IMPLEMENTASI *AUTOGLUON* DALAM EFISIENSI MODEL PREDIKTIF *MACHINE LEARNING* PADA *DATASET INTERNATIONAL BUSINESS MACHINES* (IBM) *HUMAN RESOURCE* (HR) *ANALYTICS EMPLOYEE ATTRITION*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata-1
Program Studi Teknik Industri - Fakultas Teknologi Industri
Universitas Islam Indonesia

Nama : Mohamad Ewo Muliyono Zees
No. Mahasiswa : 19522292

PROGRAM STUDI TEKNIK INDUSTRI PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2023

PERNYATAAN KEASLIAN

Saya mengakui bahwa tugas akhir ini adalah hasil karya saya sendiri kecuali kutipan dan ringkasan yang seluruhnya sudah saya jelaskan sumbernya. Jika dikemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yang sah maka saya bersedia ijazah yang telah saya terima ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, 19 – Oktober – 2023

(Mohamad Ewo Muliyono Zees)
NIM 19522292

LEMBAR PENGESAHAN PEMBIMBING

IMPLEMENTASI *AUTOGLUON* DALAM EFISIENSI MODEL PREDIKTIF *MACHINE LEARNING* PADA *DATASET INTERNATIONAL BUSINESS MACHINES* (IBM) *HUMAN RESOURCE* (HR) *ANALYTICS EMPLOYEE ATTRITION*

TUGAS AKHIR

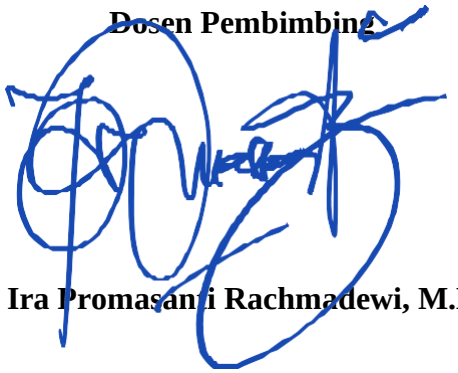
Disusun Oleh :

Nama : Mohamad Ewo Muliyono Zees

No. Mahasiswa : 19522292

Yogyakarta, 7 November 2023

Dosen Pembimbing :



(Ir. Ira Promasanti Rachmadewi, M.Eng.)

LEMBAR PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI *AUTOGLUON* DALAM EFISIENSI MODEL PREDIKTIF *MACHINE LEARNING* PADA DATASET *INTERNATIONAL BUSINESS MACHINES* (IBM) *HUMAN RESOURCE* (HR) *ANALYTICS EMPLOYEE ATTRITION*

TUGAS AKHIR

Disusun Oleh :

Nama : Mohamad Ewo Muliyono Zees

No. Mahasiswa : 19 522 292

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 Teknik Industri Fakultas Tekonologi Industri Universitas Islam Indonesia

Yogyakarta, *17 – November – 2023*

Tim Penguji

Ir. Ira Promasanti Rachmadewi, M.Eng. _____
Ketua

Ir. Ali Parkhan, M.T. _____
Anggota I

Yuli Agusti Rochman, S.T., M.Eng. _____
Anggota II

Mengetahui,
Ketua Program Studi Teknik Industri Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

Ir. Muhammad Ridwan Andi Purnomo, S. T., M. Sc., Ph.D., IPM.
NIK 015220101

HALAMAN PERSEMBAHAN

Alhamdulillah, atas izin Allah SWT penugasan ini dapat terselesaikan dengan baik. Laporan ini saya persembahkan kepada orang tua serta keluarga besar yang selalu memberikan dukungan dan motivasi. Laporan ini juga saya persembahkan untuk teman-teman seperjuangan dan civitas akademika teknik industri Universitas Islam Indonesia yang membantu dalam kemudahan dan kelancaran selama penugasan ini

MOTTO

"*Maka sesungguhnya beserta kesulitan ada kemudahan, sesungguhnya beserta kesulitan itu ada kemudahan*"
(Q.S Al Insyirah: 5-6)

"*Barang siapa menempuh suatu jalan untuk mencari ilmu, maka Allah memudahkannya mendapat jalan ke surga*"
(HR. Muslim)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur penulis panjatkan atas kehadiran Allah SWT, karena dengan rahmat, kuasa, karunia, dan hidayah-Nya penulis dapat menyelesaikan laporan ini. Shalawat serta salam penulis panjatkan kepada junjungan kita Nabi Muhammad SAW dan juga kepada para keluarga, sahabat, dan para pengikutnya hingga akhir zaman. Laporan Tugas Akhir/Skripsi yang berjudul "Implementasi *AutoGluon* Dalam Efisiensi Model Prediktif *Machine Learning* Pada *Dataset International Business Machines* (IBM) *Human Resource* (HR) *Analytics Employee Attrition*" dapat terlaksana dan terselesaikan dengan baik. Adapun Tugas Akhir ini disusun sebagai salah satu syarat utama untuk menyelesaikan studi Strata-1 pada Jurusan Teknik Industri, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Laporan Tugas Akhir ini kiranya tidak akan selesai tanpa bantuan dari beberapa pihak yang terus membantu dan mendorong penulis untuk menyelsaikannya. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., IPU., ASEAN, Eng selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Dr. Drs. Imam Djati Widodo, M.Eng, Sc selaku Ketua Jurusan Teknik Industri Fakultas Teknologi Universitas Islam Indonesia.
3. Bapak Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM selaku Ketua Program Studi Teknik Industri Program Sarjana Fakultas Teknologi Universitas Islam Indonesia
4. Ibu Ir. Ira Promasanti Rachmadewi, M.Eng. selaku Dosen Pembimbing Tugas Akhir penulis yang telah memberikan bimbingan kepada penulis selama masa pembuatan laporan Tugas Akhir
5. Bapak Dr. Qurtubi, S.T., M.T., selaku Dosen Pembimbing Akademik yang selalu memberikan dukungan dan bantuan kepada penulis selama menjalani masa aktif kuliah

6. Kedua orang tua penulis Yusuf H. Zees dan Marleni Asiari serta saudari penulis Dewi Sartika T. Zees dan Sri Indriyati Zees yang telah memberikan doa, semangat, motivasi dan kasih sayang yang diberikan kepada saya sehingga penyelsaian Tugas Akhir ini dapat terlaksana dengan lancar
7. Teman-teman seperjuangan Teknik Industri 2019 dan semua pihak yang turut membantu dalam penulisan Tugas Akhir

Semoga Allah SWT selalu memberikan kemudahan dan balasan setimpal atas bantuan yang telah diberikan kepada penulis. Penulis memohon maaf atas segala kekurangan di dalam laporan tugas akhir ini, semoga laporan ini dapat bermanfaat.

Wassalamualaikum Warahmatullahi Wabarakatuh

Yogyakarta, 7 – November – 2023

(Mohamad Ewo Muliyono Zees)

19522292

ABSTRAK

Perusahaan menginvestasikan banyak waktu dan sumber daya dalam perekrutan dan pelatihan karyawan, sesuai dengan kebutuhan strategis mereka. Karyawan merupakan representasi dari investasi yang nyata bagi organisasi di perusahaan. Ketika seorang karyawan meninggalkan perusahaan, organisasi tidak hanya kehilangan seorang karyawan yang berharga, tetapi juga sumber daya lain termasuk uang dan upaya serta waktu yang dihabiskan oleh bagian SDM perusahaan dalam merekrut, memilih serta melatih karyawan tersebut untuk melakukan pekerjaan yang spesifik. Dalam banyak kasus, keputusan terhadap pengurangan karyawan tidak hanya dari manajemen perusahaan, tetapi juga dari keputusan karyawan itu sendiri yang diakibatkan oleh banyak faktor salah satunya adalah tingkat kepuasan karyawan. Oleh karena itu, perusahaan harus dapat melakukan prediksi terhadap pengurangan karyawan (*employee attrition*) dengan mempertimbangkan banyak faktor. Salah satu pendekatan yang dapat dilakukan untuk mencapai tujuan ini adalah dengan menggunakan *machine learning* sebagai alat bantu analisis, khususnya *AutoGluon* yang merupakan salah satu teknologi *automated machine learning* yang dapat menghasilkan banyak model *machine learning* dalam satu waktu. Hasil yang diperoleh dari implementasi *AutoGluon* pada *dataset* IBM HR Data *Employee Attrition* bahwa model *WeightedEnsemble_L2* atau algoritma *ensemble* yang dipadukan dengan *regularization* L2 merupakan model terbaik dengan tingkat akurasi prediksi terhadap karyawan yang akan atrisi dengan skor 86%. Implementasi *AutoGluon* dalam melakukan *train* dan *test* data membutuhkan waktu yang lebih singkat sekitar 1 detik lebih cepat dibandingkan dengan pada model *machine learning* tradisional. Model yang dihasilkan dapat dijadikan sebagai merupakan usulan yang dapat dijadikan sebagai alternatif penyelesaian masalah oleh pihak manajemen dalam mengatasi permasalahan terkait *employee attrition*

Kata Kunci: *Attrition, AutoGluon, Machine Learning*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

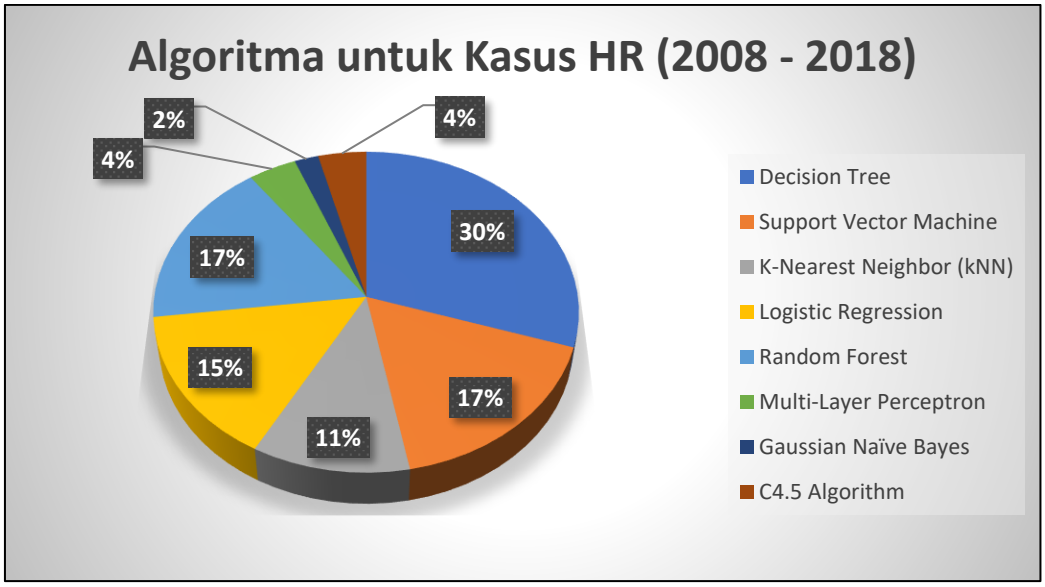
Dalam beberapa tahun terakhir, semua perusahaan tertarik pada analisis data terkait Sumber Daya Manusia (*human resource*) yang dianggap sebagai faktor utama yang mempengaruhi perkembangan perusahaan dan seluruh aktivitasnya di semua tingkat kebijakan sumber daya manusia (Berhil et al., 2019). Pada dasarnya, hal tersebut disebabkan oleh perusahaan yang ingin berusaha untuk memaksimalkan keuntungannya. Dalam memperoleh keuntungan yang maksimal tersebut, tentunya perusahaan perlu memperhatikan banyak hal, salah satunya adalah beban biaya yang terkait dengan sumber daya manusia. Perusahaan biasanya memberikan kontrak yang bersifat sementara, atau seasonal (bersifat musiman atau hanya pada periode waktu tertentu) pada karyawan yang melakukan tugas-tugas sederhana. Di sisi lain, karyawan yang mempunyai tugas yang lebih spesifik, atau membutuhkan suatu spesialisasi di bidang tertentu dan dengan jangka waktu yang relatif berkesinambungan merupakan aset yang penting bagi perusahaan. (Fallucchi et al., 2020)

Perusahaan menginvestasikan banyak waktu dan sumber daya dalam perekrutan dan pelatihan karyawan, sesuai dengan kebutuhan strategis mereka (Boushey, H., 2012). Oleh karena itu, karyawan merupakan representasi dari investasi yang nyata bagi organisasi di perusahaan. Ketika seorang karyawan meninggalkan perusahaan, organisasi tidak hanya kehilangan seorang karyawan yang berharga, tetapi juga sumber daya lain termasuk uang dan upaya serta waktu yang dihabiskan oleh bagian SDM perusahaan dalam merekrut, memilih serta melatih karyawan tersebut untuk melakukan pekerjaan yang spesifik. Konsekuensinya, organisasi harus terus berinvestasi dalam perekrutan, pelatihan dan pengembangan staf baru untuk mengisi posisi pekerjaan yang kosong. Melatih karyawan baru adalah proses yang panjang dan mahal. Namun, hal tersebut merupakan hal yang penting bagi perusahaan. Selain itu, mengendalikan dan mengurangi tingkat pengurangan karyawan (*employee attrition*) juga penting bagi perusahaan.

Pengurangan karyawan didefinisikan sebagai karyawan yang mengundurkan diri atau pensiun dari suatu perusahaan. Disisi lain, karyawan yang mempunyai motivasi dan tingkat kepuasan tinggi, serta loyal merupakan inti dari sebuah perusahaan. Hal tersebut berdampak pada produktivitas organisasi, karena karyawan seperti itu, cenderung lebih kreatif, produktif, dan berkinerja lebih baik, yang pada akhirnya menghasilkan dan dapat membantu peningkatan kinerja perusahaan (Martin, L., 2018).

Oleh karena itu, faktor ketidakpuasan kerja atau data terkait kepuasan kerja dalam banyak literatur ekonomi dijadikan sebagai prediktor yang kuat dalam melakukan prediksi terhadap karyawan dalam hal perpisahan maupun pengunduran diri, mengendalikan upah, dan jam kerja. Dalam hal ini, penerapan kecerdasan buatan (*artificial intelligence*/AI) di bidang SDM memungkinkan perusahaan untuk mengubah data menjadi pengetahuan dengan menerapkan model prediktif. Model tersebut memungkinkan prediksi terhadap karyawan menggunakan data yang dikumpulkan oleh perusahaan selama tahun-tahun sebelumnya, sehingga mengurangi masalah kritis dan mengoptimalkan seluruh aktivitas SDM (Mishra, S, et al., 2016).

Dalam artikel *paper review* yang dilakukan oleh (Fallucchi et al., 2020). Terdapat beberapa solusi IT yang telah diusulkan untuk memecahkan berbagai masalah terkait sumber daya manusia. Banyak solusi AI telah diterapkan, menggunakan metode dan algoritma yang berbeda. Algoritma AI yang paling terkenal seperti *Decision tree, Random Forest, Support Vector Machine, Multi-Layer Perceptron, K-Nearest Neighbor (KNN), Gaussian Naïve Baye, Logistic Regression, algoritma C4.5*. Persentase dari penggunaan algoritma tersebut adalah sebagai berikut.



Gambar 1. 1 Algoritma Yang Sering Digunakan Untuk Kasus HR

Algoritma diatas yang merupakan solusi AI terhadap masalah HR juga merupakan bagian dari *machine learning*. Dalam penerapan algoritma tersebut, tentunya ada beberapa hal yang menjadi pertimbangan, seperti kompleksitas dari masalah dan solusi yang akan coba diberikan. Namun, dalam penerapan setiap algoritma dalam kasus HR tersebut juga masih terdapat beberapa kekurangan. Seperti *decision tree* dan *random forest* yang rawan terhadap *overfitting* dan biaya komputasi yang tinggi, SVM yang tidak mampu menangani kumpulan data berdimensi tinggi, ataupun *logistic regression* yang memerlukan asumsi linearitas antara variabel independen dan dependen (Al Akasheh et al., 2023)

Selain itu, menurut (Al Akasheh et al., 2023) juga algoritma tingkat lanjut (*advance algorithm*) seperti *Neural network*, *Gradient Boosting* dan *Ensemble* juga masih memiliki kelemahan. *Neural network* membutuhkan biaya komputasi yang tinggi, *Gradient Boosting* masih rawan terhadap *overfitting* dan sensitif terhadap *noice* serta *ensemble* yang kurang terhadap kemampuan interpretasi. Disisi lain, tingkat pemahaman atau kepakaran dari seseorang yang membangun model *machine learning* dengan menggunakan kombinasi dari beberapa algoritma sangat menentukan hasil atau akurasi dari model tersebut. Sehingga, dalam mengimplementasikan algoritma *machine learning* secara tradisional (umum) terdapat dua permasalahan besar yaitu dari akurasi model yang akan dibangun dan dari tingkat ekspertis dari *user* atau pengguna yang akan membangun model *machine learning* tersebut. Lebih lanjut lagi, data yang digunakan sangat vital dalam membangun model *machine learning*.

Dalam kasus seperti pengurangan karyawan, tidak jarang data yang digunakan mempunyai banyak variabel, sehingga implementasi dari setiap algoritma dijalankan atau di eksekusi dengan cara yang berbeda. Salah satu sampel *dataset* yang dapat merepresentasikan permasalahan ini yaitu terdapat pada *dataset*s IBM HR *Analytics Employee Attrition*, yang merupakan *dataset* fiksional dengan input 35 variabel dan 1.470 data yang diterbitkan oleh perusahaan IBM dan tersedia pada *platform kaggle.com*. *Dataset* tersebut telah banyak digunakan sebagai referensi dari penelitian terkait HR *employee attrition*, yang mana dalam *platform kaggle.com* banyak *user* telah mencoba untuk mempublikasikan hasil kerja atau analisisnya dengan menggunakan beragam algoritma, metode, bahkan bahasa pemrograman yang berbeda-beda. Namun, mayoritas dari *user kaggle.com* menggunakan pendekatan *machine learning* dengan bahasa pemrograman *python* dalam menyelesaikan permasalahan tersebut. Pendekatan *machine learning* dengan algoritma yang berbeda-beda tentunya memperoleh *output* yang berbeda serta tingkat akurasi prediksi yang berbeda.

