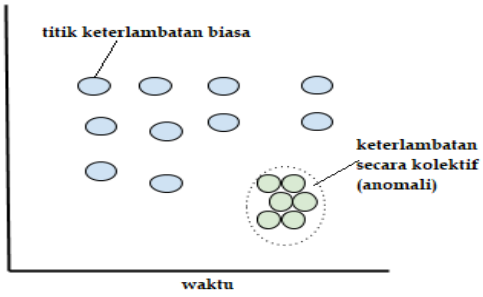
Gambar 2.5. Plot time-series cuaca pada Contextual Anomaly Detection

Gambar diatas merupakan data deret waktu tertentu selama periode tertentu yang menggambarkan keadaan suhu pada musim panas. Selama musim

panas tersebut cuaca tiap hari dirasakan sama, hingga pada hari tertentu cuaca dirasakan lebih hangat dibandingkan hari lainnya (digambarkan dengan warna orange).

3. Anomali Kolektif (*Collective anomalies*) : penyimpangan data pada anomali kolektif tidak hanya dengan melihat titik data secara individual, tetapi juga menganalisis perilaku secara kolektif. penggunaan serangkaian data secara kolektif untuk menimbulkan anomali.

   Gambar dibawah ini menganalogikan keadaan anomali pada proses pengiriman barang atau pasokan pada sebuah industri tekstil. Pengiriman yang terlambat sangat umum di industri seperti ini. Tetapi pada hari tertentu, jika ada banyak keterlambatan pengiriman pada pesanan maka mungkin perlu penyelidikan lebih lanjut. Pengiriman yang tertunda tidak berkontribusi terhadap hal ini secara individu tetapi ringkasan kolektif dipertimbangkan ketika menganalisis situasi seperti ini.

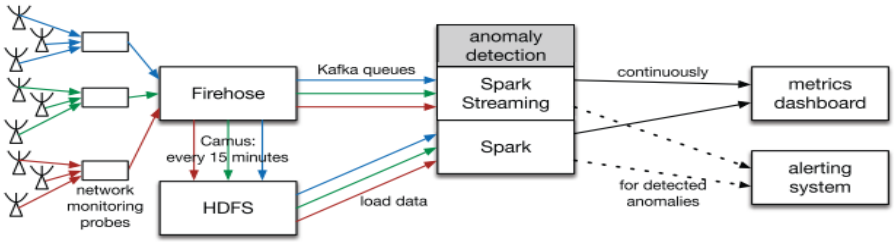


Gambar 2.6. Plot keterlambatan distribusi barang dalam Collective Anomaly Detection

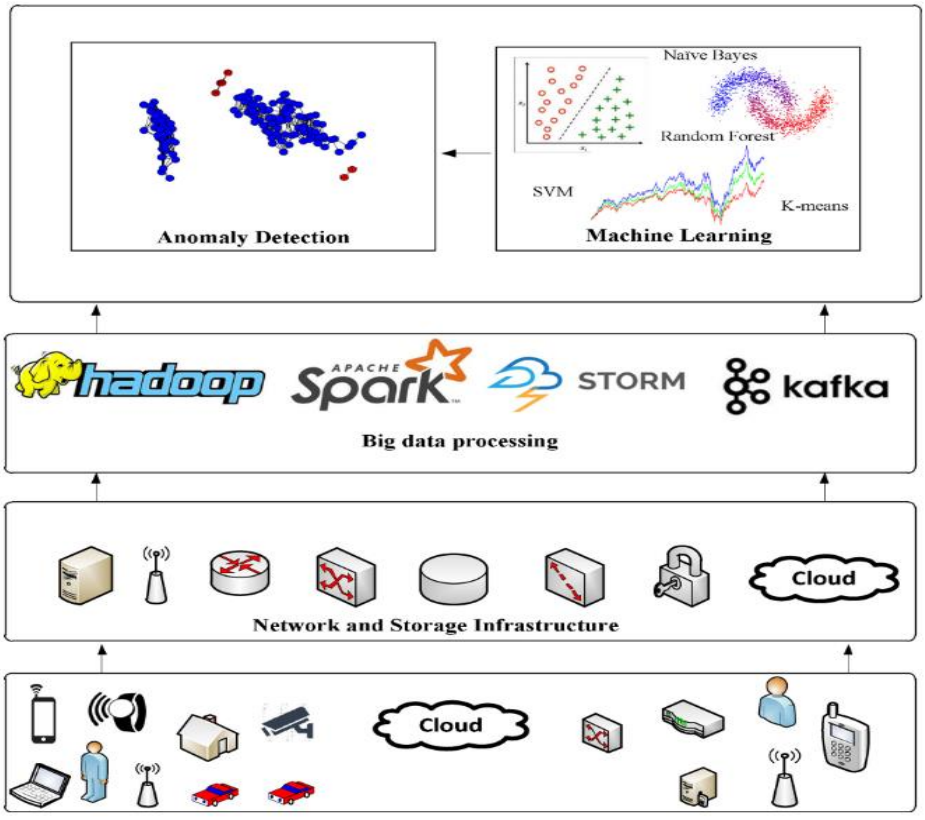
Anomaly detection sangat penting hampir di semua bidang disiplin ilmu (contoh, fisika, keuangan, machine learning dan cyber security). dalam machine learning atau disiplin ilmu manapun, kualitas data sama pentingnya dengan kualitas model prediksi ataupun klasifikasi.

Pada umumnya, anomaly detection ini digunakan pada kumpulan data yang besar (big data) dimana kualitas data menjadi acuan pada informasi yang dihasilkan. Jaminan atas kualitas data harus dengan melakukan identifikasi awal terhadap pola-pola yang tidak biasa yang terjadi di dalam data (Rettig *et al.*, 2015).

Penelitian tentang big data ini pernah dilakukan dengan mendeteksi sensor pada berbagai perangkat pintar yang dikomunikasikan pada jaringan, perangkat tersebut melakukan penyimpanan pada cloud dan media penyimpanan lainnya lalu diproses dengan teknologi big data dan hasilnya digunakan untuk analisis anomali dengan menggunakan machine learning (Luque *et al.*, 2019).



Gambar 2.7. Integrasi Anomaly Detection pada Big Data



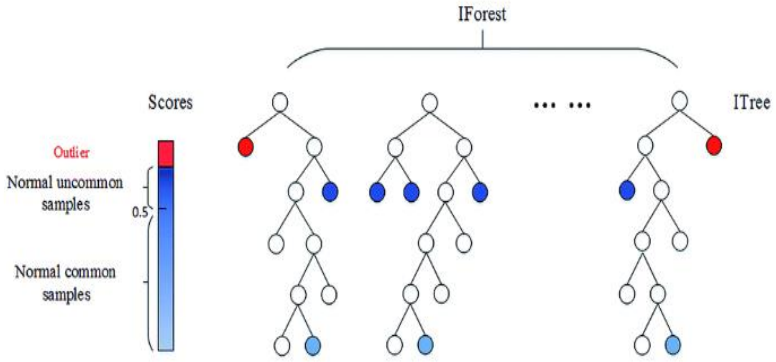
Gambar 2.8. Penerapan Anomaly Detection pada Big Data

Dalam penggunaan metode anomali detection dalam machine learning, kita perlu memastikan dan menyelidiki secara menyeluruh bahwa model yang digunakan mampu mengidentifikasi anomali secara efektif dan konsisten (Ferreira

et al., 2015). Model algoritma yang digunakan dalam mendetteksi anomali adalah isolation forest dan local outlier factor.

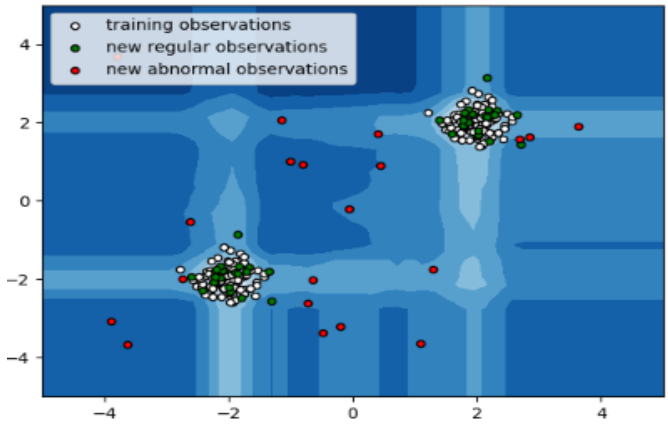
2.4.1. Algoritma Isolation Forest

Isolation forest merupakan salah satu model algoritma terbaru dalam melakukkan klasifikasi anomaly detection. Algoritma tersebut didasarkan pada fakta bahwa terdapat data yang sedikit dan sangat berbeda dari data yang dominan, dimana berdasarkan hal tersebut dapat dijelaskan bahwa anomali bersifat rentan terhdap mekanisme yang disebut isolasi. Metode ini sangat bermanfaat secara fundamental karena memperkenalkan penggunaan isolasi sebagai cara yang efektif dan efisien dalam mendeteksi anomali. Selain itu, metode ini algoritma dengan kompleksitas waktu linier rendah dan kebutuhan memori yang kecil yang dapat membangun model berkinerja baik dengan menggunakan sub sampel kecil ukuran tetap, terlepas dari ukuran kumpulan data (Yao *et al.*, 2019).



Gambar 2.9. Ilustrasi Tree Isolation Forest

Gagasan inti dari algorima isolation forest adalah bahwa jumlah titik yang abnormal biasanya kecil, dan ada perbedaan yang signifikan antara titik normal dan atribut. Algoritma ini memiliki pola dasar pada model decision tree yang dapat mem breakdown proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga proses pengambilan keputusan akan lebih menginterpretasikan solusi dari permasalahan (Zhang, Wang and Zhang, 2019).



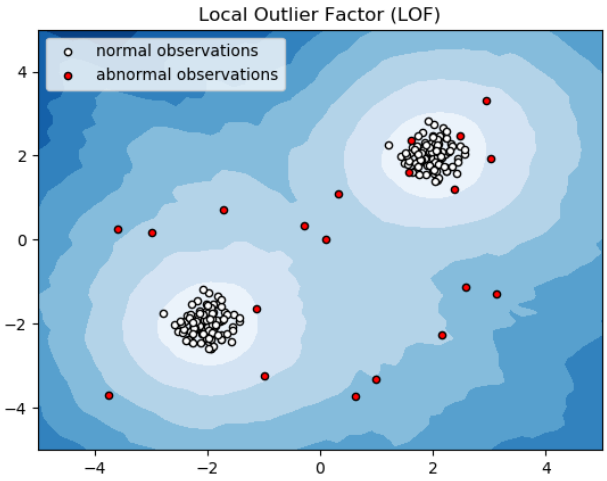
Gambar 2.10. Ilustrasi Algoritma Isolation Forest

Isolation forest merupakan salah satu cara yang efisien untuk melakukan deteksi outlier atau anomaly dalam dataset yang berdimensi tinggi. Algoritma ini melakukan pengamatan dengan memilih fitur secara acak dan selanjutnya memilih nilai split diantara nilai maksimum dan minimum dari fitur yang dipilih (Qin and Lou, 2019; Yao *et al.*, 2019).

2.4.2. Algoritma Local Outlier Factor

Local Outlier Factor (LOF) merupakan algoritma pendeteksian data anomali atau outlier yang berbasis pada kepadatan data. Algoritma ini menemukan data outlier dengan melakukan perhitungan dari kelompok data yang menyimpang dari titik data normal yang diberikan. Algoritma ini menjadi salah satu solusi pada pendekatan deteksi anomali pada sekelompok dataset yang sangat imbalance. Metode penghitungan kepadatan data pada algoritma ini dilakukan berdasarkan kepadatan antara masing-masing titik data dengan titik data berikutnya atau tetangganya, dimana semakin rendah kepadatan data pada suatu titik maka akan semakin besar kemungkinan diidentifikasi sebagai outlier (Cheng, Zou and Dong, 2019).

Sebagai algoritma yang populer dalam proses pendeteksian outlier, LOF sangat sering digunakan pada pada klasifikasi aliran data statis yang di dalam aliran data tersebut menunjukkan adanya sekelompok data yang berbeda dari keseluruhan data pada umumnya (Yang *et al.*, 2019).



Gambar 2.11. Ilustrasi Algoritma Local Outlier Factor

Berdasarkan gambar 2.10. dapat dijelaskan bahwa algoritma ini melakukan proses pendeteksian pada aliran data atau kumpulan data yang ada terlebih dahulu dengan melakukan penilaian terhadap kepadatan data. Ketika data lebih rendah dibandingkan keseluruhan data secara umum, maka diidentifikasi sebagai outlier.

Secara umum pola-pola pendeteksian dalam algoritma LOF dapat dibagi menjadi 2 macam (John and Naaz, 2019), yaitu :

1. Global outlier, jika objek data yang secara signifikan memiliki jarak yang besar dibandingkan dengan barisan data tetangganya, baik data sebelum ataupun setelahnya.
2. Local outlier, ketika objek data memiliki jarak yang relatif besar dibandingkan pada jarak rata-rata tetangganya.

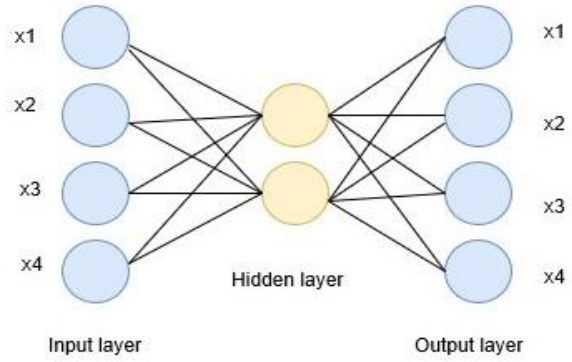
Pada penelitian lainnya, algoritma LOF juga digunakan dalam proses deteksi intrusi pada suatu struktur jaringan. Deteksi dilakukan dengan melacak bentuk barisan data atau sinyal atau sekumpulan data yang memiliki karakteristik yang berbeda secara umum dari keseluruhan data atau sinyal di dalam jaringan tersebut (Kharitonov and Zimmermann, 2019).

Dari keterangan tersebut diatas dapat disimpulkan bahwa LOF merupakan skor atau nilai yang menunjukkan seberapa besar kemungkinan suatu titik data menjadi anomali atau outlier. Atau dengan kata lain, algoritma ini mengkonfirmasi dan mengidentifikasi prilaku data yang tidak diharapkan. Metode ini lebih banyak

diasumsikan dengan menghitung skor poin dari barisan distribusi data dengan probabilitas yang sama.

2.4.3. Autoencoder

Teknik *autoencoder* disebut juga teknik *dimentionality reduction*. Dimana *autoencoder* mampu merepresentasikan data kemudian merekonstruksi nya kembali. Karena tujuan *encoder* untuk kompresi, bentuk terkompresi haruslah memiliki dimensi lebih kecil dari dimensi *input*. *Neural network* mampu melakukan kompresi dengan baik karena *neural network* mampu menemukan *hidden structure* dari data. Berikut arsitektur *autoencoder* dari penelitian Kunang dkk. (2019) pada gambar 2.1



Gambar 2.12. Arsitektur *Autoencoder* Kunang dkk. (2019)

Ukuran *performance measure* untuk *autoencoder* adalah mengukur *loss* Naway dan Li (2019). *Input* matriks X pada *autoencoder*, kemudian ingin *autoencoder* tersebut menghasilkan matriks yang sama, maka *output* harus sama dengan *input*. Perhitungan *autoencoder* didapat dari rumus Kunang dkk. (2019).

$$Y = f(X) = s(WX + b_X) \quad (2.1)$$

$$X' = g(Y) = s(W'Y + b_Y) \quad (2.2)$$

Keterangan :

$Y = f(X)$ = fungsi *encoded*
$X' = g(Y) =$ fungsi *decoded*
s = fungsi aktivasi
W = bobot *encoded*
W' = bobot *decoded*

b_X = bias *encoded* b_Y= bias *decoded*

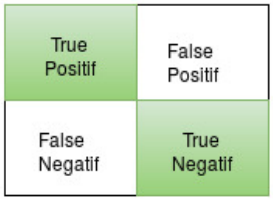
Pada konsep dan metode anomali detection, autoencoder digunakan dengan cara un-supervised learning atau tanpa pembelajaran. Ini dilakukan dengan menarik kesimpulan dengan terlebih dahulu menentukan treshold atas batas titik data yang bisa dikategorikan sebagai anomali.

Bentuk paling sederhana dari autencoder adalah feedforward, jaringan saraf non-berulang sangat mirip dengan banyak perceptron lapisan tunggal yang membuat perceptron multilayer (MLP) - memiliki lapisan input, lapisan output dan satu atau lebih lapisan tersembunyi yang menghubungkan mereka - tetapi dengan layer output memiliki jumlah node yang sama dengan layer input, dan dengan tujuan merekonstruksi inputnya sendiri (alih-alih memprediksi nilai target Y yang diberikan input X). Oleh karena itu, autoencoder adalah model pembelajaran yang tidak diawasi.

Dan dalam proses klasifikasi anomaly detection dengan menggunakan algoritma autoencoder, model yang dipakai adalah model paling sederehana untuk mendapatkan kesimpulan di titik mana barisan data tersebut dapat dikenali sebagai anomali.

2.5. Confusion Matrix

Proses Validasi bertujuan untuk mengetahui apakah simulasi dari sistem klasifikasi *author matching* ini sesuai dengan prediksi yang telah dilakukan. Proses validasi ini akan dilakukan dengan menggunakan *confusion matrix*. Matrix ini menggambarkan kinerja klasifikasi dari data percobaan yang berisi informasi prediksi data. Berikut gambar 2.4 menunjukkan *confusion matrix* untuk klasifikasi dua kelas (biner) (Luque *et al.*, 2019).



Gambar 2.12 *Confusion Matrix*

Dalam proses klasifikasi mempunyai dua label data yaitu positif dan negatif. Klasifikasi dari dua label ini akan menghasilkan empat nilai dari *confusion matrix* yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN). Sistem klasifikasi data *author matching* akan menghasilkan nilai pada *confusion matrix*. TP adalah jumlah data author yang benar (positif) ketika proses prediksi dicocokkan dengan jumah data yang benar (positif) secara aktual, TN adalah jumlah *author* yang tidak sama (negative) pada saat prediksi dibandingkan dengan jumlah yang tidak sama (negative) secara aktual. FP adalah jumlah *author* yang benar namun terdeteksi *terdeteksi sebagai author yang tidak sama*, dan FN adalah jumlah *author yang tidak sama* terdeteksi sebagai author yang sama. Berikut beberapa parameter evaluasi dari hasil *confusion matrix* (Luque *et al.*, 2019).

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.3}$$

Parameter akurasi pada persamaan 2.3 untuk mengukur keakuratan sistem dari proses klasifikasi yaitu dengan menjumlahkan nilai *true positive* dan *true negative* dibagi dengan seluruh data matriks yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN).

$$Presisi = \frac{TP}{TP+FP} \tag{2.4}$$

Parameter presisi pada persamaan 2.4 untuk mengukur tingkat presisi dari sistem klasifikasi yaitu dengan membagi *true positive* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP) .

$$Sensitivitas = \frac{TP}{TP+FN} \tag{2.5}$$

Parameter sensitivitas pada persamaan 2.5 untuk mengukur tingkat sensitivitas dari sistem klasifikasi yaitu dengan membagi *true positive* dengan menjumlahkan *true positive* (TP) dan *false negative* (FN) .

$$Spesifisitas = \frac{TN}{TN+FP} \quad (2.6)$$

Parameter spesifisitas pada persamaan 2.6 untuk mengukur tingkat spesifisitas dari sistem klasifikasi yaitu dengan membagi *true negative* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP) .

$$F1\ Score = \frac{2\times Presisi \times Sensitvitas}{Presisi+Sensitivitas} \quad (2.7)$$

Parameter F1 Score pada persamaan 2.7 untuk mengukur perbandingan rata-rata presisi dan sensitivitas dari sistem klasifikasi yaitu dengan membagi *true negative* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP).

2.6. Kurva ROC

Dalam banyak aplikasi, kurva karakteristik operator penerima (ROC) digunakan untuk menunjukkan bagaimana prediktor membandingkan dengan hasil yang sebenarnya. Salah satu keuntungan besar dari analisis ROC adalah bahwa ia adalah ambang batas agnostik; kinerja alat prediksi diperkirakan tanpa ambang batas tertentu dan juga memberikan kriteria untuk memilih ambang batas optimal berdasarkan fungsi atau tujuan biaya tertentu. Biasanya, analisis ROC menunjukkan bagaimana sensitivitas (tingkat positif sejati) berubah dengan spesifisitas yang bervariasi (tingkat negatif sejati atau 1 - tingkat positif palsu) untuk ambang yang berbeda. Analisis juga biasanya menimbang positif palsu dan negatif palsu sama. Dalam analisis ROC, kemampuan prediktif suatu variabel biasanya dirangkum oleh area di bawah kurva (AUC), yang dapat ditemukan dengan mengintegrasikan area di bawah segmen garis. (Muschelli, 2019)

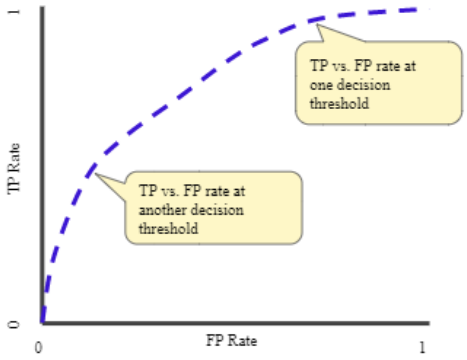
2.6.1 Kurva ROC

Kurva ROC (kurva karakteristik operasi penerima) adalah grafik yang menunjukkan kinerja model klasifikasi di semua ambang klasifikasi. Kurva ini memplot dua parameter:

- True Positive Rate

- False Positive Rate

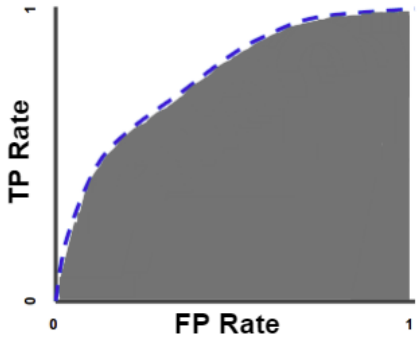
Kurva ROC memplot TPR vs FPR pada threshold klasifikasi yang berbeda. Menurunkan threshold klasifikasi mengklasifikasikan lebih banyak item sebagai positif, sehingga meningkatkan False Positive dan True Positive. Gambar berikut menunjukkan kurva ROC yang khas.



Gambar 2.13 Kurva False Positive Rate

2.6.2 AUC: Area Di Bawah Kurva ROC

AUC singkatan dari "Area di bawah Kurva ROC." Artinya, AUC mengukur seluruh area dua dimensi di bawah seluruh kurva ROC dari (0,0) hingga (1,1).

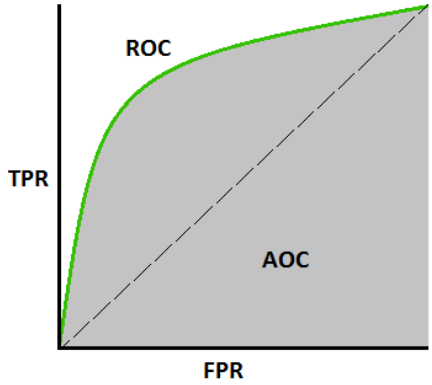


Gambar 2.14. Kurva AUC

AUC menyediakan ukuran kinerja agregat di semua threshold klasifikasi yang memungkinkan. Salah satu cara menafsirkan AUC adalah sebagai probabilitas bahwa model tersebut memeringkat contoh positif acak lebih tinggi daripada contoh negatif acak.

Kurva AUC - ROC adalah pengukuran kinerja untuk masalah klasifikasi di berbagai pengaturan thresholds. ROC adalah kurva probabilitas dan AUC mewakili derajat atau ukuran keterpisahan. Ini memberitahukan berapa banyak model yang mampu membedakan antar kelas. Semakin tinggi AUC, semakin baik modelnya dalam memprediksi 0s sebagai 0s dan 1s sebagai 1s. Dengan analogi, semakin tinggi AUC, semakin baik modelnya membedakan antara pasien dengan penyakit dan tanpa penyakit.

Kurva ROC diplot dengan TPR terhadap FPR di mana TPR berada pada sumbu y dan FPR pada sumbu x.



Gambar 2.15 Kurva ROC

2.6.3 Istilah yang digunakan dalam Kurva AUC dan ROC

TPR (True Positive Rate) / Recall /Sensitivity $= \frac{TP}{TP+FN}$

Specificity $= \frac{TN}{TN+FP}$

FPR $= 1 - Specificity$

$= \frac{FP}{TN+FP}$

2.6.4 Spekulasi kinerja model

Model yang sangat baik memiliki AUC dekat dengan 1 yang berarti memiliki ukuran pemisahan yang baik. Model yang buruk memiliki AUC mendekati 0 yang berarti memiliki ukuran pemisahan terburuk. Contohnya memprediksi 0s sebagai 1s dan 1s sebagai 0s. Dan ketika AUC adalah 0,5, itu berarti model tidak memiliki kapasitas pemisahan kelas sama sekali. Ketika dua kurva tidak *overlap* sama sekali model berarti memiliki ukuran pemisahan yang ideal. Bearti benar-benar mampu

membedakan antara kelas positif dan kelas negatif. Ketika AUC sekitar 0, model sebenarnya membalas kelas. Artinya, model memprediksi kelas negatif sebagai kelas positif dan sebaliknya.

2.6.5 Hubungan antara Sensitivitas, Spesifisitas, FPR, dan Threshold

Sensitivitas dan Spesifisitas berbanding terbalik satu sama lain. Jadi ketika kita meningkatkan Sensitivitas, kekhususan berkurang dan sebaliknya.

Sensitivity⬆, Specificity⬇ and Sensitivity⬇, Specificity⬆

Ketika menurunkan threshold, akan mendapatkan nilai lebih positif sehingga meningkatkan sensitivitas dan mengurangi spesifisitas. Demikian pula, ketika meningkatkan threshold, akan mendapatkan lebih banyak nilai negatif sehingga mendapatkan spesifisitas yang lebih tinggi dan sensitivitas yang lebih rendah. Seperti diketahui FPR adalah 1 - spesifisitas. Jadi ketika meningkatkan TPR, FPR juga meningkat dan sebaliknya.

TPR⬆, FPR⬆ and TPR⬇, FPR⬇

2.6.6 menggunakan kurva AUC ROC untuk model multi-kelas

Dalam model multi-kelas, dapat memplot angka N Kurva ROC AUC untuk kelas nomor N menggunakan metodologi One vs ALL. Jadi sebagai contoh, jika memiliki tiga kelas bernama X, Y dan Z, Disana akan memiliki satu ROC untuk X diklasifikasikan terhadap Y dan Z, ROC lain untuk Y diklasifikasikan ke X dan Z, dan yang ketiga dari Z diklasifikasikan ke Y dan X.

BAB III
METODOLOGI PENELITIAN

Bab ini berisi mengenai kerangka kerja dari penelitian dan analisis kebutuhan perangkat lunak. Metodologi yang digunakan pada bab ini adalah berfokus pada klasifikasi *author matching* dengan menggunakan pendekatan anomali *detection*. Metodologi dalam penelitian ini adalah penelusuran pustaka, persiapan data, pra-pengolahan dengan *pairless combination*, proses pengklasifikasian dengan menggunakan algoritma yang dipakai dalam pendekatan *anomaly detection* yaitu *IsolationForest, Local Outlier Factor (LOF)* dan dengan menggunakan algoritma *AutoEncoder.* Tahapan selanjutnya adalah analisis hasil dan penarikan kesimpulan. Tahapan dalam penyelesaian penelitian Tesis ini adalah bagaimana pengklasifikasian *author matching* dengan menggunakan pendekatan *anomaly detection* dalam 3 algoritma yang telah disebutkan diatas.

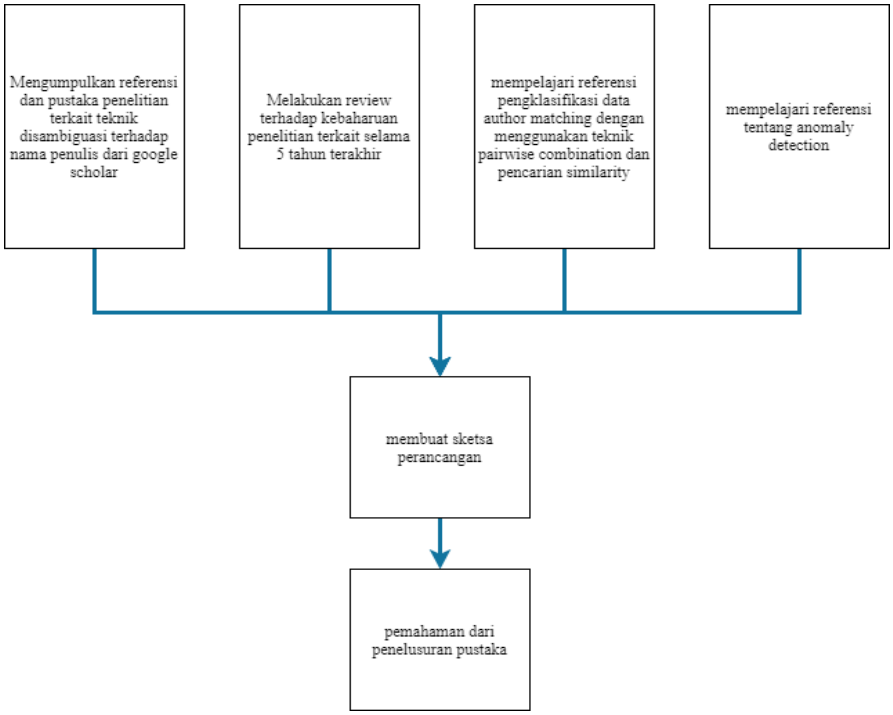
3.1 Kerangka Kerja Penelitian

Secara garis besar, adapun langkah-langkah pada metodologi penelitian yang digunakan untuk membantu dalam penyusunan tesis dan hasil dari penelitian ini, memerlukan susunan kerangka kerja yang jelas tahapan-tahapannya sehingga mampu menghasilkan prediksi dari proses klasifikasi yang terukur dan akurat. Kerangka kerja penelitian yang digunakan seperti pada Gambar 3.1, dimulai dari kegiatan tinjauan pustaka untuk mengumpulkan referensi, proses persiapan data dimulai dengan pemilihan dataset yang sesuai, pra-pengolahan data dengan *converting and deletion data*, *concatenation, remove stopword* dan *pairwise combination,* dan *year difference.* Dilanjutkan dengan proses ekstraksi fitur dengan teknik pencarian kemiripan *(similarity)* dan perbandingan *(comparation).* Kemudian dilakukan proses pengklasifikasian dengan Algoritma *Isolation Forest*, *Local Outlier Factor (LOF)* dan *AutoEncoder.* Meudian dilanjutkan dengan analisis hasil dari proses klasifikasi dan terakhir dilakukan penarikan kesimpulan atas penelitian ini.

Gambar 3.1. Kerangka Kerja Penelitian

3.2 Penelusuran Pustaka

Tahap ini dilakukan pencarian dan pembelajaran dari beberapa pustaka dan literatur yang relevan terkait landasan-landasan teori yang diperoleh dari berbagai publikasi ilmiah dengan penelitian yang akan dikerjakan. Tahapan dalam pembelajaran pustaka dapat dilihat pada Gambar 3.2.



Gambar 3.2 Tahapan Penelusuran Pustaka

3.3 Persiapan Data

Dalam tahapan ini, dilakukan survei terhadap ketersediaan dataset yang berisi data author lengkap (beserta ide, judul penelitian, tahun terbit dan lain-lain). Dari beberapa referensi yang digunakan, terdapat 2 kelompok data (dataset) yang dapat digunakan untuk melakukan klasifikasi author matching yaitu dengan menggunakan dataset *vietnamnese author* (Tran, Huynh and Do, 2014) dan *SCAD-zBMATH dataset* (Müller, Reitz and Roy, 2017). Dari penelusuran terhadap kedua dataset tersebut, dataset SCAD-zBMATH dapat dikategorikan sebagai dataset yang lengkap dan paling homogen untuk dilakukan proses klasifikasi terhadap nama author. Dataset ini menyediakan beberapa file publikasi lengkap dengan format yang berbeda-beda pada setiap filenya dimana masing-masing file dapat menjadi level pengujian dari algoritma klasifikasi yang akan digunakan. Selain itu, dataset ini memenuhi syarat struktur data dalam proses disambiguasi nama penulis (Müller, Reitz and Roy, 2017) yaitu :

1. Kasus dimana satu nama penulis muncul di barisan berikutnya dengan perbedaan yang cukup signifikan (misalnya dengan menyingkat nama depan atau nama tengah)
2. kasus di mana nama penulis yang sebenarnya ditulis dalam alfabet non-barat (mis., Asia atau Sirilik) dan muncul di *header* publikasi dalam beberapa versi, dapat memunculkan contoh kasus 1.
3. kasus-kasus publikasi oleh penulis yang kurang produktif atau penulis dengan hanya sedikit kolaborator, di mana informasi penulis bersama yang tidak tersedia, dan
4. kasus-kasus publikasi dari bidang ilmiah atau komunitas yang umumnya cenderung memiliki jumlah penulis bersama yang lebih sedikit.