Oleh karena itu, dibutuhkan suatu teknologi atau *framework automated machine learning* (*AutoML*) yang dapat melakukan *deploy* model prediksi melalui algoritma yang beragam dalam satu waktu. *AutoGluon Tabular*, yang merupakan salah satu *framework AutoML opensource* yang hanya memerlukan satu baris *Python* untuk melatih model pembelajaran mesin yang sangat akurat pada kumpulan data *tabular* yang belum diproses seperti file CSV dapat menjadi solusi untuk permasalahan ini. *Framework* ini jika dibandingkan dengan *framework* lain, dapat menghasilkan prediksi yang akurat dengan waktu yang relatif singkat, sebagai pada tabel berikut

Tabel 1. 1 *Benchmarking Framework AutoML*

Framework	***Wins***	***Losses***	***Failures***	***Champion***	***Avg. Rank***	***Avg. Rescaled Loss***	***Avg. Time (min)***
AutoGluon	-	-	1	23	1,8438	0.1385	201
H2O AutoML	4	26	8	2	3,1250	0.2447	220
TPOT	6	27	5	5	3,3750	0.2034	235
GCP-Tables	5	20	14	4	3,7500	0.3336	195
auto-sklearn	6	27	6	3	3,8125	0.3197	240
Auto-WEKA	4	28	6	1	5,0938	0.8001	244

AutoGluon sebagai *framework AutoML* juga menawarkan beberapa hal *fundamental* dalam pengembangannya seperti Simplicity dimana *user* dapat melatih model pada data mentah secara langsung tanpa mengetahui detail tentang data dan model *machine learning*. Oleh karena itu, peneltian "Implementasi *AutoGluon* Dalam Efisiensi Penerapan Model Prediktif *Machine Learning* Pada *Dataset International Business Machine*s (IBM) *Human Resource* (HR) *Analytics Employee Attrition*" mempunyai peranan penting tidak hanya dari segi akademis, tetapi juga untuk menyelesaikan permasalahan HR *employee attrition* serta menjawab kebutuhan industri di masa kini maupun mendatang.

1.2 Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini berdasarkan uraian latar belakang diatas adalah sebagai berikut:

1. Berapa hasil tingkat akurasi prediksi yang dihasilkan oleh model *machine learning* melalui implementasi *AutoGluon* pada *dataset* IBM HR *Data Employee Attrition*?
2. Apakah waktu traning dan *test* model *machine learning* dengan menggunakan *AutoGluon* lebih efisien jika dibandingkan dengan *machine learning* tradisional?
3. Apakah algoritma terbaik yang dihasilkan dari *AutoGluon* dapat digunakan untuk membantu perusahaan dalam pengambilan keputusan pada kasus *Employee Attrition*?

1.3 Tujuan Penelitian

Adapun tujuan dalam penelitian ini berdsarkan rumusan masalah diatas adalah sebagai berikut:

1. Memperoleh hasil tingkat akurasi prediksi dari model *machine learning* melalui implementasi *AutoGluon* pada *dataset* IBM HR *Data Employee Attrition*

2. Memperoleh hasil perbandingan waktu *training* dan *test* model *machine learning* antara model *Machine learning* tradisional dengan *AutoGluon*
3. Mengetahui apakah algoritma terbaik yang dihasilkan dari *AutoGluon* dapat digunakan untuk membantu perusahaan dalam pengambilan keputusan pada kasus *Employee Attrition*

1.4 Manfaat Penelitian

Manfaat pada penelitian ini adalah sebagai berikut:

1. Mengetahui cara implementasi *AutoGluon* dalam analisis *dataset* IBM HR *Data Employee Attrition*
2. Hasil model *Machine learning* dengan menggunakan *AutoGluon* dapat dijadikan sebagai alternatif dalam menganalisis *dataset* IBM HR *Data Employee Attrition* untuk penelitian kedepannya
3. Memudahkan pihak yang memiliki kepentingan untuk memperoleh informasi terkait penerapan *AutoGluon* sebagai alat bantu analisis pada kasus HR.

1.5 Batasan Penelitian

Pada penelitian ini terdapat beberapa batasan yakni sebagai berikut:

1. Data yang digunakan merupakan data yang diperoleh melalui website kaggle.com pada tautan https://www.kaggle.com/pavansubhasht/ibmhr-*analytics-attrition-dataset*
2. Penelitian ini menggunakan alat bantu berupa software *google colab* dengan bahasa pemrograman *Python* dan *microsoft excel*

BAB II

TINJAUAN PUSTAKA

2.1 Kajian Literatur

Penelitian dengan judul "*AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data*" yang dilakukan oleh (Erickson et al., 2020), yang membahas terkait komparasi performa antara *AutoGluon* dengan teknologi *automated machine learning (AutoML*) lainnya. Dalam penelitian ini, teknologi *AutoGluon-Tabular* digunakan untuk melatih *model machine learning* pada data *tabular* yang belum diproses seperti file CSV. Pengujian model tersebut dilakukan pada serangkaian 50 tugas klasifikasi dan regresi dari Kaggle dan OpenML *AutoML Benchmark.* Hasil yang diperoleh dari pengujian ini mengungkapkan bahwa *AutoGluon* lebih cepat, lebih tangguh, dan jauh lebih akurat dibandingkan *framework AutoML* lainnya seperti *TPOT, H2O, AutoWEKA, auto sklearn,* dan *Google AutoML Tables.* Selain itu, cara kerja *AutoGluon* tidak seperti beberapa *framework AutoML* yang umumnya berfokus pada pemilihan model/hiperparameter, *AutoGluon-Tabular* berhasil dengan menggabungkan beberapa model dan menumpuknya dalam beberapa lapisan.

Pada penelitian yang dilakukan oleh (Felix et al., 2020) yang berjudul "*A proposed Model for Predicting Employees' Performance Using Data Mining Techniques: Egyptian Case Study*" , dimana pendekatan dari implementasi data mining yang digunakan adalah *Naïve bayes, support vector machine*, dan *Decision tree*. Hasil dalam penelitian diperoleh melalui WEKA (*Waikato Environment for Knowledge Analysis*) dengan 10-*fold cross* yaitu dengan tingkat akurasi 70% konvergen dan sedang. Dalam penelitian ini juga disarankan untuk menggunakan teknik klasifikasi lainnya seperti *Neural network* (NN), logika *fuzzy* dan banyak lainnya juga harus diujicobakan untuk memvalidasi serta membantu meningkatkan akurasi dalam pemilihan model.

Penelitian dengan judul "*Predicting Yarn Breaks in Textile Fabrics: A Machine Learning Approach*" yang dilakukan oleh (Azevedo et al., 2022) yang membahas tentang penyebab terjadinya *downtime delay* yang diakibatkan oleh kerusakaan kain/tekstil pada saat proses produksi. Dalam menyelesaikan masalah tersebut, pendekatan yang dilakukan adalah dengan implementasi *machine learning* untuk melakukan prediksi terhadap jadwal penghentian mesin sebagai tindakan pencegahan untuk terjadinya kerusakan pada saat produksi kain/tekstil. Implementasi *machine learning* yang dimaksud yaitu dengan menggunakan teknologi *Automated machine learning* (*AutoML*) seperti H2O, *AutoKeras, AutoGluon* sebagai pendekatan dalam menyelesaikan masalah prediksi penghentian mesin jahit/tenun.

Penelitian dengan judul "*Comparison of traditional and automated machine learning approaches in predicting the compressive strength of graphene oxide/cement composites*" yang dilakukan oleh (J. Yang et al., 2023), yang menggunakan pendekatan *machine learning* sebagai percobaan untuk mencari hubungan kompleks antara *compressive strength* (CS) dari *graphene oxide reinforced cement composites* (GORCCs) terhadap beberapa faktor penggandengan (*multiple coupling factors*). Data yang digunakan untuk menguji model *machine learning* pada penelitian ini berasal dari 260 data hasil eksperimen. Hasil yang diperoleh dari membandingkan antara implementasi model *machine learning* tradisional menggunakan *support vector machine* (SVM), *Random Forest*, *Artificial Neural network* (ANN) dengan *AutoGluon-Tabular* (AGT) sebagai *automated machine learning* (*AutoML*) menunjukkan bahwa tingkat kepercayaan tertinggi dalam hasil prediksi diperoleh dengan AGT. Selain itu, AGT juga memberikan interpretasi hasil yang lebih baik, serta lebih efisien dibandingkan alur kerja ML tradisional, menghindari proses penyetelan *hyperparameter* yang memakan waktu.

Penelitian yang dilakukan oleh (Raj et al., 2023), dengan judul "*StrokeViT with AutoML for brain stroke classification*" yang membahas terkait usulan dalam diagnosa stroke dengan melakukan integrasi antara *Convolutional Neural network* (CNN), *Vision Transformers* (ViT), dan *AutoML*. Pada penelitian ini juga diusulkan untuk melakukan tahapan ekstraksi stroke-spesifik dari setiap prediksi *slice-wise* untuk mendapatkan prediksi terhadap kondisi pasien menggunakan *AutoML* sebagai alternatif menerapkan metode klasifikasi stroke yang didasarkan pada mekanisme prediksi tingkat *single-slice*. Hasil yang diperoleh dari implementasi *AutoML* dalam prediksi kondisi pasien dengan mempertimbangkan 13 algoritme ML yang berbeda, diperoleh 3 di antaranya mencapai akurasi lebih dari 90%.

Penelitian dengan judul "*Application of Data Mining Classification in Employee Performance Prediction*" yang dilakukan oleh (Kirimi & Moturi, 2016) dengan implementasi *data mining* serta CRISP-DM (*Cross Industry Standard Process for Data Mining*) yang di aplikasikan sebagai alat bantu untuk *predictive analysis.* Dari penelitian ini diperoleh hasil bahwa kinerja dari seorang karyawan sangat dipengaruhi oleh faktor pengalaman, pelatihan profesional, usia, kualifikasi akademik, status perkawinan, jenis kelamin dan nilai penilaian kinerja sebelumnya. Dalam penelitian ini model prediksi yang diusulkan adalah dengan menggunakan *decision tree* dengan *cross fold validation.*

Penelitian yang dilakukan oleh (Zhang et al., 2022) dengan judul "*A Tool for Non-Coding RNA Identification in Plants Based on an Automated Machine Learning Framework*" yang melakukan implementasi *AutoGluon* dalam *train* suatu model yang bertujuan untuk mengidentifikasi *non-coding* RNA (ncRNA) dari *dataset* yang terkonstruksi untuk empat species tumbuhan. Identifikasi terhadap ncRNA yang dimaksud adalah dimulai dengan menentukan tingkat akurasi dalam menentukan fungsi dari RNA. Dalam hal ini, sudah banyak *tool machine learning* yang dikembangkan untuk melakukan identifikasi terhadap ncRNA namun, tidak spesifik untuk melakukan identifikasi hal tersebut pada tumbuhan. Hal inilah yang mendasari implementasi dari teknologi *AutoGluon* dalam menciptakan suatu alat yang diberi nama PINC yang dapat membantu dalam identifikasi terhadap ncRNA pada tumbuhan secara berurutan. Tahapan yang dilakukan dalam melakukan identifikasi tersebut diawali dengan melakukan ekstraksi terhadap 91 *feature* yang dikombinasikan dengan F-*test* dan *variance* hingga menghasilkan hanya 10 *feature*. Hasil yang diperoleh dari penelitian ini, bahwa validasi independen pada sembilan *dataset* pengujian, akurasi PINC berkisar antara 92,74% hingga 96,42%.

Penelitian dengan judul "*Predictive model of employee attrition based on stacking ensemble learning*" yang dilakukan oleh (Chung et al., 2023) dengan tujuan untuk mengusulkan suatu model untuk prediksi pengurangan karyawan (*employee attrition*). Dalam penelitian ini, model prediksi dibangun dengan menggunakan 30 *variable* data dari *dataset IBM HR Analytics Employee Attrition & Performance* dengan entri sebanyak 1470 baris. Terdapat delapan model prediktif yang dibangun dalam penelitian ini, termasuk Regresi Logistik, *Random Forest*, *XGBoost*, SVM, model Jaringan Syaraf Tiruan, serta *ensembled model*. Model tersebut dibangun, dan kinerjanya dievaluasi seperti, dampak variabel terhadap pengurangan karyawan, variabel kepuasan lingkungan, dan kerja lembur. Hasil yang diperoleh menyatakan bahwa

faktor ‘Kepuasan Lingkungan’ menunjukkan kontribusi tertinggi dengan persentase (4,2 %) terhadap model prediksi yang dibangun. Selain itu, faktor lain seperti ‘Lembur’ juga menghasilkan persentase yang cukup tinggi (4,2 %) dan ‘Kepuasan Hubungan’ (3,9 %) juga diprediksi sebagai variabel utama dalam hal pengurangan karyawan. Hal ini menunjukkan bahwa kepuasan psikologis karyawan seperti kepuasan lingkungan dan kepuasan hubungan harus dimasukkan untuk memprediksi pengurangan karyawan.

Penelitian dengan judul “*Off-targetP ML: an open source machine learning framework for off-target panel safety assessment of small molecules*” yang dilakukan oleh (Naga et al., 2022), dengan tujuan untuk membantu ahli kimia dalam proses perancangan obat sebelum disintesis dan mempercepat penemuan obat. Teknik yang akan digunakan dalam menyelesaikan masalah ini adalah dengan menerapkan *machine learning* dan *deep learning* dalam membangun suatu model untuk melakukan prediksi terhadap bioaktivitas, seperti ketidakseimbangan data, pengukuran duplikat *inter-target*, dan duplikat pengidentifikasi senyawa publik. Objek dalam model tersebut adalah lima puluh kelas protein yang berbeda. Dalam pengembangan model yang dirancang, dilakukan tahapan eksplorasi untuk membandingkan kemampuan *Neural networks* dan *AutoML* dalam menghasilkan model prediksi. Hasil yang diperoleh dari pembangunan model dengan eksplorasi terhadap implementasi *Neural network* bahwa model prediksi yang dihasilkan dapat mengungguli tiga metode *AutoML* (*AutoGluon*, *H2O*, dan *Auto-Sklearn*), namun dengan performa yang serupa untuk *off-target*. Selain itu, hal ini dapat memungkinkan menjadi suatu memberikan solusi alternatif, lebih praktis, dan *user-friendly* untuk pembuatan model manual di masa depan.

Penelitian dengan judul “*Investigation of early career teacher attrition and the impact of induction programs in Western Australia*” oleh (Wyatt & O’Neill, 2021) yang membahas terkait keterbatasan akan pengetahuan tentang ECT (*Early Career Teacher*) di Australia. Batasan dari penelitian ini adalah hanya ECT yang bekerja di Departemen Pendidikan Australia Barat dari tahun 2004 – 2018. Hasil yang diperoleh dari penelitian ini mengungkapkan bahwa, mereka yang berkarir di level pendidikan tingkat menengah mempunyai kemungkinan besar untuk mengalami atrisi (pengurangan jumlah guru) atau ECT baik dengan alasan mengundurkan diri ataupun alasan lainnya. Namun, hal ini belum bisa memastikan sepenuhnya bahwa pada setiap kasus pengurangan jumlah guru atau karyawan disebabkan oleh pengunduran diri dari guru atau karyawan itu sendiri.

Penelitian yang dilakukan oleh, (Siahaan, 2021) yang berjudul "*An Analysis of Contract Employee Performance Assessment Using Machine Learning*" dengan menggunakan pendekatan *algoritma machine learning* seperti *K-Nearest Neighbors, Random Forest, Decision tree, Naïve Bayes*, dan *Logistic Regression*. Dalam penelitian ini, pendekatan dengan menggunakan lebih dari satu algoritma dilakukan untuk mencari algoritma yang tepat untuk aplikasi assesment kinerja karyawan kontrak dengan nilai input *attidude, skill, responsibility, absent* dan membandingkan nilai akurasi, *recall*, presisi, dan F1 *score*. Hasil yang diperoleh dari penelitian ini bahwa model algoritma *Random Forest* merupakan model yang terbaik. prediksi *assesment* kinerja karyawan kontrak dengan menggunakan model algoritma ini, nilai akurasinya mencapai 90.62 %, dengan tingkat presisi 72.22 %, dan *recall* 75%.

Penelitian dengan judul "Prediksi Kinerja Pegawai sebagai Rekomendasi Kenaikan Golongan dengan Metode *Decision tree* dan Regresi Logistik" yang dilakukan oleh (Anggara et al., 2022) menjelaskan bahwa peningkatan atau penurunan kinerja karyawan berpengaruh terhadap kenaikan gaji, promosi dan atau penurunan jabatan. Dalam hal ini, penelitian yang diangkat terkait promosi karyawan pada prasama bhakti *foundation* yang masih dilakukan dengan pendekatan manual, yang mana hal ini menyebabkan terjadinya pengambilan keputusan yang inkonsisten dari sisi manajemen. Penerapan *machine learning* dengan metode klasifikasi untuk melakukan prediksi terhadap karyawan yang layak untuk promosi di aplikasikan dalam menyelesaikan masalah ini. Metode klasifikasi yang digunakan di implementasikan melalui algoritma *decision tree* dan *logistic regression*. Hasil yang diperoleh dari model prediksi tersebut menyatakan bahwa variabel yang paling berpengaruh terhadap promosi karyawan adalah "LamaKerja".

Penelitian dengan judul "*Employee Attrition Prediction Using Logistic Regression Feature Selection*" oleh (Wardhani & Lhaksmana, 2022) yang bertujuan untuk melakukan prediksi terhadap pengurangan karyawan (*employee attrition*) pada suatu perusahaan dengan menggunakan metode regresi logistik. Pendekatan *machine learning* digunakan dalam melakukan prediksi tersebut, karena pendekatan ini bersifat tidak bias atau terdapat campur tangan manusia. Dalam penelitian ini juga diusulkan untuk menerapkan metode *feature selection* untuk mengidentifikasi faktor-faktor yang berpengaruh dalam model prediksi dan melakukan tahap menyederhanakan *data training*. *Feature selection* yang digunakan meliputi *information gain*, pemilihan *k-best*, dan *recursive feature elimination* (RFE). Pada penelitian ini juga dilakukan *10-fold-cross-validation* sebagai metode evaluasi. Hasil yang diperoleh dari

implementasi model prediksi untuk *employee attrition* menggunakan metode regresi logistik tanpa menerapkan *feature selection* mendapatkan nilai akurasi sebesar 0,865 dan skor AUC sebesar 0,932. Namun, dengan penerapan fitur *feature selection* RFE, diperoleh hasil evaluasi tertinggi dengan nilai akurasi sebesar 0,853 dan skor AUC sebesar 0,92 yang mana nilai akurasi tersebut adalah nilai tertinggi diantara *feature selection* lain seperti *information gain* dan *select k-best*.

Selain itu, peneltian yang dilakukan oleh (Effendi et al., 2023) terkait "Prediksi Guru Kemungkinan Tetap Bekerja di Sekolah Al Uswah Surabaya Menggunakan *Machine Learning*" yang bertujuan untuk melakukan prediksi terhadap potensi apakah guru di Sekolah Al Uswah Surabaya tetap bekerja atau tidak. Penelitian ini juga bertujuan untuk meminimalisir tingkat *turnover* baik pada sekolah Al Uswah Surabaya itu sendiri maupun pada tempat lainnya untuk penelitian yang lebih lanjut. Implementasi *machine learning* melalui metode klasifikasi dan algoritma terkait, seperti *decision tree* dan *logistic regression* merupakan pendekatan yang dilakukan dalam membangun model prediksi pada penelitian ini. Hasil yang diperoleh berdasarkan tabel hasil prediksi maupun *confusion matrix*, bahwa algoritma *Logistic Regression* memperoleh nilai performa paling tinggi. Tingkat *Presicion* mencapai 80,8%, untuk data *testing* dengan *split data* 80:20 yang mana nilai ini lebih tinggi dibanding tiga algoritma lainnya yang menghasilkan nilai di bawah 80%.

Penelitian dengan judul "Sistem Prediksi Awal Terhadap Atrisi Karyawan Menggunakan Algoritma C4.5" oleh (Lamramot et al., 2022) menyatakan bahwa karyawan dengan bakat dan talenta yang meninggalkan suatu perusahaan dapat menjadi masalah yang krusial bagi perusahaan itu sendiri. Hal tersebut yang membuat penelitian ini dilakukan dengan tujuan untuk merancang sistem prediksi awal terhadap atrisi karyawan. Metode prediksi yang digunakan dalam penelitian ini yaitu metode *decision tree* dengan algoritma C4.5. Total 1010 sampel data karyawan diperoleh dari perusahaan PT. Indorama Petrochemicals digunakan dalam penelitian ini sebagai populasi. Hasil yang diperoleh menunjukan bahwa tingkat akurasi model prediksi sebesar 92,9% dengan algoritma C4.5 sehingga mendapatkan hasil yang baik pada prediksi atrisi karyawan tersebut.

Tabel 2. 1 Penelitian terdahulu

No	Penulis	Kata Kunci											Kelemahan
		1	2	3	4	5	6	7	8	9	10	11	
1	(Erickson et al., 2020)	✓											-
2	(Felix et al., 2020)		✓		✓						✓		Hasil akurasi yang diperoleh pada penelitian ini masih relatif rendah, yaitu diangka 70%, dan data yang digunakan relatif kurang banyak serta dibutuhkan implementasi algoritma yang kompatibel dengan masalah klasifikasi seperti *neural network* untuk menghasilkan akurasi model yang lebih tinggi
3	(Azevedo et al., 2022)	✓											-
4	(J. Yang et al., 2023)	✓											-
5	(Raj et al., 2023)	✓											-
6	(Kirimi & Moturi, 2016)		✓										Hasil yang diperoleh dari penelitian ini adalah faktor terpenting yang mempengaruhi model prediksi menggunakan *decision tree* adalah pengalaman karyawan dan 6 *variable* lainnya yang berkaitan dengan performa karyawan, namun kedepannya dibutuhkan variabel yang lebih banyak lagi serta alasan yang kuat dalam pemilihan algoritma
7	(Zhang et al., 2022)	✓											-
8	(Chung et al., 2023)				✓	✓			✓	✓			Model yang disajikan dalam penelitian ini tidak dapat menjelaskan hubungan spesifik antar variabel. Perlu dilakukan analisis secara rinci pengaruh variabel utama terhadap variabel terikat (dependent).
9	(Naga et al., 2022)	✓											-

No	Penulis	Kata Kunci											Kelemahan
		1	2	3	4	5	6	7	8	9	10	11	
10	(Wyatt & O'Neill, 2021)											✓	Penelitian ini menggunakan alat bantu berupa SPSS 25 yang merupakan software berbayar. Pendekatan atau teknik yang digunakan adalah *logistic regression*, pearson corellation, dan beberapa teknik statistika deskriptif lainnya. Dimana hal ini dapat juga dilakukan didalam *Python* yang merupakan software *open-source*
11	(Siahaan, 2021)		✓	✓			✓	✓					Pada penelitian ini diterapkan lebih dari 1 algoritma *machine learning* secara manual/konvensional. Sementara hasil akurasi yang diperoleh ternyata pada salah satu algoritma (kNN) hasil akurasinya sangat rendah.
12	(Anggara et al., 2022)		✓	✓									Hasil yang diperoleh pada penelitian ini bahwa model prediksi yang dihasilkan oleh *decision tree* mempunyai akurasi yang lebih baik dibandingkan dengan regresi logistik. Disisi lain implementasi dari kedua algoritma dilakukan secara manual, yang artinya jika salah satu algoritma yang digunakan mempunyai akurasi yang lebih baik dibandingkan dengan lainnya, maka perlu terdapat kemungkinan bahwa masih ada algoritma yang dapat menghasilkan model prediksi yang lebih akurat
13	(Wardhani & Lhaksmana, 2022)			✓									Pada penelitian ini, akurasi model yang dihasilkan tidak hanya dengan mengimplementasikan *logistic regression*, namun ditambah dengan teknik *feature* selection
14	(Effendi et al., 2023)			✓									Prediksi yang dihasilkan pada penelitian ini memiliki tingkat akurasi yang sangat tinggi yaitu

No	Penulis	Kata Kunci											Kelemahan
		1	2	3	4	5	6	7	8	9	10	11	
													94,6%. Artinya bahwa impelementasi *decision tree* tipe C 4.5 mempunyai performa yang sangat baik, namun disisi lain hal ini dapat berpotensi *overfitting*.
15	(Lamramot et al., 2022)		✓										Dalam penelitian ini, model yang dihasilkan belum dapat menganalisa pengaruh tiap atribut terhadap prediksi yang dihasilkan (potensi guru akan bekerja di sekolah lebih lama atau sebaliknya).
16	(Zees, 2023)	✓											-

Keterangan :

1. *AutoGluon/AutoML*
2. *Decision tree*
3. *Logistic Regression*
4. *SVM*
5. *Neural network*
6. *kNN*
7. *Random Forest*
8. *Gradient Boosting*
9. *Ensemble Learning*
10. *Naïve Bayes*
11. *Non-ML Method*

2.2 Landasan Teori

Dalam landasan teori ini, memuat tentang istilah, teori atau formula yang *Automated Machine Learning* (*AutoML*), *Dataset Employee Attrition*, dan hal-hal lainnya yang menyangkut terkait *machine learning*

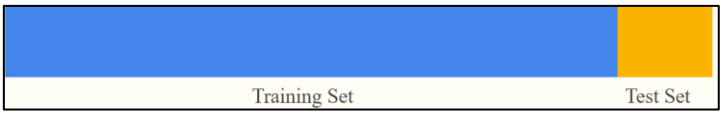
2.2.1 *Employee Attrition*

Menurut (Alduayj & Rajpoot, 2018) dalam (Chung et al., 2023) *Employee Attrition* atau pengurangan karyawan dianggap sebagai masalah utama di semua organisasi karena berdampak buruk terhadap kinerja organisasi. Ketika karyawan keluar, alur pekerjaan yang sedang berjalan terganggu dan menghabiskan alokasi waktu serta sumber daya tambahan dalam merekrut dan mendidik karyawan baru (Yedida et al., 2018). Hal ini dapat menyebabkan potensi risiko kebocoran data atau teknologi inti akibat (pada beberapa kasus di perusahaan teknologi). Menurut (S. Yang & Islam, 2020) pengurangan karyawan perlu diminimalkan agar organisasi dapat melanjutkan aktivitas bisnisnya dan mendapatkan keunggulan kompetitif yang tinggi. Dalam mencapai tujuan tersebut, penting bagi para pemimpin bisnis untuk mengidentifikasi pengurangan karyawan secara tepat waktu sehingga tindakan yang relevan dapat segera diambil untuk meningkatkan produktivitas perusahaan, alur kerja secara keseluruhan, dan kinerja bisnis.

2.2.2 *Dataset*

Menurut (Snijders et al., 2013) data set (atau *dataset*) adalah kumpulan dari suatu data. Dalam konteks data *tabular*, kumpulan data ini berhubungan dengan satu atau lebih tabel *database*, di mana setiap kolom tabel mewakili variabel tertentu, dan setiap baris berhubungan dengan catatan tertentu dari *dataset* tersebut. *Dataset* mencantumkan nilai untuk setiap variabel, misalnya tinggi dan berat suatu benda, untuk setiap anggota *dataset*. Selain itu, *dataset* juga dapat terdiri dari kumpulan dokumen atau file.

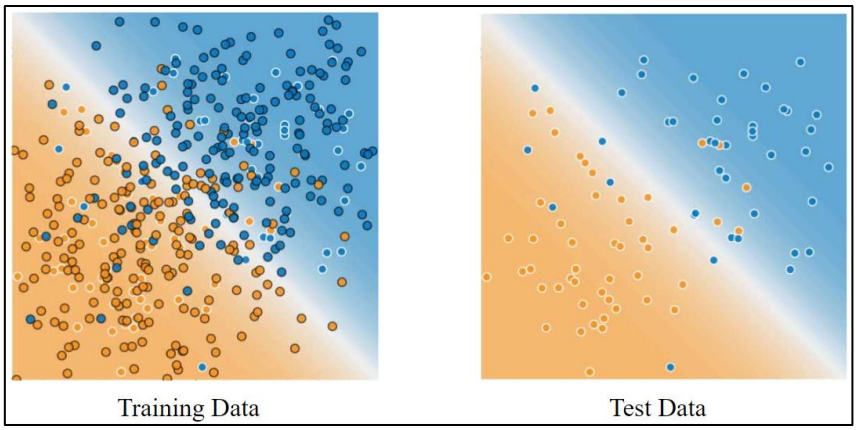
Dalam penerapan model *machine learning*, *dataset* terbagi atas 2 bagian yaitu *training* set dan *test* set. *Training set* adalah *subset* untuk melatih model, sementara *test set* adalah *subset* untuk menguji model yang terlatih (developers.*google*.com, 2022). Secara grafik hal ini dapat dilihat pada gambar berikut.



Gambar 2. 1 *Splitting* Data

Dalam pembagian (*splitting*) data tersebut perlu diperhatikan 2 hal sebagai berikut.

1. Cukup besar untuk memberikan hasil yang bermakna secara statistik.
2. Dapat merepresentasikan atau mewakili kumpulan data secara keseluruhan.



Gambar 2. 2 *Train* and *Test* Data

Hal lainnya yang perlu diperhatikan terkait *train* dan *test* data ini menurut (developers.*google*.com, 2022) adalah tidak melatih *test* data. Hal ini dapat menyebabkan hasil yang sangat bagus pada metrik evaluasi, namun hal ini kemungkinan disebabkan oleh *training* terhadap *test* data. Selain itu, dalam kasus memprediksi apakah suatu email adalah *spam*, menggunakan baris subjek, isi *email*, dan alamat email pengirim sebagai fitur, yang mana dilakukan pembagian data ke dalam set pelatihan dan pengujian, dengan pembagian 80 20 dan model yang dihasilkan mencapai presisi 99% pada set pelatihan dan set pengujian.

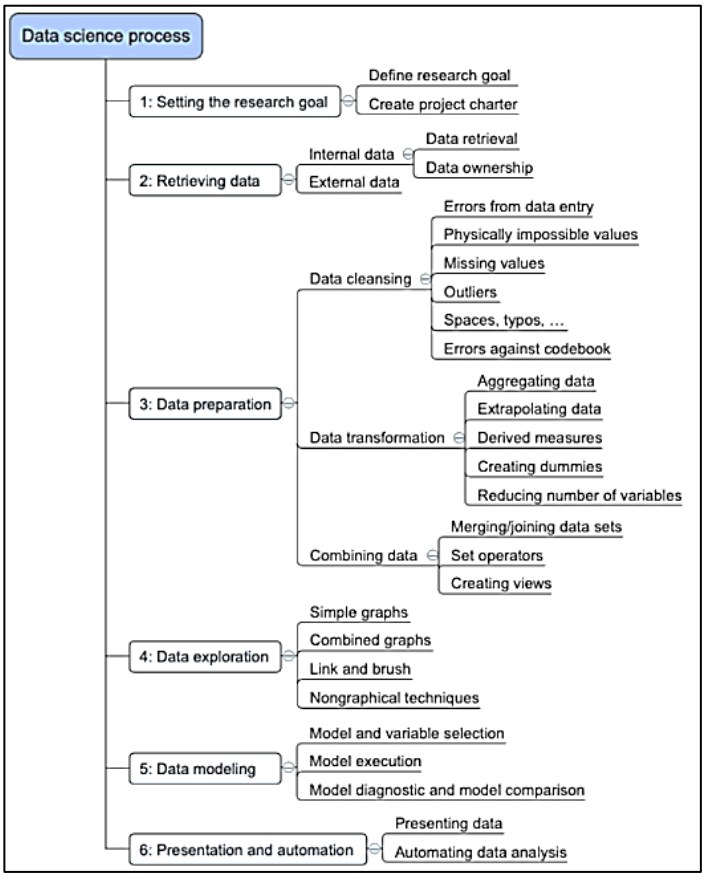
2.2.3 *Data science*

Data science menggabungkan matematika dan statistik, pemrograman khusus, analitik tingkat lanjut, kecerdasan buatan (AI), dan *machine learning* dengan keahlian pada subjek tertentu untuk mengungkap wawasan atau pengetahuan yang dapat ditindaklanjuti yang tersembunyi dalam data organisasi. Wawasan ini dapat digunakan untuk memandu pengambilan keputusan dan perencanaan strategis. (IBM, 2020). Menurut (Hartatik et al., 2023) *data science* adalah

bidang interdisipliner yang melibatkan pengumpulan, pengolahan, dan analisis data untuk memecahkan masalah dan menghasilkan wawasan yang berguna.

Pada tahapan pengumpulan data, dapat dilakukan dengan melakukan survei, atau sensor, namun dapat pula berupa sumber lain seperti *database*. Sementara pada tahapan pengolahan data, kegiatan yang dilakukan yaitu mulai dari membersihkan data, formatting, pemilihan fitur, hingga normalisasi data. Selanjutnya, pada tahapan analisis data terdapat banyak pendekatan atau teknik analisis yang digunakan seperti regresi, clustering, dan klasifikasi yang bertujuan untuk menemukan suatu tren atau pola tertentu yang dapat membantu dalam membuat keputusan. Terakhir, melakukan komunikasi hasil dan pengambilan keputusan berdasarkan tahapan analisis yang dilakukan.

Selain itu, menurut (Hartatik et al., 2023) terdapat 6 tahapan dalam proses *data science* seperti pada gambar berikut.



Gambar 2. 3 *Data science Process*

2.2.4 *Machine Learning*

Machine Learning adalah subbidang dari ilmu komputer yang berkaitan dengan pembangunan algoritma yang berdasarkan suatu fenomena, dan mengandalkan kumpulan contoh dari beberapa fenomena tersebut untuk digunakan dalam suatu fenomena. Contoh tersebut dapat berasal dari alam, dibuat dengan tangan oleh manusia, atau dihasilkan oleh algoritma lain. *Machine learning* juga dapat didefinisikan sebagai proses pemecahan masalah praktis dengan mengumpulkan kumpulan data dan secara algoritmik membangun model statistik berdasarkan kumpulan data tersebut. Model statistik tersebut diasumsikan digunakan untuk memecahkan masalah praktis.(Burkov, 2019)

a. Type of *Learning*

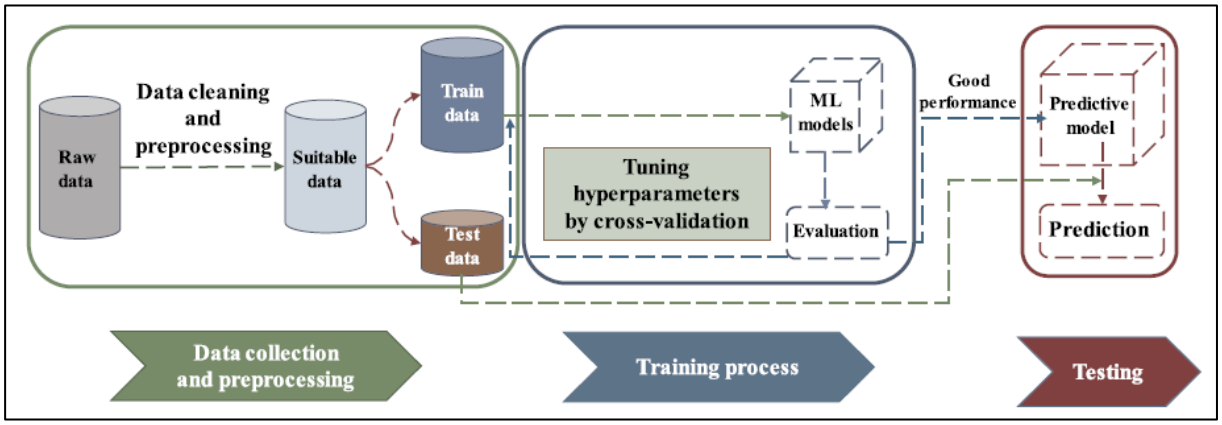
Machine learning terdiri atas beberapa jenis atau tipe seperti *supervised*, semi-*supervised, unsupervised* dan *reinforcement learning*. Pada jenis *supervised learning*, *dataset*-nya memuat kumpulan data yang memiliki label yang setiap elemennya disebut *feature vector*. *Feature vector* pada tiap dimensi memiliki suatu nilai yang mendeskripsikan contoh yang dimaksud, dimana nilai tersebut yang disebut dengan *feature*. Tujuan dari algoritma *supervised learning* ini adalah menggunakan kumpulan data untuk menghasilkan model yang mengambil vektor fitur x sebagai informasi input dan *output* yang memungkinkan deduksi label untuk vektor fitur ini. Misalnya, model yang dibuat menggunakan kumpulan data orang dapat mengambil vektor fitur yang mendeskripsikan seseorang sebagai masukan dan menghasilkan kemungkinan bahwa orang tersebut menderita kanker.