Dari penjelasan diatas dapat disimpulkan bahwa dataset yang dapat memenuhi syarat untuk dilakukan proses disambiaguasi adalah dataset yang memiliki data lengkap mulai dari author yang sangat heterogen hingga author yang jarang sekali muncul dalam barisan data, hingga data judul dan author pendamping yang juga bisa muncul sebagai author utama di judul publikasi yang berbeda.

Berdasarkan hal tersebut, maka dataset yang digunakan adalah data nama-nama penulis lengkap dengan judul jurnal serta tahun terbit yang diambil dari dataset SCAD-zBMATH dengan judul featured-dataset-merged yang terdiri dari 10.160 baris data publikasi (Müller, Reitz and Roy, 2017). Dataset ditunjukkan pada Tabel 3.1.

Tabel 3.1

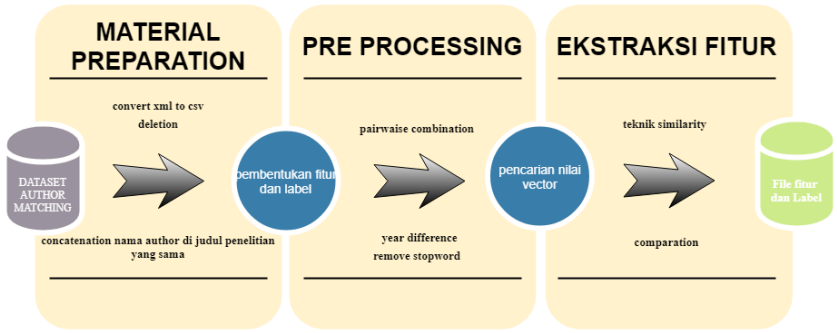
Spesifikasi Dataset

Dataset	Jumlah Data
SCAD-zBMATH	11.924

3.4 Pengolahan Data *Author Matching*

Pada tahapan ini dilakukan proses pengolahan data untuk mempersiapkan data sebelum dimasukkan kedalam proses klasifikasi. Pada penelitian ini menggunakan klasifikasi biner untuk pencocokan nama *author*. Jumlah data yang

digunakan berjumlah 11.924 baris data. Tahapan dari proses pengolahan data pada penelitian ini dapat dilihat pada gambar 3.3 berikut ini.



Gambar 3.3 Alur Pengolahan Data *Author Matching*

Pada gambar 3.3 diatas menggambarkan langkah-langkah dalam melakukan pengolahan data dalam penelitian ini sebagai berikut:

3.4.1. Mempersiapkan Data (Material Preparation)

Tahapan yang dilakukan untuk menyiapkan data *author matching* sebelum masuk ke proses lainnya. Pada gambar 3.4 dibawah ini menunjukkan bentuk data author matching awal atau *RAW* data dari dataset *SCAD-ZbMath* dengan judul file *featured-dataset-merged* yang masih dalam bentuk xml.

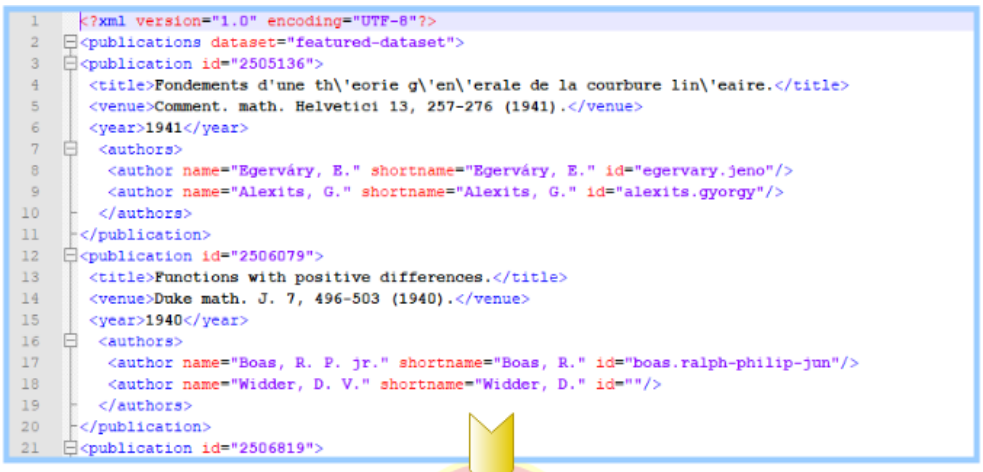
```
<?xml version="1.0" encoding="UTF-8"?>
<publications dataset="featured-dataset">
<publication id="2505136">
 <title>Fondements d'une th\'eorie g\'en\'erale de la courbure lin\'eaire.</title>
 <venue>Comment. math. Helvetici 13, 257-276 (1941).</venue>
 <year>1941</year>
  <authors>
   <author name="Egerváry, E." shortname="Egerváry, E." id="egervary.jeno"/>
   <author name="Alexits, G." shortname="Alexits, G." id="alexits.gyorgy"/>
  </authors>
</publication>
<publication id="2506079">
 <title>Functions with positive differences.</title>
 <venue>Duke math. J. 7, 496-503 (1940).</venue>
 <year>1940</year>
  <authors>
   <author name="Boas, R. P. jr." shortname="Boas, R." id="boas.ralph-philip-jun"/>
   <author name="Widder, D. V." shortname="Widder, D." id=""/>
  </authors>
</publication>
<publication id="2506819">
```

Gambar 3.4 Dataset SCAD-zBMATH.xml

Bentuk dari dataset diatas tidak dapat langsung masuk ke dalam proses selanjutnya baik itu pra pengolahan, ekstrasksi fitur maupun langsung masuk ke dalam proses klasifikasi. Oleh karena itu, diperlukan langkah selanjutnya dalam tahapan persiapan data atau *material preparation* dengan tujuan terbentuknya struktur data yang dapat diolah pada proses selanjutnya.

3.4.1.1.Converting Dataset dan Deletion Dataset

Proses convert dilakukan untuk mengubah ekstensi file dari dataset yang semula dalam bentuk xml menjadi bentuk csv dengan tujuan membentuk file menjadi kolom-kolom atribut sehingga bisa diterjemahkan dengan jelas untuk proses pra pemrosesan data selanjutnya. Proses dan hasil dari proses convert tersebut dapat dilihat pada Gambar 3.5 dan tabel 3.2.



tion/#id	/publication/@id	/publication/authors/author/@id	/publication/authors/auth...ame	/publication/authors/author/@shortname	/publication/title	/publication/venue
1	2505136	egervary.jeno	Egerváry, E.	Egerváry, E.	Fondements d'une th\'eorie g\'en\'erale de la ...	Comment. math. Helvetici 13, 257-276 (1941).
1	2505136	alexits.gyorgy	Alexits, G.	Alexits, G.	Fondements d'une th\'eorie g\'en\'erale de la ...	Comment. math. Helvetici 13, 257-276 (1941).
2	2506079	boas.ralph-philip-jun	Boas, R. P. jr.	Boas, R.	Functions with positive differences.	Duke math. J. 7, 496-503 (1940).
2	2506079	NaN	Widder, D. V.	Widder, D.	Functions with positive differences.	Duke math. J. 7, 496-503 (1940).
3	2506819	boas.ralph-philip-jun	Boas jr., R. P.	Boas jr., R.	A correction to ``A note on linear functionals''.	Bull. Amer. math. Soc. 46, 566 (1940).
3	2506819	NaN	Tukey, J. W.	Tukey, J.	A correction to ``A note on linear functionals''.	Bull. Amer. math. Soc. 46, 566 (1940).
4	2509078	boas.ralph-philip-jun	Boas, R. P. jr.	Boas, R.	An inversion formula for the Laplace integral.	Duke math. J. 6, 1-26 (1940).
4	2509078	NaN	Widder, D. V.	Widder, D.	An inversion formula for the Laplace integral.	Duke math. J. 6, 1-26 (1940).

Gambar 3.5 Proses dan Hasil Converting Dataset

Tabel 3.2

Hasil Convert dataset dari xml ke csv

id_author	name	shortname	tittle	venue
Egevary.jeno	Egervary.E	Egervary.E.	Fondements d'une	Comment. Math. Helvetici 13, 257-276 (1941)
Alexits.gyorgy	Alexits, G	Alexits, G	Fondements d'une	Comment. Math. Helvetici 13, 257-276 (1941)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, R	Function with positive differences	Duke math. J.7, 496-503 (1940)
NaN	Widder, D.V.	Widder, D.	Function with positive differences	Duke math. J.7, 496-503 (1940)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, jr. R.	A corrections to "A note on linear	Bull Amer. Math Soc. 46. 566 (1940)
NaN	Tukey, J.W.	Tukey J	A corrections to "A note on linear	Bull Amer. Math Soc. 46. 566 (1940)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, R	An inversion formula for the Laplace integral	Duke math. J. 6 1-26 (1940)

Proses diatas menghasilkan dataset sejumlah 11.924 baris dengan 10 kolom (*dataset, publication_id, publication_id2, author_id, author_name, author_shortname, publication_title, publication_venue, publication_year, dan publication_year2*). Setelah itu, dilakukan penghapusan (*deletion*) kolom yang tidak akan memberikan informasi dalam proses klasifikasi. Hasil dari penghapusan kolom tersebut menyisakan 6 kolom yaitu *id_author, name, authors, title, venue dan year.*

3.4.1.2.Penggabungan Data (Concatenate)

Proses selanjutnya setelah dilakukan *convert* adalah dengan melakukan penggabungan baris atau kolom dengan tujuan mengurutkan data publikasi menjadi 1 baris pada 1 judul publikasi. Hanya 1 kolom yang akan dilakukan penggabungan yaitu kolom authors sehingga setiap baris akan menampilkan 1 judul publikasi

dengan jumlah author yang bisa lebih dari 1dengan tujuan agar proses klasifikasi dapat menghasilkan hasil yang lebih baik (tabel 3.3).

Tabel 3.3

Hasil Penggabungan Data atau Concatenate

Id	Name	Authors	Title	Venue	Year
Egervary.jeno	Egervary.E	Egervary.D.Alexits,G	Fondements d'une th	Comment math. Helvetici 13, 257-276 (1941).	1941
alexits.gyorgy	Alexits,G	Egervary.E.Alexits,G	Fondements d'une th	Comment math. Helvetici 13, 257-276 (1941).	1941
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Widder,D.V	Functions with positive differences	Duke math J.7, 496-503 (1940)	1940
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Tukey,D.V	A corrections to "A note on linear functionals	Bull Amer. math. Soc 46, 566 (1940)	1940
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Widder,D.V	An inversion formula for the laplace integral	Duke math, J.6, 1-26 (1940)	1940

Hasil dari proses *concatenate* tersebut dapat dijadikan sebagai bentuk dataset baru yang memberikan informasi kuat dan terstruktur siap untuk diolah ke proses berikutnya dalam pengolahan data *author matching*. Proses digambarkan secara ringkas ke dalam urutan algoritma pada gambar 3.6, dan proses ini menghasilkan jumlah baris data yang baru yaitu 10.160 baris data.

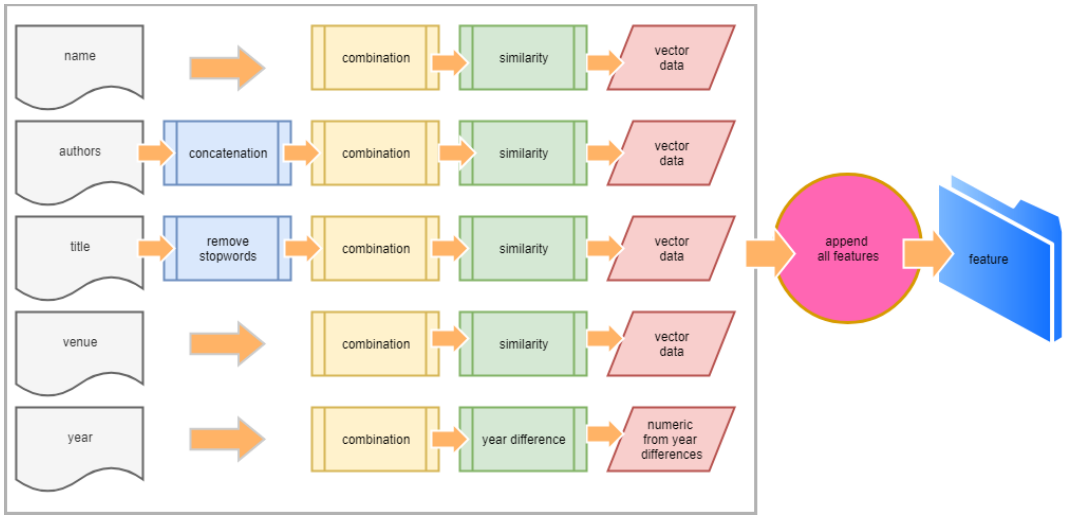
```
// declaration
dataset featured-dataset-merged.xml string;

// description
open (dataset);

xlsdataset <-- dataset Convert.ToXls;

open (xlsdataset);
// declaration:
begin
    title $prev_row;
    title $now;
    $prev_row = $this->title->CurrentValue;

for i <--0; i < 11924; i++;
{
    // perbandingan antar baris dan menggabungkan:
    if ($this->RowCnt > 0)
    {
        echo $this->RowCnt . ". now - prev = " . $prev_row;
        echo concatenate (now,prev);
        echo $now;
        echo "<br>";
    }

    //
    if ($this->title->CurrentValue <> $prev_row)
    {
        $now = $this->Tahun->CurrentValue;
    } else {
        $now = $prev_row;
    }

}
end
```

Gambar 3.6 Struktur Algoritma Sederhana Tahapan Persiapan Data

3.4.2. Pra Pemrosesan (Pre Processing)

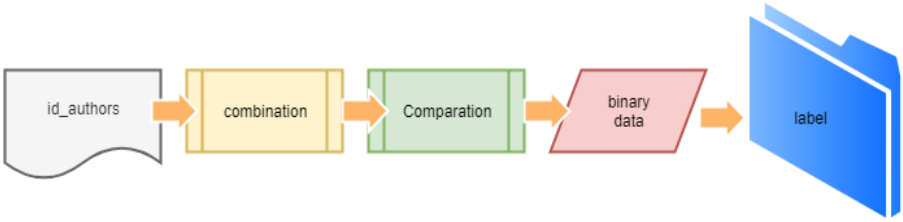
Pada tahap ini, dilakukan proses untuk membentuk data yang terstruktur dengan baik sehingga dapat memberikan informasi dan mudah dipelajari dalam proses klasifikasi. Tahap ini dilakukan dengan tujuan membentuk dataset menjadi terpisah 2 file yang masing-masing akan difungsikan sebagai fitur dan label. Proses pemisahan file tersebut dilakukan agar pendefisian label sebagai target pembelajaran *(learning)* menjadi lebih mudah.

Untuk fitur, kolom yang akan digunakan dan dilakukan pra pemrosesan adalah name, authors, title, venue dan year (gambar 3.5) sedangkan untuk label hanya menggunakan kolom ide_authors (gambar3.6). Setiap kolom akan melakukan tahapan pra pemrosesan yang sama demi menghasilkan informasi yang bernilai dalam tahap klasifikasi.



Gambar 3.7 Tahapan Pra Pemrosesan Fitur

Berdasarkan gambar diatas dapat disimpulkan bahwa untuk menemukan informasi dalam data author matching pada fitur data, dapat dilakukan dengan kombinasi yang tujuannya adalah memasangkan setiap baris pada masing-masing kolom.



Gambar 3.8 Tahapan Pra Pemrosesan Label

Dan untuk pra pemrosesan data label juga dilakukan dengan kombinasi untuk memberikan pasangan (*pairwise)* pada setiap baris pada kolom ide_author.

3.4.2.1.Pairwise Combination

Proses pengolahan dataset yang telah dihasilkan diatas dilanjutkan dengan melakukan pemasangan atau kombinasi agar dapat dilakukan proses komparasi antara baris pertama dengan baris kedua, baris pertama dengan baris ketiga dan seterusnya. Hasil kombinasi pada dataset yang digunakan yang memiliki 10.160 baris data akan menghasilkan jumlah baris data yang baru sebanyak 51.607.720 baris data pada setiap kolom baik pada fitur maupun label. Dalam rumus matematika, jumlah kombinasi didapatkan dengan rumus :

$$\frac{\boldsymbol{n}!}{\boldsymbol{r}!(\boldsymbol{n}-\boldsymbol{r})!} = \binom{\boldsymbol{n}}{\boldsymbol{r}} \tag{3.1}$$

Dimana **n** adalah jumlah objek yang bisa dipilih dan **r** adalah jumlah yang harus dipilih. Dan hasil dari dari proses pairwise dengan menggunakan teknik kombinasi tersebut dapat dilihat pada tabel 3.4.

Tabel 3.4

Hasil Proses Pairwise Combination

0	1
Egervary, E.	Alexits, G.
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Maller, R
Egervary, E.	Izumi, S

3.4.2.2.Remove Stopwords

Remove stopword merupakan proses yang dilakukan untuk mengurangi kata-kata yang tidak bernilai informasi dalam suatu data teks. Dalam proses ini, remove stopword hanya dilakukan pada kolom atau fitur title dengan maksud menghilangkan kata-kata yang dinilai tidak bermanfaat pada dataset agar tidak mengurangi informasi pada proses klasifikasi berikutnya.

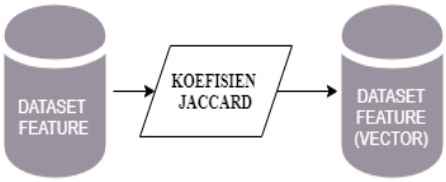
3.4.2.3. Nilai Selisih Tahun Terbit (*Year Difference*)

Proses ini dilakukan pada kolom year yang telah dikombinasikan. Teknik yang dilakukan adalah mencari perbedaan atau selisih dari dua data tersebut. Tujuan dilakukan hal tersebut adalah mencari jarak dari setiap tahun terbitnya publikasi, dengan asumsi jika pada nama author terdapat kesamaan namun jarak tahun terbit publikasi sangat jauh, maka dapat disimpulkan author tersebut berbeda.

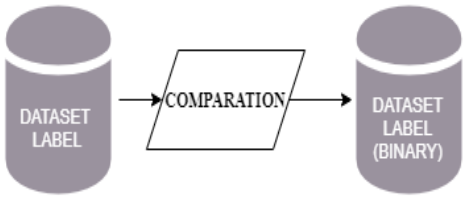
$$y = y1 - y2 \tag{3.2}$$

3.4.3. Ekstraksi Fitur

Ekstraksi fitur merupakan proses pengambilan ciri sebuah objek yang dapat menggambarkan karakteristik dari objek tersebut. Dalam tahapan ini, proses ekstraksi fitur dilakukan dengan menggunakan dua cara yang berbeda. Untuk fitur, proses pengambilan ciri dilakukan dengan teknik similarity menggunakan koefisien jaccard (gambar 3.7), sedangkan pada label dilakukan dengan menggunakan comparation antara data id_author yang telah dipasangkan atau dikombinasikan (gambar 3.8).



Gambar 3.9 Tahapan Estraksi untuk Feature



Gambar 3.10 Tahapan Ekstraksi untuk Label

3.4.3.1. Nilai Kemiripan (*Similarity*)

Proses similarity adalah tahapan menghitung jarak antara dua data yang telah dikombinasikan dengan tujuan menemukan kemiripan data. Teknik yang digunakan untuk menghitung jarak antar data yang telah dipasangkan dengan menggunakan Koefisien Jaccard. Pada persamaan 3.2 proses perhitungan yang dilakukan dalam proses koefisien Jaccard berikut ini.

$$\boldsymbol{jaccard}\,(\boldsymbol{a},\boldsymbol{b}) = \frac{|\boldsymbol{a}\cap\boldsymbol{b}|}{|\boldsymbol{a}\cup\boldsymbol{b}|} = \frac{|\boldsymbol{a}\cap\boldsymbol{b}|}{|\boldsymbol{a}|+|\boldsymbol{b}|-|\boldsymbol{a}\cap\boldsymbol{b}|} \tag{3.3}$$

Pada tahapan ini, koefisien jaccard dilakukan pada kolom-kolom yang akan digunakan sebagai feature yaitu name, authors, title, venue, lalu dilakukan penggabungan fitur year yanng telah dicari selisihnya pada proses sebelumnya. Hasil dari penggunaan koefisien dan penggabungan tersebut dapat dilihat pada tabel 3.5.

Tabel 3.5

Hasil Proses Similarity menggunakan Koefisien Jaccard

	name	authors	title	venue	year
0	0.33	1	1	1	0
1	0.21	0.14	0.35	0.5	1
2	0.28	0.26	0.24	0.56	1
3	0.21	0.14	0.25	0.54	1
4	0.28	0.25	0.26	0.47	2
5	0.21	0.15	0.38	0.53	3
6	0.21	0.15	0.23	0.53	3
7	0.21	0.15	0.23	0.48	3
8	0.21	0.15	0.38	0.5	3
9	0.41	0.4	0.33	0.5	3
.	.	.	.	.	.
.	.	.	.	.	.
51607720	0.27	0.35	0.26	0.44	7

3.4.3.2.Nilai Pembandingan (*Comparation*)

Tahapan komparasi atau perbandingan dilakukan untuk menemukan informasi pada kolom *id_author* yang akan digunakan sebagai contoh output dari semua input yang diajarkan atau biasa disebut label. Hasil dari kombinasi adalah sebuah angka binary yang prosesnya dilanjutkan dengan membandingkan satu sama lain, jika sama akan diberi nilai 1 dan jika tidak sama akan diberi nilai 0.

$$A \rightarrow \text{'0'} \; if \; \text{id1} \neq \text{id2} \tag{3.4}$$

$$A \rightarrow \text{'1'} \; if \; \text{id1} = \text{id2} \tag{3.5}$$

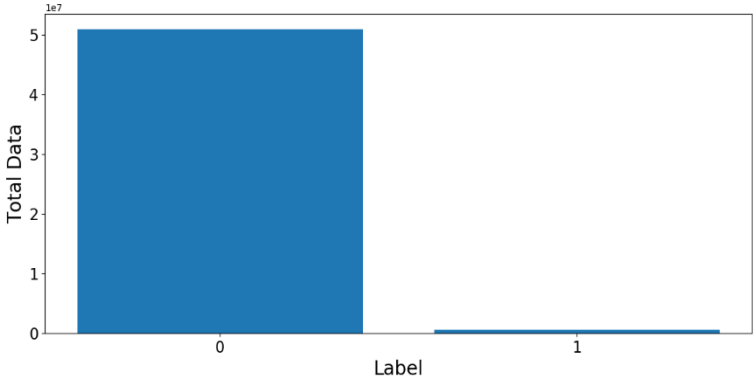
Hasil dari perbandingan tersebut menjadi satu buah tabel yang akan digunakan sebagai label pada algoritma klasifikasi atau pada tahapan berikutnya dan hasil perbandingan tersebut dapat dilihat pada tabel 3.6.

Tabel 3.6

Hasil Proses Comparison

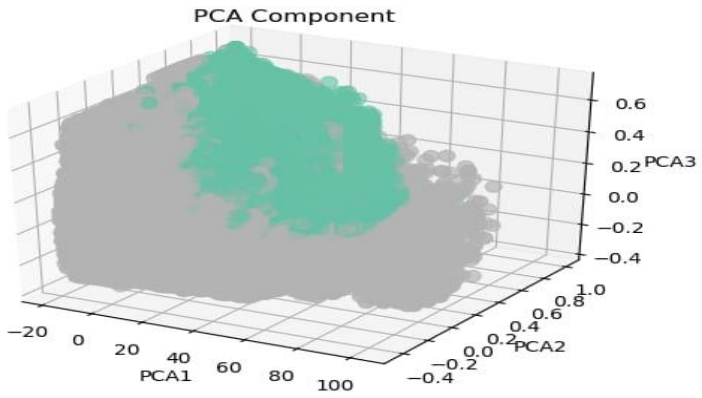
	id_authors
0	0
1	0
2	0
3	0
4	0
5	0
6	0
.	.
.	.
51607720	0

Hasil dari label inilah yang akan digunakan sebagai prediksi dalam proses klasifikasi data author matching. Nilai 0 didefinisikan sebagai nilai positive (data yang tidak sama) atau dengan kata lain author yang berbeda dan 1 sebagai data negative (data yang sama) atau bisa disimpulkan merupakan author yang sama. Dari hasil kombinasi dan dilanjutkan pada proses comparation terhadap label ini, maka terjadilah ketidakseimbangan data (*imbalance data*) yang sangat tinggi antara dimana data positive berjumlah 583.942 dan data negative sejumlah 51.023.778, atau sejumlah 1,1 % berbanding 98,9 % (gambar 3.8).



Gambar 3.11 Plot Imbalance Data Hasil Pra Pemrosesan Author Matching

Pada pengamatan diatas dapat digambarkan terjadinya imbalanced data yang sangat ekstream sehingga proses klasifikasi harus dilakukan dengan menggunakan metode yang mampu mengenali feature dan label dengan sangat baik. Gambaran sebaran data pada hasil pra pemrosesan maupun ekstraksi fitur dapat dilihat pada gambar 3.9.



Gambar 3.12 Plot Sebaran Imbalance Data Hasil Pra Pemrosesan Author Matching

Setiap tahapan pada pra pemrosesan data ini sebelum masuk ke dalam proses klasifikasi dengan algoritma classifier, di gambarkan secara ringkas dalam urutan algoritma yang dapat dilihat pada gambar 3.13

```
//declaration
dataset xlsdataset String;

//description
open (xlsdataset);
dataset  <-- xlsdataset
begin
{
    id      <-- expand.Column dataset (0)
    name    <-- expand.Column dataset (1)
    author  <-- expand.Column dataset (2)
    title   <-- expand.Column dataset (3)
    venue   <-- expand.Column dataset (4)
    year    <-- expand.Column dataset (1)

import numpy as np
import itertools
from itertools import combinations, product
from difflib import SequenceMatcher
comb_all = []

// proses kombinasi dan penggunaan koefisien jaccard

comb_all = None
feature = None
for i in range(7):
    comb = list(combinations(df.iloc[:,i], 2))
    comb = np.array(comb)
    x = []
    if comb.dtype == np.int64:
        distance = abs(comb[:,0] - comb[:,1])
        x = distance
    else:
        for c in comb:
            distance = SequenceMatcher(None, c[0], c[1]).ratio()
            x.append(distance)
    x = np.array(x)
    x = x.reshape(len(x), 1)

    if feature is None:
        feature = x
        comb_all = comb
    else:
        feature = np.concatenate((feature, x), axis = 1)
        comb_all = np.concatenate((comb_all, comb), axis = 1)

}
```

Gambar 3.13 Struktur Algoritma Sederhana Proses Pra Pengolahan Data

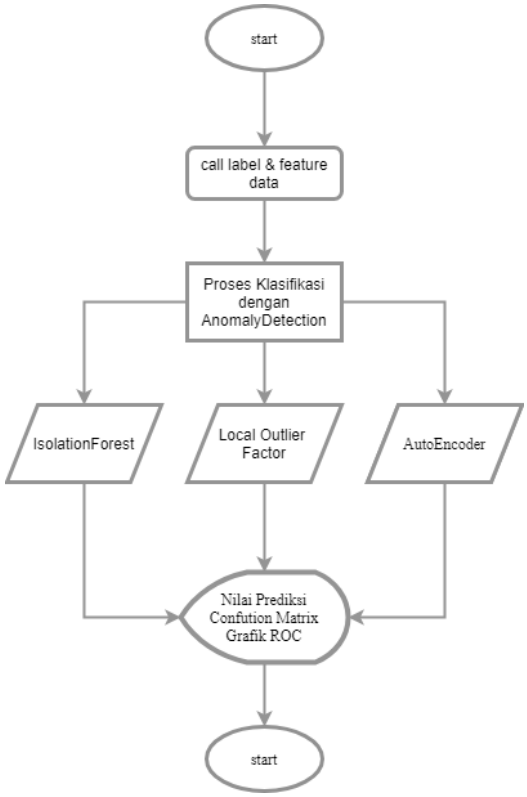
3.5 Klasifikasi menggunakan Menggunakan Pendekatan Anomaly Detection

Dengan merujuk pada hasil dari dari tahapan pre processing, dimana data menjadi sangat imbalance atau berdimensi tinggi, maka proses klasifikasi yang akan dilakukan dengan menggunakan data tersebut adalah dengan melakukan pendekatan anomaly detection. Anomaly detection digunakan karena proses yang dilakukan dalam algoritma nya memiliki kemiripan dengan bentuk dari dasil pre processing yang dilakukan dalam penelitian ini.

Anomaly detection merupakan sebuah teknik yang digunakan untuk mengidentifikasi pola-pola yang tidak biasa atau tidak sesuai perilaku yang diharapkan, dan hal tersebut lebih sering dikenal dengan outlier. Dengan didasarkan hal tersebut pula pendekatan ini dipakai dalam proses klasifikasi data author matching ini.

Ide dasar teknik anomali berbasis isolasi adalah bahwa anomali lebih mudah diisolasi daripada keadaan normal, dimana data anomali dapat dibuat sebagai data dengan profil berbeda dari keadaan normal. Metode pertama yang berbasi isolasi adalah Itree (Isolation Tree).

Adapun teknik anomaly detection memiliki beberapa algoritma yang sering digunakan untuk melakukan proses klasifikasi data yaitu Isolayion Forest, Local Outlier Factor (LOF) dan dengan menggunakan Auto Encoder. Masing-masing algoritma tersebut memiliki karakter dan pola yang berbeda dan akan menghasilkan angka prediksi dan kesimpulan yang berbeda pula. Adapun proses dan tahapan klasifikasi yang dilakukan dalam penelitian ini digambarkan pada proses flowchart Klasifikasi anomaly detection (Gambar 3.11).



Gambar 3.14 Flowchart Klasifikasi Anomaly Detection

3.5.1 Klasifikasi menggunakan Menggunakan Isolation Forest

Tahapan klasifikasi dengan menggunakan algoritma isolation forest dilakukan dengan gagasan inti yaitu dengan menemukan jumlah titik yang abnormal yang biasanya kecil atau bisa dianggap minority, dan ada perbedaan yang signifikan antara titik normal dan atribut. Perbedaan signifikan tersebut dalam data author matching digambarkan dengan nilai vector antara data feature dan label yang memiliki jarak atau nilai yang sangat berbeda. Dalam proses tersebut, pengamatan

terhdap semua data diberi skor dan ditandai sebagai anomali bila memenuhi beberapa kriteria sebagai berikut :

- Skor dekat dengan 1 terindikasi atau dimasukkan dalam kategori anomaly.
- Skor lebih kecil dari 0.5 terindikasi sebagai data normal.
- Jika seluruh skor dekat 0.5 kemudian seluruh sampel tidak terlihat memiliki baris anomali yang teratur, maka data yang bernilai 1 lah yang akan dianggap sebagai data anomali.

Pada contoh uji *x*, panjang lintasan *h(x)* dikumpulkan pertama kali dan rata-rata panjang lintasan *E((h(x))* dihitung. Jika nilai dari *x* lebih dari 1, maka panjang lintasan *h(x)* akan ditambahkan lagi ke subtree untuk membuat barisan data yang baru. Untuk data dengan ukuran *n*, panjang lintasan rata-rata yang diharapkan *(n)* dapat dinyatakan sebagai berikut :

$$\boldsymbol{c(n)} = \boldsymbol{2H}(n-1) - \frac{2(n-1)}{n} \tag{3.2}$$

$$\boldsymbol{H(i)} = \boldsymbol{ln}(i) + 0.5772156649 \tag{3.3}$$

Seperti metode *outlier detection* atau *anomaly detection* lainnya, skor anomali diperlukan untuk pengambilan keputusan dan hasil dari pengamatan skor anomali tersebut dinyatakan dalam satu persamaan. Persamaan 3.2 untuk algoritma Isolation Forest, didefinisikan sebagai (Yao *et al.*, 2019) **:**

$$\boldsymbol{s(x,n)} = \boldsymbol{2}^{-\frac{E(h(x))}{c(n)}} \tag{3.4}$$

Dimana dijelaskan bahwa :
h(x) merupakan panjang lintasan data dari pengamatan *x*,
c(n) merupakan panjang lintasan pencarian yang gagal di pohon pencarian biner
n merupakan jumlah dari node (simpul) eksternal

3.5.2 Klasifikasi menggunakan Menggunakan Local Outlier Factor (LOF)

Algoritma selanjutnya yang digunakan dalam pendekatan anomaly detection adalah local aoulier factor (LOF). Karakteristik algoritma ini adalah dengan menghitung deviasi atau penyimpangan data berdasarkan kepadatan data tetangganya (data berikutnya) di dalam barisan atau lajur data yang ada. Algoritma ini menemukan data outlier dengan melakukan perhitungan dari kelompok data yang menyimpang dari titik data normal yang diberikan. Algoritma ini menjadi salah satu solusi pada pendekatan deteksi anomali pada sekelompok dataset yang sangat imbalance. Metode penghitungan kepadatan data pada algoritma ini dilakukan berdasarkan kepadatan antara masing-masing titik data dengan titik data berikutnya atau tetangganya, dimana semakin rendah kepadatan data pada suatu titik maka akan semakin besar kemungkinan diidentifikasi sebagai outlier.