Pada jenis *unsupervised learning*, *dataset*-nya memuat data yang tidak memiliki label. Tujuan dari algoritma ini, yaitu membuat model yang mengambil *feature vector* sebagai masukan dan mengubahnya menjadi vektor lain atau menjadi nilai yang dapat digunakan untuk memecahkan masalah praktis. Misalnya, dalam clustering, model mengembalikan *id cluster* untuk setiap *feature vector* dalam kumpulan data. Sementara pada jenis *semi-supervised learning*, *dataset*-nya memuat kumpulan data yang berlabel dan tidak berlabel. Biasanya jumlah yang tidak diberi label jauh lebih banyak dibandingkan jumlah data yang diberi label. Tujuan dari algoritma *semi-supervised* ini sama dengan tujuan dari algoritma *supervised*, namun dengan harapan bahwa menggunakan lebih banyak data yang tidak berlabel dapat membantu algoritma *machine learning* untuk menemukan (menghasilkan atau menghitung) model yang lebih baik.

Jenis *reinforcement learning*, adalah jenis *machine learning* dimana mesinnya hidup di suatu environment dan mampu memahami state dari environment sebagai *feature vector*. Mesin tersebut dapat mengeksekusi tindakan di setiap bagian state. Tujuan dari algoritma *reinforcement* ini adalah untuk mempelajari kebijakan. Kebijakan yang dimaksud adalah suatu fungsi yang mirip dengan model dalam *supervised learning* yang menggunakan *feature vector* suatu keadaan sebagai input dan *output* berupa tindakan optimal untuk dieksekusi dalam keadaan tersebut. *Reinforcement learning* ini biasanya dapat memecahkan jenis masalah tertentu di mana pengambilan keputusan dilakukan secara berurutan, dan tujuannya bersifat jangka panjang, seperti bermain game, robotika, pengelolaan sumber daya, atau logistik (Burkov, 2019)

b. *Traditional Machine Learning*

Menurut (J. Yang et al., 2023), dalam pengembangannya, model *machine learning* yang dibangun dapat dilakukan dengan dua cara yaitu cara tradisional dan otomasi (*Automated Machine Learning* atau *AutoML*). Adapun alur dalam pembangunan model *machine learning* tradisional adalah sebagai berikut.



Gambar 2. 4 Alur *Machine Learning* Tradisional

Dalam penerapannya, model *machine learning* tradisional ini biasanya menggunakan *Support Vector Machine* (SVM) untuk meningkatkan akurasi dari model prediksi yang dirancang. *Artificial Neural network* (ANN) juga biasanya di implementasikan dalam model tradisional ini dengan tujuan untuk dirancang untuk mendesain model atau memodelkan hubungan kompleks antara input dan *output*. Selain itu, algoritma *Random Forest* juga sering menjadi opsi dalam model tradisional karena kemampuan yang dimiliki algoritma ini seperti dapat menghasilkan performa yang tinggi, biaya rendah, dan efektivitas dalam menangani kumpulan data besar.

c. *Automated Machine Learning* (*AutoML*)

Automated Machine Learning atau biasa disebut sebagai *AutoML*, adalah proses mengotomatisasi tugas berulang (iterative task) dalam pengembangan model pembelajaran mesin yang memakan waktu (Microsoft, 2023b). Menurut (Frank et al., 2022), bidang *AutoML* ini bertujuan untuk membuat keputusan dengan cara yang berdasarkan data, obyektif, dan otomatis dimana pengguna cukup menyediakan data, dan sistem *AutoML* secara otomatis menentukan pendekatan dengan performa terbaik untuk pengaplikasian pada suatu masalah tertentu. Oleh karena itu, *AutoML* membuat pendekatan *machine learning* yang canggih dapat diakses oleh mereka yang tertarik untuk menerapkan *machine learning* tetapi tidak memiliki sumber daya untuk mempelajari teknologi di baliknya secara mendetail. Hal ini dapat dilihat sebagai demokratisasi terhadap *machine learning*, dimana dengan adanya *AutoML* implementasi dari *machine learning* yang canggih dapat disesuaikan dapat diakses oleh semua orang.

Pendekatan *AutoML* sudah cukup matang untuk menyaingi dan terkadang bahkan mengungguli pakar atau *expert* di bidang *machine learning*. Sederhananya, *AutoML* dapat menghasilkan peningkatan kinerja sekaligus menghemat banyak waktu dan uang, karena pakar di bidang *machine learning* sulit ditemukan dan mahal. Selama beberapa tahun terakhir, beberapa *package* siap pakai yang menyediakan *AutoML* telah dikembangkan (*AutoML*.org, 2023). *Package* yang dimaksud antara lain sebagai berikut:

1. AutoWEKA,
2. Auto-sklearn,
3. Auto-*Pytorch*,
4. *AutoGluon*,
5. H2O *AutoML*,
6. MLBoX,
7. TPOT,
8. TransmogrifAI

2.2.5 *AutoGluon*

AutoGluon merupakan salah satu *framework Automated Machine Learning* (*AutoML*) yang dapat di akses secara gratis atau *open-source*. Saat ini, *AutoGluon* memiliki tiga jenis yaitu *Tabular*, *Multimodal*, dan *Time Series*. *AutoGluon-Tabular* adalah *framework AutoML* yang hanya membutuhkan satu baris *Python* untuk melatih model *machine learning* dengan tingkat akurasi yang tinggi, pada *dataset* bentuk *tabular* yang belum diproses seperti file CSV. *Framework* tersebut tidak seperti *framework AutoML* lainnya yang biasanya berfokus pada pemilihan model/hiperparameter. *AutoGluon-Tabular* ini dapat melakukan implementasi

metode *ensembling* untuk beberapa model dan menumpuknya dalam beberapa *layer*. (Erickson et al., 2020)

2.2.6 *Fundamental Algorithms*

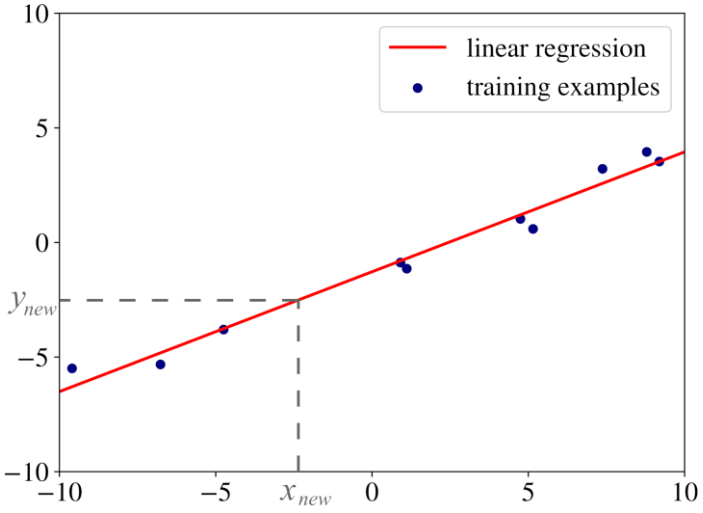
Menurut (Burkov, 2019), terdapat lima algoritma yang tidak hanya populer tetapi juga sering digunakan dalam membangun algortima pembelajaran (*learning algorithms*) karena efektivitas dari algoritma itu sendiri. Adapun kelima algoritma yang dimaksud adalah sebagai berikut.

a. *Linear Regression*

Linear Regression atau regresi linier adalah algoritma regresi populer yang mempelajari model sebagai kombinasi linier dari fitur-fitur contoh masukan (input example). Dalam penerapannya, formula dari algoritma ini di representasikan oleh persamaan berikut.

$$f_{w,b}(x) = wx + b \tag{2.1}$$

Dimana w adalah D-dimensi vektor dari suatu parameter dan b adalah bilangan real. Notasi $f_{w,b}$ artinya bahwa model f diparametrikan oleh dua nilai yaitu w dan b. Persamaan diatas sangat mirip dengan bentuk model *Support Vector Machine* (SVM). Satu-satunya perbedaan adalah tidak adanya tanda operator pada persamaan tersebut. Kedua model tersebut memang mirip, namun, *hyperplane* di SVM berperan sebagai batas keputusan (*decision boundary*) atau digunakan untuk memisahkan dua kelompok contoh satu sama lain, yang mana, jaraknya harus sejauh mungkin dari setiap kelompok. Disisi lain, *hyperplane* dalam regresi linier dipilih sedekat mungkin dengan semua contoh pelatihan seperti pada gambar dibawah ini



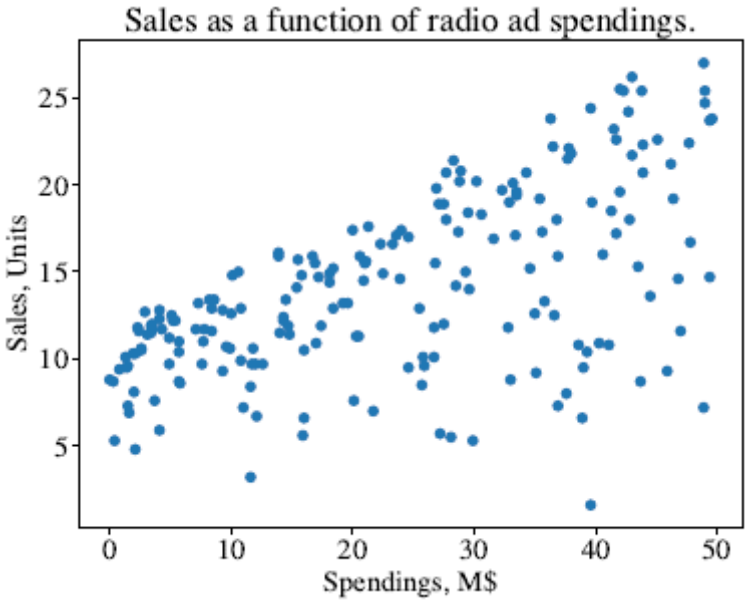
Gambar 2. 5 Contoh Regresi Linier *one-dimensional*

Dalam menemukan nilai yang optimal untuk w dan b pada persamaan diatas, dapat dilakukan melalui persamaan *objective function* seperti berikut.

$$\frac{1}{N}\sum_{i=1\ldots N}\left(f_{w,b}(x) = wx + b\right)^2 \tag{2.2}$$

Dalam matematika, meminimalkan atau maksimalkan disebut dengan fungsi objektif. Persamaan 2.2 diatas juga disebut dengan *loss function*. Persamaan tersebut digunakan untuk menghitung kesalahan klasifikasi dari sampel *i*. Sementara untuk pilihan tertentu dari *loss function* ini disebut dengan *squared error* loss.

Dalam penerapannya, untuk melakukan optimalisasi dari algoritma regresi linier ini dilakukan dengan pendekatan *gradient descent* seperti pada contoh sampel data dibawah ini

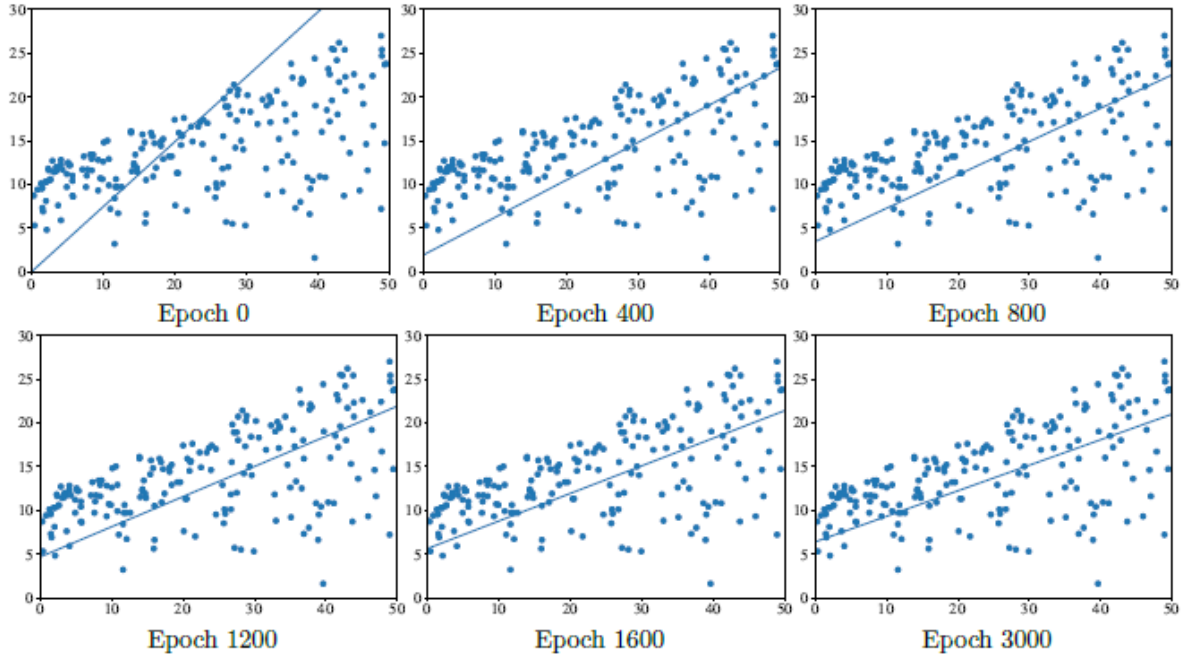


Gambar 2. 6 Contoh Optimalisasi Regresi Linier

Gradient descent diawali dengan melakukan perhitungan turunan sebagian (partial derivation) pada tiap parameter seperti pada persamaan berikut

$$\frac{\partial l}{\partial w} = \frac{1}{N}\sum_{i=1} f_{w,b}(x) = -2x_i\,(y_i - (wx_i + b)) \tag{2.3}$$

Persamaan diatas membuat *gradient descent* ini akan diproses kedalam *epochs* seperti pada gambar dibawah ini



Gambar 2. 7 Perubahan Garis Regresi Melalui *epoch gradient descent*

b. *Logistic Regression*

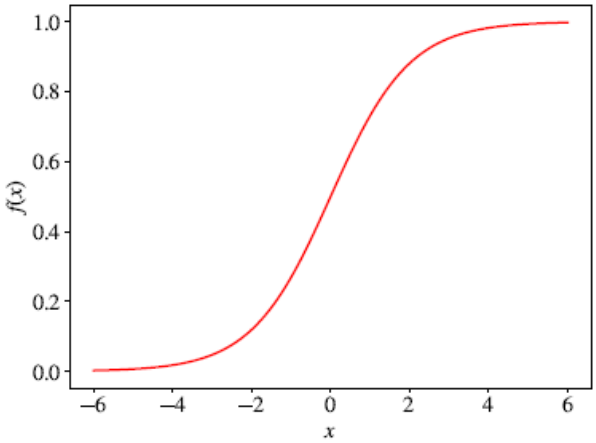
Logistic regression secara fomula matematis mirip dengan regresi linier, namun sebenarnya algoritma ini merupakan salah satu algoritma klasifikasi. Pada regresi logistik kombinasi fitur-fiture $wx_i + b$ adalah fungsi yang terbentang dari minus tak terhingga (-∞) sampai plus tak terhingga (+∞). Dalam hal ini dibutuhkan suatu fungsi yang dapat mengembalikan nilai dari model yang mempunyai nilai input x mendekati 0. Salah satu fungsi tersebut adalah standar *logistic function* atau yang dikenal dengan *sigmoid function*.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

Dimana *e* adalah konstanta logaritma natural atau yang biasa disebut dengan bilangan Euler. Sementara untuk persamaan logistik regresi di ekspresikan oleh persamaan berikut.

$$f(x) = \frac{1}{1 + e^{-(wx+b)}} \tag{2.5}$$

Dari persamaan diatas, diperoleh graphic seperti dibawah ini.