Ada 2 kondisi dari data author matching ini yang akan dikategorikan sebagai outlier dengan menggunakan algoritma Local Outlier Factor (LOF), yaitu sebagai berikut :

- Jumlah tetangga yang dipertimbangkan, (parameter n_neighbors) biasanya dipilih 1 lebih besar dari jumlah minimum objek yang harus dikandung sebuah cluster, sehingga objek lain dapat berupa outlier lokal relatif terhadap cluster ini.
- lebih kecil dari jumlah maksimum dekat dengan objek yang berpotensi menjadi outlier lokal.

namun, kondisi kedua sangat jarang ditemukan, karena informasi nilai data yang lebih kecil dari jumlah maksimum akan sangat sulit diidentifikasi.

Secara matematis, urutan untuk pendefinisian outlier atau bukan di dalam algoritma LOF ditentukan dengan sepanjang apa data tetangga yang akan dipertimbangkan untuk menjadi cluster data (parameter n), kemudian dilakukan dengan mencari nilai maksimum dari jarang jangkauan atau *Reachbility Distance (RD)* pada cluster tersebut, lalu mencari nilai rata-rata dari jarak jangkaun tersebut atau *Average Reachibility Distance (ARD)* dan hasilnya dari rata-rata tersebut didefinisikan sebagai *Local Reachibility Distance (LRD) n,* setelah semua nilai didapat baru akan dijumlahkan dari seluruh *LRD n* dan dibagi dengan *LRD a* (jika yang dicari adalah nilai a) untuk mendapatkan nilai dari *Local Outlier Factor (LOF)*

dari *n* tersebut. Hal tersebut dapat digambarkan dengan urutan persamaan dibawah ini.

- *n* adalah jumlah tetangga atau baris data yang ditentukan untuk titik

- Menentukan nilai maksimum dari *RD (Reachbility Distance)* pada poin *a*

$$RD\ a = \max(n\,\mathrm{dist}_{\mathrm{a}}\,, dist\ (a,n)) \quad (3.5)$$

- Lalu, menentukan nilai rata-rata dari *Local Reachibility Distance (LRD)*

$$LRD\ a = Average\ (RD\ a + \cdots . + RD\ n) \quad (3.6)$$

dan selanjutnya melakukan pencarian nilai sampai $LRD\ ke\ n$

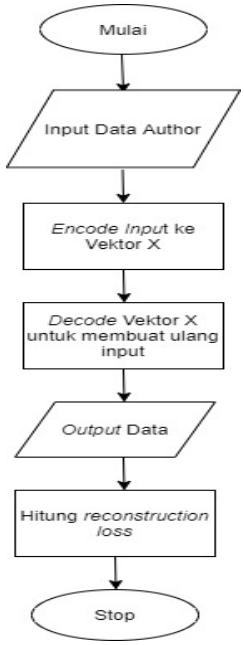
- Terakhir kita dapat menentukan nilai LOF dari *a* dengan cara mencari nilai rata-rata dari panjang *LRD* dalam 1 cluster dibagi dengan nilai *LRD* yang dicari (dalam hal ini *LRD a*)

$$LOF\ a = \frac{average\ (LRD\ b + \cdots . + LRD\ n)}{LRD\ a} \quad (3.7)$$

Setelah semua nilai LOF didapatkan, naru akan didefinisikan nilai berapa yang akan dikategorikan sebagai outlier. Dalam hal data author matching ini, nilai yang akan dikategorikan sebagai outlier bernilai antara 0,9 s.d. 1.

3.5.3 Klasifikasi menggunakan Menggunakan AutoEncoder

Algoritma selanjutnya adalah dengan menggunakan auto encoder dengan tujuan merekonstruksi data terlebih dahulu untuk mendapatkan value data yang lebih sebelum masuk dalam proses klasifikasi. Proses *autoencoder* diawali dengan *input* data *author*. Lalu *input* di *encode* ke vektor x. Setelah di *encode* maka masuk ke proses *decode* untuk memuat ulang *input* yang merupakan *output* data yang sudah direduksi dimensinya. Setelah itu hitung *reconstruction loss*.



Gambar 3.15 Flowchart Klasifikasi Auto Encoder

Proses klasifikasi dengan menggunakan auto encoder ini dimaksudkan untuk menjadi pembanding dalam metode anomaly detection dimana algoritma machine learning dan deep learning akan menghasilkan nilai prediksi dari proses klasifikasi yang seperti apa, terutama ketika digunakan pada data yang berdimensi tinggi atau tingkat imbalance yang tinggi.

3.6 Analisa Hasil

Pada tahap ini akan dijelaskan mengenai analisis dari keseluruahan kinerja sistem klasifikasi *author matching* menggunakan metode Anomaly Detection dengan menggunakan 3 model algoritma itu akan mencakup akurasi, presisi, sensitivitas, dan F1-score. Hasil sementara dalam penelitian ini akan dijelaskan di Bab IV.

3.7 Kesimpulan

Tahap ini dilakukan untuk menarik kesimpulan terhadap metode *anomaly detection* dengan menggunakan 3 model algoritma. Penarikan kesimpulan merupakan jawaban dari tujuan penelitian yang ingin dicapai. Kesimpulan akan dijelaskan di Bab V.

BAB IV
HASIL DAN ANALISA

Bab ini berisi hasil dari penelitian dari sistem klasifikasi binary untuk data positif dan negatif pada *author matching*. Pada bab ini akan mengukur performa dari klasifikasi seperti akurasi, presisi, sensitivitas, spesifisitas, dan *F1-score*. Hasil validasi untuk bab IV ini menggunakan dua skenario pada satu model klasifikasi. Skenario pertama memvalidasi hasil dengan langsung menggunakan data secara keseluruhan pada model *isolationforest, Local Outlier Factor dan AutoEncoder*. Skenario kedua yaitu pembagian data *training* dan *testing* yang berbeda pada model *Isolation Forest*. Ini bertujuan untuk melihat apakah metode *anomaly detection* layak digunakan dan apakah model nya mampu memberikan hasil validasi yang baik pada proses klasifikasi *author matching*.

4.1 Hasil Validasi Menggunakan Isolation Forest (Langsung pada keseluruhan Data / Percobaan Pertama)

Proses klasifikasi pada data author matching menggunakan *isolation forest* dilakukan dengan mendefinisikan *n_estimator* yaitu panjang baris data yang akan dijadikan acuan untuk melakukan pencarian data anomaly, dan dalam percobaan ini ditentukan adalah keseluruhan data. Selanjutnya adalah pendefinisian *contamination* dengan menggunakan *'auto'* yaitu seberapa besar porsi *outlier* pada data anomali yang akan diklafisikasi atau bisa dikatakan seberapa besar rasio jumlah data anomali yang akan dipakai atau ditemukan. Langkah berikutnya adalah menentukan *max_feature* yang merupakan angka maksimum tertinggi dari *feature* yang ada dan dalam penelitian ini angka tersebut adalah 1.

Selanjutnya dengan menggunakan decision function pada format behavoiur untuk data string dengan memilih '*new*' sebagai poin *behaviour* yang berfungsi untuk menetapkan *threshold* menjelaskan proses pengenalan anomali dimulai dari awal data untuk mendeteksi *outlier*. Selain itu, digunakan juga *random generator* untuk dapat melakukan pengenalan dan pembelajaran secara acak dengan *random_state* sebanyak 42 kali.

Proses diatas mampu menghasilkan nilai akurasi yang tinggi serta menyajikan hasil *precision* dan *recall* yang cukup baik untuk data yang anomali. Hasil dari proses klasifikasi menunjukkan bahwa pendekatan anomaly detection dengan menggunakan algoritma IsolatioForest dapat digunakan pada data yang berdimensi tinggi. Selain itu, dengan menggunakan keseluruhan data (tanpa pembagian jumlah data training dan testing) mampu menghasilkan nilai *precision* dan *recall* pada kedua identitas binary dari label baik itu data positif maupun data negatif (tabel 4.1).

Tabel 4.1

Hasil Kinerja Model Isolation Forest Secara Langsung (Keseluruhan Data tanpa Data Training dan Testing)

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,995**		
Precision	**0,78**	*1.00*	*0,78*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,79**	*1.00*	*0,79*
F1-Score	**0,78**	*1.00*	*0,78*
Data Support		***51,023,778***	***583,942***

Pada hasil kinerja diatas dapat dijelaskan per kategori dimana *Accuracy* menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Maka, *accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Lalu, *Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka, *precision*merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. Dari semua kelas positif yang telah di prediksi dengan benar, berapa banyak data yang benar-benar positif.

Kategori berikutnya adalah recall atau specificity, dimana *Recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Maka, *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

Hasil confusion matrix dari proses klasifikasi secara keseluruhan data juga mampu disajikan dengan baik sehingga ketika dilakukan perhitungan manual dengan menggunakan hasil dari confusion matrix tersebut, maka hasil performance yang ditunjukkan dalam proses klasifikasi menggunakan algoritma ini dapat dibuktikan (tabel 4.2)

Tabel 4.2

Hasil Confusion Matrix Model Isolation Forest

TP 458,920	FP 125,022
FN 131,704	TN 50,892,074

Berikut adalah perhitungan manual berdasarkan hasil dari confusion matrix untuk membuktika hasil accuracy, presition dan menampilkan hasil perhitungan manual sensitivity, spesivicity, dan f1 score.

1. Accuracy $= \frac{458920+50892074}{458920+125022+131704+50892074} = \frac{51350994}{51607720} = 0{,}995$

2. Precision $= \frac{458920}{458920+125022} = \frac{458920}{583924} = 0{,}785$

3. Sensitivity $= \frac{458920}{458920+131704} = \frac{458920}{590624} = 0{,}787$

4. Spesificity $= \frac{50892074}{125022+50892074} = \frac{50892074}{51017926} = 0{,}997$

5. F1 score $= 2 \text{ x } \frac{0{,}785 \text{ x } 0{,}777}{0{,}785+0{,}777} = \frac{0{,}61}{1{,}56} = 0{,}78$

4.2 Hasil Validasi Menggunakan Isolation Forest (Split data untuk Training dan Testing/Percobaan Kedua)

Hasil percobaan selanjutnya adalah dengan menerapkan pengujian berdasarkan pembagian data dengan skenario 80% untuk training dan 20% untuk testing. Jumlah pembagian ini merupakan persentase umum yang sering digunakan pada proses data latih dan data uji (training dan testing). Sama seperti sebelumnya, mendefinisikan *n_estimator* yaitu panjang baris data yang akan dijadikan acuan untuk melakukan pencarian data anomaly, dan dalam percobaan ini ditentukan adalah keseluruhan data. Selanjutnya adalah pendefinisian *contamination* dengan menggunakan *'auto'* yaitu seberapa besar porsi *outlier* pada data anomali yang akan diklafisikasi atau bisa dikatakan seberapa besar rasio jumlah data anomali yang akan dipakai atau ditemukan. Langkah berikutnya adalah menentukan *max_feature* yang merupakan angka maksimum tertinggi dari *fetature* yang ada dan dalam penelitian ini angka terebut adalah 1.0

percobaan ini dilakukan dengan menggunakan decision function pada format behavoiur untuk data string dengan memilih '*new*' sebagai poin *behaviour* dan juga *random generator* untuk dapat melakukan pengenalan dan pembelajaran secara acak dengan *random_state* sebanyak 42 kali.

Proses ini dimaksudkan untuk memberikan tahapan pengenalan dan pembelajaran (learning) pada algoritma sebelum memutuskan pada data sebenarnya atau pada data testing. Pada proses ini, pengenalan dan pembelajaran dilakukan menggunakan random generator yang memilih data secara acak sepanjang baris data yang ada. Pemilihan random state sebanyak 42 kali adalah untuk mencari jumlah kelompok data paling lengkap dan diharapkan mampu menghasilkan proses klasfikasi yang terbaik.

Selain data diatas, proses pada isolation forest juga dilakukan dengan menentukan terlebih dahulu n_estimators yang berfungsi sebagai rujukan jumlah kelompok data yang ditaksir sebagai anomali, type data secara umum adalah integer dan nilai standarnya adalah 100. Lalu dilanjutkan dengan menentukan max_sample sebagai jumlah yang akan digunakan atau diambil untuk melatih setiap label dan feature.

Lalu dilanjutkan dengan menentukan contamination, dimana variabel ini digunakan untuk menentukan ambang batas atau threshold yang akan dipakai sebagai acuan untuk menetakan data anomali. Namun, dalam prosesnya nilai ambang batas tersebut ditentukan secara otomatis di dalam sistem. Dan terakhir adalah max_feature yang digunakan untuk menentukan jumlah feature yang digunakan dalam proses latihan, agar tidak semua feature yang identik digunakan lebih dari satu kali.

Dari semua proses tersebut, didapatkan hasil prediksi yang sangat baik di kategori akurasi dan spesificity, dan sama seperti percobaan sebelumnya pada keseluruhan data, hasil yang cukup baik atau dapat dikategorikan baik dalam klasifikasi data yang super imbalance didapatkan pada kategori presisi, sensitivity dan juga dengan ditunjukkan pada persentase capaian di F1-Score (Tabel 4.3).

Pada tabel tersebut menampilkan hasil pada percobaan kedua (menggunakan split data) dimana hasil presiksi pada data training dibandingkan dengan hasil prediksi pada data testing. Selain itu ditampilkan juga hasil dari setiap kelas dengan tujuan memberikan penjelasan bahwa setiap kelas berhasil ditemukan datanya dalam percobaan kedua ini.

Tabel 4.3

Hasil Pengujian Data Training (80%) dan Testing (20%)

Kategori	Training			Testing		
	80% Data	Kelas		20% Data	Kelas	
		0	1		0	1
Accuracy	0,995			0,995		
Presisi	0,793	1.00	0,793	0,799	1.00	0,788
Sensitivity	0,798	1.00	0,798	0,826	1.00	0,797
Specificity	0,997	1.00	0,997	0,997	1.00	0,997
F1 Score	0,782	1.00	0,782	0,782	1.00	0,782
Data Support		***40,819,022***	***467,154***		***10,204,756***	***116,788***

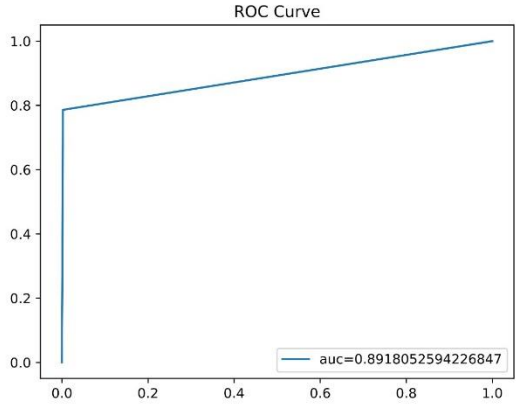
Hasil diatas menunjukkan tidak ada perbedaan hasil yang signifikan dari proses sebelumnya. Lalu pada proses validasi pengujian atau *testing* juga mendapatkan hasil yang tidak jauh berbeda dan dapat juga dilihat dari hasil *confusion matrix* (Tabel 4.5). Hal ini menunjukkan performa yang stabil dari model Isolation Forest dalam proses klasifikasi *author matching.*

Tabel 4.4

Hasil Confusion Matrix Data Testing (20%)

TP 93,177	FP 23,611
FN 25,033	TN 10,179,723

Selain hasil diatas, performa dari metode anomaly detection dengan menggunakan model Isolation Forest dapat dikatakan sangat baik dapat dilihat dari hasil perhitungan confusion matrix yang digambarkan melalui kurva ROC (Gambar 4.1). Kurva tersebut digunakan untuk meringkaskan data *confusiton matrix* yang bernilai binary atau 2 class dan menggambarkan hasil dari nilai *confusion matrix* dan menggambarkan *True Positive Rate (TPR)* dan nilai *False Positif Rate (FPR).* TPR menggambarkan seberapa besar data positif yang ditemukan dibandingkan dengan data positif sebernarnya, begitu pula pada FPR. Pada kurva ROC tersebut, dapat disimpulkan bahwa nilai TPR naik mulai dari 0.0 sampai ke 0.8. hal tersebut menjelaskan bahwa sensitivitas dari model algoritma ini hanya sampai pada 0.8, dan nilai dapat menampilkan hasil prediksi dengan cukup akurat. Sedangkan nilai FPR dapat disimpulkan bahwa semakin mendekati nilai 1.0.



Gambar 4.1. Kurva ROC Isolation Forest

4.3 Hasil Validasi Menggunakan Isolation Forest (Fine Tunning Parameter / Percobaan Ketiga)

Pada percobaan berikutnya adalah dengan melakukan perubahan paramater pada algoritma di python yaitu dengan merubah pilihan *contamination* dari *'auto'* menjadi lebih spesifik yaitu *'outlier fraction'.* Parameter itu digunakan untuk merekam proses pencarian data anomali untuk dibandingkan dengan data yang ditemui berikutnya. Selain itu, dalam percobaan ketiga ini juga ditambahkan parameter *max_samples* untuk mencoba membatasi jumlah sample yang diambil dalam proses klasifikasi, dan parameter yang digunakan adala *áuto'.* Percobaan ketiga ini dimaksudkan untuk meningkatkan hasil validasi dari data *author matching* ini menggunakan algoritma Isolation Forest. Adapun pilihan parameter yang dilakukan perubahan adalah memang parameter yang memiliki alternatif untuk dilakukan *fine tunning.*

Dalam percobaan ketiga ini, dilakukan proses klasifikasi sama seperti pada percobaan pertama dimana kami melakukan pada keseluruhan data secara langsung tanpa ada proses split data. Dan hasil yang ditampilkan pada percobaan ini menjadi sama persis dengan hasil pada percobaan pertama, tanpa ada perbedaan sedikitpun dimana nilai validasi dan *confusion matrix* menampilkan nilai yang sama persis. Bahkan pada data support dan hasil validasi pada setiap class juga menampilkan hasil yang sama persis (Tabel 4.5).

Tabel 4.5

Hasil Kinerja Model Isolation Forest dengan Fine Tunning Parameter

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,995**		
Precision	**0,78**	*1.00*	*0,78*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,79**	*1.00*	*0,79*
F1-Score	**0,78**	*1.00*	*0,78*
Data Support		***51,023,778***	***583,942***

4.4 Hasil Validasi Menggunakan *Local Outlier Factor (LOF)* / Percobaan Pertama

Proses klasifikasi pada data author matching menggunakan *Local Outlier Factor (LOF)* dilakukan menggunakan python dimulai dengan menetapkan jumlah *n_neighbors* terbelih dahulu atau jumlah jumlah baris tetangga dalam satu cluster. Secara default, biasanya jumlahnya ditetapkan 20, namun karena jumlah data hasil pre processing *author matching* ini sangat banyak, maka dalam penelitian ini kami menetapkan jumlah dari *n_neighbors* menjadi 2000. Hal tersebut dilakukan agar proses klasifikasi tidak terlalu lama namun selutuh data atau sampel tetap dapat diklasifikan.

Selain itu, hal berikutnya yang ditentukan dalam algoritma ini adalah *algorithm.* Secara default dalam percobaan pertama ini, kami menggunakan pilihan *auto* pada keselurahan data feature dan label. Hal ini dimaksudkan untuk melihat seberapa besar nilai prediksi yang dihasilkan pada pilihan struktur standar pada algoritma ini.

Selanjutnya adalah pemilihan besar *leaf_size*, yaitu kecepatan penggunaan memory dalam proses klasifikasi ini. Secara default biasanya nilai yang digunakan adalah 30, namuan dalam penelitian ini digunakan nilai 150 untuk mempercepat proses dan menampilkan hasil dari prediksi. Kemudian, yang dilakukan adalah pemilihan *metric* yang digunakan sebagai parameter jarak atau panjang dalam perhitungan komputasi, dan dalam penelitian ini digunakan *'minkowski'.*

Langkah selanjutnya di python adalah dengan menentukan *metric_param* yang dalam penelitian ini dipilih *none*, dikarenakan hal tersebut sudah diwakili oleh metric pada proses sebelumnya. Kemudian adalah penentuan *contamination* yang merupakan proses pendefinisian nilai atau jumlah yang akan dikategorikan sebagai outlier atau anomali. Dalam percobaan pertama ini, kami menggunakan pilihan *auto* agar hasil prdiksi menjadi lebih baik.

Proses diatas mampu menghasilkan nilai akurasi yang cukup tinggi serta menyajikan hasil *precision* dan *recall* yang cukup baik untuk data yang anomali meskipun tidak melebihi dari proses pada algoritma Isolation Forest. Sekali lagi, hasil dari proses klasifikasi menunjukkan bahwa pendekatan anomaly detection dengan menggunakan algoritma IsolatioForest dapat digunakan pada data yang berdimensi tinggi. Selain itu, proses tersebut mampu menghasilkan nilai *precision* dan *recall* pada kedua identitas binary dari label baik itu data positif maupun data negatif (tabel 4.6).

Tabel 4.6

Hasil Kinerja Model Local Outlier Factor Secara Langsung

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,994**		
Precision	**0,76**	*1.00*	*0,76*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,76**	*1.00*	*0,76*
F1-Score	**0,76**	*1.00*	*0,76*
Data Support		***51,022,724***	***584,996***

Pada hasil kinerja diatas dapat dijelaskan per kategori dimana *Accuracy* menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Maka, *accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan

nilai prediksi dengan nilai aktual (sebenarnya). Lalu, *Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka, *precision*merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. Dari semua kelas positif yang telah di prediksi dengan benar, berapa banyak data yang benar-benar positif.

Kategori berikutnya adalah recall atau specificity, dimana *Recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Maka, *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

Hasil confusion matrix yang mampu menunjukkan nilai dari hasil prediksi dan dari proses klasifikasi secara keseluruhan data juga mampu disajikan dengan baik (tabel 4.7)

Tabel 4.7

Hasil Confusion Matrix Model Local Outlier Factor

TP	FP
447,620	137,376
FN	**TN**
139,004	50,883,720

Berikut adalah perhitungan manual berdasarkan hasil dari confusion matrix untuk membuktikan hasil accuracy, presition dan menampilkan hasil perhitungan manual sensitivity, spesivicity, dan f1 score.

1. Accuracy $= \frac{447620+50883720}{447620+137376+139004+50883720} = \frac{51331340}{51607720} = 0{,}994$

2. Precision $= \frac{447620}{447620+137376} = \frac{447620}{584996} = 0{,}76$

3. Sensitivity $= \frac{447620}{447620+139004} = \frac{447620}{586624} = 0{,}76$

4. Spesificity $= \frac{50883720}{137376+50883720} = \frac{50883720}{51021096} = 0{,}997$

5. F1 score = $2 \times \frac{0,76 \times 0,76}{0,76+0,76} = \frac{0,61}{1,56} = 0,76$

4.5 Hasil Validasi Menggunakan Local Oulier Factor (Split data untuk Training dan Testing / Percobaan Kedua)

Hasil percobaan selanjutnya adalah dengan menerapkan pengujian berdasarkan pembagian data dengan skenario 80% untuk training dan 20% untuk testing. Semua teknik di python yang digunakan pada percobaan kedua ini sama seperti sebelumnya. Proses ini dimaksudkan untuk memberikan tahapan pengenalan dan pembelajaran (learning) pada algoritma sebelum memutuskan pada data sebenarnya atau pada data testing.

Pada proses ini, pengenalan dan pembelajaran dilakukan menetapkan terlebih dahulu panjang tetangga atau barisan data yang akan dijadikan acuan sebagai pendefinisian outlier sepanjang baris data yang ada. menggunakan python dimulai dengan menetapkan jumlah *n_neighbors terbelih dahulu* atau jumlah jumlah baris tetangga dalam satu cluster. Secara default, biasanya jumlahnya ditetakan 20, namun karena jumlah data hasil pre processing *author matching* ini sangat banyak, maka dalam penelitian ini kami menetapkan jumlah dari *n_neighbor* menjadi 2000. Hal tersebut dilakukan agar proses klasifikasi tidak terlalu lama namun selutuh data atau sampel tetap dapat diklasifikan.

Selain itu, semua teknik yang digunakan dalam python sama seperti pada percobaan sebelumnya. Hal itu dikarenakan ingin memperoleh kesimpulan apakah dengan metode klasifikasi yang berbeda, algoritma ini mampu memberikan hasil validasi yang lebih baik atau tidak. Hasil dari percobaan kedua pada algoritma ini dapat dilihat pada tabel 4.8.

Tabel 4.8

Hasil Pengujian Data Training (80%)

Kategori	Training			Testing		
	80% Data	Kelas		20% Data	Kelas	
		0	1		0	1
Accuracy	0,994			0,994		
Presisi	0,76	1.00	0,76	0,76	1.00	0,76
Sensitivity	0,76	1.00	0,76	0,76	1.00	0,76
Specificity	0,99	1.00	0,99	0,99	1.00	0,99
F1 Score	0,76	1.00	0,76	0,76	1.00	0,76
Data Support		***40,819,022***	***467,154***		***10,001,274***	***110,894***

Hasil diatas menunjukkan ada perubahan hasil validasi pada percobaan kedua ini, dimana hasil pada kategori accuracy meningkat serta terjadi peningkatan cukup signifikan pada kategori presisi, sensitivity, spesificity, dan f1 score. Perubahan menunjukkan ada proses yang berpengaruh pada hasil validasi, dan dapat disimpulkan bahwa penentuan jumlah *n_neighbors* menjadi penting dalam proses klasifikasi menggunakan algoritma LOF ini. Panjang jumlah data yang akan dihitung menjadi titik seberapa baik hasil validasi dari algoritma LOF ini.

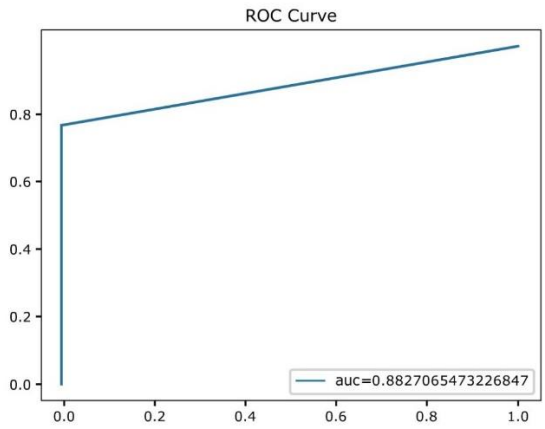
Parameter lain yang digunakan pada percobaan tidak berpengaruh pada nilai hasil dari validasi dikarenakan teknik yang laen dalam python yang digunakan tidak mempengaruhi proses aritmatik yang menentukan nilai validasi outlier. Dan dapat disimpulkan juga bahwa nilai outlier yang ditentukan dalam percobaan kedua ini adalah data yang bernilai 1.

Tabel 4.9

Hasil Confusion Matrix Data Testing (20%)

TP	FP
89,524	27,475
FN	**TN**
27,801	10,176,744

Selain hasil diatas, performa dari metode anomaly detection dengan menggunakan model Algoritma Local Outlier Factor dengan percobaan split data dapat dikatakan sangat baik dapat dilihat dari hasil perhitungan confusion matrix yang digambarkan melalui kurva ROC (Gambar 4.2) dimana kurva digambarkan mendekakati nilai 0.79 pada TPR dan mendekati titik 1.0. pada nilai FPR.



Gambar 4.2. Kurva ROC Local Outlier Factor

4.6 Hasil Validasi Menggunakan Local Oulier Factor (Fine Tunning parameter / Percobaan Ketiga)

Pada percobaan berikutnya adalah dengan melakukan perubahan paramater pada algoritma di python yaitu dengan merubah pilihan *algorithm* dari *'auto'* menjadi *'ball_tree'*, lalu mencoba merubah parameter *contamination* dari *'auto'* menjadi lebih spesifik yaitu *'float'*. Percobaan ketiga ini dimaksudkan untuk meningkatkan hasil validasi dari data *author matching* ini menggunakan algoritma LOF. Adapun pilihan parameter yang dilakukan perubahan adalah memang parameter yang memiliki alternatif untuk dilakukan *fine tunning.*

Dalam percobaan ketiga ini, kami melakukan proses klasifikasi sama seperti pada percobaan pertama dimana kami melakukan pada keseluruhan data secara langsung tanpa ada proses split data. Dan hasil yang ditampilkan pada percobaan ini menjadi sama persis dengan hasil pada percobaan pertama, tanpa ada perbedaan sedikitpun dimana nilai validasi dan *confusion matrix* menampilkan nilai yang sama persis. Bahkan pada data support dan hasil validasi pada setiap class juga menampilkan hasil yang sama persis (Tabel 4.10).

Tabel 4.10

Hasil Pengujian Data Training (80%)

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,994**		
Precision	**0,76**	*1.00*	*0,76*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,76**	*1.00*	*0,76*
F1-Score	**0,76**	*1.00*	*0,76*
Data Support		***51,022,724***	***584,996***

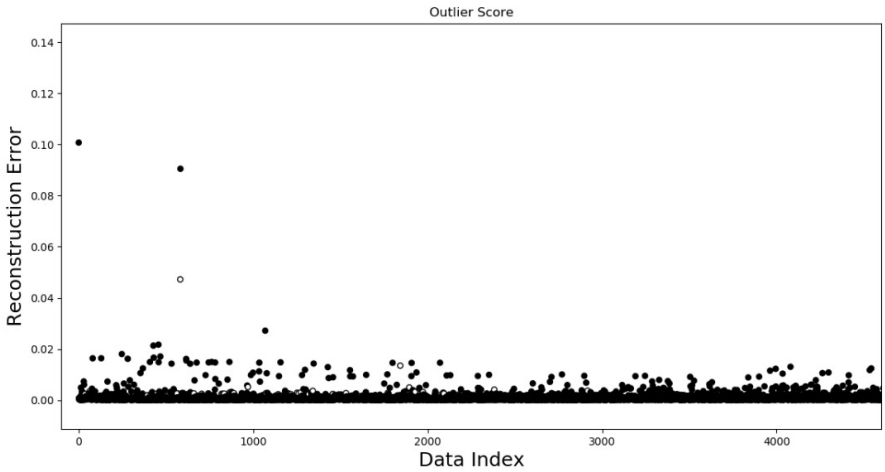
Dari hasil diatas dapat disimpulkan bahwa algoritma LOF dengan berbagai parameter yang dimiliki di dalam python mempunyai kestabilan dan cukup handal menampilkan hasil validasi jika digunakan pada data jelas dan memiliki struktur value yang seragam. Sama halnya dengan Isolation Forest, algoritma ini mampu menampilkan hasil yang cukup baik pada data *author matching*, dan juga pendekatan anomaly detection yang digunakan untuk mengklasifikan data *author matching* yang highly imbalance ini dirasa menjadi sangat tepat.

4.7 Hasil Validasi Menggunakan Auto Encoder

Pada tahap ini adalah proses klasfikasi dengan menggunakan algoritma ketiga yaitu auto encoder. Auto encoder merupakan salah satu algoritma yang dapat digunakan dalam pendekatan namoly detection denngan menggunakan konsep

neural network. Dimana algoritma ini akan digunakan dalam *un-supervised* proses dengan diawali proses ekstraksi data hasil pre prosessing dengan melakukan uncode dan decode, lalu dilakukan tahap visualisasi data untuk menentukan nilai threshold yang akan dijadikan acuan sebagai kelompok anomaly atau outlier.

Proses ini dilakukan dengan merepresentasikan data awal lalu dilanjutkan dengan merekinstruksikan kembali data tersebut untuk dapat diolah menjadi lebih jelas pada neural network. Hasil dari proses tersebut dilanjutkan dengan proses visualisasi bentuk data untuk dilankutkan proses penetuan threshold secara manual sebelum dilanjutkan dengan proses validasi. Dan hasil dari visualisasi data tersebut dapat dilihat pada Gambar 4.3.



Gambar 4.3. Visualisasi Auto Encoder pada Data Author Matching

Dari hasil visualisasi diatas, kami menyimpulkan bahwa proses penentuan nilai threshold yang menjadi acuan pendefinisian data namolay atau outlier menjadi tidak dapat dilakukan, karena semua data yang memiliki nilai ‘1’ atau divisulisasikan dengan lingkaran yang kosong, hampir semuanya tertutupi oleh data yang memiliki nilai ‘0’ atau lingkaran yang hitam. Sebagaimana telah kami jelaskan pada proses sebelumnya (pre processing) bahwa data 0 adalah data author yang tidak sama, sedangkan data 1 adalah data author yang sama.

Dengan gagalnya proses penentuan threshold tersebut, maka proses validasi tidak dapat kami lanjutkan karena bisa dipastikan bahwa class 1 tidak akan terdeteksi dalam proses klasifikasi atau validasi tersebut sehingga nilai TP pada

confusion matrix yang menjadi data jadi acuan data benar menjadi bernilai ‘Nan’. Untuk menjelaskan hal tersebut, proses penelitian ini dilanjutkan ke dalam tahap analisa data pada algoritma ini berdasarakan hasil analisis penelitian sebelumnya.

Pada penelitian sebelumnya dijelaskan bahwa, untuk proses klasifikasi data imbalance menggunakan deep learning lebih baik dengan cara menyeimbangkan data terlebih dahulu menggunakan proses over sampling atau under sampling (Amin *et al.*, 2016; Gong and Chen, 2016). Teknik tersebut tidak digunakan dalam penelitian ini dikarenakan pada data author matching klasfikasi data membutuhkan data realyang tidak terkontaminasi apapun agar dapat menghasilkan nilai validasi yang akurat sesuai dengan data yang ada. Selain itu, teknik oversampling yg dilakukan pada data imbalance dengan selisih data 99% dirasakan tidak tepat karena menambahkan terlalu banyak informasi baru yang sebenarnya palsu.

Pada proses klasifikasi data author matching dengan menggunakan auto encoder ini, hasil validasi yang tidak dapat ditampilkan kemungkinan dikarenakan jumlah data yang terlalu banyak dengan selisih jumlah data yang imbalance tertalu banyak pula. Barisan data yang terlalu banyak menyebabkan proses algoritma deep learning menjadi sulit untuk menentukan nilai validasi dengan baik (Leevy *et al.*, 2018). Oleh karena itulah, dalam algoritma ketiga ini tidak dapat menentukan nilai threshold dan tidak dapat dilanjutkan ke dalam proses validasi.

4.8 Studi Perbandingan

Evaluasi kinerja metode anomaly detection dengan menggunakan Isolation Forest, Local Outlier Factor dan AutoEncoder dapat dilihata pada tabel 4.11. dan hasil penelitian ini dibandingkan dengan beberapa penelitian sebelumnya tentang pengklasifikasian author matching. Hasil dari perbandingan tersebut bisa dilihat pada tabel 4.12.

Tabel 4.11

Tabel Perbandingan Hasil Validasi antar Algoritma

Kategori	Isolation Forest			LOF			Auto Encoder
	Percobaan			Percobaan			
	1	2	3	1	2	3	
Accuracy	0,995	0,995	0,995	0,994	0,994	0,994	-
Presisi	0,78	0,79	0,78	0,81	0,81	0,81	-
Sensitivity	0,78	0,79	0,79	0,76	0,76	0,76	-
Specificity	0,99	0,99	0,99	0,99	0,99	0,99	-
F1 Score	0,78	0,78	0,78	0,76	0,76	0,76	-

Tujuan awal dari penelitian ini adalah untuk menguji apakah metode anomaly detection dapat digunakan pada klasifikasi terhadap author matching yang diawali dengan proses pairwise combination dan string simlarity pada proses ekstraksi fiturnya. Pendekatan atau metode yang digunakan pada penelitian ini belum pernah digunakan sebelumnya, terutama pada klasifikasi data teks seperti data author matching. Selain itu, tujuan dari penelitian ini adalah untuk menguji juga model dari anomaly detection yaitu Isolation Forest, Local Outlier Factor (LOF) mampu menghasilkan nilai klasifikasi yang baik. Hasil yang ditunjukkan dari penggunaan model ini bisa dikatakan sangat baik dari sisi akurasi, meskipun belum optimal dari sisi precision dan recall yang sering jadi poin utama penilaian untuk klasifikasi data teks. Meskipun demikian, untuk data dengan tingkat imbalance yang sangat tinggi, hasil secara keseluruhan dari model ini bisa dikatakan baik. Tabel 4.6 dapat menunjukkan perbandingan dari penelitian sebelumnya.

Tabel 4.12

Tabel Perbandingan Dengan Penelitian Sebelumnya

No.	Metode	author	Implementasi	Teknik	Akurasi
1	Heuristic Based Hirarchical	AA. Ferreira, et al (2017)	Klasifikasi Biner antar author	Pairwise dan Similarity	94,3
2	Associative classifier	Marcos André Gonçalves, et al (2015)	Klasifikasi Biner	Pairwise	90,4
3	K-NN	Ming Son, et al (2015)	Klasifikasi Biner	Pairwise dan Similarity	98,17
4	Random Forest	kim, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	98,13
5	C4.5	kim, et al (2018)		Pairwise dan Similarity	98,2
6	SVM	kim, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	97,45
7	Naive Bayes	kim and diesner (2016)	Klasifikasi Biner	Pairwise dan Similarity	96,68
8	Boosted tree Classification	wang jian, et al (2012)	Klasifikasi Biner	Pairwise dan Similarity	89,3
9	Dempster-Shafer Theory	Hao wu, et al (2014)		Pairwise	
10	Graph Structural	Ijaz Husain, et al (2017)	Klasifikasi Biner	Pairwise dan Similarity	
11	Deep Neural Network	Tin Huynh, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	99,31
12	Anomaly Detection	2019	Klasifikasi Biner	Pairwise dan Similarity	99,56

Dari semua hasil percobaan, analisis dan perbandingan dengan masing-masing percobaan maupun dibandingkan dengan hasil klasifikasi penelitian sebelumnya, dapat disimpulkan bahwa proses klasifikasi data author pada database publikasi penelitian dapat dilakukan dengan mnggunakan pendekatan anonaly detection. Dimana pendekatan tersebut digunakan ketika proses pra prosesing data set dilakukan dengan menggunakan teknik pairwise dengan model kombinasi. Model tersebut menghasilkan data yang memiliki kesenjangan yang sangat tinggi (*super imbalance)* sehingga beberapa model algoritma classifier tidak dapat memberikan hasil dari proses klasifikasi data tersebut.

Namun, harus diakui bahwa hasil yang digunakan pada pendekatan ini, belum dapat menghasilkan nilai klasifikasi yang sempurna. Hal tersebut dikarenakan diperlukan proses modifikasi pada algoritma klasifikasi yang digunakan untuk memberikan hasil klasifikasi yang lebih baik dibandingkan yang dihasilkan pada penelitian ini.

BAB V

KESIMPULAN DAN SARAN

5.1. KESIMPULAN

Kesimpulan yang dihasilkan dalam penelitian mengenai pengklasifikasian author mathcing dengan menggunakan metode anomaly detection adalah sebagai berikut:

1. Proses pra pengolahan data set author name disambiguation dapat dilakukan dengan menggunakan pencocokan nama penulis (*author matching)* dengan melakukan pairwise combination dan string similarity dengan coeficient jaccard sehingga mampu menginterpretasikan kecocokan nama author dalam proses klasifikasi.
2. Metode anomaly detection dapat digunakan dalam proses klasifikasi pada data *author matching* yang memiliki tingkat *imbalance* yang sangat tinggi (1% data positive berbanding 99% data negative), hasil akurasi yang baik (akurasi 99,5 %) ditunjukkan pada metode ini dengan menggunakan model Algoritma *Isolation Forest dan Local Outlier Factor*.
3. Kinerja yang dihasilkan pada proses klasifikasi data author matching bisa disimpulkan sangat baik dari sisi akurasi dan spesifitas dibandingkan penelitian sebelumnya, serta cukup baik pada presisi, sensitivitas dan F1 score mengingat data yang digunakan memiliki tingkat imbalance yang sangat tinggi.

5.2. SARAN

Agar penelitian selanjutnya yang berkaitan dengan penelitian ini dapat lebih lebih baik dan mampu menghasilkan klasifikasi yang lebih baik pula, dapat disarankan sebagai berikut :

1. Dataset raw author pada database publikasi memerlukan beberapa tahapan lagi dalam pra pemrosesan untuk mampu menghasilkan pengenalan yang lebih baik pada tahapan pembelajaran data uji dan data latih ketika masuk ke dalam classifier.

2. Pendekatan anolamy detection memerlukan proses atau tahapan modifikasi model algoritma yang berkaitan dengan anomaly detection. Hal tersebut dikarenakan pada format model algoritma default, tidak mampu meningkatkan hasil prediksi walaupun sudah merubah beberapa fungsi dalam algoritma tersebut.

DAFTAR PUSTAKA

Amin, A. *et al.* (2016) 'Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study', *IEEE Access*, 4(Ml), pp. 7940–7957. doi: 10.1109/ACCESS.2016.2619719.

Berzins, K. *et al.* (2012) 'A boosted-trees method for name disambiguation', *Scientometrics*, 93(2), pp. 391–411. doi: 10.1007/s11192-012-0681-1.

de Carvalho, A. P. *et al.* (2011) 'Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries', *Journal of Information and Data Management*, 2(573871), p. 289.

Cheng, Z., Zou, C. and Dong, J. (2019) 'Outlier detection using isolation forest and local outlier', *Proceedings of the 2019 Research in Adaptive and Convergent Systems, RACS 2019*, pp. 161–168. doi: 10.1145/3338840.3355641.

Cota, R. G. *et al.* (2010) 'An Unsupervised Heuristic-Based Hierarchical Method for Name Disambiguation in Bibliographic Citations', 61(May), pp. 1853–1870. doi: 10.1002/asi.

Dawoud, A., Shahristani, S. and Raun, C. (2019) 'Dimensionality Reduction for Network Anomalies Detection: A Deep Learning Approach', in *Advances in Intelligent Systems and Computing*. Springer Verlag, pp. 957–965. doi: 10.1007/978-3-030-15035-8_94.

Ferreira, A. A. *et al.* (2010) 'Effective self-training author name disambiguation in scholarly digital libraries', *Proceedings of the ACM International Conference on Digital Libraries*, pp. 39–48. doi: 10.1145/1816123.1816130.

Ferreira, A. A., Gonçalves, M. A. and Laender, A. H. F. (2012) 'A brief survey of automatic methods for author name disambiguation', *ACM SIGMOD Record*, 41(2), p. 15. doi: 10.1145/2350036.2350040.

Ferreira, V. O. *et al.* (2015) 'A model for anomaly classification in intrusion detection systems', in *Journal of Physics: Conference Series*. doi: 10.1088/1742-6596/633/1/012124.

Firdaus, F. (2018) 'Improving Data Integrity of Individual-based Bibliographic Repository Using Clustering Techniques', *Computer Engineering and Applications Journal*, 7(1), pp. 49–56. doi: 10.18495/comengapp.v7i1.223.

Fugate, M. and Gattiker, J. R. (2002) 'Anomaly detection enhanced classification in computer intrusion detection', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 186–197. doi: 10.1007/3-540-45665-1_15.

Gong, Z. and Chen, H. (2016) 'Model-based oversampling for imbalanced

sequence classification', *International Conference on Information and Knowledge Management, Proceedings*, 24-28-October-2016, pp. 1009–1018. doi: 10.1145/2983323.2983784.

Gu, S. *et al.* (2016) 'Name Disambiguation Method Based on Multi-step Clustering', *Procedia Computer Science*, 83(Ant), pp. 488–495. doi: 10.1016/j.procs.2016.04.237.

Han, H. *et al.* (2004) 'Two supervised learning approaches for name disambiguation in author citations', *Proceedings of the ACM IEEE International Conference on Digital Libraries, JCDL 2004*, pp. 296–305.

Hazra, R. *et al.* (2016) 'An efficient technique for author name disambiguation', *2016 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2016*. doi: 10.1109/ICCTAC.2016.7567344.

Hussain, I. and Asghar, S. (2017) 'A survey of author name disambiguation techniques: 2010–2016', *The Knowledge Engineering Review*, 32, p. e22. doi: 10.1017/S0269888917000182.

Hussain, I. and Asghar, S. (2018a) 'Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity', *Arabian Journal for Science and Engineering*. Springer Berlin Heidelberg, 43(12), pp. 7421–7437. doi: 10.1007/s13369-018-3099-0.

Hussain, I. and Asghar, S. (2018b) 'LUCID: Author name disambiguation using graph Structural Clustering', *2017 Intelligent Systems Conference, IntelliSys 2017*, 2018-Janua(September), pp. 406–413. doi: 10.1109/IntelliSys.2017.8324326.

John, H. and Naaz, S. (2019) 'Credit Card Fraud Detection using Local Outlier Factor and Isolation Forest International Journal of Computer Sciences and Engineering Open Access Credit Card Fraud Detection using Local Outlier Factor and Isolation', (September). doi: 10.26438/ijcse/v7i4.10601064.

Kharitonov, A. and Zimmermann, A. (2019) 'Intrusion detection using growing hierarchical self-organizing maps and comparison with other intrusion detection techniques', *CPSS 2019 - Proceedings of the 5th ACM Cyber-Physical System Security Workshop, co-located with AsiaCCS 2019*, pp. 13–23. doi: 10.1145/3327961.3329531.

Kim, J. (2018) 'Evaluating author name disambiguation for digital libraries: a case of DBLP', *Scientometrics*. Springer International Publishing, 116(3), pp. 1867–1886. doi: 10.1007/s11192-018-2824-5.

Kim, Jinseok and Kim, Jenna (2018) 'The impact of imbalanced training data on machine learning for author name disambiguation', *Scientometrics*. Springer International Publishing, 117(1), pp. 511–526. doi: 10.1007/s11192-018-2865-9.

Kunang, Y. N. *et al.* (2019) 'Automatic Features Extraction Using Autoencoder in Intrusion Detection System', *Proceedings of 2018 International Conference on Electrical Engineering and Computer Science, ICECOS 2018*. IEEE, (June 2019), pp. 219–224. doi: 10.1109/ICECOS.2018.8605181.

Leevy, J. L. *et al.* (2018) 'A survey on addressing high-class imbalance in big data', *Journal of Big Data*. Springer International Publishing, 5(1). doi: 10.1186/s40537-018-0151-6.

Li, N. and Han, J. (2017) 'The application of naive bayes classifier in name disambiguation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10603 LNCS, pp. 611–618. doi: 10.1007/978-3-319-68542-7_52.

Lu, J. *et al.* (no date) 'Tutorial Proposal : Synergy of Database Techniques and Machine Learning Models for String Similarity Search and Join'.

Luque, A. *et al.* (2019) 'The impact of class imbalance in classification performance metrics based on the binary confusion matrix', *Pattern Recognition*, 91, pp. 216–231. doi: 10.1016/j.patcog.2019.02.023.

Milojević, S. (2013) 'Accuracy of simple, initials-based methods for author name disambiguation', *Journal of Informetrics*, 7(4), pp. 767–773. doi: 10.1016/j.joi.2013.06.006.

Mitra, P. *et al.* (2007) 'Are your citations clean?', *Communications of the ACM*, 50(12), pp. 33–38. doi: 10.1145/1323688.1323690.

Mozafari, F. and Tahayori, H. (2019) 'Emotion Detection by Using Similarity Techniques', *2019 7th Iranian Joint Congress on Fuzzy and Intelligent Systems, CFIS 2019*. IEEE, pp. 1–5. doi: 10.1109/CFIS.2019.8692152.

Müller, M. C., Reitz, F. and Roy, N. (2017) 'Data sets for author name disambiguation: an empirical analysis and a new resource', *Scientometrics*, 111(3), pp. 1467–1500. doi: 10.1007/s11192-017-2363-5.

Naway, A. and Li, Y. (2019) 'Android Malware Detection Using Autoencoder', pp. 1–9.

Nicholson, S. W. and Bennett, T. B. (2016) 'Dissemination and Discovery of Diverse Data: Do Libraries Promote Their Unique Research Data Collections?', *International Information and Library Review*, 48(2), pp. 85–93. doi: 10.1080/10572317.2016.1176448.

On, B. W., Lee, I. and Lee, D. (2012) 'Scalable clustering methods for the name disambiguation problem', *Knowledge and Information Systems*, 31(1), pp. 129–151. doi: 10.1007/s10115-011-0397-1.

Palfrey, J. (2016) ‘Design choices for libraries in the digital-plus era’, *Daedalus*, 145(1), pp. 79–86. doi: 10.1162/DAED_a_00367.

Qin, Y. and Lou, Y. (2019) ‘Hydrological time series anomaly pattern detection based on isolation forest’, *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*. IEEE, (Itnec), pp. 1706–1710. doi: 10.1109/ITNEC.2019.8729405.

Rettig, L. *et al.* (2015) ‘Online anomaly detection over Big Data streams’, *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 1113–1122. doi: 10.1109/BigData.2015.7363865.

Shah, N. B., Balakrishnan, S. and Wainwright, M. J. (2016) ‘Feeling the bern: Adaptive estimators for Bernoulli probabilities of pairwise comparisons’, *IEEE International Symposium on Information Theory - Proceedings*, 2016-Augus, pp. 1153–1157. doi: 10.1109/ISIT.2016.7541480.

Shen, Q. *et al.* (2017) ‘NameClarifier: A Visual Analytics System for Author Name Disambiguation’, *IEEE Transactions on Visualization and Computer Graphics*, 23(1), pp. 141–150. doi: 10.1109/TVCG.2016.2598465.

Song, M., Kim, E. H. J. and Kim, H. J. (2015) ‘Exploring author name disambiguation on PubMed-scale’, *Journal of Informetrics*. Elsevier Ltd, 9(4), pp. 924–941. doi: 10.1016/j.joi.2015.08.004.

Tran, H. N., Huynh, T. and Do, T. (2014) ‘Author name disambiguation by using deep neural network’, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8397 LNAI(PART 1), pp. 123–132. doi: 10.1007/978-3-319-05476-6_13.

Vluymans, S. (2019) ‘Dealing with imbalanced and weakly labelled data in machine learning using fuzzy and rough set methods’, *Studies in Computational Intelligence*, 807, pp. 1–249. doi: 10.1007/978-3-030-04663-7_1.

Wang, J.-P. *et al.* (2013) ‘Effective string processing and matching for author disambiguation’, 15, pp. 1–9. doi: 10.1145/2517288.2517295.

Weiss, A. (2016) ‘Examining Massive Digital Libraries (MDLs) and Their Impact on Reference Services’, *Reference Librarian*, 57(4), pp. 286–306. doi: 10.1080/02763877.2016.1145614.

Wressnegger, C. *et al.* (2013) ‘A close look on n-grams in intrusion detection: Anomaly detection vs. classification’, in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 67–76. doi: 10.1145/2517312.2517316.

Wu, H. *et al.* (2014) ‘Unsupervised author disambiguation using Dempster–Shafer theory’, *Scientometrics*, 101(3), pp. 1955–1972. doi: 10.1007/s11192-014-1283-x.

Yang, X. *et al.* (2019) ‘A fast and efficient local outlier detection in data streams’, *ACM International Conference Proceeding Series*, Part F1477, pp. 111–116. doi: 10.1145/3317640.3317653.

Yao, C. *et al.* (2019) ‘Distribution Forest: An Anomaly Detection Method Based on Isolation Forest’, in, pp. 135–147. doi: 10.1007/978-3-030-29611-7_11.

Zhang, T., Wang, E. and Zhang, D. (2019) ‘Predicting failures in hard drivers based on isolation forest algorithm using sliding window’, *Journal of Physics: Conference Series*, 1187(4). doi: 10.1088/1742-6596/1187/4/042084.

Zhao, J., Wang, P. and Huang, K. (2013) ‘A semi-supervised approach for author disambiguation in KDD CUP 2013’, in *Proceedings of the 2013 KDD Cup 2013 Workshop on - KDD Cup ’13*. doi: 10.1145/2517288.2517298.

Zhu, Y. and Li, Q. (2013) ‘Enhancing object distinction utilizing probabilistic topic model’, *Proceedings - 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013*, pp. 177–182. doi: 10.1109/CLOUDCOM-ASIA.2013.61.

AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

OLEH :

ZAQQI YAMANI A

09042611822005

PROGRAM MAGISTER ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJYA

TAHUN 2019

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab ini menjelaskan tentang latar belakang diangkatnya tema permasalahan tesis ini, yang kemudian dari masalah dilakukan perumusan masalah. Di dalam bab ini pula dibahas mengenai batasan masalah, tujuan penelitian serta metodologi penulisan dalam penelitian ini.

1.1. Latar Belakang

Digital Library (DL) saat ini merupakan salah satu pusat literasi yang paling berkembang pesat, salah satunya pada bidang akademik. Hal ini dipengaruhi berbagai faktor seperti pemotongan anggaran untuk perpustakaan tradisional, ruang penyimpanan yang hampir tak terbatas dengan biaya yang jauh lebih rendah, kemudahan penggunaan dan tidak ada batas fisik dari penelitian (Palfrey, 2016; Weiss, 2016). Beberapa DL, antara lain DBLP1, MEDLINE2, CiteSeer3 arXiv4, MAS5, Google Scholar6, dan BDBComp7 secara luas digunakan oleh para peneliti untuk menemukan literatur ilmiah untuk penelitian dan penemuan mereka (Nicholson and Bennett, 2016). Selain memberikan beberapa informasi dan analisis yang berguna bagi pengambilan keputusan dan menyediakan konten berkualitas tinggi pada penelitian (Mitra *et al*., 2007), DL juga memiliki beberapa sumber kesalahan antara lain adalah tipografi, pemindaian dan konversi data, menemukan dan mengganti, menyalin dan menempel, meta data, perangkat lunak pengumpulan kutipan yang tidak sempurna, format kutipan yang berbeda, nama-nama penulis yang ambigu, pembuatan konten terdesentralisasi dan singkatan dari judul tempat publikasi yang ambigu, dll (Ferreira, Gonçalves and Laender, 2012).

Ambiguitas Nama Penulis atau yang lebih dikenal dengan *Author Name Disambiguation* (AND) adalah salah satu masalah yang menurunkan kualitas dan keandalan informasi yang diperoleh dari DL (Tran, Huynh and Do, 2014). Konten DL dan kualitas layanan sangat dipengaruhi oleh masalah ambiguitas nama penulis dalam kutipan dan dianggap sebagai salah satu masalah tersulit yang dihadapi oleh para peneliti perpustakaan digital dari penelitian (Hussain and Asghar, 2017). AND menjadi sebuah masalah ketika satu set catatan publikasi berisi nama penulis yang

menimbulkan lebih dari satu interpretasi, yaitu penulis yang sama dapat muncul dengan nama yang berbeda (Ferreira, Gonçalves and Laender, 2012). Hal tersebut menjadi poin yang mengurangi kualitas dari informasi serta mengurangi pula keandalan informasi tersebut karena berdampak pada informasi terhadap penulis, organisasi dan hal lain yang ditampilkan sebagai bagian dari catatan publikasi tersebut (Müller, Reitz and Roy, 2017).

Beberapa penelitian telah dilakukan dalam rangka mengklasifikasikan AND, terutama pada proses klasifikasi *author matching*. Proses klasifikasi tersebut mengedepankan proses *pairwise* dan *distance* dari nama-nama author (Wang *et al*., 2013). Klasifikasi terhadap data *author* diharapkan memberikan interpretasi yang tepat dan prediksi serta akurasi yang tinggi ketika dijalankan pada setiap dataset AND dan *bibliography*.

Dalam perkembangannya, AND dalam klasifikasi *author matching* menciptakan tantangan yang menakutkan dalam teknik disambiguasi karena sering menarik kesimpulan yang salah pada data publikasi yang tidak lengkap (Song, Kim and Kim, 2015). Ada sejumlah solusi yang dijalankan terhadap hal tersebut, diantaranya dengan *un-supervised* yaitu dilakukan berdasarkan kesamaan catatan bibliografi atau pola penulisan yang bersifat umum (Milojević, 2013), model NDMC atau multi step clustering yang dilakukan dengan menyamaratakan nama penulis atau menggabungkan karakteristik singkat dari informasi data publikasi (Gu *et al*., 2016). Selain itu, ada teknik lain yang pernah digunakan yaitu LUCID, di mana dilakukan dengan menggunakan algoritma pendeteksian komunitas dan operasi grafik yang di akhir fase teknik ini tetap menggunakan fungsi kemiripan dari data publikasi tersebut (Hussain and Asghar, 2018b). Lalu ada teknik sistem analisis visual yang disebut NameClarifier yang secara interaktif mengelompokkan nama penulis dalam publikasi di dalam lingkaran tertentu, lalu menghitung dan memvisualisasikan kesamaan antara nama ambigu dan yang telah dikonfirmasi di Digital Library (Shen *et al*., 2017). Namun, keempat metode tersebut tidak mementingkan akurasi dari proses klasifikasi data publikasi. Semua metode tersebut memberikan gambaran yang sama dalam teknik menuju klasifikasi yaitu dengan melakukan *pairwise* lalu menemukan kemiripan dari data yang dipasangkan tersebut.

Di sisi lain, *Machine Learning* juga telah banyak digunakan untuk melakukan proses klasifikasi AND dan menghasilkan kinerja yang memuaskan (Hussain and Asghar, 2017). Diantaranya dengan *supervised AND techniques* dengan menggunakan *boosted tree classification* yang fokus pada proses pemfilteran dan pencocokan nama dan afiliasi dalam sebuah publikasi (Wang *et al*., 2013). *un supervised AND techniques* dengan pendekatan teori DST (*Dempster-Shafer Theory)* yaitu menghitung kesamaan fitur tingkat tinggi seperti afiliasi, tempat, content, rekan penulis, kutipan, korelasi Web(Wu *et al*., 2014). Selanjutnya semi *supervised* AND *techniques* dengan menggunakan Algoritma mendeteksi kesamaan di antara objek. Mereka membangun matriks dua dimensi untuk penulisan bersama dan hubungan topik dan menghitung jarak antara dua simpul dengan bantuan jarak *Euclidean* (Zhu and Li, 2013). Selain itu, ada *Graph-Based AND techniques* dengan menggunakan algoritma *multi-level Graph Partinging* (MGP), dan algoritma *Multi-Level Graph Partinging* dan *Merging* (MGPM) (On, Lee and Lee, 2012). dan *Graph Based AND techniques* yang menggunakan kesamaan antara catatan *bibliografi* dan kelompok catatan baru untuk penulis dengan catatan kutipan yang sama di DL, atau untuk penulis baru ketika bukti kesamaan tidak cukup kuat. Beberapa *heuristik* khusus digunakan untuk memeriksa apakah referensi dari catatan kutipan baru milik penulis yang sudah ada di DL atau milik penulis yang baru (yaitu, penulis tanpa catatan kutipan di DL), menghindari menjalankan proses disambiguasi di seluruh DL (de Carvalho *et al*., 2011). Namun, metode diatas menghasilkan kinerja akurasi, presisi, spesifisitas dan sensitivitas kurang memuaskan. Dan sama seperti sebelumnya, semua metode tersebut dilakukan dengan melakukan *pairwise,* lalu dilanjutkan dengan menghitung jarak untuk menghasilkan *similarity* dari data author yang dipasangkan.

Selain itu, metode lain yang pernah dilakukan untuk melakukan klasifikasi terdapat *author matching* yaitu dengan menggunakan *deep structure*. salah satu metode yang menggunakan struktur tersebut adalah metode *Deep Neural Network*. Arsitektur ini memiliki dua komponen utama. Dalam komponen pertama, data diambil sebagai input dan representasi data dihitung dengan mencari kemiripan dari data. Komponen kedua mengambil set fitur dasar sebagai inputnya dan mempelajari fitur-fitur di dalamnya adalah lapisan tersembunyi untuk menyamarkan nama

pembuatnya (Tran, Huynh and Do, 2014). Metode ini telah menghasilkan akurasi tinggi pada nilai 99,31%. Sama seperti penelitian sebelumnya, proses yang dilakukan adalah dengan *pairwise*. Namun, penelitian ini dilakukan dengan data yang sedikit dimana *pairwise* data menghasilkan hanya sekitar 30.537 data.

Dari berbagai metode yang telah dilakukan, dapat ditarik kesimpulan bahwa pra pemrosesan sebelum masuk pada proses klasifikasi adalah dengan melakukan *pairwise* dan dilanjutkan dengan menghitung jarak (*similarity)* dari data yang telah dipasangkan. Dalam prosesnya, teknik *pairwise* pada data author akan menimbulkan *imbalanced* data yang tinggi, dimana semakin banyak data yang dipasangkan akan menimbulkan tingkat *imbalanced* yang semakin tinggi pula atau dengan kalimat lain dimana data negatif menjadi sangat dominan dibanding data yang positif (Kim and Kim, 2018). Hal itu menjadikan hasil proses klasifikasi menjadi meragukan karena proses pencarian data yang bernilai positif menjadi sulit dan bisa dikatakan seperti mencari data yang langka.

Proses diatas sama halnya dengan dengan proses klasifikasi pada deteksi intrusi pada sistem komputer, dimana deteksi pada gangguan sistem komputer seperti mencari barang langka di dalam proses yang secara dominan berjalan normal (Ferreira, Gonçalves and Laender, 2012). Selain itu, kasus pada deteksi gangguan pada *network* pun memiliki masalah yang sama, dimana hal-hal yang dianggap gangguan atau *instrusion* merupakan hal yang sulit dikenali atau dicari karena persentase keberadaannya sangat kecil dibandingkan proses yang ada dan dianggap bukan merupakan sebuah *intrusion* (Dawoud, Shahristani and Raun, 2019). Adapun teknik yang dilakukan untuk mengklasifikan dua kasus diatas adalah dengan menggunakan metode *machine learning* pada pendekatan *Anomaly Detection* yang dapat menghasilkan nilai akurasi pada data negatif maupun data positif.

Berdasarkan hal tersebut, tingkat *imbalanced* yang tinggi pada proses klasifikasi *Author matching* dapat dikategorikan sama dengan yang terjadi pada proses *intrusi* atau gangguan pada sistem komputer dan *network*. Oleh karena itu, pendekatan *anomaly detection* akan menghasilkan tingkat akurasi yang tinggi pada proses klasifikasi. Sama seperti pada penggunaan *anomaly detection* di intrusi sistem komputer, model yang digunakan untuk proses klasifikasi pada penelitian

ini juga dengan menggunakan model algoritma *isolationForest* dan *local Outlier Factor (LOF)*.

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka perumusan masalah yang diambil pada penelitian ini adalah tentang "Bagaimana membuat sistem pengklasifikasi ambiguitas nama penulis (*author name disambiguation)* dalam *author matching* dengan menggunakan *Machine Learning* pada pendekatan *Anomaly Detection?*".

Dalam penelitian ini, perumusan masalah dijabarkan dalam bentuk pertanyaan sebagai berikut :

a. Bagaimana proses pra pengolahan data *Author Matching* dari suatu *digital library?*
b. Bagaimana menemukan model *Anomaly Detection* yang terbaik dengan menggunakan algoritma *IsolationForest* dan *Local Outlier Factor (LOF)* untuk melakukan klasifikasi pada data yang berdimensi tinggi?
c. Bagaimana mengukur hasil kinerja dari model *Anomaly Detection* berdasarkan parameter akurasi, sensitivitas, spesifisitas, presisi dan F1 score?

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah :

1. System hanya berupa simulasi untuk melakukan proses klasifikasi data penulis (*author matching)* pada data di DL.
2. Dataset yang digunakan adalah data nama-nama penulis lengkap dengan judul jurnal serta tahun terbit yang diambil dari dataset SCAD-zBMATH dengan judul featured-dataset-merged yang terdiri dari 11.924 baris data publikasi (Müller, Reitz and Roy, 2017).
3. Pra pengolahan akan menggunakan teknik *pairwise* dan *similarity*.

1.4. Tujuan

Adapun tujuan dari penelitian ini adalah :

1. Melakukan proses pra pengolahan data publikasi yang memiliki disambiguitas untuk menginterpretasi dan mengelompokkan penulis berdasarkan kemiripan nama, judul, tahun dan nama kecil dari penulis.
2. Menganalisis model klasifikasi dari Anomaly Detection dalam mengklasifikasikan penulis yang sama atau bukan pada judul tulisan dan tahun terbit yang berbeda.
3. Mengukur kinerja klasifikasi Anomaly Detection berdasarkan parameter akurasi, sensitivitas, spesifisitas, presisi dan F1 score.