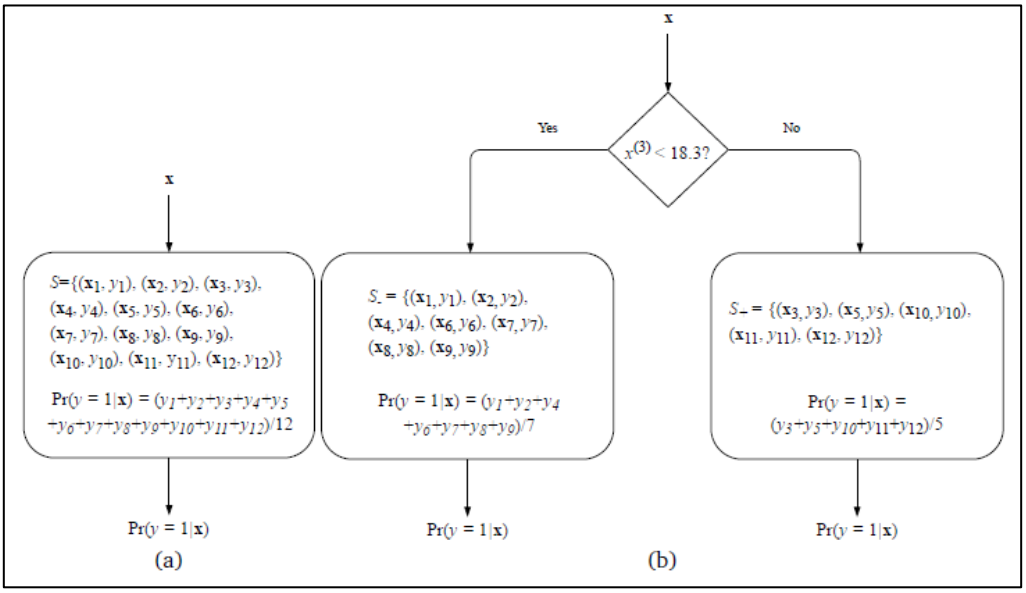


Gambar 2. 8 Fungsi Logistik Standar

c. *Decision tree*

Decision tree adalah grafik asiklik yang dapat digunakan untuk mengambil keputusan. Setiap simpul percabangan grafik memiliki fitur vektor *j* tertentu dari fitur yang diperiksa. Jika nilai fitur berada di bawah ambang batas tertentu, maka cabang kiri diikuti, jika tidak cabang kanan akan diikuti. Ketika node daun tercapai, keputusan dibuat terkait kelas dimana contoh tersebut berada. Menurut (Burkov, 2019) terdapat beberapa varian formula *decision tree*, adapun untuk formula jenis ID3 adalah sebagai berikut.

$$\frac{1}{N}\sum_{i=1}[y_i \ln(1-y_i) + (1-y_i)\ln(1-f_{ID3}(x_i))] \tag{2.6}$$



Gambar 2. 9 Ilustrasi Contoh Pembangunan Algoritma *Decision tree*

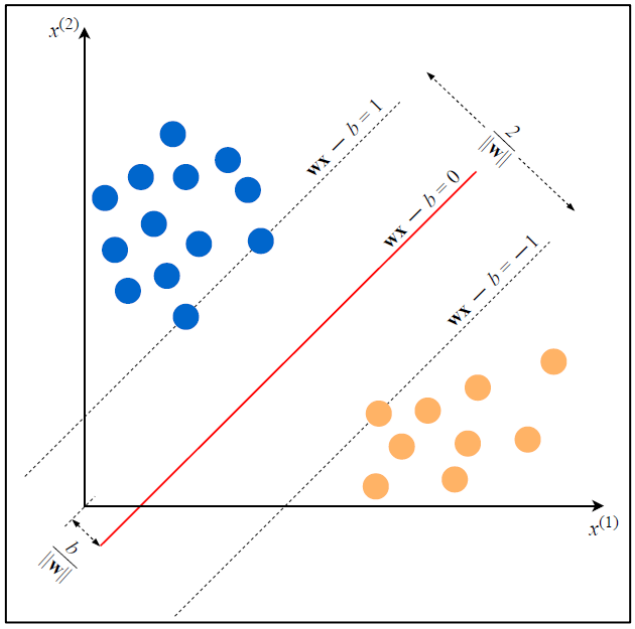
d. *Support Vector Machine*

Support Vector Machine (SVM) merupakan salah satu algoritma dalam *supervised learning* yang mengharuskan label positif (dalam kasus kita, "*spam*") memiliki nilai numerik +1 (satu), dan label negatif ("*not_spam*") memiliki nilai −1 (minus satu). SVM melihat setiap vektor fitur sebagai titik dalam ruang berdimensi tinggi. Algoritma ini menempatkan semua vektor fitur pada plot imajiner berdimensi 20.000 (contoh) dan menggambar garis imajiner berdimensi 19.999 (*hyperlane*) yang memisahkan contoh berlabel positif dari contoh berlabel negatif. Hal ini disebut sebagai *decision boundary* dalam *machine learning*.

Adapun persamaan *hyperplane* diberikan oleh dua parameter, sebuah vektor bernilai riil w dengan dimensi yang sama dengan vektor fitur masukan kita x, dan bilangan real b seperti berikut:

$$wx - b = 0 \tag{2.7}$$

Dalam mencari nilai optimal untuk w* dan b* pada persamaan diatas untuk contoh label yang dimaksud, dilakukan dengan memprediksi label 10.000 contoh dengan benar. Dengan ketentuan bahwa setiap contoh i = 1, . . . , 10000 diberikan oleh pasangan (xi, yi), di mana xi adalah vektor fitur dari contoh i dan yi adalah labelnya yang mengambil nilai −1 atau +1. Jadi fungsi kendala dapat digambarakan seperti berikut:



Gambar 2. 10 Contoh SVM untuk Vektor 2 Dimensi

$$\begin{aligned} wx_i - b &\geq +1 \quad jika \quad y_i = +1 \\ wx_i - b &\leq -1 \quad jika \quad y_i = -1 \end{aligned} \tag{2.8}$$

Menurut (Burkov, 2019) akan lebih baik jika *hyperplane* memisahkan contoh positif dari contoh negatif dengan margin terbesar. Margin adalah jarak antara contoh terdekat dari dua kelas, sebagaimana ditentukan oleh batas keputusan. Margin yang besar berkontribusi pada generalisasi yang lebih baik, yaitu seberapa baik model mengklasifikasikan contoh-contoh baru di masa depan.

e. K-*Nearest Neighbors*

k-*Nearest Neighbors* (kNN) merupakan algoritma pembelajaran non-parametrik. Berbeda dengan algoritma pembelajaran lain yang memungkinkan pembuangan data pelatihan setelah model dibuat, kNN menyimpan semua contoh pelatihan dalam memori. Ketika contoh x baru yang belum pernah dilihat sebelumnya masuk, algoritma kNN menemukan k contoh pelatihan yang paling dekat dengan x dan mengembalikan label mayoritas, dalam kasus klasifikasi, atau label rata-rata, dalam kasus regresi. (Burkov, 2019)

Kedekatan dua contoh diberikan oleh fungsi jarak. Misalnya, jarak *Euclidean* yang terlihat di atas sering digunakan dalam praktik. Pilihan fungsi jarak populer lainnya adalah *negative cosine similarity* yakni sebagai berikut

$$\cos\left(\angle\left(x_i\,,\,x_k\right)\right) = \frac{\sum_{j=1}^{D} x_i^{(j)} x_k^{(j)}}{\sqrt{\sum_{j=1}^{D}\left(x_i^{(j)}\right)^2}\sqrt{\sum_{j=1}^{D}\left(x_k^{(j)}\right)^2}} \tag{2.9}$$

Persamaan diatas mirip dengan pengukuran arah dua vektor. Jika sudut antara dua vektor adalah 0 derajat, maka dua vektor menunjuk ke arah yang sama, dan kesamaan kosinusnya sama dengan 1. Jika vektor-vektornya ortogonal, maka kesamaan kosinusnya adalah 0. Untuk vektor-vektor yang berlawanan arah, kesamaan kosinusnya adalah −1. Untuk memperoleh kesamaan kosinus sebagai metrik jarak, maka perlu mengalikannya dengan −1.

2.2.7 *Neural Network*

Jaringan saraf atau *Neural network* (NN), seperti halnya regresi atau model SVM, adalah fungsi matematika:

$$\mathrm{y} = f_{NN}(x) \tag{2.10}$$

Fungsi f_{NN} memiliki bentuk tertentu: yaitu fungsi *nested function*. Jika terdapat untuk jaringan saraf 3 lapis yang mengembalikan skalar, fNN terlihat seperti ini:

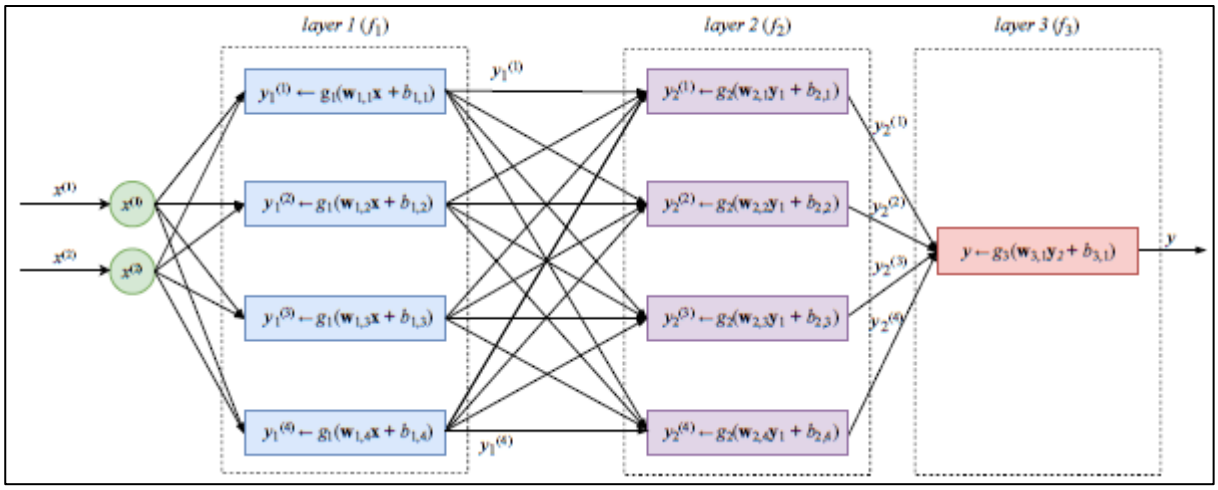
$$\mathrm{y} = f_{NN}(x) = f_3(f_2(f_1(x))) \tag{2.11}$$

Pada persamaan diatas, f_1 dan f_2 adalah fungsi vektor pada persamaan berikut.

$$f_1(z) = gl(W_l z + b_l)) \tag{2.12}$$

Dimana l disebut indeks lapisan dan dapat berkisar dari 1 hingga sejumlah lapisan, sementara fungsi gl disebut fungsi aktivasi atau *activation function*. Fungsi tersebut adalah fungsi tetap, biasanya *nonlinier* yang dipilih oleh analis data sebelum pembelajaran dimulai. Parameter Wl (matriks) dan bl (vektor) untuk setiap lapisan dipelajari menggunakan *gradient descent* yang sudah dikenal dengan mengoptimalkan, bergantung pada tugasnya, fungsi biaya tertentu (seperti MSE).

Pada satu konfigurasi jaringan saraf tertentu yang disebut *feed-forward neural network* (FFNN), dan lebih khusus lagi arsitekturnya disebut *multilayer perceptron* (MLP). Dalam kasus MLP dengan tiga lapisan, jaringan yang terbentuk mengambil vektor fitur dua dimensi sebagai input dan *output* yang berupa angka. FFNN ini dapat berupa model regresi atau klasifikasi, bergantung pada fungsi aktivasi yang digunakan pada lapisan keluaran ketiga seperti pada gambar dibawah ini



Gambar 2. 11 Contoh *Multilayer Perceptron* Dengan 2 *Layer* 4 Unit dan 1 *Output Layer*

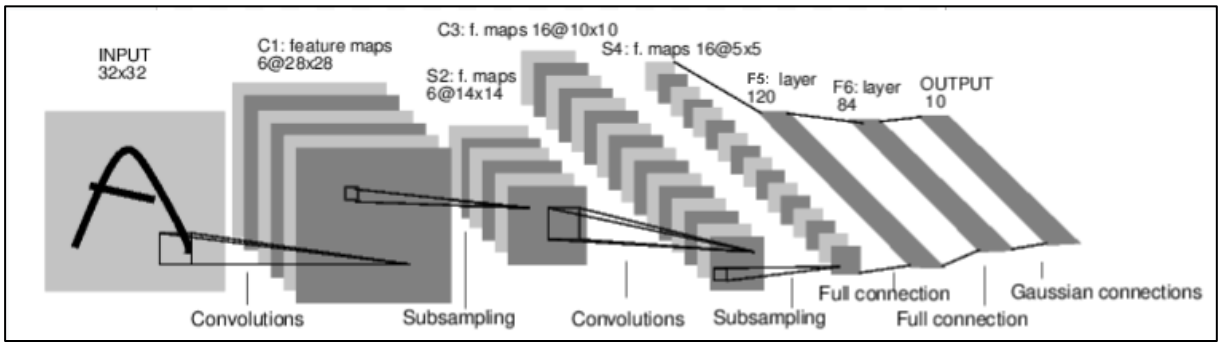
Jaringan saraf direpresentasikan secara grafis sebagai kombinasi yang terhubung unit disusun secara logis menjadi satu atau lebih lapisan. Setiap unit diwakili oleh lingkaran atau persegi panjang. Panah masuk mewakili masukan suatu unit dan menunjukkan dari mana masukan tersebut berasal. Panah keluar menunjukkan keluaran suatu unit. Keluaran setiap satuan merupakan hasil operasi matematika yang tertulis di dalam persegi panjang. Unit lingkaran tidak melakukan apa pun dengan input, hanya mengirimkan masukannya langsung ke *output*. Pada gambar diatas, fungsi aktivasi *gl* memiliki satu indeks *l* yaitu indeks lapisan

tempat unit tersebut berada. Biasanya, semua unit lapisan menggunakan fungsi aktivasi yang sama, tetapi hal ini bukanlah sebuah aturan. Setiap lapisan dapat memiliki jumlah unit yang berbeda. Setiap unit memiliki parameternya sendiri.

a. *Pytorch*

Pytorch adalah *framework open-source* karya *developers* di Facebook AI *Research* dan beberapa lab lainnya yang memiliki fitur lengkap untuk membangun model *deep learning* yang merupakan jenis *machine learning* yang umum digunakan dalam aplikasi seperti *image recognition* dan *language process*sing. *Pytorch* memiliki ciri khas karena dukungannya yang luar biasa untuk GPU dan penggunaan diferensiasi otomatis mode terbalik, yang memungkinkan grafik komputasi dimodifikasi dengan cepat. Hal ini menjadikannya pilihan populer untuk eksperimen cepat dan pembuatan prototipe.(NVIDIA, 2023)

Neural network terdiri dari lapisan/modul yang melakukan operasi pada data. Dalam *Pytorch* sebuah torch.nn menyediakan semua blok penyusun yang dibutuhkan untuk membangun jaringan saraf yang diinginkan. Setiap modul di *Pytorch* membuat subkelas nn.Module, yang mana jaringan saraf adalah modul itu sendiri yang terdiri dari modul-modul lain (lapisan). Struktur bersarang ini memungkinkan untuk membangun dan mengelola arsitektur yang kompleks dengan mudah. Misalnya, pada jaringan yang mengklasifikasikan gambar digit dibawah ini:



Gambar 2. 12 Contoh *Network* Untuk Klasifikasi Digit *Images*

Gambar diatas merupakan contoh feed-forward *network* yang sederhana. Dimana dalam mekanisme kerjanya, ia mengambil input, kemudian memasukkannya melalui beberapa lapisan satu demi satu, dan akhirnya memberikan keluaran. Prosedur pelatihan umum untuk jaringan saraf adalah sebagai berikut:

1. Menentukan jaringan saraf yang memiliki beberapa parameter (atau bobot) yang bersifat learnable
2. Iterasi kumpulan data input
3. Memproses input melalui jaringan (*network*)
4. Melakukan perhitungan terhadap *loss* (seberapa jauh keluarannya tidak benar)
5. Menyebarkan kembali gradien kedalam parameter jaringan
6. Perbarui bobot jaringan

b. FastAI

FastAI adalah *library deep learning* yang menyediakan komponen tingkat tinggi kepada para praktisi yang dapat dengan cepat dan mudah memberikan hasil canggih dalam standar domain *deep learning*, dan juga dapat memberi para peneliti komponen tingkat rendah yang dapat dipadukan dan dicocokkan untuk membangun suatu pendekatan yang baru (Howard & Gugger, 2020). Dua pendekatan ini bertujuan untuk melakukan kedua hal tersebut tanpa kompromi besar dari segi kemudahan penggunaan, fleksibilitas, atau kinerja. Abstraksi ini dapat di ekspresikan secara ringkas dan jelas dengan memanfaatkan dinamisme bahasa *Python* yang mendasarinya dan fleksibilitas *library Pytorch*. Aplikasi yang disediakan oleh *library* ini meliputi *vision, text, tabular,* dan *Collaborative Filtering*.

2.2.8 *Ensemble Learning*

Ensemble learning adalah paradigma pembelajaran yang lebih berfokus pada melatih sejumlah besar model dengan akurasi rendah dan kemudian menggabungkan prediksi yang diberikan oleh model lemah tersebut untuk mendapatkan meta-model dengan akurasi tinggi dari pada mencoba mempelajari satu model yang super akurat (Burkov, 2019). Model dengan akurasi rendah, yaitu algoritma pembelajaran yang tidak dapat mempelajari model yang kompleks, sehingga biasanya cepat dalam pelatihan dan waktu prediksi seperti algoritma *decision tree* di mana dalam penerapannya sering berhenti memisahkan set pelatihan setelah beberapa kali iterasi saja.

Model *Ensemble* dikembangkan dengan tujuan menggabungkan beberapa model untuk mencapai kinerja yang lebih baik dari satu model (Chung et al., 2023). Model *ensemble* menggunakan banyak model untuk mengurangi tingkat kesalahan secara keseluruhan dan meningkatkan akurasi prediksi, seperti halnya pengambilan keputusan dengan mendengarkan

banyak ahli dari pada mengandalkan satu pendapat, berkontribusi untuk meminimalkan kemungkinan keputusan yang buruk atau hasil yang tidak diinginkan.