1.5. Manfaat

Manfaat dari penelitian ini adalah menjadi landasan penggunaan metode anomaly detection dalam klasifikasi Author Names Disambiguation terutama dalam konsep Author Matching. Selain itu, manfaat dari penelitian ini adalah sebagai berikut :

1. Anomaly Detection dapat menjadi metode tambahan baru dalam proses disambiguasi terhadap nama penulis.
2. Model klasifikasi yang dipakai dalam penelitian ini dapat digunakan lebih lanjut untuk meningkatka hasil accuracy dan presisi dalam proses AND terutama pada konsep Author Matching.

1.6. Metodologi Penulisan

Metodologi penulisan pada tesis ini terdiri dari lima bab sebagai berikut:

BAB I : PENDAHULUAN

Bab I berisi pendahuluan berupa latar belakang, perumusan masalah, tujuan dan manfaat dari topik yang dipilih.

BAB II: TINJAUAN PUSTAKA

Bab II berisi kerangka teori dan pustaka yang berhubungan dengan klasifikasi *Author Matching* pada data DL dengan menggunakan metode *Anomaly Detection* yang mengacu pada beberapa penelitian jurnal publikasi.

BAB III : METODOLOGI PENELITIAN

Bab III berisi metodologi yang menjelaskan secara bertahap dan terperinci tentang langkah-langkah yang digunakan untuk mencari, mengumpulkan dan menganalisa yang berkaitan dengan *author Matching*. Metodologi ini menjelaskan pendekatan atau algoritma *author matching*, serta model yang digunakan sehingga tujuan dari penulisan dapat tercapai.

BAB IV: HASIL DAN ANALISA SEMENTARA

Bab IV berisi hasil pengujian yang telah dilakukan, data-data yang diambil dari pengujian tersebut akan dianalisa menggunakan berbagai macam teknik, selain itu di bab ini juga membahas kevalidasian dari sistem yang telah dibuat.

BAB V: KESIMPULAN

BAB V berisi tentang kesimpulan apa yang diperoleh oleh penulis serta merupakan jawaban dari setiap tujuan yang ingin dicapai.

BAB II
TINJAUAN PUSTAKA

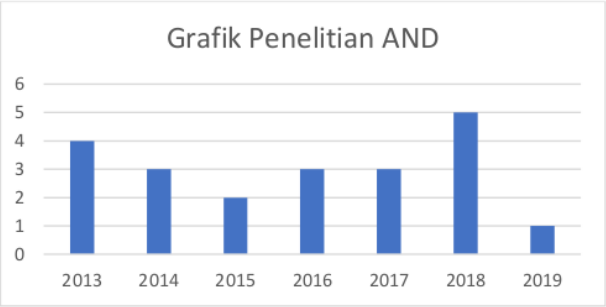
Bab ini berisi mengenai tinjauan pustaka yang berhubungan dan relevan dengan *Author Names Disambiguation* dalam Klasifikasi *Author Matching* dengan Menggunakan Metode *Machine Learning* Pada Pendekatan *Anomaly Detection*. Bab ini juga akan menjabarkan penelitian-penelitian terkait dan membandingkan hasil penelitian tersebut dari sisi kinerja algoritma klasifikasi, pembahasan teori *Author Names Disambiguation*, *Author Matching, Pairwise and Similarity Technique,* dan metode *anomaly detction*.

2.1. Tinjauan Penelitian

Ambiguitas Nama Penulis atau yang lebih dikenal dengan *Author Name Disambiguation* (AND) adalah salah satu masalah yang menurunkan kualitas dan keandalan informasi yang diperoleh dari *Digital Library* (DL) (Tran, Huynh and Do, 2014) dan (Hazra *et al*., 2016). Konten DL dan kualitas layanan sangat dipengaruhi oleh masalah ambiguitas nama penulis dalam kutipan dan dianggap sebagai salah satu masalah tersulit yang dihadapi oleh para peneliti perpustakaan *digital* (Hussain and Asghar, 2017). AND menjadi sebuah masalah ketika satu set catatan publikasi berisi nama penulis yang menimbulkan lebih dari satu interpretasi, yaitu penulis yang sama bisa muncul dengan nama yang berbeda (Ferreira, Gonçalves and Laender, 2012) dan (Firdaus, 2018), hal itu disebabkan inkonsistensi dalam penyajian nama *author* (Milojević, 2013). Konten DL yang menimbulkan lebih dari satu interpretasi tersebut menjadi tantangan dalam proses klasifikasi terutama pada pengolahan data *author matching* (Song, Kim and Kim, 2015).

Pembahasan mengenai AND belum mendapat perhatian secara khusus dalam beberapa penelitian terutama dalam penentuan akurasi , hal ini dikarenakan kebutuhan akan *resource* yang besar dalam pengolahan data yang memiliki jumlah yang sangat besar pula (Milojević, 2013) dan jumlah penelitian spesifik terhdapa AND dalam 6 tahun terakhir dapat dilihat pada gambar 2.1 . Beberapa evaluasi telah dilakukan terhadap penelitian mengenai AND, metode dan teknik yang dilakukan pada sarnya memiliki kesamaan yaitu dengan melakukan pemasangan dari data

authors dan lebih lanjut dengan melakukan pencarian kemiripan dati hasil pemesangan tersebut (Kim, 2018).



Gambar 2.1. Grafik Jumlah Penelitian AND

Penelitian terhadap AND ini selanjutnya dipisahkan menjadi 3 bentuk penelitian yaitu *author matching, author grouping* dan *author assignment* dimana masing-masing tingkatan memiliki karakteristik atau interpretasi yang berbeda-beda dalam hasil akurasi (Ferreira, Gonçalves and Laender, 2012). Masing-masing pola tersebut telah dilakukan penelitian lebih lanjut untuk menghasilkan tingkat akurasi dalam klasifikasi data *authors* (Hussain and Asghar, 2017).

AND atau lebih spesifiknya adalah *author matching* dalam perkembangannya menjadi topik yang menarik dalam dunia klasifikasi terhadap data publikasi dalam DL. Hal itu disebabkan karena AND menjadi poin yang mengurangi kualitas dari informasi serta mengurangi pula keandalan informasi tersebut karena berdampak pada informasi terhadap penulis, organisasi dan hal lain yang ditampilkan sebagai bagian dari catatan publikasi tersebut (Müller, Reitz and Roy, 2017).

2.1.1. Klasifikasi Author Matching

Author matching merupakan langkah awal dalam proses disambiguasi terhadap konten DL (Wang *et al*., 2013). Beberapa penelitian yang dilakukan terhadap AND, hampir semuanya berpijak atau diawali pada proses *author matching* tersebut. Beberapa penelitian dengan berbagai macam metode telah

dilakukan untuk menemukan cara terbaik dan metode terbaik dalam proses klasifikasi disambiguasi *author matching*.

Beberapa metode yang pernah dilakukan antara lain, *heuristic based hierarchical methode* (Cota *et al*., 2010), dengan menggunakan *associative classifier* (Ferreira, Gonçalves and Laender, 2012), lalu ada pula metode K-NN, *Random forest*, C4.5 (Ferreira, Gonçalves and Laender, 2012), selanjutnya dengan metode SVM dan *Naive Bayes* (Han *et al*., 2004). Selain itu, ada teknik *boosted tree classification* (Hussain and Asghar, 2018b) (Berzins *et al*., 2012). Ada lagi dengan menggunakan teori dan algoritma *dempster shafer* (Wu *et al*., 2014) dan *Graph structural* dan *hybrid simlarity* (Hussain and Asghar, 2018a). Dan ada pula penelitian yang menggunakan teknik deep learning dengan menggunakan pendekatan deep neural network (DNN) (Tran, Huynh and Do, 2014).

Tabel 2.1

Penelitian Terhadap Author Matching

No.	Peneliti/Jurnal	Metode	Implementasi	Teknik
1	(Cota *et al*., 2010)	Heuristic Based Hirarchical	Klasifikasi Biner antar author	Pairwise dan Similarity
2	(Ferreira *et al*., 2010)	Associative classifier	Klasifikasi Biner	Pairwise
3	(Ferreira, Gonçalves and Laender, 2012)	K-NN	Klasifikasi Biner	Pairwise dan Similarity
4	(Ferreira, Gonçalves and Laender, 2012)	Random Forest	Klasifikasi Biner	Pairwise dan Similarity
5	(Ferreira, Gonçalves and Laender, 2012)	C4.5		Pairwise dan Similarity
6	(Han *et al*., 2004)	SVM	Klasifikasi Biner	Pairwise dan Similarity

7	(Li and Han, 2017)	Naive Bayes	Klasifikasi Biner	Pairwise dan Similarity
8	(Hussain and Asghar, 2018b) (Berzins *et al*., 2012)	Boosted tree Classification	Klasifikasi Biner	Pairwise dan Similarity
9	(Wu *et al*., 2014)	Dempster-Shafer Theory		Pairwise
10	(Hussain and Asghar, 2018a)	Graph Structural	Klasifikasi Biner	Pairwise dan Similarity
11	(Tran, Huynh and Do, 2014)	Deep Neural Network	Klasifikasi Biner	Pairwise dan Similarity

2.2. Teknik Pemasangan (*Pairwise*) dan Penghitungan Kemiripan (*Similarity*)

Dalam teknik disambiguasi terhadap data penulis atau *author* terutama pada *author matching,* teknik pengolahan data menjadi penting sebelum masuk ke dalam algoritma atau mesin *classifier*. Proses pengolahan data atau pra processing ini menjadi titik awal pengenalan atau pengelompokkan fitur dan label yang akan menjadi mode pembelajaran dari classifier yang akan digunakan (Wu *et al*., 2014).

Di beberapa penelitian yang telah dilaksanakan teknik pengolahan terhadap data author di dominasi oleh konsep pemasangan atau *pairwise*. Dan teknik pemasangan tersebut banyak dilakukan dengan teknik matematika sederhana yaitu kombinasi. Lalu setelah semua data memiliki pasangannya, pra pengolahan data author dilaksanakan dengan mencari jarak antar data tersebut atau sering dikenal dengan teknik *similarity*. Teknik tersebut bertujuan memberi nilai pada data author yang telah dipasangkan dengan harapan memberi prediksi kemiripan dari data tersebut (Kim, 2018), (Tran, Huynh and Do, 2014), (Ferreira, Gonçalves and Laender, 2012), (Hussain and Asghar, 2017).

Kedua teknik diatas menjadi sangat penting dalam proses pra pengolahan data *author matching*. Keduanya akan menjadi titik awal untuk meningkatkan nilai akurasi pada proses klasifikasi *author matching*.

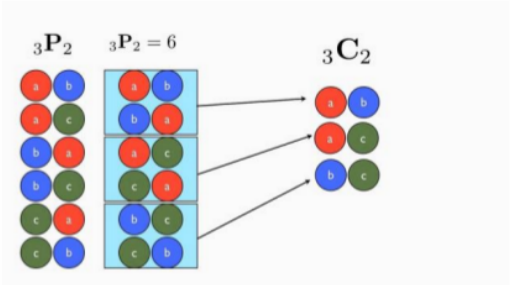
2.2.1. Pemasangan dengan Kombinasi (*Pairwise combination*)

Ada beberapa teknik untuk mendapatkan hasil probabilitas dari perbandingan data atau item yang dikoleksi, antara lain dengan *pariwise combinations*. Metode ini menjadi efektif ketika kita akan mendapat hasil dari suatu perbandingan dengan tujuan probabilitas di masa mendatang (Shah, Balakrishnan and Wainwright, 2016). Teknik ini mengedepankan proses perbandingan berpasangan yang dapat digunakan untuk memperoleh kecenderungan terkait dari setiap kriteria yang dibandingkan.

Dalam *pairwise combination,* teknik yang biasa digunakan untuk meningkatkan hasil dari perbandingan yang dilakukan adalah dengan melakukan kombinasi dari data yang ingin dipasangkan. Tujuan dari dilakukannya kombinasi adalah agar setiap baris data memiliki pasangan atau berdampingan dengan data di baris berikutnya dan seterusnya. Dan di dalam pengolahan dataset untuk *author matching,* proses kombinasi akan sangat membantu dalam menyajikan baris data demi meningkatnya akurasi dalam proses klasifikasi (Zhao, Wang and Huang, 2013). Di dalam rumus matematika sederhana, kombinasi adalah menggabungkan beberapa objek dari suatu grup tanpa memperhatikan urutan (rumus 2.1).

$$\frac{n!}{r!(n-r)!} = \binom{n}{r} \qquad \textbf{(2.1)}$$

Dimana ***n*** adalah jumlah objek yang bisa dipilih dan ***r*** adalah jumlah yang harus dipilih. Hasil dari pengolahan data dengan kombinasi akan menyajikan data yang berpasangan secara menyeluruh, dan akan memberikan jumlah data yang lebih banyak dari data awal.



Gambar 2.2. teknis berpasangan menggunakan kombinasi

Proses ini akan menghasilkan baris data yang lebih banyak dari jumlah baris data sebenarnya dengan tujuan dapat dibandingkan antara baris data pertama dengan baris data kedua, baris pertama dengan baris ketiga dan seterusnya sampai setiap baris bertemu. Saat semua baris telah dipasangkan, akan diambil cara untuk menghitung atau menganalisis data tersebut berdasarkan output yang akan dihasilkan dan digunakan.

2.2.2. Penghitungan Kemiripan (*Similarity*)

Similarity merupakan proses dalam *feature extraxtion* dengan melakukan pencarian nilai kemiripan atau kesamaan dari data. Dalam metode *machine learning,* untuk kasus pencarian kemiripan dari data *string*, teknik *similarity* menjadi teknik yang paling dominan dipakai. Teknik ini mampu membangun sebuah informasi melalui nilai atau dengan kata lain menerjemahkan sebuah data *string* menjadi data yang memiliki nilai (Mozafari and Tahayori, 2019) Lu dkk. (2019).

Dalam penerapannya, teknik ini dilakukan setelah terbentuknya data yang berpasangan. Selain untuk data *author matching,* teknik ini dipakai pula pada proses pengenalan emosi berbasis gambar. Teknik ini dilakukan dengan menbandingkan dua set data untuk melihat mana data yang sama dan mana data yang berbeda.

Di banyak penelitian tentang AND, teknik *similarity* yang banyak digunakan adalah dengan menggunakan koefisien jaccard, jaro, euclidean dan levenshtein. Adapun koefisien jaccard yang paling umum digunakan untuk mengukur kesamaan antara dua set data. Hasil yang dimunculkan adalah rentang 0% sampai dengan 100%. Semakin tinggi persentasenya, semakin mirip populasi kedua data tersebut (Ferreira, Gonçalves and Laender, 2012), (Tran, Huynh and Do, 2014), (Song, Kim and Kim, 2015), (Shen *et al*., 2017), (Zhu and Li, 2013).

Keofisien jaccard didefinisikan dengan rumus matematis untuk menghasilkan pengukuran jarak antar string (rumus 2.2).

$$\boldsymbol{jaccard}\ (\boldsymbol{a},\boldsymbol{b}) = \frac{|\boldsymbol{a} \cap \boldsymbol{b}|}{|\boldsymbol{a} \cup \boldsymbol{b}|} = \frac{|\boldsymbol{a} \cap \boldsymbol{b}|}{|\boldsymbol{a}|+|\boldsymbol{b}|-|\boldsymbol{a} \cap \boldsymbol{b}|} \qquad \textbf{(2.2)}$$

Koefisien jaccard memiliki kelemahan dimana koefisien ini tidak memperhatikan term frequency (berapa kali suatu term terdapat di dalam suatu dokumen). Perlu diketahui, bahwa terms yang jarang muncul dalam suatu koleksi sangat bernilai dari sisi informasi, tetapi koefisien Jaccard tidak mempertimbangkan hal ini. Jadi kita butuh cara lain untuk menormalisasikannya. Namun untuk pengolahan data AND terutama pada poin *author matching,* teknik perhitungan koefisien jaccard dinilai lebih baik untuk digunakan (Ferreira, Gonçalves and Laender, 2012).

2.3. Ketidakseimbangan Data (Imbalance Data)

Ketidakseimbangan data atau biasa disebut *imbalance data* merupakan kondisi dimana ada satu kelas minoritas yang memiliki jumlah data yang sangat sedikit dibandingkan kelas lainnya (Vluymans, 2019). Hal ini merupakan permasalahan umum dalam proses klasifikasi data *machine learning,* dimana terjadi rasio pengamatan yang tidak seimbang pada setiap kelas. Kondisi ini sering ditemukan pada data diagnosis medis, penyaringan spam dan deteksi pada penipuan kartu kredit. Keadaan imbalanced mengakibatkan kinerja algoritma classifier standar menurun secara signifikan karena kebanyakan algoritma classifier standar mengasumsikan distribusi instance dalam kelas adalah balanced sehingga hasil klasifikasi lebih cenderung ke kelas mayoritas (Luque *et al*., 2019).

Metode utama pada pendekatan level classifier adalah menyesuaikan operasi algoritma yang ada untuk membuat classifier lebih konduktif terhadap klasifikasi dalam kelas minoritas. Metode-metode yang digunakan pada level classifier adalah one-class learning, metode ensemble dan cost sensitive learning. Pendekatan pada level data merujuk pada berbagai teknik resampling (oversampling dan undersampling) dan sintesis data untuk memperbaiki kecondongan distribusi kelas data. Salah satu metode yang digunakan pada level data adalah Synthetic Minority Oversampling Technique atau lebih dikenal dengan nama SMOTE.

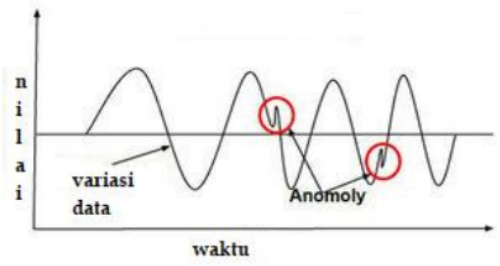
Pada dataset disambiguasi nama penulis atau AND, ketidakseimbangan atau imbalance data tersebut tidak dapat dilihat. Namun, teknik yang dipilih dalam proses pengolahan data sebelum masuk ke classifier, akan menentukan apakah data

menjadi seimbang atau tidak seimbang. Dalam hal ini, pemilihan teknik pairwise menggunakan kombinasi membuat data author yang positif (benar) menjadi lebih sedikit dibandingkan dengan data author yang negatif (bukan sebenarnya).

2.4. Deteksi Data Anomali (*Anomaly Detection*)

Anomaly detection merupakan sebuah teknik yang digunakan untuk mengidentifikasi pola-pola yang tidak biasa atau tidak sesuai perilaku yang diharapkan, dan hal tersebut lebih sering dikenal dengan outlier. Hal ini telah dilakukan dalam banyak kasus, mulai dari deteksi instrusi pada sistem komputer, lalu intrusi pada pada network (dilakukan dengan mendeteksi pola aneh dalam lalu lintas jaringan data untuk mengetahui jaringan terkena hacking atau tidak), pemantauan kesehatan (mendeteksi tumor ganas dalam pemindaian MRI), hingga penipuan dalan transaksi kartu kredit. (Fugate and Gattiker, 2002; Ferreira *et al*., 2015; Dawoud, Shahristani and Raun, 2019).

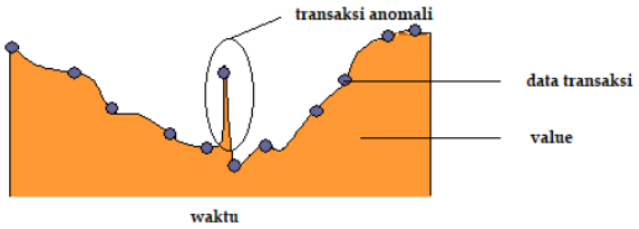
Dalam semua kasus yang pernah dipakai di atas, terdapat satu kesamaan dimana titik data itu berbeda dari item lain di dalam kumpulan data dan tidak dapat diidentifikasi dengan cepat karena tidak berada pada jalur yang diharapkan. Anomali detection dipusatkan pada pengenalan pada data yang langka atau sulit dikenali dalam barisan data.



Gambar 2.2. Pola Data Pada Konsep Anomaly Detection

Dalam proses klasifikasi data, anomali detection mengenali atau mengidentifikasi barang langka, peristiwa atau pengamatan yang menimbulkan kecurigaan dengan berbeda dari signifikan data mayoritas data (Wressnegger *et al*., 2013). Anomali dapat dikategorikan menjadi :

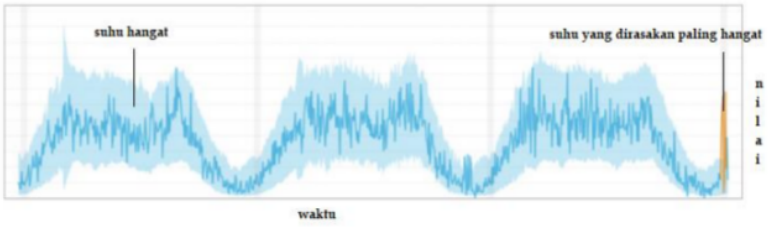
1. Anomali Terpusat (*Point anomalies*) : contoh data yang terlalu jauh atau yang terlalu berbeda dari signifikan data, misalnya pada penipuan kartu kredit.



Gambar 2.3. Plot Point Anomaly Detection

gambar diatas menjelaskan satu titik data yang sangat berbeda dari pola data atau alur data yang ada dalam penggunaan kartu kredit. Titik tersebut berjarak sangat dekat dengan nilai yang sangat signifikan dan hanya sekali dari beberapa kali penggunaan. Dalam ha ini, poin anomali merupakan data yang sangat berbeda dari kebiasaan data yang berjalan secara normal.

2. Anomali Kontekstual (*Contextual anomalies*) : penyimpagan data yang mengarah ke anomali didasarkan pada sesuatu yang kontekstual, misalnya pada kondisi cuaca, dima satu wilayah sedang merasakan puncak cuaca hangat (yang tidak pernah dirasakan sebelumnya) pada musim panas yang sedang berlangsung. Tapi hal tersebut tidak berlaku di wilayah lain pada musim yang sama. Hal tersebut dinamakan anomali karena berdasarkan kontekstual.



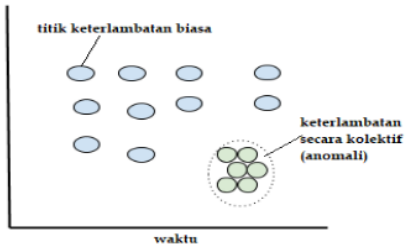
Gambar 2.4. Plot time-series cuaca pada Contextual Anomaly Detection

Gambar diatas merupakan data deret waktu tertentu selama periode tertentu yang menggambarkan keadaan suhu pada musim panas. Selama musim

panas tersebut cuaca tiap hari dirasakan sama, hingga pada hari tertentu cuaca dirasakan lebih hangat dibandingkan hari lainnya (digambarkan dengan warna orange).

3. Anomali Kolektif (*Collective anomalies*) : penyimpangan data pada anomali kolektif tidak hanya dengan melihat titik data secara individual, tetapi juga menganalisis perilaku secara kolektif. penggunaan serangkaian data secara kolektif untuk menimbulkan anomali.

   Gambar dibawah ini menganalogikan keadaan anomali pada proses pengiriman barang atau pasokan pada sebuah industri tekstil. Pengiriman yang terlambat sangat umum di industri seperti ini. Tetapi pada hari tertentu, jika ada banyak keterlambatan pengiriman pada pesanan maka mungkin perlu penyelidikan lebih lanjut. Pengiriman yang tertunda tidak berkontribusi terhadap hal ini secara individu tetapi ringkasan kolektif dipertimbangkan ketika menganalisis situasi seperti ini.

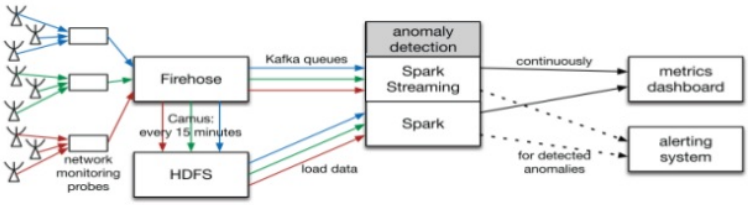


Gambar 2.5. Plot keterlambatan distribusi barang dalam Collective Anomaly Detection

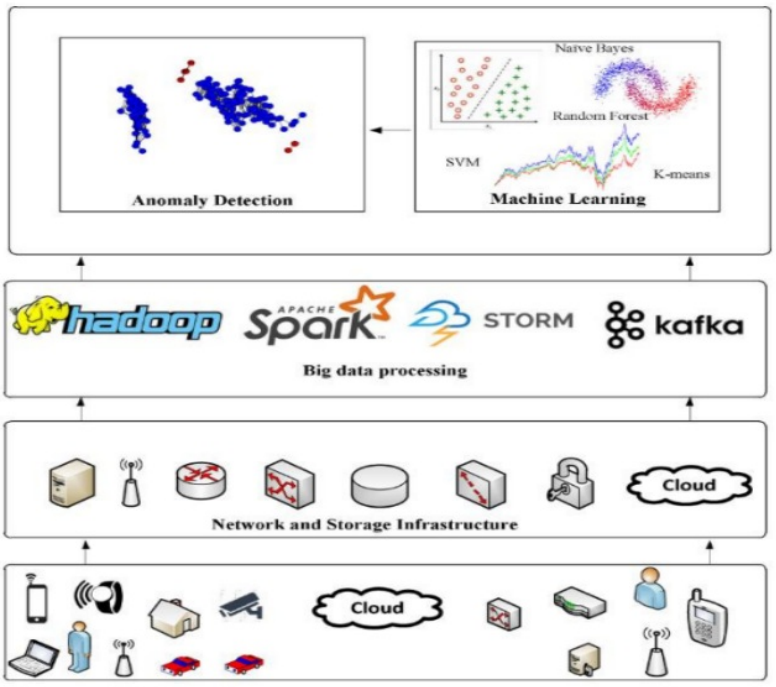
Anomaly detection sangat penting hampir di semua bidang disiplin ilmu (contoh, fisika, keuangan, machine learning dan cyber security). dalam machine learning atau disiplin ilmu manapun, kualitas data sama pentingnya dengan kualitas model prediksi ataupun klasifikasi.

Pada umumnya, anomaly detection ini digunakan pada kumpulan data yang besar (big data) dimana kualitas data menjadi acuan pada informasi yang dihasilkan. Jaminan atas kualitas data harus dengan melakukan identifikasi awal terhadap pola-pola yang tidak biasa yang terjadi di dalam data (Rettig *et al*., 2015).

Penelitian tentang big data ini pernah dilakukan dengan mendeteksi sensor pada berbagai perangkat pintar yang dikomunikasikan pada jaringan, perangkat tersebut melakukan penyimpanan pada cloud dan media penyimpanan lainnya lalu diproses dengan teknologi big data dan hasilnya digunakan untuk analisis anomali dengan menggunakan machine learning (Luque *et al*., 2019).



Gambar 2.6. Integrasi Anomaly Detection pada Big Data



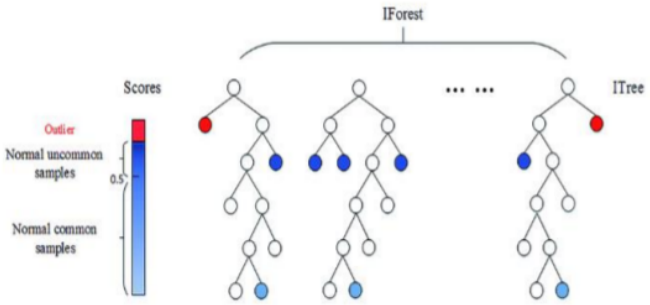
Gambar 2.7. Penerapan Anomaly Detection pada Big Data

Dalam penggunaan metode anomali detection dalam machine learning, kita perlu memastikan dan menyelidiki secara menyeluruh bahwa model yang digunakan mampu mengidentifikasi anomali secara efektif dan konsisten (Ferreira

et al., 2015). Model algoritma yang digunakan dalam mendetteksi anomali adalah isolation forest dan local outlier factor.

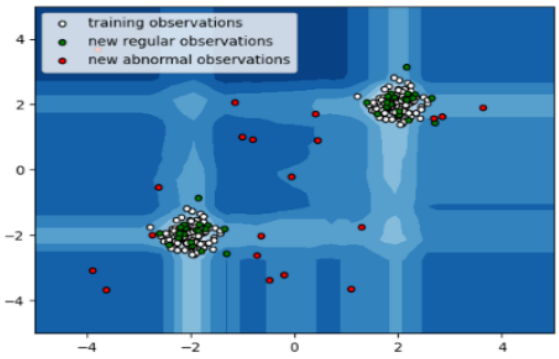
2.4.1. Algoritma Isolation Forest

Isolation forest merupakan salah satu model algoritma terbaru dalam melakukkan klasifikasi anomaly detection. Algoritma tersebut didasarkan pada fakta bahwa terdapat data yang sedikit dan sangat berbeda dari data yang dominan, dimana berdasarkan hal tersebut dapat dijelaskan bahwa anomali bersifat rentan terhdap mekanisme yang disebut isolasi. Metode ini sangat bermanfaat secara fundamental karena memperkenalkan penggunaan isolasi sebagai cara yang efektif dan efisien dalam mendeteksi anomali. Selain itu, metode ini algoritma dengan kompleksitas waktu linier rendah dan kebutuhan memori yang kecil yang dapat membangun model berkinerja baik dengan menggunakan sub sampel kecil ukuran tetap, terlepas dari ukuran kumpulan data (Yao *et al*., 2019).



Gambar 2.8. Ilustrasi Tree Isolation Forest

Gagasan inti dari algorima isolation forest adalah bahwa jumlah titik yang abnormal biasanya kecil, dan ada perbedaan yang signifikan antara titik normal dan atribut. Algoritma ini memiliki pola dasar pada model decision tree yang dapat mem breakdown proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga proses pengambilan keputusan akan lebih menginterpretasikan solusi dari permasalahan (Zhang, Wang and Zhang, 2019).



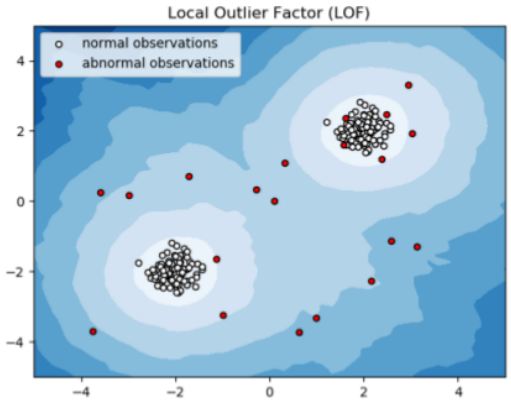
Gambar 2.9. Ilustrasi Algoritma Isolation Forest

Isolation forest merupakan salah satu cara yang efisien untuk melakukan deteksi outlier atau anomaly dalam dataset yang berdimensi tinggi. Algoritma ini melakukan pengamatan dengan memilih fitur secara acak dan selanjutnya memilih nilai split diantara nilai maksimum dan minimum dari fitur yang dipilih (Qin and Lou, 2019; Yao *et al*., 2019).

2.4.2. Algoritma Local Outlier Factor

Local Outlier Factor (LOF) merupakan algoritma pendeteksian data anomali atau outlier yang berbasis pada kepadatan data. Algoritma ini menemukan data outlier dengan melakukan perhitungan dari kelompok data yang menyimpang dari titik data normal yang diberikan. Algoritma ini menjadi salah satu solusi pada pendekatan deteksi anomali pada sekelompok dataset yang sangat imbalance. Metode penghitungan kepadatan data pada algoritma ini dilakukan berdasarkan kepadatan antara masing-masing titik data dengan titik data berikutnya atau tetangganya, dimana semakin rendah kepadatan data pada suatu titik maka akan semakin besar kemungkinan diidentifikasi sebagai outlier (Cheng, Zou and Dong, 2019).

Sebagai algoritma yang populer dalam proses pendeteksian outlier, LOF sangat sering digunakan pada pada klasifikasi aliran data statis yang di dalam aliran data tersebut menunjukkan adanya sekelompok data yang berbeda dari keseluruhan data pada umumnya (Yang *et al*., 2019).



Gambar 2.10. Ilustrasi Algoritma Local Outlier Factor

Berdasarkan gambar 2.10. dapat dijelaskan bahwa algoritma ini melakukan proses pendeteksian pada aliran data atau kumpulan data yang ada terlebih dahulu dengan melakukan penilaian terhadap kepadatan data. Ketika data lebih rendah dibandingkan keseluruhan data secara umum, maka diidentifikasi sebagai outlier.

Secara umum pola-pola pendeteksian dalam algoritma LOF dapat dibagi menjadi 2 macam (John and Naaz, 2019), yaitu :

1. Global outlier, jika objek data yang secara signifikan memiliki jarak yang besar dibandingkan dengan barisan data tetangganya, baik data sebelum ataupun setelahnya.
2. Local outlier, ketika objek data memiliki jarak yang relatif besar dibandingkan pada jarak rata-rata tetangganya.

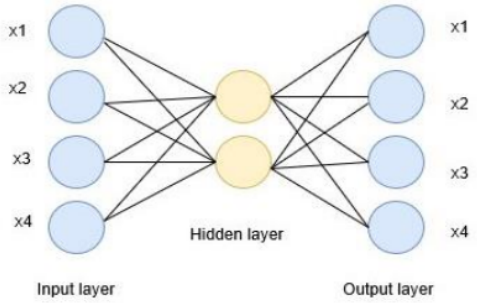
Pada penelitian lainnya, algoritma LOF juga digunakan dalam proses deteksi intrusi pada suatu struktur jaringan. Deteksi dilakukan dengan melacak bentuk barisan data atau sinyal atau sekumpulan data yang memiliki karakteristik yang berbeda secara umum dari keseluruhan data atau sinyal di dalam jaringan tersebut (Kharitonov and Zimmermann, 2019).

Dari keterangan tersebut diatas dapat disimpulkan bahwa LOF merupakan skor atau nilai yang menunjukkan seberapa besar kemungkinan suatu titik data menjadi anomali atau outlier. Atau dengan kata lain, algoritma ini mengkonfirmasi dan mengidentifikasi prilaku data yang tidak diharapkan. Metode ini lebih banyak

diasumsikan dengan menghitung skor poin dari barisan distribusi data dengan probabilitas yang sama.

2.4.3. Autoencoder

Teknik *autoencoder* disebut juga teknik *dimentionality reduction*. Dimana *autoencoder* mampu merepresentasikan data kemudian merekonstruksi nya kembali. Karena tujuan *encoder* untuk kompresi, bentuk terkompresi haruslah memiliki dimensi lebih kecil dari dimensi *input*. *Neural network* mampu melakukan kompresi dengan baik karena *neural network* mampu menemukan *hidden structure* dari data. Berikut arsitektur *autoencoder* dari penelitian Kunang dkk. (2019) pada gambar 2.1



Gambar 2.11 Arsitektur *Autoencoder* Kunang dkk. (2019)

Ukuran *performance measure* untuk *autoencoder* adalah mengukur *loss* Naway dan Li (2019). *Input* matriks X pada *autoencoder*, kemudian ingin *autoencoder* tersebut menghasilkan matriks yang sama, maka *output* harus sama dengan *input*. Perhitungan *autoencoder* didapat dari rumus Kunang dkk. (2019).

$$Y = f(X) = s(WX + b_X) \quad (2.1)$$

$$X' = g(Y) = s(W'Y + b_Y) \quad (2.2)$$

Keterangan :

$Y = f(X)$ = fungsi *encoded*

$X' = g(Y)$ = fungsi *decoded*

s = fungsi aktivasi

W = bobot *encoded*

W' = bobot *decoded*

b_X = bias *encoded* b_Y= bias *decoded*

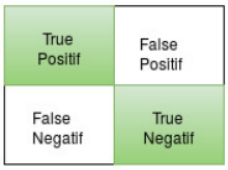
Pada konsep dan metode anomali detection, autoencoder digunakan dengan cara un-supervised learning atau tanpa pembelajaran. Ini dilakukan dengan menarik kesimpulan dengan terlebih dahulu menentukan treshold atas batas titik data yang bisa dikategorikan sebagai anomali.

Bentuk paling sederhana dari autencoder adalah feedforward, jaringan saraf non-berulang sangat mirip dengan banyak perceptron lapisan tunggal yang membuat perceptron multilayer (MLP) - memiliki lapisan input, lapisan output dan satu atau lebih lapisan tersembunyi yang menghubungkan mereka - tetapi dengan layer output memiliki jumlah node yang sama dengan layer input, dan dengan tujuan merekonstruksi inputnya sendiri (alih-alih memprediksi nilai target Y yang diberikan input X). Oleh karena itu, autoencoder adalah model pembelajaran yang tidak diawasi.

Dan dalam proses klasifikasi anomaly detection dengan menggunakan algoritma autoencoder, model yang dipakai adalah model paling sederehana untuk mendapatkan kesimpulan di titik mana barisan data tersebut dapat dikenali sebagai anomali.

2.5. Confusion Matrix

Proses Validasi bertujuan untuk mengetahui apakah simulasi dari sistem klasifikasi *author matching* ini sesuai dengan prediksi yang telah dilakukan. Proses validasi ini akan dilakukan dengan menggunakan *confusion matrix*. Matrix ini menggambarkan kinerja klasifikasi dari data percobaan yang berisi informasi prediksi data. Berikut gambar 2.4 menunjukkan *confusion matrix* untuk klasifikasi dua kelas (biner) (Luque *et al.*, 2019).



Gambar 2.12 *Confusion Matrix*

Dalam proses klasifikasi mempunyai dua label data yaitu positif dan negatif. Klasifikasi dari dua label ini akan menghasilkan empat nilai dari *confusion matrix* yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN). Sistem klasifikasi data *author matching* akan menghasilkan nilai pada *confusion matrix*. TP adalah jumlah data author yang benar (positif) ketika proses prediksi dicocokkan dengan jumah data yang benar (positif) secara aktual, TN adalah jumlah *author* yang tidak sama (negative) pada saat prediksi dibandingkan dengan jumlah yang tidak sama (negative) secara aktual. FP adalah jumlah *author* yang benar namun terdeteksi *terdeteksi sebagai author yang tidak sama*, dan FN adalah jumlah *author yang tidak sama* terdeteksi sebagai author yang sama. Berikut beberapa parameter evaluasi dari hasil *confusion matrix* (Luque *et al.*, 2019).

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.3)$$

Parameter akurasi pada persamaan 2.3 untuk mengukur keakuratan sistem dari proses klasifikasi yaitu dengan menjumlahkan nilai *true positive* dan *true negative* dibagi dengan seluruh data matriks yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN).

$$Presisi = \frac{TP}{TP+FP} \quad (2.4)$$

Parameter presisi pada persamaan 2.4 untuk mengukur tingkat presisi dari sistem klasifikasi yaitu dengan membagi *true positive* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP) .

$$Sensitivitas = \frac{TP}{TP+FN} \quad (2.5)$$

Parameter sensitivitas pada persamaan 2.5 untuk mengukur tingkat sensitivitas dari sistem klasifikasi yaitu dengan membagi *true positive* dengan menjumlahkan *true positive* (TP) dan *false negative* (FN) .

$$Spesifisitas = \frac{TN}{TN+FP} \quad (2.6)$$

Parameter spesifisitas pada persamaan 2.6 untuk mengukur tingkat spesifisitas dari sistem klasifikasi yaitu dengan membagi *true negative* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP) .

$$F1\ Score = \frac{2 \times Presisi \times Sensitvitas}{Presisi+Sensitivitas} \quad (2.7)$$

Parameter F1 Score pada persamaan 2.7 untuk mengukur perbandingan rata-rata presisi dan sensitivitas dari sistem klasifikasi yaitu dengan membagi *true negative* dengan menjumlahkan *true negative* (TN) dan *false positive* (FP).

2.6. Kurva ROC

Dalam banyak aplikasi, kurva karakteristik operator penerima (ROC) digunakan untuk menunjukkan bagaimana prediktor membandingkan dengan hasil yang sebenarnya. Salah satu keuntungan besar dari analisis ROC adalah bahwa ia adalah ambang batas agnostik; kinerja alat prediksi diperkirakan tanpa ambang batas tertentu dan juga memberikan kriteria untuk memilih ambang batas optimal berdasarkan fungsi atau tujuan biaya tertentu. Biasanya, analisis ROC menunjukkan bagaimana sensitivitas (tingkat positif sejati) berubah dengan spesifisitas yang bervariasi (tingkat negatif sejati atau 1 - tingkat positif palsu) untuk ambang yang berbeda. Analisis juga biasanya menimbang positif palsu dan negatif palsu sama. Dalam analisis ROC, kemampuan prediktif suatu variabel biasanya dirangkum oleh area di bawah kurva (AUC), yang dapat ditemukan dengan mengintegrasikan area di bawah segmen garis. (Muschelli, 2019)

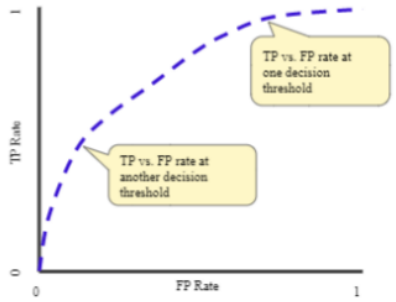
2.6.1 Kurva ROC

Kurva ROC (kurva karakteristik operasi penerima) adalah grafik yang menunjukkan kinerja model klasifikasi di semua ambang klasifikasi. Kurva ini memplot dua parameter:

- True Positive Rate

- False Positive Rate

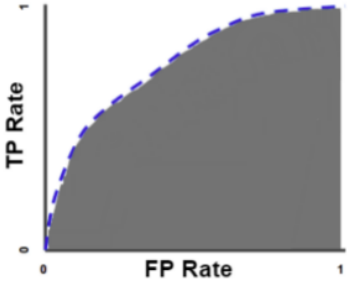
Kurva ROC memplot TPR vs FPR pada threshold klasifikasi yang berbeda. Menurunkan threshold klasifikasi mengklasifikasikan lebih banyak item sebagai positif, sehingga meningkatkan False Positive dan True Positive. Gambar berikut menunjukkan kurva ROC yang khas.



Gambar 2.13 Kurva False Positive Rate

2.6.2 AUC: Area Di Bawah Kurva ROC

AUC singkatan dari "Area di bawah Kurva ROC." Artinya, AUC mengukur seluruh area dua dimensi di bawah seluruh kurva ROC dari (0,0) hingga (1,1).

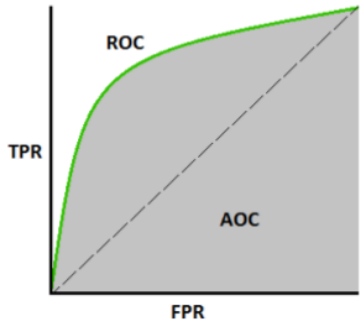


Gambar 2.14. Kurva AUC

AUC menyediakan ukuran kinerja agregat di semua threshold klasifikasi yang memungkinkan. Salah satu cara menafsirkan AUC adalah sebagai probabilitas bahwa model tersebut memeringkat contoh positif acak lebih tinggi daripada contoh negatif acak.

Kurva AUC - ROC adalah pengukuran kinerja untuk masalah klasifikasi di berbagai pengaturan thresholds. ROC adalah kurva probabilitas dan AUC mewakili derajat atau ukuran keterpisahan. Ini memberitahukan berapa banyak model yang mampu membedakan antar kelas. Semakin tinggi AUC, semakin baik modelnya dalam memprediksi 0s sebagai 0s dan 1s sebagai 1s. Dengan analogi, semakin tinggi AUC, semakin baik modelnya membedakan antara pasien dengan penyakit dan tanpa penyakit.

Kurva ROC diplot dengan TPR terhadap FPR di mana TPR berada pada sumbu y dan FPR pada sumbu x.



Gambar 2.15 Kurva ROC

2.6.3 Mendefinisikan istilah yang digunakan dalam Kurva AUC dan ROC

TPR (True Positive Rate) / Recall /Sensitivity $= \frac{TP}{TP+FN}$

Specificity $= \frac{TN}{TN+FP}$

FPR $= 1 - Specificity$

$= \frac{FP}{TN+FP}$

2.6.4 Bagaimana cara berspekulasi kinerja model?

Model yang sangat baik memiliki AUC dekat dengan 1 yang berarti memiliki ukuran pemisahan yang baik. Model yang buruk memiliki AUC mendekati 0 yang berarti memiliki ukuran pemisahan terburuk. Contohnya memprediksi 0s sebagai 1s dan 1s sebagai 0s. Dan ketika AUC adalah 0,5, itu berarti model tidak memiliki kapasitas pemisahan kelas sama sekali. Ketika dua kurva tidak *overlap* sama sekali model berarti memiliki ukuran pemisahan yang ideal. Bearti benar-benar mampu

membedakan antara kelas positif dan kelas negatif. Ketika AUC sekitar 0, model sebenarnya membalas kelas. Artinya, model memprediksi kelas negatif sebagai kelas positif dan sebaliknya.

2.6.5 Hubungan antara Sensitivitas, Spesifisitas, FPR, dan Threshold

Sensitivitas dan Spesifisitas berbanding terbalik satu sama lain. Jadi ketika kita meningkatkan Sensitivitas, kekhususan berkurang dan sebaliknya.

Sensitivity↑, Specificity↓ and Sensitivity↓, Specificity↑

Ketika menurunkan threshold, akan mendapatkan nilai lebih positif sehingga meningkatkan sensitivitas dan mengurangi spesifisitas. Demikian pula, ketika meningkatkan threshold, akan mendapatkan lebih banyak nilai negatif sehingga mendapatkan spesifisitas yang lebih tinggi dan sensitivitas yang lebih rendah. Seperti diketahui FPR adalah 1 - spesifisitas. Jadi ketika meningkatkan TPR, FPR juga meningkat dan sebaliknya.

TPR↑, FPR↑ and TPR↓, FPR↓

2.6.6 Bagaimana cara menggunakan kurva AUC ROC untuk model multi-kelas?

Dalam model multi-kelas, dapat memplot angka N Kurva ROC AUC untuk kelas nomor N menggunakan metodologi One vs ALL. Jadi sebagai contoh, jika memiliki tiga kelas bernama X, Y dan Z, Disana akan memiliki satu ROC untuk X diklasifikasikan terhadap Y dan Z, ROC lain untuk Y diklasifikasikan ke X dan Z, dan yang ketiga dari Z diklasifikasikan ke Y dan X.

BAB III
METODOLOGI PENELITIAN

Bab ini berisi mengenai kerangka kerja dari penelitian dan analisis kebutuhan perangkat lunak. Metodologi yang digunakan pada bab ini adalah berfokus pada klasifikasi *author matching* dengan menggunakan pendekatan anomali *detection*. Metodologi dalam penelitian ini adalah penelusuran pustaka, persiapan data, pra-pengolahan dengan *pairless combination*, proses pengklasifikasian dengan menggunakan algoritma yang dipakai dalam pendekatan *anomaly detection* yaitu *IsolationForest*, *Local Outlier Factor (LOF)* dan dengan menggunakan algoritma *AutoEncoder*. Tahapan selanjutnya adalah analisis hasil dan penarikan kesimpulan. Tahapan dalam penyelesaian penelitian Tesis ini adalah bagaimana pengklasifikasian *author matching* dengan menggunakan pendekatan *anomaly detection* dalam 3 algoritma yang telah disebutkan diatas.

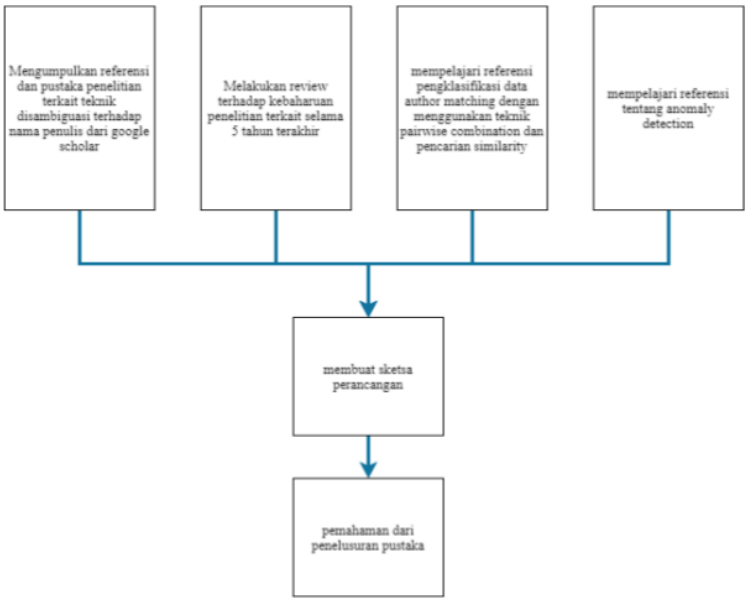
3.1 Kerangka Kerja Penelitian

Secara garis besar, adapun langkah-langkah pada metodologi penelitian yang digunakan untuk membantu dalam penyusunan tesis dan hasil dari penelitian ini, memerlukan susunan kerangka kerja yang jelas tahapan-tahapannya sehingga mampu menghasilkan prediksi dari proses klasifikasi yang terukur dan akurat. Kerangka kerja penelitian yang digunakan seperti pada Gambar 3.1, dimulai dari kegiatan tinjauan pustaka untuk mengumpulkan referensi, proses persiapan data dimulai dengan pemilihan dataset yang sesuai, pra-pengolahan data dengan *converting and deletion data*, *concatenation, remove stopword* dan *pairwise combination,* dan *year difference*. Dilanjutkan dengan proses ekstraksi fitur dengan teknik pencarian kemiripan *(similarity)* dan perbandingan *(comparation)*. Kemudian dilakukan proses pengklasifikasian dengan Algoritma *Isolation Forest*, *Local Outlier Factor (LOF)* dan *AutoEncoder*. Meudian dilanjutkan dengan analisis hasil dari proses klasifikasi dan terakhir dilakukan penarikan kesimpulan atas penelitian ini.

Gambar 3.1. Kerangka Kerja Penelitian

3.2 Penelusuran Pustaka

Tahap ini dilakukan pencarian dan pembelajaran dari beberapa pustaka dan literatur yang relevan terkait landasan-landasan teori yang diperoleh dari berbagai publikasi ilmiah dengan penelitian yang akan dikerjakan. Tahapan dalam pembelajaran pustaka dapat dilihat pada Gambar 3.2.



Gambar 3.2 Tahapan Penelusuran Pustaka

3.3 Persiapan Data

Dalam tahapan ini, dilakukan survei terhadap ketersediaan dataset yang berisi data author lengkap (beserta ide, judul penelitian, tahun terbit dan lain-lain). Dari beberapa referensi yang digunakan, terdapat 2 kelompok data (dataset) yang dapat digunakan untuk melakukan klasifikasi author matching yaitu dengan menggunakan dataset *vietnamnese author* (Tran, Huynh and Do, 2014) dan *SCAD-zBMATH dataset* (Müller, Reitz and Roy, 2017). Dari penelusuran terhadap kedua dataset tersebut, dataset SCAD-zBMATH dapat dikategorikan sebagai dataset yang lengkap dan paling homogen untuk dilakukan proses klasifikasi terhadap nama author. Dataset ini menyediakan beberapa file publikasi lengkap dengan format yang berbeda-beda pada setiap filenya dimana masing-masing file dapat menjadi level pengujian dari algoritma klasifikasi yang akan digunakan. Selain itu, dataset ini memenuhi syarat struktur data dalam proses disambiguasi nama penulis (Müller, Reitz and Roy, 2017) yaitu :

1. Kasus dimana satu nama penulis muncul di barisan berikutnya dengan perbedaan yang cukup signifikan (misalnya dengan menyingkat nama depan atau nama tengah)
2. kasus di mana nama penulis yang sebenarnya ditulis dalam alfabet non-barat (mis., Asia atau Sirilik) dan muncul di *header* publikasi dalam beberapa versi, dapat memunculkan contoh kasus 1.
3. kasus-kasus publikasi oleh penulis yang kurang produktif atau penulis dengan hanya sedikit kolaborator, di mana informasi penulis bersama yang tidak tersedia, dan
4. kasus-kasus publikasi dari bidang ilmiah atau komunitas yang umumnya cenderung memiliki jumlah penulis bersama yang lebih sedikit.

Dari penjelasan diatas dapat disimpulkan bahwa dataset yang dapat memenuhi syarat untuk dilakukan proses disambiaguasi adalah dataset yang memiliki data lengkap mulai dari author yang sangat heterogen hingga author yang jarang sekali muncul dalam barisan data, hingga data judul dan author pendamping yang juga bisa muncul sebagai author utama di judul publikasi yang berbeda.

Berdasarkan hal tersebut, maka dataset yang digunakan adalah data nama-nama penulis lengkap dengan judul jurnal serta tahun terbit yang diambil dari dataset SCAD-zBMATH dengan judul featured-dataset-merged yang terdiri dari 10.160 baris data publikasi (Müller, Reitz and Roy, 2017). Dataset ditunjukkan pada Tabel 3.1.

Tabel 3.1

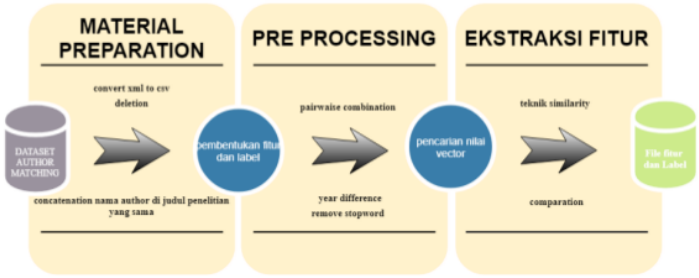
Spesifikasi Dataset

Dataset	Jumlah Data
SCAD-zBMATH	10.610

3.4 Pengolahan Data *Author Matching*

Pada tahapan ini dilakukan proses pengolahan data untuk mempersiapkan data sebelum dimasukkan kedalam proses klasifikasi. Pada penelitian ini menggunakan klasifikasi biner untuk pencocokan nama *author*. Jumlah data yang

digunakan berjumlah 11.924 baris data. Tahapan dari proses pengolahan data pada penelitian ini dapat dilihat pada gambar 3.3 berikut ini.



Gambar 3.3 Alur Pengolahan Data *Author Matching*

Pada gambar 3.3 diatas menggambarkan langkah-langkah dalam melakukan pengolahan data dalam penelitian ini sebagai berikut:

3.4.1. Mempersiapkan Data (Material Preparation)

Tahapan yang dilakukan untuk menyiapkan data *author matching* sebelum masuk ke proses lainnya. Pada gambar 3.4 dibawah ini menunjukkan bentuk data author matching awal atau *RAW* data dari dataset *SCAD-ZbMath* dengan judul file *featured-dataset-merged* yang masih dalam bentuk xml.

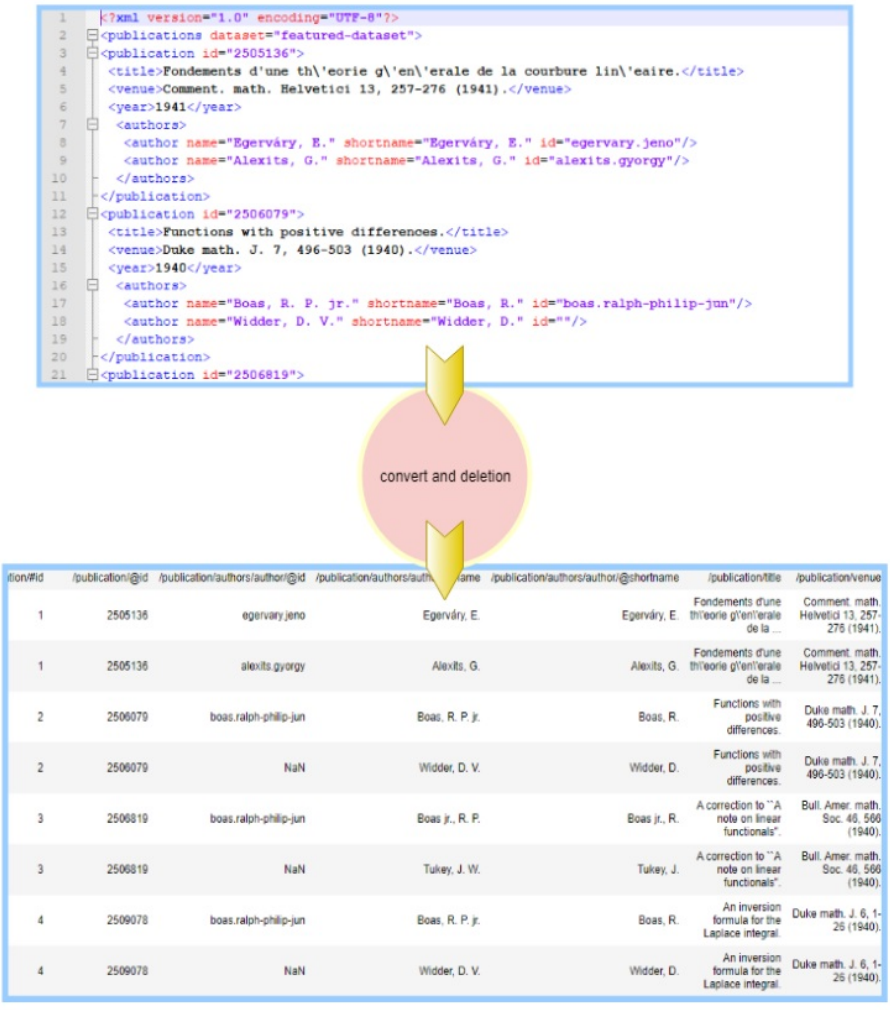
```
<?xml version="1.0" encoding="UTF-8"?>
<publications dataset="featured-dataset">
<publication id="2505136">
 <title>Fondements d'une th\'eorie g\'en\'erale de la courbure lin\'eaire.</title>
 <venue>Comment. math. Helvetici 13, 257-276 (1941).</venue>
 <year>1941</year>
 <authors>
  <author name="Egerváry, E." shortname="Egerváry, E." id="egervary.jeno"/>
  <author name="Alexits, G." shortname="Alexits, G." id="alexits.gyorgy"/>
 </authors>
</publication>
<publication id="2506079">
 <title>Functions with positive differences.</title>
 <venue>Duke math. J. 7, 496-503 (1940).</venue>
 <year>1940</year>
 <authors>
  <author name="Boas, R. P. jr." shortname="Boas, R." id="boas.ralph-philip-jun"/>
  <author name="Widder, D. V." shortname="Widder, D." id=""/>
 </authors>
</publication>
<publication id="2506819">
```

Gambar 3.4 Dataset SCAD-zBMATH.xml

Bentuk dari dataset diatas tidak dapat langsung masuk ke dalam proses selanjutnya baik itu pra pengolahan, ekstrasksi fitur maupun langsung masuk ke dalam proses klasifikasi. Oleh karena itu, diperlukan langkah selanjutnya dalam tahapan persiapan data atau *material preparation* dengan tujuan terbentuknya struktur data yang dapat diolah pada proses selanjutnya.

3.4.1.1.Converting Dataset dan Deletion Dataset

Proses convert dilakukan untuk mengubah ekstensi file dari dataset yang semula dalam bentuk xml menjadi bentuk csv dengan tujuan membentuk file menjadi kolom-kolom atribut sehingga bisa diterjemahkan dengan jelas untuk proses pra pemrosesan data selanjutnya. Proses dan hasil dari proses convert tersebut dapat dilihat pada Gambar 3.5 dan tabel 3.2.



tion/#id	/publication/@id	/publication/authors/author/@id	/publication/authors/auth...ame	/publication/authors/author/@shortname	/publication/title	/publication/venue
1	2505136	egervary.jeno	Egerváry, E.	Egerváry, E.	Fondements d'une th\'eorie g\'en\'erale de la ...	Comment. math. Helvetici 13, 257-276 (1941).
1	2505136	alexits.gyorgy	Alexits, G.	Alexits, G.	Fondements d'une th\'eorie g\'en\'erale de la ...	Comment. math. Helvetici 13, 257-276 (1941).
2	2506079	boas.ralph-philip-jun	Boas, R. P. jr.	Boas, R.	Functions with positive differences.	Duke math. J. 7, 496-503 (1940).
2	2506079	NaN	Widder, D. V.	Widder, D.	Functions with positive differences.	Duke math. J. 7, 496-503 (1940).
3	2506819	boas.ralph-philip-jun	Boas jr., R. P.	Boas jr., R.	A correction to "A note on linear functionals".	Bull. Amer. math. Soc. 46, 566 (1940).
3	2506819	NaN	Tukey, J. W.	Tukey, J.	A correction to "A note on linear functionals".	Bull. Amer. math. Soc. 46, 566 (1940).
4	2509078	boas.ralph-philip-jun	Boas, R. P. jr.	Boas, R.	An inversion formula for the Laplace integral.	Duke math. J. 6, 1-26 (1940).
4	2509078	NaN	Widder, D. V.	Widder, D.	An inversion formula for the Laplace integral.	Duke math. J. 6, 1-26 (1940).

Gambar 3.5 Proses dan Hasil Converting Dataset

Tabel 3.2

Hasil Convert dataset dari xml ke csv

id_author	name	shortname	tittle	venue
Egevary.jeno	Egervary.E	Egervary.E.	Fondements d'une	Comment. Math. Helvetici 13, 257-276 (1941)
Alexits.gyorgy	Alexits, G	Alexits, G	Fondements d'une	Comment. Math. Helvetici 13, 257-276 (1941)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, R	Function with positive differences	Duke math. J.7, 496-503 (1940)
NaN	Widder, D.V.	Widder, D.	Function with positive differences	Duke math. J.7, 496-503 (1940)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, jr. R.	A corrections to "A note on linear	Bull Amer. Math Soc. 46. 566 (1940)
NaN	Tukey, J.W.	Tukey J	A corrections to "A note on linear	Bull Amer. Math Soc. 46. 566 (1940)
Boas.ralph-philip-jun	Boas, R.P.jr	Boas, R	An inversion formula for the Laplace integral	Duke math. J. 6 1-26 (1940)

Proses diatas menghasilkan dataset sejumlah 11.924 baris dengan 10 kolom (*dataset, publication_id, publication_id2, author_id, author_name, author_shortname, publication_title, publication_venue, publication_year, dan publication_year2*). Setelah itu, dilakukan penghapusan (*deletion*) kolom yang tidak akan memberikan informasi dalam proses klasifikasi. Hasil dari penghapusan kolom tersebut menyisakan 6 kolom yaitu *id_author, name, authors, title, venue dan year*.

3.4.1.2.Penggabungan Data (Concatenate)

Proses selanjutnya setelah dilakukan *convert* adalah dengan melakukan penggabungan baris atau kolom dengan tujuan mengurutkan data publikasi menjadi 1 baris pada 1 judul publikasi. Hanya 1 kolom yang akan dilakukan penggabungan yaitu kolom authors sehingga setiap baris akan menampilkan 1 judul publikasi

dengan jumlah author yang bisa lebih dari 1dengan tujuan agar proses klasifikasi dapat menghasilkan hasil yang lebih baik (tabel 3.3).

Tabel 3.3

Hasil Penggabungan Data atau Concatenate

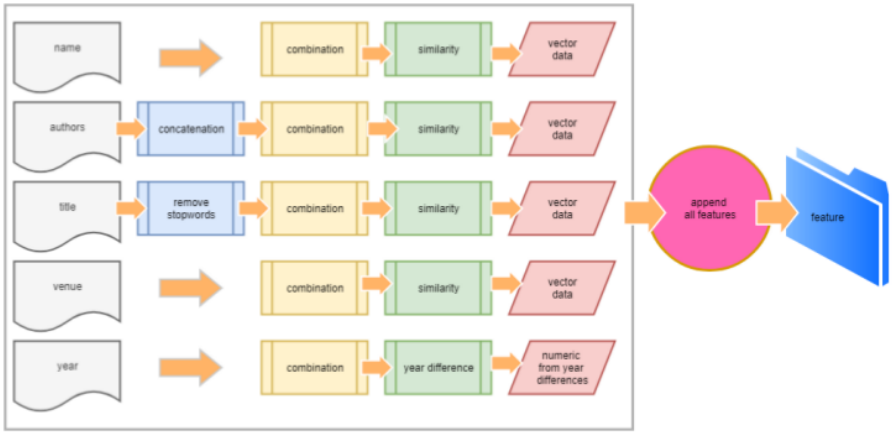
Id	Name	Authors	Title	Venue	Year
Egervary.jeno	Egervary.E	Egervary.D.Alexits,G	Fondements d'une th	Comment math. Helvetici 13, 257-276 (1941).	1941
alexits.gyorgy	Alexits,G	Egervary E.Alexits,G	Fondements d'une th	Comment math. Helvetici 13, 257-276 (1941).	1941
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Widder,D.V	Functions with positive differences	Duke math J.7, 496-503 (1940)	1940
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Tukey,D.V	A corrections to "A note on linear functionals	Bull Amer. math. Soc 46, 566 (1940)	1940
boas.ralph-philip-jun	Boas, R.P.jr	Boas,R.P.jr.Widder,D.V	An inversion formula for the laplace integral	Duke math, J.6, 1-26 (1940)	1940

Hasil dari proses *concatenate* tersebut dapat dijadikan sebagai bentuk dataset baru yang memberikan informasi kuat dan terstruktur siap untuk diolah ke proses berikutnya dalam pengolahan data *author matching*. Proses ini menghasilkan jumlah baris data yang baru yaitu 10.160 baris data.