Dalam memperoleh suatu prediksi dengan input x, prediksi setiap model lemah digabungkan menggunakan semacam pemungutan suara tertimbang (*weighted voting*). Bentuk spesifik dari pembobotan suara bergantung pada algoritma, namun, terlepas dari algoritma tersebut idenya tetap sama. Jika council dari model lemah menyatakan bahwa suatu pesan dikatakan spam, maka hal ini ditetapkan label spam kepada x. Dalam *ensemble learning* terdapat dua metode utama yaitu *boosting* and *bagging*.

a. *Boosting and Bagging*

Boosting terdiri dari penggunaan data pelatihan asli dan pembuatan beberapa model secara berulang dengan menggunakan algoritma yang lemah. Setiap model baru akan berbeda dari model sebelumnya dalam arti bahwa algoritma yang lemah, dengan membangun setiap model baru, mencoba "memperbaiki" *error* yang dibuat oleh model sebelumnya. Final Ensamble model adalah kombinasi tertentu dari beberapa model lemah yang dibangun secara iteratif. Sementara *Bagging* terdiri dari pembuatan banyak "salinan" data pelatihan (setiap salinan sedikit berbeda satu sama lain) dan kemudian menerapkan algoritma yang lemah ke setiap salinan untuk mendapatkan beberapa model lemah dan kemudian menggabungkannya. Algoritma *machine learning* yang banyak digunakan dan efektif berdasarkan gagasan *bagging* adalah *random forest*.

b. *Random Forest*

Random Forest adalah model yang diusulkan oleh Breiman pada tahun 1996 dengan melakukan utilisasi pada metode bootstrap, model yang dihasilkan dari beberapa sampel acak dan menghitung keluaran secara komprehensif melalui *decision tree*. Pada proses *fitting Decision tree*, pemilihan variabel penjelas untuk setiap node juga dilakukan secara acak. Melalui metode ini, keacakan dimaksimalkan dan ketika keacakan semakin maksimal, korelasi antara *decision tree* menurun, yang mengarah pada minimalisasi kesalahan prediksi (Liaw & Wiener, 2002). Menurut (Vladimir et al., 2003) *Random Forest* menjamin kekuatan prediksi dan stabilitas yang lebih tinggi sekaligus mengurangi *overfitting* dibandingkan dengan *Decision tree*s konvensional. Selain itu, ini sangat berguna ketika distribusi data tidak merata dan terdapat lebih banyak data dari kelas tertentu.

Pada suatu *training* pelatihan dengan B sampel acak Sb (untuk setiap b = 1, ... ,B) dari *training* set dan membangun model *decision tree* menggunakan setiap sampel Sb sebagai set

pelatihan. Untuk mengambil sampel Sb pada beberapa b, kita melakukan pengambilan sampel dengan penggantian. Artinya proses ini dimulai dari himpunan kosong, lalu memilih secara acak sebuah contoh dari set pelatihan dan memasukkan salinan persisnya ke Sb dengan menyimpan contoh asli di set pelatihan asli. Sehingga untuk contoh secara acak |Sb| = N diperoleh *decision tree* B setelah pelatihan. Adapun prediksi untuk contoh baru x diperoleh sebagai rata-rata dari prediksi B:

$$y \leftarrow \hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x) \qquad (2.13)$$

Formula diatas dapat digunakan pada kasus regresi, atau mayoritas kasus klasifikasi. Selain itu, *random forest* adalah salah satu algoritma ensamble *learning* yang paling banyak digunakan. Alasan dari penggunaan algoritma ini karena dengan menggunakan beberapa sampel dari kumpulan data asli, algoritma ini dapat mengurangi varians model akhir. (Burkov, 2019)

c. *Extremely Randomized Trees Classifier* (*Extra Tree*s Classifier)

Pengklasifikasi Pohon Ekstra) adalah jenis teknik *ensemble learning* yang menggabungkan hasil beberapa *decision tree* yang tidak berkorelasi yang dikumpulkan di "*forest*" untuk menghasilkan hasil klasifikasinya. Secara konsep, teknik ini sangat mirip dengan *random forest* classifier dan hanya berbeda dalam cara pembuatan *decision tree* didalam *forest*. Setiap *decision tree* di *extra tree*s *forest* dibuat dari sampel pelatihan asli. Kemudian, pada setiap node pengujian, setiap *tree* diberikan sampel acak sebanyak *k* fitur dari kumpulan fitur yang kemudian setiap pohon keputusan harus memilih fitur terbaik untuk membagi data berdasarkan beberapa kriteria matematika (biasanya Indeks *Gini*). Sampel fitur yang acak ini mengarah pada pembuatan beberapa pohon keputusan yang tidak berkorelasi. Untuk melakukan seleksi fitur menggunakan struktur hutan di atas selama konstruksi *forest*, untuk setiap fitur, dilakukan reduksi total yang dinormalisasi dalam kriteria matematis yang digunakan dalam pengambilan keputusan pemisahan fitur (Indeks Gini jika Indeks Gini digunakan dalam konstruksi *forest*). Nilai ini disebut Gini *importance* dari fitur.

Untuk melakukan pemilihan fitur, setiap fitur diurutkan dalam urutan menurun sesuai dengan Gini *importance* pada setiap fitur dan pengguna memilih k fitur teratas sesuai pilihannya. Pada kasus lima *decision tree* dan nilai k yang menentukan jumlah fitur dalam sampel fitur acak menjadi dua perlu dipertimbangkan hipotetis dari *extra tree forest*. Dalam hal

ini, kriteria keputusan yang digunakan adalah Information Gain, yang dapat diperoleh dengan melakuan perhitungan entropi data terlebih dahulu seperti pada persamaan dibawah

$$entropy = \sum_{i=1}^{C} -p_i \log_2(p_i) \tag{2.14}$$

d. *Gradient Boosting*

Gradient boosting adalah algoritma lain yang efektif dari ensamble *learning*. Pertama-tama mari kita lihat peningkatan gradien untuk regresi. *Boosting* model dapat didefinisikan dengan persamaan berikut:

$$f = f_0 + \alpha f_1 \tag{2.15}$$

Dimana f_1 sadalah suatu model *decision tree* dengan α sebagai *learning* rate atau sebuah *hyperparameter*. Jika dilakukan perhitungan ulang, dan mengganti label pada data pelatihan sekali lagi, latih model *decision tree* baru f_2, maka definisikan ulang model peningkatan sebagai berikut

$$f = f_0 + \alpha f_1 + \alpha f_2 \tag{2.16}$$

Proses ini akan berlanjut hingga M pada *tree*s maksimum yang telah ditentukan digabungkan. Dalam gradien *boosting* terdapat tiga *hyperparameter* utama yang perlu disesuaikan yaitu *number of tree*s, *learning rate*, dan *depth of tree*s. Ketiga hal ini dapat memengaruhi akurasi model. Kedalaman pohon (*depth of tree*s) mempengaruhi kecepatan latihan dan prediksi, dimana semakin pendek, semakin cepat. *Gradient boosting* pada klasifikasi juga menerapkan hal serupa, tetapi langkah-langkahnya sedikit berbeda.(Burkov, 2019)

Selain itu, terdapat beberapa bentuk algoritma lain dari *Gradient Boosting* yaitu sebagai berikut.

1. *eXtream Gradient Boosting* (*XGBoost*)

eXtream Gradient Boosting (*XGBoost*) merupakan model yang memanfaatkan teknik *Tree Boosting*, dan merupakan salah satu dari sekian banyak teknik *boosting* itu sendiri (Sukhpreet et al., 2018). *XGBoost* ini berfungsi hampir setara dengan model *Random Forest* tetapi, tidak seperti *Random Forest*, yang terus memperbarui parameter bobot sedemikian rupa sehingga mengurangi nilai *error* pada prediksi. *XGBoost* menghasilkan beberapa model yang digabungkan secara berurutan menggunakan *gradient descent*, dan juga menghasilkan model optimal akhir dengan memodifikasi model sebelumnya. Secara khusus, *XGBoost* ini

diimplementasikan untuk mengatasi masalah kecepatan *Gradient Boosting Machine* (GBM) melalui pemrosesan paralel.

2. *LightGBM*

LightGBM adalah kerangka kerja peningkatan gradien yang menggunakan algoritma pembelajaran berbasis pohon. Ini dirancang untuk didistribusikan dan efisien dengan kelebihan seperti kecepatan pelatihan lebih cepat dan efisiensi lebih tinggi, penggunaan memori lebih rendah, akurasi yang lebih baik, dukungan pembelajaran paralel, terdistribusi, dan GPU, serta mampu menangani data berskala besar (Microsoft, 2023a).

3. *CatBoost*

CatBoost adalah algoritma untuk *gradient boosting* pada *decision tree* yang dikembangkan oleh peneliti dan insinyur Yandex. Algoritma ini digunakan untuk pencarian, sistem rekomendasi, asisten pribadi, mobil tanpa pengemudi, prediksi cuaca, dan banyak tugas lainnya di Yandex dan perusahaan lain, termasuk CERN, *Cloudflare*, taksi Careem. *CatBoost* dapat di akses secara open source dan dapat digunakan oleh siapa saja.(Yandex, 2023). Adapun fitur yang ditawarkan oleh *CatBoost* adalah sebagai berikut.

- Kualitas hebat tanpa parameter *tuning*, dapat mengurangi waktu yang dihabiskan untuk parameter *tuning*, karena *CatBoost* memberikan hasil luar biasa dengan parameter default
- Dukungan fitur kategoris, meningkatkan hasil pelatihan yang memungkinkan *user* dapat menggunakan faktor non numerik, daripada harus memproses data terlebih dahulu atau menghabiskan waktu dan tenaga untuk mengubahnya menjadi angka.
- Versi GPU yang lebih cepat dan terukur, melakukan *train* model dengan implementasi gradien *boosting* untuk GPU dan dapat menggunakan konfigurasi *multi-card* untuk *dataset* yang lebih besar.
- Peningkatan akurasi, mengurangi *overfitting* saat membuat model dengan skema novel *gradient boosting*.
- Prediksi cepat, menerapkan model terlatih dengan cepat dan efisien bahkan pada tugas-tugas kritis latensi menggunakan penerapan model *CatBoost*

2.2.9 *Feature Engineering*

Menurut (Burkov, 2019) *feature engineering* adalah suatu permasalahan dalam mengubah data mentah menjadi kumpulan data. Pada sebagian besar masalah praktis, *feature engineering*

adalah proses labor-intensive yang menuntut seorang data analis mempunyai banyak kreativitas dan lebih diutamakan yang mempunyai *domain knowledge*. Dalam hal ini, peran dari seorang data analis adalah menciptakan fitur informatif yang memungkinkan algoritma pembelajaran (*learning algorithm*) membangun model yang berfungsi dengan baik dalam memprediksi label data yang digunakan untuk pelatihan.

a. *One-hot Encoding*

Pada beberapa algoritma pembelajaran hanya dapat bekerja dengan vektor fitur numerik. Jika beberapa fitur dalam suatu kumpulan data bersifat kategoris, seperti "warna" atau "hari dalam seminggu", maka data ini dapat diubah menjadi beberapa fitur biner. Sebagai contoh jika terdapat suatu data yang memiliki fitur kategorikal "warna" dan fitur ini memiliki tiga kemungkinan nilai yaitu "merah", "kuning", "hijau", maka fitur tersebut dapat diubah menjadi vektor dengan tiga nilai numerik seperti berikut

$$\begin{aligned} red &= [1,0,0] \\ yellow &= [0,1,0] \\ green &= [0,0,1] \end{aligned} \tag{2.17}$$

Ketika warna merah diubah menjadi 1, kuning menjadi 2, dan hijau menjadi 3 hal ini dapat menyiratkan bahwa ada urutan di antara nilai-nilai dalam kategori ini dimana urutan khusus tersebut dianggap penting untuk pengambilan keputusan, sehingga hal tersebut harus dihindari dengan melakukan hal seperti pada persamaan diatas.

b. Normalization

Normalisasi (*normalization*) adalah proses mengubah rentang nilai aktual yang dapat diambil oleh fitur numerik menjadi rentang nilai standar, yang biasanya dalam interval [-1,1] atau [0,1].

$$\bar{x}^{(j)} = \frac{\bar{x}^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}} \tag{2.18}$$

Dimana $min^{jJ)}$ dan $maks^{(j)}$ masing-masing adalah nilai minimum dan maksimum fitur *j* dalam kumpulan data. Normalisasi data bukanlah suatu tahapan prasyarat yang ketat, namun dalam praktiknya, hal ini dapat menyebabkan peningkatan kecepatan dari suatu pembelajaran. Selain itu, tahapan ini juga berguna untuk memastikan bahwa input yang diberikan kira-kira berada dalam kisaran yang relatif kecil untuk menghindari masalah yang dialami komputer ketika bekerja dengan bilangan yang sangat kecil atau sangat besar (atau *numerical overflow*).

c. Standardization

Standardisasi (atau normalisasi z-*score*) adalah prosedur di mana nilai fitur diubah skalanya sehingga memiliki properti distribusi normal yang standar dengan $\mu = 0$ dan $\sigma = 1$, dimana μ adalah mean (nilai rata-rata fitur) dan σ adalah standar deviasi dari mean. Standar skor (atau skor-z) fitur dihitung menggunakan rumus sebagai berikut:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}} \tag{2.19}$$

Dalam penerapannya, terdapat beberapa pertimbangan dalam memilih antara normalisasi atau standarisasi yaitu sebagai berikut.

1. Pada *unsupervised algorithm*s, manfaat menggunakan standardisasi lebih banyak dibandingkan normalisasi
2. Standardisasi juga lebih diutamakan untuk kasus dimana suatu nilai yang diambil fitur ini terdistribusi mendekati distribusi normal (disebut kurva lonceng)
3. Standardisasi juga lebih diutamakan untuk suatu fitur jika fitur tersebut terkadang memiliki nilai yang sangat tinggi atau rendah (*outlier*), hal ini karena normalisasi akan "memeras" nilai normal ke dalam kisaran yang sangat kecil
4. Dalam kasus lainnya, normalisasi lebih baik atau diutamakan dibandingkan standarisasi.

d. *Handling Missing Feature*

Dalam beberapa kasus, nilai beberapa fitur pada suatu kumpulan data mungkin hilang. Hal ini sering terjadi ketika kumpulan data dibuat dengan cara handcraft, dan orang yang mengerjakannya lupa mengisi beberapa nilai atau tidak mengukurnya sama sekali (human *error*). Pendekatan umum dalam menangani nilai fitur yang hilang adalah sebagai berikut:

1. Menghapus contoh data yang fiturnya hilang dari kumpulan data (hal ini dapat dilakukan jika kumpulan data cukup besar sehingga dapat mengorbankan beberapa contoh pelatihan atau *train* data)
2. Menggunakan suatu algoritma pembelajaran yang dapat menangani nilai fitur yang hilang (tergantung pada *library* dan implementasi spesifik dari algoritma tersebut)
3. Menggunakan sebuah teknik data imputasi.

e. Data *Imputation*

Salah satu teknik imputasi data yaitu dengan mengganti nilai fitur yang hilang dengan nilai rata-rata fitur tersebut dalam suatu kumpulan data:

$$\hat{x}^{(j)} \leftarrow \frac{1}{M}\sum_{i=1}^{N} x_i{}^{(j)} \tag{2.20}$$

Di mana M < N adalah jumlah contoh yang terdapat nilai fitur *j*, sedangkan penjumlahannya tidak termasuk contoh yang tidak terdapat nilai fitur *j*. Teknik lainnya adalah mengganti nilai yang hilang dengan nilai di luar kisaran nilai normal.

2.2.10 *Feature Importance*

Menurut (Nuraliza et al., 2022) dalam (Li dan Liu 2017), *Feature importance* atau pemilihan fitur merupakan salah satu teknik pada tahapan data preparation di lingkup data reduction. *Feature importance* bekerja dengan cara melakukan pemilihan atribut-atribut yang paling mempengaruhi hasil dari klasifikasi. Dengan dilakukannya sistem pemilihan ini, maka atribut-atribut yang terdapat pada *dataset* akan berkurang (reduksi data). Namun, informasi yang penting tetap dipertahankan pada saat proses pemilihan atribut, sehingga proses ini tidak akan mempengaruhi hasil klasifikasi secara signifikan.

2.2.11 *Learning Algorithm Selection*

Memilih algoritma *machine learning* dapat menjadi tugas yang sulit karena keterbatasan waktu. Dalam hal ini, terdapat beberapa poin yang dapat dipertimbangkan untuk memilih model terbaik yakni sebagai berikut.

1. *Explainability*, apakah model yang dirancang harus dapat dijelaskan kepada audiens non-teknis?. Algoritma pembelajaran yang paling akurat biasanya disebut "*black boxes*". Dimana algoritma tersebut mempelajari model yang menghasilkan sedikit nilai *error*, namun membuat prediksi tertentu bisa jadi sangat sulit untuk dipahami dan bahkan lebih sulit untuk dijelaskan
2. *In-memory vs out-of-memori*, apakah suatu kumpulan data dapat dimuat sepenuhnya ke dalam RAM server atau komputer pribadi? Jika ya, maka kita dapat memilih beragam algoritma. Jika tidak, maka pilihan terbaik adalah algoritma pembelajaran tambahan yang dapat meningkatkan model dengan menambahkan lebih banyak data secara bertahap.