3.4.2. Pra Pemrosesan (Pre Processing)

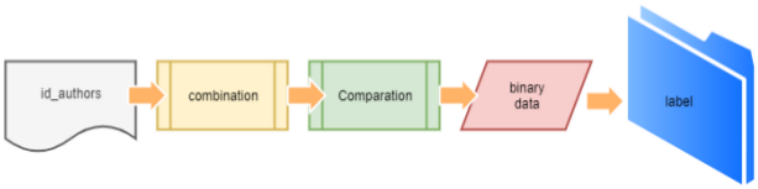
Pada tahap ini, dilakukan proses untuk membentuk data yang terstruktur dengan baik sehingga dapat memberikan informasi dan mudah dipelajari dalam proses klasifikasi. Tahap ini dilakukan dengan tujuan membentuk dataset menjadi terpisah 2 file yang masing-masing akan difungsikan sebagai fitur dan label. Proses pemisahan file tersebut dilakukan agar pendefisian label sebagai target pembelajaran *(learning)* menjadi lebih mudah.

Untuk fitur, kolom yang akan digunakan dan dilakukan pra pemrosesan adalah name, authors, title, venue dan year (gambar 3.5) sedangkan untuk label hanya menggunakan kolom ide_authors (gambar3.6). Setiap kolom akan melakukan tahapan pra pemrosesan yang sama demi menghasilkan informasi yang bernilai dalam tahap klasifikasi.



Gambar 3.5 Tahapan Pra Pemrosesan Fitur

Berdasarkan gambar diatas dapat disimpulkan bahwa untuk menemukan informasi dalam data author matching pada fitur data, dapat dilakukan dengan kombinasi yang tujuannya adalah memasangkan setiap baris pada masing-masing kolom.



Gambar 3.6 Tahapan Pra Pemrosesan Label

Dan untuk pra pemrosesan data label juga dilakukan dengan kombinasi untuk memberikan pasangan (*pairwise)* pada setiap baris pada kolom ide_author.

3.4.2.1.Pairwise Combination

Proses pengolahan dataset yang telah dihasilkan diatas dilanjutkan dengan melakukan pemasangan atau kombinasi agar dapat dilakukan proses komparasi antara baris pertama dengan baris kedua, baris pertama dengan baris ketiga dan seterusnya. Hasil kombinasi pada dataset yang digunakan yang memiliki 10.160 baris data akan menghasilkan jumlah baris data yang baru sebanyak 51.607.720

baris data pada setiap kolom baik pada fitur maupun label. Dalam rumus matematika, jumlah kombinasi didapatkan dengan rumus :

$$\frac{n!}{r!(n-r)!} = \binom{n}{r} \quad (3.1)$$

Dimana **n** adalah jumlah objek yang bisa dipilih dan **r** adalah jumlah yang harus dipilih. Dan hasil dari dari proses pairwise dengan menggunakan teknik kombinasi tersebut dapat dilihat pada tabel 3.4.

Tabel 3.4

Hasil Proses Pairwise Combination

0	1
Egervary, E.	Alexits, G.
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Boas, R.P. jr
Egervary, E.	Maller, R
Egervary, E.	Izumi, S

3.4.2.2.Remove Stopwords

Remove stopword merupakan proses yang dilakukan untuk mengurangi kata-kata yang tidak bernilai informasi dalam suatu data teks. Dalam proses ini, remove stopword hanya dilakukan pada kolom atau fitur title dengan maksud menghilangkan kata-kata yang dinilai tidak bermanfaat pada dataset agar tidak mengurangi informasi pada proses klasifikasi berikutnya.

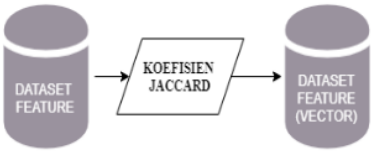
3.4.2.3. Nilai Selisih Tahun Terbit (*Year Difference*)

Proses ini dilakukan pada kolom year yang telah dikombinasikan. Teknik yang dilakukan adalah mencari perbedaan atau selisih dari dua data tersebut. Tujuan dilakukan hal tersebut adalah mencari jarak dari setiap tahun terbitnya publikasi, dengan asumsi jika pada nama author terdapat kesamaan namun jarak tahun terbit publikasi sangat jauh, maka dapat disimpulkan author tersebut berbeda.

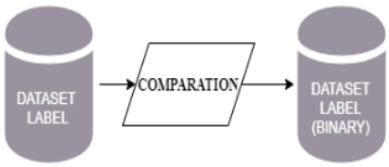
$$y = y1 - y2 \quad (3.2)$$

3.4.3. Ekstraksi Fitur

Ekstraksi fitur merupakan proses pengambilan ciri sebuah objek yang dapat menggambarkan karakteristik dari objek tersebut. Dalam tahapan ini, proses ekstraksi fitur dilakukan dengan menggunakan dua cara yang berbeda. Untuk fitur, proses pengambilan ciri dilakukan dengan teknik similarity menggunakan koefisien jaccard (gambar 3.7), sedangkan pada label dilakukan dengan menggunakan comparation antara data id_author yang telah dipasangkan atau dikombinasikan (gambar 3.8).



Gambar 3.7 Tahapan Estraksi untuk Feature



Gambar 3.8 Tahapan Ekstraksi untuk Label

3.4.3.1. Nilai Kemiripan (*Similarity*)

Proses similarity adalah tahapan menghitung jarak antara dua data yang telah dikombinasikan dengan tujuan menemukan kemiripan data. Teknik yang digunakan untuk menghitung jarak antar data yang telah dipasangkan dengan menggunakan Koefisien Jaccard. Pada persamaan 3.2 proses perhitungan yang dilakukan dalam proses koefisien Jaccard berikut ini.

$$\boldsymbol{jaccard}\ (\boldsymbol{a}, \boldsymbol{b}) = \frac{|\boldsymbol{a} \cap \boldsymbol{b}|}{|\boldsymbol{a} \cup \boldsymbol{b}|} = \frac{|\boldsymbol{a} \cap \boldsymbol{b}|}{|\boldsymbol{a}| + |\boldsymbol{b}| - |\boldsymbol{a} \cap \boldsymbol{b}|} \quad (3.3)$$

Pada tahapan ini, koefisien jaccard dilakukan pada kolom-kolom yang akan digunakan sebagai feature yaitu name, authors, title, venue, lalu dilakukan penggabungan fitur year yanng telah dicari selisihnya pada proses sebelumnya. Hasil dari penggunaan koefisien dan penggabungan tersebut dapat dilihat pada tabel 3.5.

Tabel 3.5

Hasil Proses Similarity menggunakan Koefisien Jaccard

	name	authors	title	venue	year
0	0.33	1	1	1	0
1	0.21	0.14	0.35	0.5	1
2	0.28	0.26	0.24	0.56	1
3	0.21	0.14	0.25	0.54	1
4	0.28	0.25	0.26	0.47	2
5	0.21	0.15	0.38	0.53	3
6	0.21	0.15	0.23	0.53	3
7	0.21	0.15	0.23	0.48	3
8	0.21	0.15	0.38	0.5	3
9	0.41	0.4	0.33	0.5	3
.	.	.	.	.	.
.	.	.	.	.	.
51607720	0.27	0.35	0.26	0.44	7

3.4.3.2.Nilai Pembandingan (*Comparation*)

Tahapan komparasi atau perbandingan dilakukan untuk menemukan informasi pada kolom *id_author* yang akan digunakan sebagai contoh output dari semua input yang diajarkan atau biasa disebut label. Hasil dari kombinasi adalah sebuah angka binary yang prosesnya dilanjutkan dengan membandingkan satu sama lain, jika sama akan diberi nilai 1 dan jika tidak sama akan diberi nilai 0.

$$A \rightarrow \text{'0'} \; if \; \text{id1} \neq \text{id2} \tag{3.4}$$

$$A \rightarrow \text{'1'} \; if \; \text{id1} = \text{id2} \tag{3.5}$$

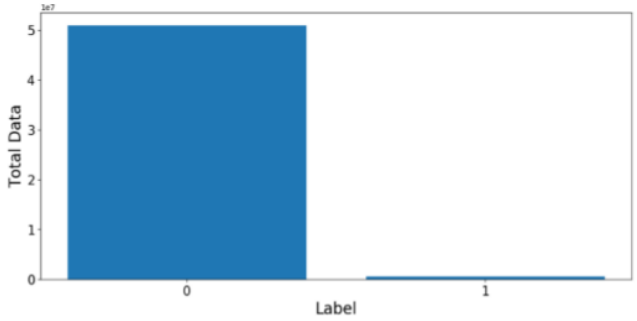
Hasil dari perbandingan tersebut menjadi satu buah tabel yang akan digunakan sebagai label pada algoritma klasifikasi atau pada tahapan berikutnya dan hasil perbandingan tersebut dapat dilihat pada tabel 3.6.

Tabel 3.6

Hasil Proses Comparation

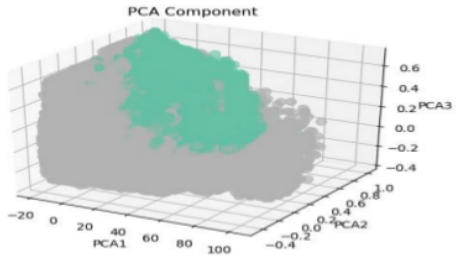
	id_authors
0	0
1	0
2	0
3	0
4	0
5	0
6	0
.	.
.	.
51607720	0

Hasil dari label inilah yang akan digunakan sebagai prediksi dalam proses klasifikasi data author matching. Nilai 0 didefinisikan sebagai nilai positive (data yang tidak sama) atau dengan kata lain author yang berbeda dan 1 sebagai data negative (data yang sama) atau bisa disimpulkan merupakan author yang sama. Dari hasil kombinasi dan dilanjutkan pada proses comparation terhadap label ini, maka terjadilah ketidakseimbangan data (*imbalance data*) yang sangat tinggi antara dimana data positive berjumlah 583.942 dan data negative sejumlah 51.023.778, atau sejumlah 1,1 % berbanding 98,9 % (gambar 3.8).



Gambar 3.9 Plot Imbalance Data Hasil Pra Pemrosesan Author Matching

Pada pengamatan diatas dapat digambarkan terjadinya imbalanced data yang sangat ekstream sehingga proses klasifikasi harus dilakukan dengan menggunakan metode yang mampu mengenali feature dan label dengan sangat baik. Gambaran sebaran data pada hasil pra pemrosesan maupun ekstraksi fitur dapat dilihat pada gambar 3.9.



Gambar 3.10 Plot Sebaran Imbalance Data Hasil Pra Pemrosesan Author Matching

3.5 Klasifikasi menggunakan Menggunakan Pendekatan Anomaly Detection

Dengan merujuk pada hasil dari dari tahapan pre processing, dimana data menjadi sangat imbalance atau berdimensi tinggi, maka proses klasifikasi yang akan dilakukan dengan menggunakan data tersebut adalah dengan melakukan pendekatan anomaly detection. Anomaly detection digunakan karena proses yang dilakukan dalam algoritma nya memiliki kemiripan dengan bentuk dari dasil pre processing yang dilakukan dalam penelitian ini.

Anomaly detection merupakan sebuah teknik yang digunakan untuk mengidentifikasi pola-pola yang tidak biasa atau tidak sesuai perilaku yang diharapkan, dan hal tersebut lebih sering dikenal dengan outlier. Dengan didasarkan hal tersebut pula pendekatan ini dipakai dalam proses klasifikasi data author matching ini.

Adapun teknik anomaly detection memiliki beberapa algoritma yang sering digunakan untuk melakukan proses klasifikasi data yaitu Isolayion Forest, Local Outlier Factor (LOF) dan dengan menggunakan Auto Encoder. Masing-masing

algoritma tersebut memiliki karakter dan pola yang berbeda dan akan menghasilkan angka prediksi dan kesimpulan yang berbeda pula. Adapun proses dan tahapan klasifikasi yang dilakukan dalam penelitian ini digambarkan pada proses flowchart Klasifikasi anomaly detection (Gambar 3.11).

Gambar 3.11 Flowchart Klasifikasi Anomaly Detection

3.5.1 Klasifikasi menggunakan Menggunakan Isolation Forest

Tahapan klasifikasi dengan menggunakan algoritma isolation forest dilakukan dengan gagasan inti yaitu dengan menemukan jumlah titik yang abnormal yang biasanya kecil atau bisa dianggap minority, dan ada perbedaan yang signifikan antara titik normal dan atribut. Perbedaan signifikan tersebut dalam data author matching digambarkan dengan nilai vector antara data feature dan label yang memiliki jarak atau nilai yang sangat berbeda. Dalam proses tersebut, pengamatan terhdap semua data diberi skor dan ditandai sebagai anomali bila memenuhi beberapa kriteria sebagai berikut :

- Skor dekat dengan 1 terindikasi atau dimasukkan dalam kategori anomaly.
- Skor lebih kecil dari 0.5 terindikasi sebagai data normal .
- Jika seluruh skor dekat 0.5 kemudian seluruh sampel tidak terlihat memiliki baris anomali yang teratur, maka data yang bernilai 1 lah yang akan dianggap sebagai data anomali.

Seperti metode *outlier detection* atau *anomaly detection* lainnya, skor anomali diperlukan untuk pengambilan keputusan dan hasil dari pengamatan skor anomali tersebut dinyatakan dalam satu persamaan . Persamaan 3.2 untuk algoritma Isolation Forest, didefinisikan sebagai (Yao *et al*., 2019) **:**

$$\boldsymbol{s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}} \qquad (3.2)$$

Dimana dijelaskan bahwa :
h(x) merupakan panjang lintasan data dari pengamatan *x,*
c(n) merupakan panjang lintasan pencarian yang gagal di pohon pencarian biner
n merupakan jumlah dari node (simpul) eksternal

3.5.2 Klasifikasi menggunakan Menggunakan Local Outlier Factor (LOF)

Algoritma selanjutnya yang digunakan dalam pendekatan anomaly detection adalah local aoulier factor (LOF). Karakteristik algoritma ini adalah dengan menghitung deviasi atau penyimpangan data berdasarkan kepadatan data tetangganya (data berikutnya) di dalam barisan atau lajur data yang ada. Algoritma ini menemukan data outlier dengan melakukan perhitungan dari kelompok data yang menyimpang dari titik data normal yang diberikan. Algoritma ini menjadi salah satu solusi pada pendekatan deteksi anomali pada sekelompok dataset yang sangat imbalance. Metode penghitungan kepadatan data pada algoritma ini dilakukan berdasarkan kepadatan antara masing-masing titik data dengan titik data berikutnya atau tetangganya, dimana semakin rendah kepadatan data pada suatu titik maka akan semakin besar kemungkinan diidentifikasi sebagai outlier.

Ada 2 kondisi dari data author matching ini yang akan dikategorikan sebagai outlier dengan menggunakan algoritma Local Outlier Factor (LOF), yaitu sebagai berikut :

- Jumlah tetangga yang dipertimbangkan, (parameter n_neighbors) biasanya dipilih 1 lebih besar dari jumlah minimum objek yang harus dikandung sebuah cluster, sehingga objek lain dapat berupa outlier lokal relatif terhadap cluster ini.
- lebih kecil dari jumlah maksimum dekat dengan objek yang berpotensi menjadi outlier lokal.

namun, kondisi kedua sangat jarang ditemukan, karena informasi nilai data yang lebih kecil dari jumlah maksimum akan sangat sulit diidentifikasi.

Secara matematis, urutan untuk pendefinisian outlier atau bukan di dalam algoritma LOF ditentukan dengan sepanjang apa data tetangga yang akan dipertimbangkan untuk menjadi cluster data (parameter n), kemudian dilakukan dengan mencari nilai maksimum dari jarang jangkauan atau *Reachbility Distance (RD)* pada cluster tersebut, lalu mencari nilai rata-rata dari jarak jangkaun tersebut atau *Average Reachibility Distance (ARD)* dan hasilnya dari rata-rata tersebut didefinisikan sebagai *Local Reachibility Distance (LRD) n,* setelah semua nilai didapat baru akan dijumlahkan dari seluruh *LRD n* dan dibagi dengan *LRD a* (jika yang dicari adalah nilai a) untuk mendapatkan nilai dari *Local Outlier Factor (LOF)* dari *n* tersebut. Hal tersebut dapat digambarkan dengan urutan persamaan dibawah ini.

- *n* adalah jumlah tetangga atau baris data yang ditentukan untuk titik

- Menentukan nilai maksimum dari *RD (Reachbility Distance)* pada poin *a*

$$RD\ a = \max(n\,\mathrm{dist_a}, dist\ (a,n)) \tag{3.3}$$

- Lalu, menentukan nilai rata-rata dari *Local Reachibility Distance (LRD)*

$$LRD\ a = Average\ (RD\ a + \cdots . + RD\ n) \tag{3.4}$$

dan selanjutnya melakukan pencarian nilai sampai $LRD\ ke\ n$

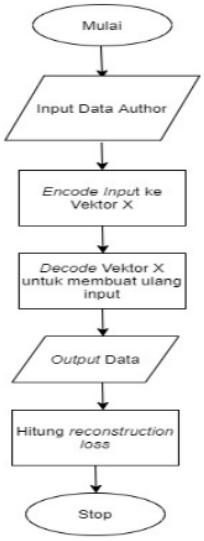
- Terakhir kita dapat menentukan nilai LOF dari *a* dengan cara mencari nilai rata-rata dari panjang *LRD* dalam 1 cluster dibagi dengan nilai *LRD* yang dicari (dalam hal ini *LRD a*)

$$LOF\ a = \frac{average\ (LRD\ b + \cdots + LRD\ n)}{LRD\ a} \quad (3.5)$$

Setelah semua nilai LOF didapatkan, naru akan didefinisikan nilai berapa yang akan dikategorikan sebagai outlier. Dalam hal data author matching ini, nilai yang akan dikategorikan sebagai outlier bernilai antara 0,9 s.d. 1.

3.5.3 Klasifikasi menggunakan Menggunakan AutoEncoder

Algoritma selanjutnya adalah dengan menggunakan auto encoder dengan tujuan merekonstruksi data terlebih dahulu untuk mendapatkan value data yang lebih sebelum masuk dalam proses klasifikasi. Proses *autoencoder* diawali dengan *input* data *author*. Lalu *input* di *encode* ke vektor x. Setelah di *encode* maka masuk ke proses *decode* untuk memuat ulang *input* yang merupakan *output* data yang sudah direduksi dimensinya. Setelah itu hitung *reconstruction loss*.



Gambar 3.12 Flowchart Klasifikasi Auto Encoder

Proses klasifikasi dengan menggunakan auto encoder ini dimaksudkan untuk menjadi pembanding dalam metode anomaly detection dimana algoritma machine learning dan deep learning akan menghasilkan nilai prediksi dari proses klasifikasi yang seperti apa, terutama ketika digunakan pada data yang berdimensi tinggi atau tingkat imbalance yang tinggi.

3.6 Analisa Hasil

Pada tahap ini akan dijelaskan mengenai analisis dari keseluruahan kinerja sistem klasifikasi *author matching* menggunakan metode Anomaly Detection dengan menggunakan 3 model algoritma itu akan mencakup akurasi, presisi, sensitivitas, dan F1-score. Hasil sementara dalam penelitian ini akan dijelaskan di Bab IV.

3.7 Kesimpulan

Tahap ini dilakukan untuk menarik kesimpulan terhadap metode *anomaly detection* dengan menggunakan 3 model algoritma. Penarikan kesimpulan merupakan jawaban dari tujuan penelitian yang ingin dicapai. Kesimpulan akan dijelaskan di Bab V.

BAB IV
HASIL DAN ANALISA

Bab ini berisi hasil dari penelitian dari sistem klasifikasi binary untuk data positif dan negatif pada *author matching*. Pada bab ini akan mengukur performa dari klasifikasi seperti akurasi, presisi, sensitivitas, spesifisitas, dan *F1-score*. Hasil validasi untuk bab IV ini menggunakan dua skenario pada satu model klasifikasi. Skenario pertama memvalidasi hasil dengan langsung menggunakan data secara keseluruhan pada model *isolationforest, Local Outlier Factor dan AutoEncoder*. Skenario kedua yaitu pembagian data *training* dan *testing* yang berbeda pada model *Isolation Forest*. Ini bertujuan untuk melihat apakah metode *anomaly detection* layak digunakan dan apakah model nya mampu memberikan hasil validasi yang baik pada proses klasifikasi *author matching*.

4.1 Hasil Validasi Menggunakan Isolation Forest (Langsung pada keseluruhan Data / Percobaan Pertama)

Proses klasifikasi pada data author matching menggunakan *isolation forest* dilakukan dengan mendefinisikan *n_estimator* yaitu panjang baris data yang akan dijadikan acuan untuk melakukan pencarian data anomaly, dan dalam percobaan ini ditentukan adalah keseluruhan data. Selanjutnya adalah pendefinisian *contamination* dengan menggunakan *'auto'* yaitu seberapa besar porsi *outlier* pada data anomali yang akan diklafisikasi atau bisa dikatakan seberapa besar rasio jumlah data anomali yang akan dipakai atau ditemukan. Langkah berikutnya adalah menentukan *max_feature* yang merupakan angka maksimum tertinggi dari *fetature* yang ada dan dalam penelitian ini angka terebut adalah 1.0

Selanjutnya dengan menggunakan decision function pada format behavoiur untuk data string dengan memilih '*new*' sebagai poin *behaviour* yang berfungsi untuk menetapkan *threshold* menjelaskan proses pengenalan anomali dimulai dari awal data untuk mendeteksi *outlier*. Selain itu, digunakan juga *random generator* untuk dapat melakukan pengenalan dan pembelajaran secara acak dengan *random_state* sebanyak 42 kali.

Proses diatas mampu menghasilkan nilai akurasi yang tinggi serta menyajikan hasil *precision* dan *recall* yang cukup baik untuk data yang anomali. Hasil dari proses klasifikasi menunjukkan bahwa pendekatan anomaly detection dengan menggunakan algoritma IsolatioForest dapat digunakan pada data yang berdimensi tinggi. Selain itu, dengan menggunakan keseluruhan data (tanpa pembagian jumlah data training dan testing) mampu menghasilkan nilai *precision* dan *recall* pada kedua identitas binary dari label baik itu data positif maupun data negatif (tabel 4.1).

Tabel 4.1

Hasil Kinerja Model Isolation Forest Secara Langsung (Keseluruhan Data tanpa Data Training dan Testing)

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,995**		
Precision	**0,78**	*1.00*	*0,78*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,79**	*1.00*	*0,79*
F1-Score	**0,78**	*1.00*	*0,78*
Data Support		***51,023,778***	***583,942***

Hasil confusion matrix dari proses klasifikasi secara keseluruhan data juga mampu disajikan dengan baik sehingga ketika dilakukan perhitungan manual dengan menggunakan hasil dari confusion matrix tersebut, maka hasil performance yang ditunjukkan dalam proses klasifikasi menggunakan algoritma ini dapat dibuktikan (tabel 4.2)

Tabel 4.2

Hasil Confusion Matrix Model Isolation Forest

TP	FP
458,920	125,022
FN	**TN**
131,704	50,892,074

Berikut adalah perhitungan manual berdasarkan hasil dari confusion matrix untuk membuktika hasil accuracy, presition dan menampilkan hasil perhitungan manual sensitivity, spesivicity, dan f1 score.

1. Accuracy $= \frac{458920+50892074}{458920+125022+131704+50892074} = \frac{51350994}{51607720} = 0{,}995$

2. Precision $= \frac{458920}{458920+125022} = \frac{458920}{583924} = 0{,}785$

3. Sensitivity $= \frac{458920}{458920+131704} = \frac{458920}{590624} = 0{,}787$

4. Spesificity $= \frac{50892074}{125022+50892074} = \frac{50892074}{51017926} = 0{,}997$

5. F1 score $= 2 \text{ x } \frac{0{,}785 \text{ x } 0{,}777}{0{,}785+0{,}777} = \frac{0{,}61}{1{,}56} = 0{,}78$

4.2 Hasil Validasi Menggunakan Isolation Forest (Split data untuk Training dan Testing/Percobaan Kedua)

Hasil percobaan selanjutnya adalah dengan menerapkan pengujian berdasarkan pembagian data dengan skenario 80% untuk training dan 20% untuk testing. Jumlah pembagian ini merupakan persentase umum yang sering digunakan pada proses data latih dan data uji (training dan testing). Sama seperti sebelumnya, mendefinisikan *n_estimator* yaitu panjang baris data yang akan dijadikan acuan untuk melakukan pencarian data anomaly, dan dalam percobaan ini ditentukan

adalah keseluruhan data. Selanjutnya adalah pendefinisian *contamination* dengan menggunakan *'auto'* yaitu seberapa besar porsi *outlier* pada data anomali yang akan diklafisikasi atau bisa dikatakan seberapa besar rasio jumlah data anomali yang akan dipakai atau ditemukan. Langkah berikutnya adalah menentukan *max_feature* yang merupakan angka maksimum tertinggi dari *fetature* yang ada dan dalam penelitian ini angka terebut adalah 1.0

percobaan ini dilakukan dengan menggunakan decision function pada format behavoiur untuk data string dengan memilih '*new*' sebagai poin *behaviour* dan juga *random generator* untuk dapat melakukan pengenalan dan pembelajaran secara acak dengan *random_state* sebanyak 42 kali.

Proses ini dimaksudkan untuk memberikan tahapan pengenalan dan pembelajaran (learning) pada algoritma sebelum memutuskan pada data sebenarnya atau pada data testing. Pada proses ini, pengenalan dan pembelajaran dilakukan menggunakan random generator yang memilih data secara acak sepanjang baris data yang ada. Pemilihan random state sebanyak 42 kali adalah untuk mencari jumlah kelompok data paling lengkap dan diharapkan mampu menghasilkan proses klasfikasi yang terbaik.

Selain data diatas, proses pada isolation forest juga dilakukan dengan menentukan terlebih dahulu n_estimators yang berfungsi sebagai rujukan jumlah kelompok data yang ditaksir sebagai anomali, type data secara umum adalah integer dan nilai standarnya adalah 100. Lalu dilanjutkan dengan menentukan max_sample sebagai jumlah yang akan digunakan atau diambil untuk melatih setiap label dan feature.

Lalu dilanjutkan dengan menentukan contamination, dimana variabel ini digunakan untuk menentukan ambang batas atau threshold yang akan dipakai sebagai acuan untuk menetakan data anomali. Namun, dalam prosesnya nilai ambang batas tersebut ditentukan secara otomatis di dalam sistem. Dan terakhir adalah max_feature yang digunakan untuk menentukan jumlah feature yang digunakan dalam proses latihan, agar tidak semua feature yang identik digunakan lebih dari satu kali.

Dari semua proses tersebut, didapatkan hasil prediksi yang sangat baik di kategori akurasi dan spesificity, dan sama seperti percobaan sebelumnya pada

keseluruhan data, hasil yang cukup baik atau dapat dikategorikan baik dalam klasifikasi data yang super imbalance didapatkan pada kategori presisi, sensitivity dan juga dengan ditunjukkan pada persentase capaian di F1-Score (Tabel 4.3).

Pada tabel tersebut menampilkan hasil pada percobaan kedua (menggunakan split data) dimana hasil presiksi pada data training dibandingkan dengan hasil prediksi pada data testing. Selain itu ditampilkan juga hasil dari setiap kelas dengan tujuan memberikan penjelasan bahwa setiap kelas berhasil ditemukan datanya dalam percobaan kedua ini.

Tabel 4.3

Hasil Pengujian Data Training (80%) dan Testing (20%)

Kategori	Training			Testing		
	80% Data	Kelas		20% Data	Kelas	
		0	1		0	1
Accuracy	0,995			0,995		
Presisi	0,793	1.00	0,793	0,799	1.00	0,788
Sensitivity	0,798	1.00	0,798	0,826	1.00	0,797
Specificity	0,997	1.00	0,997	0,997	1.00	0,997
F1 Score	0,782	1.00	0,782	0,782	1.00	0,782
Data Support		***40,819,022***	***467,154***		***10,204,756***	***116,788***

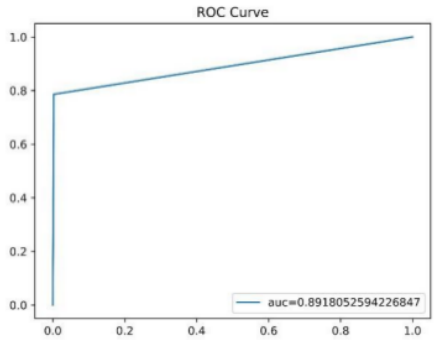
Hasil diatas menunjukkan tidak ada perbedaan hasil yang signifikan dari proses sebelumnya. Lalu pada proses validasi pengujian atau *testing* juga mendapatkan hasil yang tidak jauh berbeda dan dapat juga dilihat dari hasil *confusion matrix* (Tabel 4.5). Hal ini menunjukkan performa yang stabil dari model Isolation Forest dalam proses klasifikasi *author matching*.

Tabel 4.4

Hasil Confusion Matrix Data Testing (20%)

TP 93,177	FP 23,611
FN 25,033	TN 10,179,723

Selain hasil diatas, performa dari metode anomaly detection dengan menggunakan model Isolation Forest dapat dikatakan sangat baik dapat dilihat dari hasil perhitungan confusion matrix yang digambarkan melalui kurva ROC (Gambar 4.1). Kurva tersebut digunakan untuk meringkaskan data *confusiton matrix* yang bernilai binary atau 2 class dan menggambarkan hasil dari nilai *confusion matrix* dan menggambarkan *True Positive Rate (TPR)* dan nilai *False Positif Rate (FPR)*. TPR menggambarkan seberapa besar data positif yang ditemukan dibandingkan dengan data positif sebernarnya, begitu pula pada FPR. Pada kurva ROC tersebut, dapat disimpulkan bahwa nilai TPR naik mulai dari 0.0 sampai ke 0.8. hal tersebut menjelaskan bahwa sensitivitas dari model algoritma ini hanya sampai pada 0.8, dan nilai dapat menampilkan hasil prediksi dengan cukup akurat. Sedangkan nilai FPR dapat disimpulkan bahwa semakin mendekati nilai 1.0.



Gambar 4.1. Kurva ROC Isolation Forest

4.3 Hasil Validasi Menggunakan Isolation Forest (Fine Tunning Parameter / Percobaan Ketiga)

Pada percobaan berikutnya adalah dengan melakukan perubahan paramater pada algoritma di python yaitu dengan merubah pilihan *contamination* dari *'auto'* menjadi lebih spesifik yaitu *'outlier fraction'*. Parameter itu digunakan untuk merekam proses pencarian data anomali untuk dibandingkan dengan data yang ditemui berikutnya. Selain itu, dalam percobaan ketiga ini juga ditambahkan parameter *max_samples* untuk mencoba membatasi jumlah sample yang diambil dalam proses klasifikasi, dan parameter yang digunakan adala *áuto'*. Percobaan ketiga ini dimaksudkan untuk meningkatkan hasil validasi dari data *author matching* ini menggunakan algoritma Isolation Forest. Adapun pilihan parameter yang dilakukan perubahan adalah memang parameter yang memiliki alternatif untuk dilakukan *fine tunning*.

Dalam percobaan ketiga ini, dilakukan proses klasifikasi sama seperti pada percobaan pertama dimana kami melakukan pada keseluruhan data secara langsung tanpa ada proses split data. Dan hasil yang ditampilkan pada percobaan ini menjadi sama persis dengan hasil pada percobaan pertama, tanpa ada perbedaan sedikitpun dimana nilai validasi dan *confusion matrix* menampilkan nilai yang sama persis. Bahkan pada data support dan hasil validasi pada setiap class juga menampilkan hasil yang sama persis (Tabel 4.5).

Tabel 4.5

Hasil Kinerja Model Isolation Forest dengan Fine Tunning Parameter

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,995**		
Precision	**0,78**	*1.00*	*0,78*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,79**	*1.00*	*0,79*
F1-Score	**0,78**	*1.00*	*0,78*
Data Support		***51,023,778***	***583,942***

4.4 Hasil Validasi Menggunakan *Local Outlier Factor (LOF)* / Percobaan Pertama

Proses klasifikasi pada data author matching menggunakan *Local Outlier Factor (LOF)* dilakukan menggunakan python dimulai dengan menetapkan jumlah *n_neighbors terbelih dahulu* atau jumlah jumlah baris tetangga dalam satu cluster. Secara default, biasanya jumlahnya ditetakan 20, namun karena jumlah data hasil pre processing *author matching* ini sangat banyak, maka dalam penelitian ini kami menetapkan jumlah dari *n_neighbor* menjadi 2000. Hal tersebut dilakukan agar proses klasifikasi tidak terlalu lama namun selutuh data atau sampel tetap dapat diklasifikan.