3. Jumlah fitur dan contoh, seberapa banyak contoh pelatihan yang dimiliki oleh suatu kumpulan data? Berapa banyak fitur yang dimiliki setiap contoh? Beberapa algoritma, termasuk *neural network* dan gradien *boosting* dapat menangani sejumlah besar contoh dan jutaan fitur.
4. *Categorical vs. numerical feature*s, apakah data yang dimiliki hanya terdiri dari fitur kategorikal, atau hanya fitur numerik, atau gabungan keduanya? Karena beberapa algoritme tidak dapat menangani kumpulan data secara langsung, dan perlu mengubah fitur kategorikal menjadi fitur numerik.
5. *Nonlinearity of the data*, apakah data yang dimiliki dapat dipisahkan secara linier atau dapatkah dimodelkan menggunakan model linier? Jika ya, SVM dengan kernel linier, logistik, atau regresi linier bisa menjadi pilihan yang baik. Jika tidak, jaringan neural dalam atau algoritme ansambel, mungkin akan bekerja lebih baik.
6. *Training Speed*, seberapa banyak waktu yang diperbolehkan untuk digunakan oleh algoritma pembelajaran dalam membangun model? Jaringan saraf diketahui lambat untuk dilatih. Algoritma sederhana seperti regresi logistik dan linier atau *decision tree* jauh lebih cepat.
7. *Prediction speed*, seberapa cepat model harus menghasilkan prediksi? Apakah model yang akan digunakan dalam produksi yang memerlukan throughput yang sangat tinggi? Algoritma seperti SVM, regresi linier dan logistik, dan (beberapa jenis) *neural network*, sangat cepat dalam waktu prediksi. Algoritma lain, seperti kNN, ensamble *algorithm*, dan *neural network* yang sangat dalam atau berulang, lebih lambat.

2.2.12 *Regularization*

Regularisasi adalah istilah umum yang mencakup metode yang memaksa algoritma pembelajaran untuk membangun model yang tidak terlalu rumit (Burkov, 2019). Dalam praktiknya, hal ini sering kali menghasilkan bias yang sedikit lebih tinggi namun secara signifikan mengurangi variansnya. Masalah ini dikenal dalam literatur sebagai tradeoff bias-varians. Dua jenis regularisasi yang paling banyak digunakan disebut regularisasi L1 dan L2. Berikut ini merupakan objektif regularisasi L1

$$min_{w,b}\left[C|w| + \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - (y_i)^2\right] \tag{2.21}$$

Di mana |w| = $\sum_{j=1}^{D}|w^{(j)}|$ dan C adalah *hyperparameter* yang mengontrol pentingnya regularisasi. Jika nilai C ditetapkan menjadi nol, model tersebut menjadi model standar non-regularized regresi linier. Sebaliknya, jika nilai C diubah menjadi nilai yang tinggi, algoritma pembelajaran akan mencoba mengatur paling banyak $w^{(j)}$ ke nilai yang sangat kecil atau nol untuk meminimalkan *objec*tive, dan model akan menjadi sangat sederhana yang dapat menyebabkan *underfitting*. Berikut ini merupakan objektif regularisasi L2

$$min_{w,b}\left[C\|w\|^2 + \frac{1}{N}\sum_{i=1}^{N}(f_{w,b}(x_i) - (y_i)^2\right] \quad (2.22)$$

Dimana

$$\|w\|^2 = \sum_{j=1}^{D}\left(w^{(j)}\right)^2$$

Dalam praktiknya, regularisasi L1 menghasilkan sparse model, model yang sebagian besar parameternya (dalam kasus model linier, sebagian besar $w^{(j)}$) sama dengan nol, selama *hyperparameter* C cukup besar. Jadi L1 melakukan *feature* selection dengan memutuskan fitur mana yang penting untuk prediksi dan mana yang tidak. Hal ini dapat berfungsi untuk meningkatkan kemampuan menjelaskan model. Namun, tujuannya adalah memaksimalkan performa model pada data ketidaksepakatan, maka L2 biasanya memberikan hasil yang lebih baik. L2 juga memiliki keuntungan karena dapat terdiferensiasi, sehingga *gradient descent* dapat digunakan untuk mengoptimalkan fungsi *objec*tive/tujuan. Metode regularisasi L1 dan L2 juga digabungkan dalam apa yang disebut elastic net *regularization* dengan regularisasi L1 dan L2 menjadi kasus khusus. Anda dapat menemukan dalam literatur nama *ridge regularization* untuk L2 dan *laso regularization* untuk L1.

2.2.13 Model *Fitting*

a. Underfitting

Menurut (Burkov, 2019) suatu model yang memiliki bias yang rendah dapat memprediksi label data pelatihan dengan baik. Jika model membuat banyak kesalahan pada data pelatihan, model tersebut dikatakan memiliki bias yang tinggi atau itu model *underfit*. Ada beberapa alasan terjadinya *underfitting*, yang paling penting adalah:

1. Model yang dirancang terlalu sederhana untuk datanya (misalnya model linier sering kali tidak sesuai)
2. Fitur direkayasa (*feature engineering*) tidak cukup informatif.

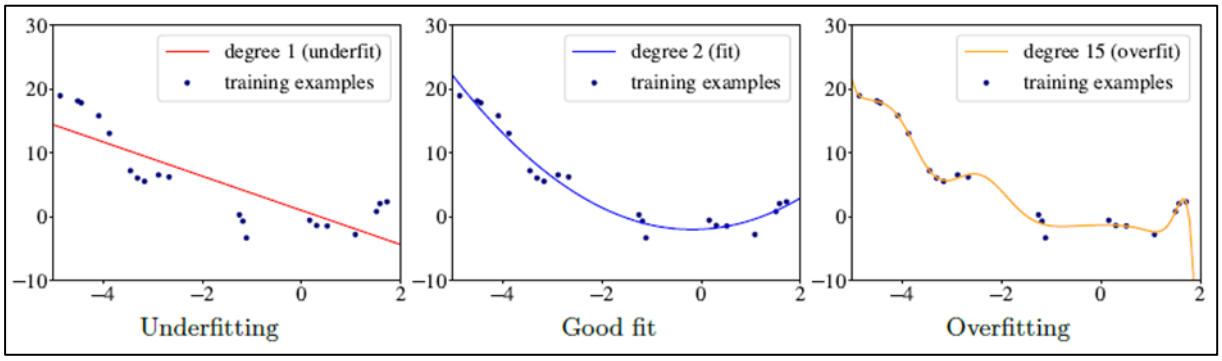
Solusi terhadap masalah *underfitting* diatas adalah dengan mencoba model yang lebih kompleks atau merekayasa fitur dengan daya prediksi yang lebih tinggi.

b. *Overfitting*

Overfitting adalah masalah lain yang dapat ditunjukkan oleh model. Model yang *overfit* memprediksi data pelatihan dengan sangat baik, namun buruk dalam memprediksi data dari satu dari dua subset data. Ada beberapa penyebab yang bisa menyebabkan terjadinya *overfitting*, yang terpenting adalah:

1. Model yang dirancang terlalu rumit atau kompleks untuk menampung data (misalnya *decision tree* yang sangat panjang dan *neural network* yang lebar sering kali mengalami overfit)
2. Terlalu banyak fitur namun jumlah contoh pelatihannya sedikit.

Berikut merupakan ilustrasi *underfitting* dan *overfitting*.



Gambar 2. 13 Contoh Model *Fitting* Pada Model Linear, Kuadrat, dan Polinomial

Dalam beberapa literatur, nama lain untuk masalah *overfitting* disebut dengan masalah high varians, yang merupakan istilah dari statistik. Varians adalah kesalahan model yang disebabkan sensitivitasnya terhadap fluktuasi kecil di set pelatihan. Artinya, jika data pelatihan yang diambil sampelnya secara berbeda, pembelajaran akan menghasilkan model yang berbeda secara signifikan. Itulah sebabnya model yang overfit memiliki performa yang buruk pada data pengujian, karena data pengujian dan pelatihan diambil sampelnya dari kumpulan data secara

independen satu sama lain. Dalam mengatasi overftting ini terdapat beberapa solusi yang dapat dilakukan sebagai berikut:

1. Mencoba model yang lebih sederhana (regresi linier, bukan regresi polinomial, atau SVM dengan kernel linier, atau *neural network* dengan lapisan/unit lebih sedikit).
2. Mengurangi dimensi contoh dalam kumpulan data
3. Menambah data pelatihan, jika memungkinkan.
4. Melakukan regularisasikan terhadap model (*regularization*)

2.2.14 Model *Performance Assessment*

Menurut (Burkov, 2019) para praktisi *Machine Learning* menggunakan berbagai metrik yang formal dan alat tertentu untuk menilai (*assessment*) performa dari suatu model. Untuk regresi, penilaian modelnya cukup sederhana, yaitu dengan mencari model yang menghasilkan nilai prediksi yang paling mendekati dengan nilai data observasi. Dalam hal ini, model rata-rata yang selalu memprediksi rata-rata label pada data *train*, umumnya akan digunakan jika tidak ada fitur informatif. Oleh karena itu, kesesuaian model regresi yang dinilai harus lebih baik daripada kesesuaian model rata-rata. Dalam melakukan hal tersebut, biasanya dilakukan dengan menghitung mean squared *error* (MSE) secara terpisah untuk data pelatihan (*train* data), dan data pengujian (*test* data). Berbeda dengan model regresi, untuk model klasifikasi terdapat beberapa metrik dan alat yang sering digunakan untuk penilaian yakni sebagai berikut

a. Confusion Matrix

Confusion matrix atau matriks konfusi adalah tabel yang merangkum seberapa sukses model klasifikasi dalam memprediksi contoh-contoh yang termasuk dalam berbagai kelas. Salah satu sumbu matriks konfusi adalah label yang diprediksi oleh model (*predicted*), dan sumbu lainnya adalah label sebenarnya (*actual*). (Burkov, 2019). Sebagai contoh, dalam masalah klasifikasi biner, terdapat dua kelas yang mana model yang akan dibangun bertujuan untuk memprediksi dua kelas yaitu "*spam*" dan "*not_spam*" pada tabel berikut

Tabel 1. 2 Contoh Matriks Konfusi

	Spam (predicted)	*Not_spam (predicted)*
Spam (actual)	23 (TP)	1 (FN)
Not_spam (actual)	12 (FP)	556 (TN)

Matriks konfusi di atas menunjukkan bahwa dari 24 contoh yang sebenarnya merupakan spam, model tersebut dengan tepat mengklasifikasikan 23 contoh sebagai spam atau 23 *True Positive* atau TP = 23 serta 1 model salah sebagai *not_spam*. Dalam kasus ini, kita mempunyai 1 *False Negative*, atau FN = 1. Demikian pula, dari 568 contoh yang sebenarnya bukan spam, 556 diklasifikasikan dengan benar (556 *True Negative* atau TN = 556), dan 12 diklasifikasikan salah (12 *False Positive*, FP = 12). Matriks konfusi untuk klasifikasi multikelas memiliki jumlah baris dan kolom yang sama banyaknya dengan jumlah kelas yang berbeda.

b. Precision/*Recall*

Dua metrik yang paling sering digunakan untuk menilai suatu model adalah presisi (precision) dan perolehan (*recall*). Presisi adalah rasio prediksi positif yang benar terhadap jumlah keseluruhan prediksi positif:

$$precision = \frac{TP}{TP + FP} \tag{2.23}$$

Recall adalah rasio prediksi positif yang benar terhadap jumlah keseluruhan contoh positif dalam kumpulan data:

$$recall = \frac{TP}{TP + FN} \tag{2.24}$$

Dalam praktiknya, hampir selalu terdapat dua pilihan yaitu high precision atau *high recall*. Biasanya tidak mungkin untuk memiliki keduanya. Hal ini hanya dapat dicapai dari berbagai cara seperti dengan memberikan bobot yang lebih tinggi pada contoh kelas tertentu (algoritma SVM menerima bobot kelas sebagai masukan) atau dengan dengan menyetel *hyperparameter* untuk memaksimalkan presisi atau perolehan kembali pada set validasi (Burkov, 2019).

c. *Accuracy*

Akurasi ditentukan oleh jumlah contoh yang diklasifikasikan dengan benar dibagi dengan jumlah total contoh yang diklasifikasikan (Burkov, 2019). Dalam matriks konfusi hal tersebut di representasikan oleh formula berikut:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.25}$$

Akurasi adalah metrik yang berguna ketika kesalahan dalam memprediksi semua kelas sama pentingnya. Dalam kasus spam or not_spam, hal ini mungkin tidak terjadi Misalnya, Anda akan lebih sedikit menoleransi hasil positif palsu dibandingkan negatif palsu. *True Positive*

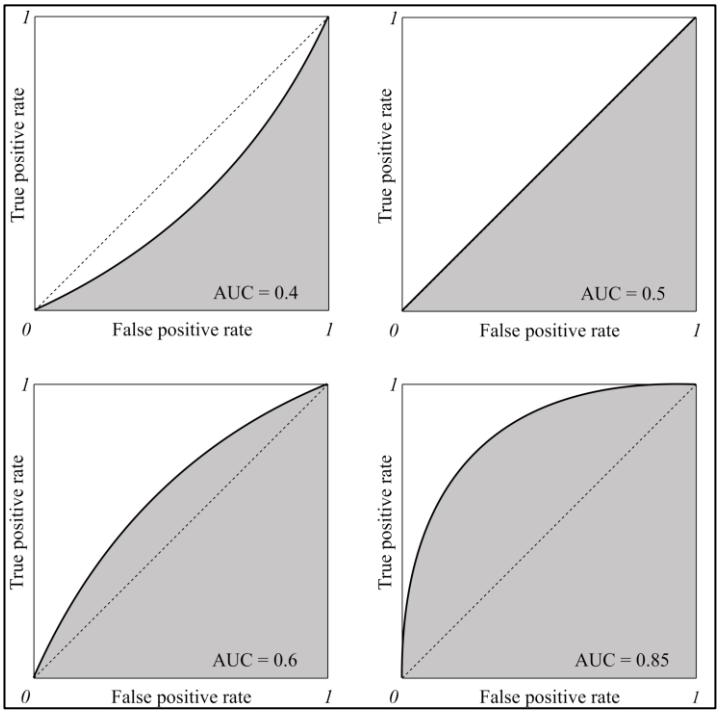
dalam deteksi spam adalah situasi di mana seseorang mengirimi email kepada kita, namun model melabelinya sebagai spam dan tidak menunjukkan notifikasi kepada kita.

d. Area Under the ROC Curve (AUC)

ROC atau *Receiver Operating* Characteristic adalah istilah yang berasal dari teknik radar dan merupakan metode yang umum digunakan untuk menilai kinerja model klasifikasi. Kurva ROC menggunakan kombinasi tingkat *True Positive* dan tingkat *False Positive* untuk membangun gambaran ringkasan kinerja klasifikasi. Tingkat positif sebenarnya (*True Positive* Rate atau TPR) dan tingkat positif palsu (*False Positive* Rate atau FPR) masing-masing didefinisikan sebagai berikut:

$$TPR = \frac{TP}{TP + FN} \qquad dan \qquad FPR = \frac{FP}{FP + TN} \tag{2.26}$$

Kurva ROC hanya dapat digunakan untuk menilai pengklasifikasi yang mengembalikan beberapa confidence *score* (atau probabilitas) dari suatu prediksi. Misalnya, *logistic regression*, *neural network*s, dan *decision tree* (dan *ensemble* model berdasarkan *decision tree*) dapat dinilai menggunakan kurva ROC.



Gambar 2. 14 Area Dibawah ROC

Sangat mudah untuk melihat bahwa jika ambang batasnya adalah 0, maka semua prediksi kita akan positif, sehingga TPR dan FPR akan menjadi 1 (pojok kanan atas). Sebaliknya jika

ambang batasnya adalah 1 maka tidak akan terjadi prediksi positif, baik TPR maupun FPR akan bernilai 0 yang sesuai dengan pojok kiri bawah. Semakin tinggi area di bawah kurva ROC (AUC), semakin baik pengklasifikasiannya. Pengklasifikasi dengan AUC lebih tinggi dari 0,5 lebih baik daripada pengklasifikasi acak. Jika AUC lebih rendah dari 0,5, berarti ada yang salah dengan model Anda. Pengklasifikasi yang sempurna akan memiliki AUC sebesar 1. Biasanya, jika model Anda berperilaku baik, Anda akan mendapatkan pengklasifikasi yang baik dengan memilih nilai ambang batas yang memberikan TPR mendekati 1 dan menjaga FPR mendekati 0 (Burkov, 2019)

e. F1-*Score*

F-*score* didefinisikan sebagai rata-rata harmonik tertimbang dari *Recall* & *Precision* (Powers, 2019). Ada beberapa alasan dibalik untuk pemilihan jenis rerata ini. Secara khusus, rata-rata harmonik umumnya sesuai ketika merata-ratakan laju atau frekuensi. Bentuk yang paling umum, F, memungkinkan pembobotan diferensial dari *Recall* dan *Precision* tetapi biasanya keduanya diberi bobot yang sama, sehingga menimbulkan F1 tetapi karena ada di mana-mana, hal ini sering dipahami ketika mengacu pada F-measure. Adapun formula dari F1 *score* ini adalag sebagai berikut.

$$F_1 score = \frac{2 * TP}{2 * TP + +FP + FN} \quad (2.27)$$

2.2.15 *Hyperparameter Tuning*

Menurut (Burkov, 2019), analis data harus melakukan "tune atau menyesuaikan" *hyperparameter* dengan cara eksperimental untuk menemukan kombinasi nilai terbaik, satu per *hyperparameter*. *Grid search* adalah jenis teknik *hyperparameter tuning* yang paling simpel. Misalnya, ketika melatih SVM dan terdapat dua *hyperparameter* untuk disetel (*tuning*) yaitu parameter penalti C (bilangan real positif) dan kernel (baik "linear" atau "rbf"). Dalam hal eksperimental, artinya mencoba semua kombinasi *hyperparameter* dapat memakan waktu, jika jumlahnya lebih dari satu, terutama untuk kumpulan data yang besar. Ada teknik yang lebih efisien, seperti *random search* dan *Bayesian Hyperparameter Optimization*.