Selain itu, hal berikutnya yang ditentukan dalam algoritma ini adalah *algorithm*. Secara default dalam percobaan pertama ini, kami menggunakan pilihan *auto* pada keselurahan data feature dan label. Hal ini dimaksudkan untuk melihat seberapa besar nilai prediksi yang dihasilkan pada pilihan struktur standar pada algoritma ini.

Selanjutnya adalah pemilihan besar *leaf_size*, yaitu kecepatan penggunaan memory dalam proses klasifikasi ini. Secara default biasanya nilai yang digunakan adalah 30, namuan dalam penelitian ini digunakan nilai 150 untuk mempercepat proses dan menampilkan hasil dari prediksi. Kemudian, yang dilakukan adalah pemilihan *metric* yang digunakan sebagai parameter jarak atau panjang dalam perhitungan komputasi, dan dalam penelitian ini digunakan *'minkowski'*.

Langkah selanjutnya di python adalah dengan menentukan *metric_param* yang dalam penelitian ini dipilih *none*, dikarenakan hal tersebut sudah diwakili oleh metric pada proses sebelumnya. Kemudian adalah penentuan *contamination* yang merupakan proses pendefinisian nilai atau jumlah yang akan dikategorikan sebagai outlier atau anomali. Dalam percobaan pertama ini, kami menggunakan pilihan *auto* agar hasil prdiksi menjadi lebih baik.

Proses diatas mampu menghasilkan nilai akurasi yang cukup tinggi serta menyajikan hasil *precision* dan *recall* yang cukup baik untuk data yang anomali meskipun tidak melebihi dari proses pada algoritma Isolation Forest. Sekali lagi, hasil dari proses klasifikasi menunjukkan bahwa pendekatan anomaly detection dengan menggunakan algoritma IsolatioForest dapat digunakan pada data yang

berdimensi tinggi. Selain itu, proses tersebut mampu menghasilkan nilai *precision* dan *recall* pada kedua identitas binary dari label baik itu data positif maupun data negatif (tabel 4.6).

Tabel 4.6

Hasil Kinerja Model Local Outlier Factor Secara Langsung

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,994**		
Precision	**0,76**	*1.00*	*0,76*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,76**	*1.00*	*0,76*
F1-Score	**0,76**	*1.00*	*0,76*
Data Support		***51,022,724***	***584,996***

Hasil confusion matrix yang mampu menunjukkan nilai dari hasil prediksi dan dari proses klasifikasi secara keseluruhan data juga mampu disajikan dengan baik (tabel 4.7)

Tabel 4.7

Hasil Confusion Matrix Model Local Outlier Factor

TP 447,620	FP 137,376
FN 139,004	TN 50,883,720

Berikut adalah perhitungan manual berdasarkan hasil dari confusion matrix untuk membuktikan hasil accuracy, presition dan menampilkan hasil perhitungan manual sensitivity, spesivicity, dan f1 score.

1. Accuracy $= \frac{447620+50883720}{447620+137376+139004+50883720} = \frac{51331340}{51607720} = 0{,}994$

2. Precision $= \frac{447620}{447620+137376} = \frac{447620}{584996} = 0{,}76$

3. Sensitivity $= \frac{447620}{447620+139004} = \frac{447620}{586624} = 0{,}76$

4. Spesificity $= \frac{50883720}{137376+50883720} = \frac{50883720}{51021096} = 0{,}997$

5. F1 score $= 2 \times \frac{0{,}76 \times 0{,}76}{0{,}76+0{,}76} = \frac{0{,}61}{1{,}56} = 0{,}76$

4.5 Hasil Validasi Menggunakan Local Oulier Factor (Split data untuk Training dan Testing / Percobaan Kedua)

Hasil percobaan selanjutnya adalah dengan menerapkan pengujian berdasarkan pembagian data dengan skenario 80% untuk training dan 20% untuk testing. Semua teknik di python yang digunakan pada percobaan kedua ini sama seperti sebelumnya. Proses ini dimaksudkan untuk memberikan tahapan pengenalan dan pembelajaran (learning) pada algoritma sebelum memutuskan pada data sebenarnya atau pada data testing.

Pada proses ini, pengenalan dan pembelajaran dilakukan menetapkan terlebih dahulu panjang tetangga atau barisan data yang akan dijadikan acuan sebagai pendefinisian outlier sepanjang baris data yang ada. menggunakan python dimulai dengan menetapkan jumlah *n_neighbors terbelih dahulu* atau jumlah jumlah baris tetangga dalam satu cluster. Secara default, biasanya jumlahnya ditetakan 20, namun karena jumlah data hasil pre processing *author matching* ini sangat banyak, maka dalam penelitian ini kami menetapkan jumlah dari *n_neighbor* menjadi 2000. Hal tersebut dilakukan agar proses klasifikasi tidak terlalu lama namun selutuh data atau sampel tetap dapat diklasifikan.

Selain itu, semua teknik yang digunakan dalam python sama seperti pada percobaan sebelumnya. Hal itu dikarenakan ingin memperoleh kesimpulan apakah dengan metode klasifikasi yang berbeda, algoritma ini mampu memberikan hasil

validasi yang lebih baik atau tidak. Hasil dari percobaan kedua pada algoritma ini dapat dilihat pada tabel 4.8.

Tabel 4.8

Hasil Pengujian Data Training (80%)

Kategori	Training			Testing		
	80% Data	Kelas		20% Data	Kelas	
		0	1		0	1
Accuracy	0,995			0,995		
Presisi	0,812	1.00	0,812	0,812	1.00	0,812
Sensitivity	0,803	1.00	0,803	0,803	1.00	0,803
Specificity	0,997	1.00	0,997	0,997	1.00	0,997
F1 Score	0,801	1.00	0,801	0,801	1.00	0,801
Data Support		***40,819,022***	***467,154***		***10,001,274***	***110,894***

Hasil diatas menunjukkan ada perubahan hasil validasi pada percobaan kedua ini, dimana hasil pada kategori accuracy meningkat serta terjadi peningkatan cukup signifikan pada kategori presisi, sensitivity, spesificity, dan f1 score. Perubahan menunjukkan ada proses yang berpengaruh pada hasil validasi, dan dapat disimpulkan bahwa penentuan jumlah *n_neighbors* menjadi penting dalam proses klasifikasi menggunakan algoritma LOF ini. Panjang jumlah data yang akan dihitung menjadi titik seberapa baik hasil validasi dari algoritma LOF ini.

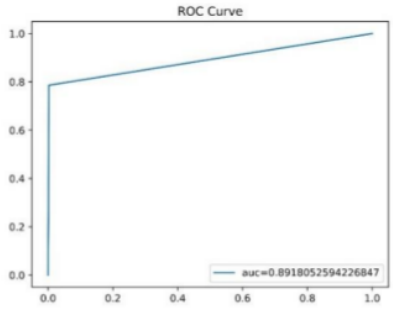
Parameter lain yang digunakan pada percobaan tidak berpengaruh pada nilai hasil dari validasi dikarenakan teknik yang laen dalam python yang digunakan tidak mempengaruhi proses aritmatik yang menentukan nilai validasi outlier. Dan dapat disimpulkan juga bahwa nilai outlier yang ditentukan dalam percobaan kedua ini adalah data yang bernilai 1.

Tabel 4.9

Hasil Confusion Matrix Data Testing (20%)

TP	FP
90,122	20,772
FN	**TN**
22,345	9,978,929

Selain hasil diatas, performa dari metode anomaly detection dengan menggunakan model Algoritma Local Outlier Factor dengan percobaan split data dapat dikatakan sangat baik dapat dilihat dari hasil perhitungan confusion matrix yang digambarkan melalui kurva ROC (Gambar 4.2) dimana kurva digambarkan mendekakati nilai 0.79 pada TPR dan mendekati titik 1.0. pada nilai FPR.



Gambar 4.2. Kurva ROC Local Outlier Factor

4.6 Hasil Validasi Menggunakan Local Oulier Factor (Fine Tunning parameter / Percobaan Ketiga)

Pada percobaan berikutnya adalah dengan melakukan perubahan paramater pada algoritma di python yaitu dengan merubah pilihan *algorithm* dari *'auto'* menjadi *'ball_tree'*, lalu mencoba merubah parameter *contamination* dari *'auto'* menjadi lebih spesifik yaitu *'float'*. Percobaan ketiga ini dimaksudkan untuk meningkatkan hasil validasi dari data *author matching* ini menggunakan algoritma LOF. Adapun pilihan parameter yang dilakukan perubahan adalah memang parameter yang memiliki alternatif untuk dilakukan *fine tunning*.

Dalam percobaan ketiga ini, kami melakukan proses klasifikasi sama seperti pada percobaan pertama dimana kami melakukan pada keseluruhan data secara

langsung tanpa ada proses split data. Dan hasil yang ditampilkan pada percobaan ini menjadi sama persis dengan hasil pada percobaan pertama, tanpa ada perbedaan sedikitpun dimana nilai validasi dan *confusion matrix* menampilkan nilai yang sama persis. Bahkan pada data support dan hasil validasi pada setiap class juga menampilkan hasil yang sama persis (Tabel 4.10).

Tabel 4.10

Hasil Pengujian Data Training (80%)

Kategori	Nilai	Kelas	
		0	1
Accuracy	**0,994**		
Precision	**0,76**	*1.00*	*0,76*
Specificity	**0,99**	*1.00*	*0,99*
Recall	**0,76**	*1.00*	*0,76*
F1-Score	**0,76**	*1.00*	*0,76*
Data Support		***51,022,724***	***584,996***

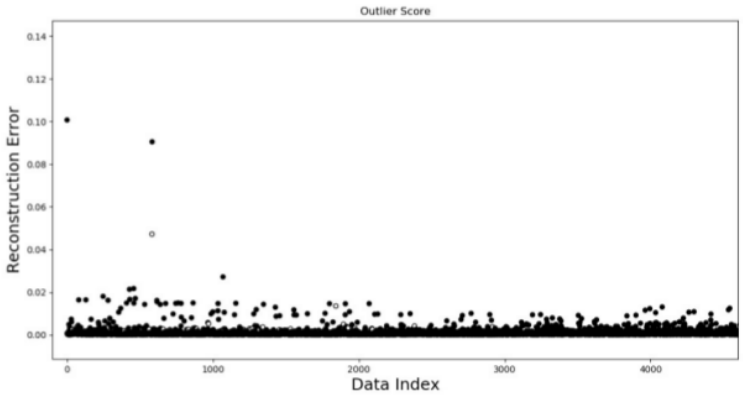
Dari hasil diatas dapat disimpulkan bahwa algoritma LOF dengan berbagai parameter yang dimiliki di dalam python mempunyai kestabilan dan cukup handal menampilkan hasil validasi jika digunakan pada data jelas dan memiliki struktur value yang seragam. Sama halnya dengan Isolation Forest, algoritma ini mampu menampilkan hasil yang cukup baik pada data *author matching*, dan juga pendekatan anomaly detection yang digunakan untuk mengklasifikan data *author matching* yang highly imbalance ini dirasa menjadi sangat tepat.

4.7 Hasil Validasi Menggunakan Auto Encoder

Pada tahap ini adalah proses klasfikasi dengan menggunakan algoritma ketiga yaitu auto encoder. Auto encoder merupakan salah satu algoritma yang dapat digunakan dalam pendekatan namoly detection denngan menggunakan konsep *neural network*. Dimana algoritma ini akan digunakan dalam *un-supervised* proses dengan diawali proses ekstraksi data hasil pre prosessing dengan melakukan

uncode dan decode, lalu dilakukan tahap visualisasi data untuk menentukan nilai threshold yang akan dijadikan acuan sebagai kelompok anomaly atau outlier.

Proses ini dilakukan dengan merepresentasikan data awal lalu dilanjutkan dengan merekinstruksikan kembali data tersebut untuk dapat diolah menjadi lebih jelas pada neural network. Hasil dari proses tersebut dilanjutkan dengan proses visualisasi bentuk data untuk dilankutkan proses penetuan threshold secara manual sebelum dilanjutkan dengan proses validasi. Dan hasil dari visualisasi data tersebut dapat dilihat pada Gambar 4.3.



Gambar 4.3. Visualisasi Auto Encoder pada Data Author Matching

Dari hasil visualisasi diatas, kami menyimpulkan bahwa proses penentuan nilai threshold yang menjadi acuan pendefinisian data namolay atau outlier menjadi tidak dapat dilakukan, karena semua data yang memiliki nilai ‘1’ atau divisulisasikan dengan lingkaran yang kosong, hampir semuanya tertutupi oleh data yang memiliki nilai ‘0’ atau lingkaran yang hitam. Sebagaimana telah kami jelaskan pada proses sebelumnya (pre processing) bahwa data 0 adalah data author yang tidak sama, sedangkan data 1 adalah data author yang sama.

Dengan gagalnya proses penentuan trheshold tersebut, maka proses validasi tidak dapat kami lanjutkan karena bisa dipastikan bahwa class 1 tidak akan terdeteksi dalam proses klasifikasi atau validasi tersebut sehingga nilai TP pada confusion matrix yang menjadi data jadi acuan data benar menjadi bernilai ‘Nan’.

Untuk menjelaskan hal tersebut, proses penelitian ini dilanjutkan ke dalam tahap analisa data pada algoritma ini berdasarakan hasil analisis penelitian sebelumnya.

Pada penelitian sebelumnya dijelaskan bahwa, untuk proses klasifikasi data imbalance menggunakan deep learning lebih baik dengan cara menyeimbangkan data terlebih dahulu menggunakan proses over sampling atau under sampling (Amin *et al*., 2016; Gong and Chen, 2016). Teknik tersebut tidak digunakan dalam penelitian ini dikarenakan pada data author matching klasfikasi data membutuhkan data realyang tidak terkontaminasi apapun agar dapat menghasilkan nilai validasi yang akurat sesuai dengan data yang ada. Selain itu, teknik oversampling yg dilakukan pada data imbalance dengan selisih data 99% dirasakan tidak tepat karena menambahkan terlalu banyak informasi baru yang sebenarnya palsu.

Pada proses klasifikasi data author matching dengan menggunakan auto encoder ini, hasil validasi yang tidak dapat ditampilkan kemungkinan dikarenakan jumlah data yang terlalu banyak dengan selisih jumlah data yang imbalance tertalu banyak pula. Barisan data yang terlalu banyak menyebabkan proses algoritma deep learning menjadi sulit untuk menentukan nilai validasi dengan baik (Leevy *et al*., 2018). Oleh karena itulah, dalam algoritma ketiga ini tidak dapat menentukan nilai threshold dan tidak dapat dilanjutkan ke dalam proses validasi.

4.8 Studi Perbandingan

Evaluasi kinerja metode anomaly detection dengan menggunakan Isolation Forest, Local Outlier Factor dan AutoEncoder dapat dilihata pada tabel 4.11. dan hasil penelitian ini dibandingkan dengan beberapa penelitian sebelumnya tentang pengklasifikasian author matching. Hasil dari perbandingan tersebut bisa dilihat pada tabel 4.12.

Tabel 4.11

Tabel Perbandingan Hasil Validasi antar Algoritma

Kategori	Isolation Forest			LOF			Auto Encoder
	Percobaan			Percobaan			
	1	2	3	1	2	3	
Accuracy	0,995			0,995			-
Presisi	0,812	1.00	0,812	0,812	1.00	0,812	-
Sensitivity	0,803	1.00	0,803	0,803	1.00	0,803	-
Specificity	0,997	1.00	0,997	0,997	1.00	0,997	-
F1 Score	0,801	1.00	0,801	0,801	1.00	0,801	-

Tujuan awal dari penelitian ini adalah untuk menguji apakah metode anomaly detection dapat digunakan pada klasifikasi terhadap author matching yang diawali dengan proses pairwise combination dan string simlarity pada proses ekstraksi fiturnya. Pendekatan atau metode yang digunakan pada penelitian ini belum pernah digunakan sebelumnya, terutama pada klasifikasi data teks seperti data author matching. Selain itu, tujuan dari penelitian ini adalah untuk menguji juga model dari anomaly detection yaitu Isolation Forest, Local Outlier Factor (LOF) mampu menghasilkan nilai klasifikasi yang baik. Hasil yang ditunjukkan dari penggunaan model ini bisa dikatakan sangat baik dari sisi akurasi, meskipun belum optimal dari sisi precision dan recall yang sering jadi poin utama penilaian untuk klasifikasi data teks. Meskipun demikian, untuk data dengan tingkat imbalance yang sangat tinggi, hasil secara keseluruhan dari model ini bisa dikatakan baik. Tabel 4.6 dapat menunjukkan perbandingan dari penelitian sebelumnya.

Tabel 4.12

Tabel Perbandingan Dengan Penelitian Sebelumnya

No.	Metode	author	Implementasi	Teknik	Akurasi
1	Heuristic Based Hirarchical	AA. Ferreira, et al (2017)	Klasifikasi Biner antar author	Pairwise dan Similarity	94,3
2	Associative classifier	Marcos André Gonçalves, et al (2015)	Klasifikasi Biner	Pairwise	90,4
3	K-NN	Ming Son, et al (2015)	Klasifikasi Biner	Pairwise dan Similarity	98,17
4	Random Forest	kim, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	98,13
5	C4.5	kim, et al (2018)		Pairwise dan Similarity	98,2
6	SVM	kim, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	97,45
7	Naive Bayes	kim and diesner (2016)	Klasifikasi Biner	Pairwise dan Similarity	96,68
8	Boosted tree Classification	wang jian, et al (2012)	Klasifikasi Biner	Pairwise dan Similarity	89,3
9	Dempster-Shafer Theory	Hao wu, et al (2014)		Pairwise	
10	Graph Structural	Ijaz Husain, et al (2017)	Klasifikasi Biner	Pairwise dan Similarity	
11	Deep Neural Network	Tin Huynh, et al (2018)	Klasifikasi Biner	Pairwise dan Similarity	99,31
12	Anomaly Detection	2019	Klasifikasi Biner	Pairwise dan Similarity	99,56

Dari semua hasil percobaan, analisis dan perbandingan dengan masing-masing percobaan maupun dibandingkan dengan hasil klasifikasi penelitian sebelumnya, dapat disimpulkan bahwa proses klasifikasi data author pada database publikasi penelitian dapat dilakukan dengan mnggunakan pendekatan anonaly detection. Dimana pendekatan tersebut digunakan ketika proses pra prosesing data set dilakukan dengan menggunakan teknik pairwise dengan model kombinasi. Model tersebut menghasilkan data yang memiliki kesenjangan yang sangat tinggi (*super imbalance)* sehingga beberapa model algoritma classifier tidak dapat memberikan hasil dari proses klasifikasi data tersebut.

Namun, harus diakui bahwa hasil yang digunakan pada pendekatan ini, belum dapat menghasilkan nilai klasifikasi yang sempurna. Hal tersebut dikarenakan diperlukan proses modifikasi pada algoritma klasifikasi yang digunakan untuk memberikan hasil klasifikasi yang lebih baik dibandingkan yang dihasilkan pada penelitian ini.

BAB V

KESIMPULAN DAN SARAN

5.1. KESIMPULAN

Kesimpulan yang dihasilkan dalam penelitian mengenai pengklasifikasian author mathcing dengan menggunakan metode anomaly detection adalah sebagai berikut:

1. Proses pra pengolahan data set author name disambiguation dapat dilakukan dengan menggunakan pencocokan nama penulis (*author matching)* dengan melakukan pairwise combination dan string similarity dengan coeficient jaccard sehingga mampu menginterpretasikan kecocokan nama author dalam proses klasifikasi.
2. Metode anomaly detection dapat digunakan dalam proses klasifikasi pada data *author matching* yang memiliki tingkat *imbalance* yang sangat tinggi (1% data positive berbanding 99% data negative), hasil akurasi yang baik (akurasi 99,5 %) ditunjukkan pada metode ini dengan menggunakan model Algoritma *Isolation Forest dan Local Outlier Factor*.
3. Kinerja yang dihasilkan pada proses klasifikasi data author matching bisa disimpulkan sangat baik dari sisi akurasi dan spesifitas dibandingkan penelitian sebelumnya, serta cukup baik pada presisi, sensitivitas dan F1 score mengingat data yang digunakan memiliki tingkat imbalance yang sangat tinggi.

5.2. SARAN

Agar penelitian selanjutnya yang berkaitan dengan penelitian ini dapat lebih lebih baik dan mampu menghasilkan klasifikasi yang lebih baik pula, dapat disarankan sebagai berikut :

1. Dataset raw author pada database publikasi memerlukan beberapa tahapan lagi dalam pra pemrosesan untuk mampu menghasilkan pengenalan yang lebih baik pada tahapan pembelajaran data uji dan data latih ketika masuk ke dalam classifier.

2. Pendekatan anolamy detection memerlukan proses atau tahapan modifikasi model algoritma yang berkaitan dengan anomaly detection. Hal tersebut dikarenakan pada format model algoritma default, tidak mampu meningkatkan hasil prediksi walaupun sudah merubah beberapa fungsi dalam algoritma tersebut.

DAFTAR PUSTAKA

Amin, A. *et al*. (2016) 'Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study', *IEEE Access*, 4(Ml), pp. 7940–7957. doi: 10.1109/ACCESS.2016.2619719.

Berzins, K. *et al*. (2012) 'A boosted-trees method for name disambiguation', *Scientometrics*, 93(2), pp. 391–411. doi: 10.1007/s11192-012-0681-1.

de Carvalho, A. P. *et al*. (2011) 'Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries', *Journal of Information and Data Management*, 2(573871), p. 289.

Cheng, Z., Zou, C. and Dong, J. (2019) 'Outlier detection using isolation forest and local outlier', *Proceedings of the 2019 Research in Adaptive and Convergent Systems, RACS 2019*, pp. 161–168. doi: 10.1145/3338840.3355641.

Cota, R. G. *et al*. (2010) 'An Unsupervised Heuristic-Based Hierarchical Method for Name Disambiguation in Bibliographic Citations', 61(May), pp. 1853–1870. doi: 10.1002/asi.

Dawoud, A., Shahristani, S. and Raun, C. (2019) 'Dimensionality Reduction for Network Anomalies Detection: A Deep Learning Approach', in *Advances in Intelligent Systems and Computing*. Springer Verlag, pp. 957–965. doi: 10.1007/978-3-030-15035-8_94.

Ferreira, A. A. *et al*. (2010) 'Effective self-training author name disambiguation in scholarly digital libraries', *Proceedings of the ACM International Conference on Digital Libraries*, pp. 39–48. doi: 10.1145/1816123.1816130.

Ferreira, A. A., Gonçalves, M. A. and Laender, A. H. F. (2012) 'A brief survey of automatic methods for author name disambiguation', *ACM SIGMOD Record*, 41(2), p. 15. doi: 10.1145/2350036.2350040.

Ferreira, V. O. *et al*. (2015) 'A model for anomaly classification in intrusion detection systems', in *Journal of Physics: Conference Series*. doi: 10.1088/1742-6596/633/1/012124.

Firdaus, F. (2018) 'Improving Data Integrity of Individual-based Bibliographic Repository Using Clustering Techniques', *Computer Engineering and Applications Journal*, 7(1), pp. 49–56. doi: 10.18495/comengapp.v7i1.223.

Fugate, M. and Gattiker, J. R. (2002) 'Anomaly detection enhanced classification in computer intrusion detection', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 186–197. doi: 10.1007/3-540-45665-1_15.

Gong, Z. and Chen, H. (2016) 'Model-based oversampling for imbalanced

sequence classification', *International Conference on Information and Knowledge Management, Proceedings*, 24-28-October-2016, pp. 1009–1018. doi: 10.1145/2983323.2983784.

Gu, S. *et al*. (2016) 'Name Disambiguation Method Based on Multi-step Clustering', *Procedia Computer Science*, 83(Ant), pp. 488–495. doi: 10.1016/j.procs.2016.04.237.

Han, H. *et al*. (2004) 'Two supervised learning approaches for name disambiguation in author citations', *Proceedings of the ACM IEEE International Conference on Digital Libraries, JCDL 2004*, pp. 296–305.

Hazra, R. *et al*. (2016) 'An efficient technique for author name disambiguation', *2016 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2016*. doi: 10.1109/ICCTAC.2016.7567344.

Hussain, I. and Asghar, S. (2017) 'A survey of author name disambiguation techniques: 2010–2016', *The Knowledge Engineering Review*, 32, p. e22. doi: 10.1017/S0269888917000182.

Hussain, I. and Asghar, S. (2018a) 'Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity', *Arabian Journal for Science and Engineering*. Springer Berlin Heidelberg, 43(12), pp. 7421–7437. doi: 10.1007/s13369-018-3099-0.

Hussain, I. and Asghar, S. (2018b) 'LUCID: Author name disambiguation using graph Structural Clustering', *2017 Intelligent Systems Conference, IntelliSys 2017*, 2018-Janua(September), pp. 406–413. doi: 10.1109/IntelliSys.2017.8324326.

John, H. and Naaz, S. (2019) 'Credit Card Fraud Detection using Local Outlier Factor and Isolation Forest International Journal of Computer Sciences and Engineering Open Access Credit Card Fraud Detection using Local Outlier Factor and Isolation', (September). doi: 10.26438/ijcse/v7i4.10601064.

Kharitonov, A. and Zimmermann, A. (2019) 'Intrusion detection using growing hierarchical self-organizing maps and comparison with other intrusion detection techniques', *CPSS 2019 - Proceedings of the 5th ACM Cyber-Physical System Security Workshop, co-located with AsiaCCS 2019*, pp. 13–23. doi: 10.1145/3327961.3329531.

Kim, J. (2018) 'Evaluating author name disambiguation for digital libraries: a case of DBLP', *Scientometrics*. Springer International Publishing, 116(3), pp. 1867–1886. doi: 10.1007/s11192-018-2824-5.

Kim, Jinseok and Kim, Jenna (2018) 'The impact of imbalanced training data on machine learning for author name disambiguation', *Scientometrics*. Springer International Publishing, 117(1), pp. 511–526. doi: 10.1007/s11192-018-2865-9.

Kunang, Y. N. *et al*. (2019) 'Automatic Features Extraction Using Autoencoder in Intrusion Detection System', *Proceedings of 2018 International Conference on Electrical Engineering and Computer Science, ICECOS 2018*. IEEE, (June 2019), pp. 219–224. doi: 10.1109/ICECOS.2018.8605181.

Leevy, J. L. *et al*. (2018) 'A survey on addressing high-class imbalance in big data', *Journal of Big Data*. Springer International Publishing, 5(1). doi: 10.1186/s40537-018-0151-6.

Li, N. and Han, J. (2017) 'The application of naive bayes classifier in name disambiguation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10603 LNCS, pp. 611–618. doi: 10.1007/978-3-319-68542-7_52.

Lu, J. *et al*. (no date) 'Tutorial Proposal : Synergy of Database Techniques and Machine Learning Models for String Similarity Search and Join'.

Luque, A. *et al*. (2019) 'The impact of class imbalance in classification performance metrics based on the binary confusion matrix', *Pattern Recognition*, 91, pp. 216–231. doi: 10.1016/j.patcog.2019.02.023.

Milojević, S. (2013) 'Accuracy of simple, initials-based methods for author name disambiguation', *Journal of Informetrics*, 7(4), pp. 767–773. doi: 10.1016/j.joi.2013.06.006.

Mitra, P. *et al*. (2007) 'Are your citations clean?', *Communications of the ACM*, 50(12), pp. 33–38. doi: 10.1145/1323688.1323690.

Mozafari, F. and Tahayori, H. (2019) 'Emotion Detection by Using Similarity Techniques', *2019 7th Iranian Joint Congress on Fuzzy and Intelligent Systems, CFIS 2019*. IEEE, pp. 1–5. doi: 10.1109/CFIS.2019.8692152.

Müller, M. C., Reitz, F. and Roy, N. (2017) 'Data sets for author name disambiguation: an empirical analysis and a new resource', *Scientometrics*, 111(3), pp. 1467–1500. doi: 10.1007/s11192-017-2363-5.

Naway, A. and Li, Y. (2019) 'Android Malware Detection Using Autoencoder', pp. 1–9.

Nicholson, S. W. and Bennett, T. B. (2016) 'Dissemination and Discovery of Diverse Data: Do Libraries Promote Their Unique Research Data Collections?', *International Information and Library Review*, 48(2), pp. 85–93. doi: 10.1080/10572317.2016.1176448.

On, B. W., Lee, I. and Lee, D. (2012) 'Scalable clustering methods for the name disambiguation problem', *Knowledge and Information Systems*, 31(1), pp. 129–151. doi: 10.1007/s10115-011-0397-1.

Palfrey, J. (2016) 'Design choices for libraries in the digital-plus era', *Daedalus*, 145(1), pp. 79–86. doi: 10.1162/DAED_a_00367.

Qin, Y. and Lou, Y. (2019) 'Hydrological time series anomaly pattern detection based on isolation forest', *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*. IEEE, (Itnec), pp. 1706–1710. doi: 10.1109/ITNEC.2019.8729405.

Rettig, L. *et al*. (2015) 'Online anomaly detection over Big Data streams', *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 1113–1122. doi: 10.1109/BigData.2015.7363865.

Shah, N. B., Balakrishnan, S. and Wainwright, M. J. (2016) 'Feeling the bern: Adaptive estimators for Bernoulli probabilities of pairwise comparisons', *IEEE International Symposium on Information Theory - Proceedings*, 2016-Augus, pp. 1153–1157. doi: 10.1109/ISIT.2016.7541480.

Shen, Q. *et al*. (2017) 'NameClarifier: A Visual Analytics System for Author Name Disambiguation', *IEEE Transactions on Visualization and Computer Graphics*, 23(1), pp. 141–150. doi: 10.1109/TVCG.2016.2598465.

Song, M., Kim, E. H. J. and Kim, H. J. (2015) 'Exploring author name disambiguation on PubMed-scale', *Journal of Informetrics*. Elsevier Ltd, 9(4), pp. 924–941. doi: 10.1016/j.joi.2015.08.004.

Tran, H. N., Huynh, T. and Do, T. (2014) 'Author name disambiguation by using deep neural network', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8397 LNAI(PART 1), pp. 123–132. doi: 10.1007/978-3-319-05476-6_13.

Vluymans, S. (2019) 'Dealing with imbalanced and weakly labelled data in machine learning using fuzzy and rough set methods', *Studies in Computational Intelligence*, 807, pp. 1–249. doi: 10.1007/978-3-030-04663-7_1.

Wang, J.-P. *et al*. (2013) 'Effective string processing and matching for author disambiguation', 15, pp. 1–9. doi: 10.1145/2517288.2517295.

Weiss, A. (2016) 'Examining Massive Digital Libraries (MDLs) and Their Impact on Reference Services', *Reference Librarian*, 57(4), pp. 286–306. doi: 10.1080/02763877.2016.1145614.

Wressnegger, C. *et al*. (2013) 'A close look on n-grams in intrusion detection: Anomaly detection vs. classification', in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 67–76. doi: 10.1145/2517312.2517316.

Wu, H. *et al*. (2014) 'Unsupervised author disambiguation using Dempster–Shafer theory', *Scientometrics*, 101(3), pp. 1955–1972. doi: 10.1007/s11192-014-1283-x.

Yang, X. *et al*. (2019) 'A fast and efficient local outlier detection in data streams', *ACM International Conference Proceeding Series*, Part F1477, pp. 111–116. doi: 10.1145/3317640.3317653.

Yao, C. *et al*. (2019) 'Distribution Forest: An Anomaly Detection Method Based on Isolation Forest', in, pp. 135–147. doi: 10.1007/978-3-030-29611-7_11.

Zhang, T., Wang, E. and Zhang, D. (2019) 'Predicting failures in hard drivers based on isolation forest algorithm using sliding window', *Journal of Physics: Conference Series*, 1187(4). doi: 10.1088/1742-6596/1187/4/042084.

Zhao, J., Wang, P. and Huang, K. (2013) 'A semi-supervised approach for author disambiguation in KDD CUP 2013', in *Proceedings of the 2013 KDD Cup 2013 Workshop on - KDD Cup '13*. doi: 10.1145/2517288.2517298.

Zhu, Y. and Li, Q. (2013) 'Enhancing object distinction utilizing probabilistic topic model', *Proceedings - 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013*, pp. 177–182. doi: 10.1109/CLOUDCOM-ASIA.2013.61.

AUTHOR NAMES DISAMBIGUATION DALAM KLASIFIKASI AUTHOR MATCHING DENGAN MENGGUNAKAN METODE MACHINE LEARNING PADA PENDEKATAN ANOMALY DETECTION

ORIGINALITY REPORT

6%	1%	1%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sriwijaya University Student Paper	5%
2	Ijaz Hussain, Sohail Asghar. "A survey of author name disambiguation techniques: 2010–2016", The Knowledge Engineering Review, 2017 Publication	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		