Random search berbeda dengan grid *search* karena dalam penerapannya tidak perlu menyediakan kumpulan nilai terpisah untuk dijelajahi untuk setiap *hyperparameter*. Sebagai gantinya, hanya perlu menyediakan distribusi statistik untuk setiap *hyperparameter* yang nilainya diambil sampelnya secara acak dan menetapkan jumlah total kombinasi yang ingin

dicoba. Teknik Bayesian juga berbeda dengan *random* seacrh atau grid karena teknik ini menggunakan hasil evaluasi masa lalu untuk memilih nilai selanjutnya yang akan dievaluasi. Idenya adalah untuk membatasi jumlah optimasi fungsi tujuan yang mahal dengan memilih nilai *hyperparameter* berikutnya berdasarkan pada mereka yang telah melakukannya dengan baik di masa lalu.

Selain itu, terdapat cara yang dapat membantu saat suatu kumpulan data tidak memiliki set validasi yang layak untuk menyesuaikan *hyperparameter*, yang disebut dengan teknik umum *cross-validation*. Cara kerja atau mekanisme dari *cross-validation* ini adalah sebagai berikut.

1. Pertama, memperbaiki nilai *hyperparameter* yang ingin Anda evaluasi.
2. Membagi set pelatihan menjadi beberapa subset dengan ukuran yang sama, (setiap subset disebut sebuah lipatan atau *fold*).
3. Biasanya, *cross-validation* yang diterapkan adalah five-*fold cross-validation*.
4. Dengan validasi silang lima kali lipat, maka data pelatihan dibagi secara acak menjadi lima bagian:{F1, F2, . . . ,F5}, dimana setiap Fk,k=1, . . . ,5 berisi 20% data pelatihan.
5. Melatih lima model dimana model pertama, F1 digunakan semua contoh dari lipatan F2, F3,F4, dan F5 seperti set pelatihan dan contoh dari F1 sebagai set validasi. Kemudian untuk melatih model kedua F2, digunakan contoh dari lipatan F1,F3,F4, dan F5 untuk melatih dan contoh dari F2 sebagai set validasi. Langkah iteraasi ini juga dilakukan untuk menghitung nilai metrik yang diinginkan pada setiap kumpulan validasi, dari F1 ke F5.
6. Kemudian dilakukan perhitungan rata-rata kelima nilai metrik untuk mendapatkan nilai akhir.

2.2.16 *Google Colaboratory*

Google Colaboratorry atau *Colab* adalah layanan yang dihosting oleh *Jupyter notebook* yang tidak membutuhkan set-up atau persiapan untuk digunakan serta dapat memberikan akses gratis ke resource komputasi, termasuk GPU dan TPU. Colab ini sangat cocok untuk *machine learning, data science,* dan pendidikan (*Google*, 2023).

BAB III

METODE PENELITIAN

3.1 Objek Penelitian

Pada penelitian ini, objek penelitiannya adalah *dataset* IBM HR *Analytics Employee Attrition* yang merupakan *dataset open-source* di *platform* kaggle.com. *Dataset* ini merupakan *dataset* fiksional dengan input 35 variabel dan 1.470 data yang diterbitkan oleh perusahaan IBM.

3.2 Metode Pengumpulan Data

Dalam penelitian ini, pengumpulan data dilakukan dengan cara mengunduh langsung *dataset* IBM HR *Analytics Employee Attrition* pada *platform* kaggle.com. Adapun spesifikasi dalam pemilihan *public dataset* ini yaitu dengan memperhatikan lisesnsi dan kualitas data, yang mana *dataset* yang digunakan pada penelitian ini sudah mempunyai lisensi *Database Contents License* (DbCL) v1.0

3.3 Jenis dan Sumber Data

Jenis data yang akan digunakan pada penelitian ini terdiri atas dua yaitu dari data primer dan data sekunder. Data primer merupakan *public dataset* IBM HR *Analytics Employee Attrition* yang di unduh secara langsung melalui website. Sedangkan data sekunder adalah data yang relevan dan kredibel yang diperoleh melalui berbagai sumber seperti jurnal, buku, dokumentasi, dan sebagainya.

3.3.1 Data Primer

Data primer dalam penelitian ini diperoleh dengan cara mengunduh langsung *public dataset* di website kaggle.com yang berjudul IBM HR *Analytics Employee Attrition.* Data tersebut memuat 35 *variable* dengan 1.470 input.

3.3.2 Data Sekunder

Data sekunder disini diperoleh melalui dari jurnal, buku, dokumentasi, *script code*, ataupun informasi yang terkait dengan permasalahan yang dengan penelitian ini.

3.4 Metode Analisis Data

Metode analisis data pada penelitian ini dilakukan sesuai dengan proses *data science* pada umumnya dengan menggunakan *framework AutoGluon* sebagai alat utama dalam membangun model *machine learning*. Adapun tahapan atau proses *data science* dalam analisis data yang dimaksud adalah sebagai berikut.

1. Data Pre-*process*ing

Pada tahapan ini, *dataset* IBM HR *Analytics Employee Attrition* dibagi kedalam 2 file csv (*train*_HR_*Employee_Attrition*.csv dan *test*_HR_*Employee_Attrition*.csv) menggunakan MS. Excel ataupun dapat dengan menggunakan cara cutting dataframe dengan menggunakan fungsi .iloc() pada pandas *library*. Adapun hal yang dipertimbangkan dalam pembagian *dataset* ini, mengikuti dokumentasi pada website auto.gluon.ai seperti pada gambar berikut.

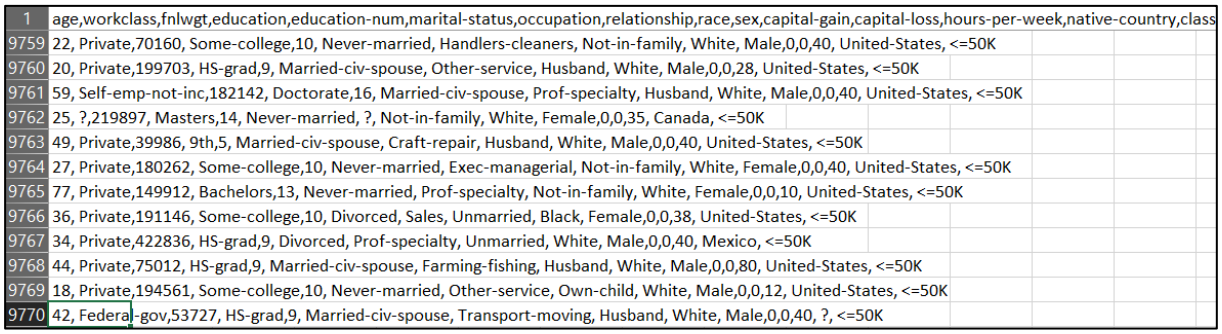
```
train_data = TabularDataset('https://autogluon.s3.amazonaws.com/datasets/Inc/train.csv')
test_data = TabularDataset('https://autogluon.s3.amazonaws.com/datasets/Inc/test.csv')
```

Gambar 3. 1 *Syntax Variable Train* and *Test Dataset*

Dari dokumentasi *syntax coding* diatas, direct link pada masing-masing file akan terlihat seperti gambar dibawah ini

	A
1	age,workclass,fnlwgt,education,education-num,marital-status,occupation,relationship,race,sex,capital-gain,capital-loss,hours-per-week,native-country,class
39065	40, Private,224232, Bachelors,13, Married-civ-spouse, Sales, Husband, White, Male,0,0,40, United-States, >50K
39066	20, Private,147523, 9th,5, Never-married, Handlers-cleaners, Not-in-family, White, Female,0,0,40, El-Salvador, <=50K
39067	51, Local-gov,194970, HS-grad,9, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,40, United-States, >50K
39068	32, Self-emp-not-inc,70985, HS-grad,9, Married-civ-spouse, Farming-fishing, Husband, White, Male,4064,0,40, United-States, <=50K
39069	27, Self-emp-not-inc,300777, HS-grad,9, Divorced, Transport-moving, Not-in-family, White, Male,0,0,70, United-States, <=50K
39070	54, Private,83103, Bachelors,13, Married-civ-spouse, Sales, Husband, White, Male,0,0,67, United-States, <=50K
39071	23, Private,172232, HS-grad,9, Never-married, Farming-fishing, Not-in-family, White, Male,0,0,53, United-States, <=50K
39072	37, Local-gov,165883, HS-grad,9, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male,0,0,40, United-States, <=50K
39073	26, Self-emp-not-inc,67240, HS-grad,9, Never-married, Handlers-cleaners, Not-in-family, White, Male,0,0,35, United-States, <=50K
39074	37, State-gov,210452, Masters,14, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,38, United-States, <=50K

Gambar 3. 2 Contoh Sampel *Dataset train*.csv AWS



1	age,workclass,fnlwgt,education,education-num,marital-status,occupation,relationship,race,sex,capital-gain,capital-loss,hours-per-week,native-country,class
9759	22, Private,70160, Some-college,10, Never-married, Handlers-cleaners, Not-in-family, White, Male,0,0,40, United-States, <=50K
9760	20, Private,199703, HS-grad,9, Married-civ-spouse, Other-service, Husband, White, Male,0,0,28, United-States, <=50K
9761	59, Self-emp-not-inc,182142, Doctorate,16, Married-civ-spouse, Prof-specialty, Husband, White, Male,0,0,40, United-States, <=50K
9762	25, ?,219897, Masters,14, Never-married, ?, Not-in-family, White, Female,0,0,35, Canada, <=50K
9763	49, Private,39986, 9th,5, Married-civ-spouse, Craft-repair, Husband, White, Male,0,0,40, United-States, <=50K
9764	27, Private,180262, Some-college,10, Never-married, Exec-managerial, Not-in-family, White, Female,0,0,40, United-States, <=50K
9765	77, Private,149912, Bachelors,13, Never-married, Prof-specialty, Not-in-family, White, Female,0,0,10, United-States, <=50K
9766	36, Private,191146, Some-college,10, Divorced, Sales, Unmarried, Black, Female,0,0,38, United-States, <=50K
9767	34, Private,422836, HS-grad,9, Divorced, Prof-specialty, Unmarried, White, Male,0,0,40, Mexico, <=50K
9768	44, Private,75012, HS-grad,9, Married-civ-spouse, Farming-fishing, Husband, White, Male,0,0,80, United-States, <=50K
9769	18, Private,194561, Some-college,10, Never-married, Other-service, Own-child, White, Male,0,0,12, United-States, <=50K
9770	42, Federa-gov,53727, HS-grad,9, Married-civ-spouse, Transport-moving, Husband, White, Male,0,0,40, ?, <=50K

Gambar 3. 3 Contoh Sampel *Dataset test*.csv AWS

Dari kedua file diatas, diperoleh persentase *split* data seperti tabel dibawah ini,

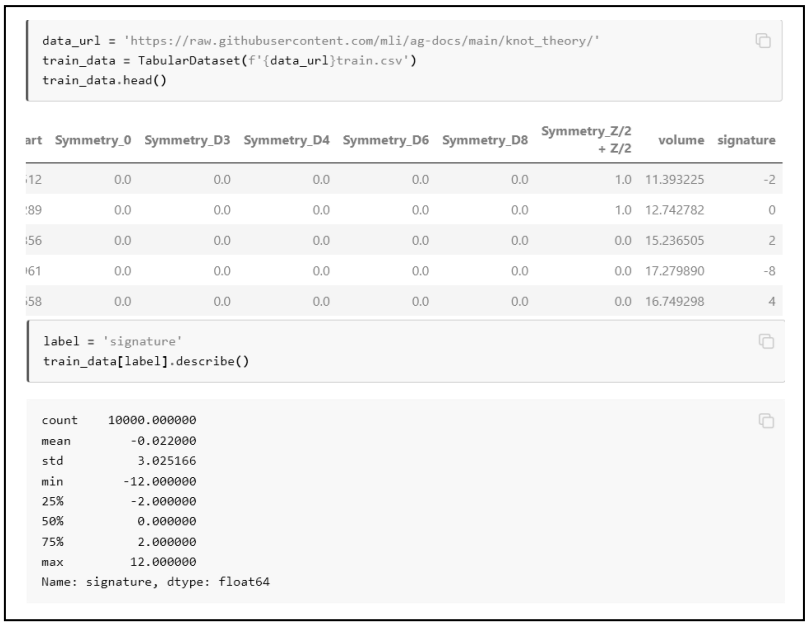
Tabel 3. 1 Contoh *Splitting* Pada *Dataset* AWS Stats

***Dataset* Source AWS**	***Entry Row***	**Persetase**
train	39073	80%
test	9769	20%
total/persentase	48842	100%

Sehingga untuk *dataset* IBM HR *Analytics Employee Attrition* akan dilakukan *split* data dengan persentase yang sama seperti *dataset* diatas yaitu 80:20. Selain tahapan input dan *splitting* data, pada tahapan ini juga nanti akan dilakukan proses eliminasi terhadap variabel atau *feature* yang memiliki *outlier*s.

2. *Exploratory Data Analysis (EDA)*

Tahapan EDA pada penelitian ini meliputi, implementasi statistika deskriptif, validasi *entry* data (apakah terdapat *entry* null/na/nan) pada tiap *variable*, dan membuat grafik pada tiap *variable* untuk mengetahui gambaran awal dari *dataset* dengan menggunakan *syntax* dibawah ini.



```
data_url = 'https://raw.githubusercontent.com/mli/ag-docs/main/knot_theory/'
train_data = TabularDataset(f'{data_url}train.csv')
train_data.head()
```

art	Symmetry_0	Symmetry_D3	Symmetry_D4	Symmetry_D6	Symmetry_D8	Symmetry_Z/2 + Z/2	volume	signature
;12	0.0	0.0	0.0	0.0	0.0	1.0	11.393225	-2
:89	0.0	0.0	0.0	0.0	0.0	1.0	12.742782	0
;56	0.0	0.0	0.0	0.0	0.0	0.0	15.236505	2
)61	0.0	0.0	0.0	0.0	0.0	0.0	17.279890	-8
;58	0.0	0.0	0.0	0.0	0.0	0.0	16.749298	4

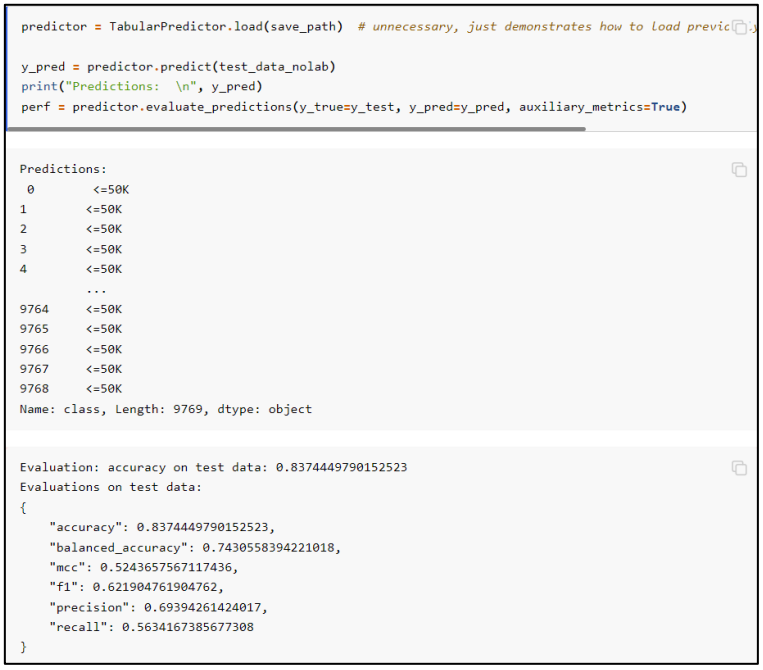
```
label = 'signature'
train_data[label].describe()
```

```
count    10000.000000
mean        -0.022000
std          3.025166
min        -12.000000
25%         -2.000000
50%          0.000000
75%          2.000000
max         12.000000
Name: signature, dtype: float64
```

Gambar 3. 4 *Syntax* EDA *AutoGluon*

3. Implementasi Model *Machine Learning*

Model *machine learning* yang akan dibangun, di eksekusi sesuai dengan ketentuan dokumentasi pada auto.gluon.ai sebagai berikut



```
predictor = TabularPredictor.load(save_path)  # unnecessary, just demonstrates how to load previc
y_pred = predictor.predict(test_data_nolab)
print("Predictions:  \n", y_pred)
perf = predictor.evaluate_predictions(y_true=y_test, y_pred=y_pred, auxiliary_metrics=True)
```

```
Predictions:
 0        <=50K
1        <=50K
2        <=50K
3        <=50K
4        <=50K
         ...
9764     <=50K
9765     <=50K
9766     <=50K
9767     <=50K
9768     <=50K
Name: class, Length: 9769, dtype: object
```

```
Evaluation: accuracy on test data: 0.8374449790152523
Evaluations on test data:
{
    "accuracy": 0.8374449790152523,
    "balanced_accuracy": 0.7430558394221018,
    "mcc": 0.5243657567117436,
    "f1": 0.621904761904762,
    "precision": 0.69394261424017,
    "recall": 0.5634167385677308
}
```

Gambar 3. 5 *Syntax* Evaluasi Model *Machine Learning AutoGluon*

Selanjutnya, melakukan eksekusi pada *syntax predictor.Leaderboard* untuk memperoleh ranking dari *test* data berdasarkan *dataset* yang dilatih seperti pada gambar berikut

```
predictor.leaderboard(test_data, silent=True)
```

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal
0	WeightedEnsemble_L2	0.9490	0.963964	0.535592	0.092418	15.158309	0.005171
1	LightGBM	0.9456	0.955956	0.180614	0.032769	3.686272	0.180614
2	XGBoost	0.9448	0.956957	0.459367	0.077025	5.076633	0.459367
3	LightGBMLarge	0.9444	0.949950	0.467929	0.074907	9.332877	0.467929
4	CatBoost	0.9432	0.955956	0.027557	0.005510	21.258693	0.027557
5	RandomForestEntr	0.9384	0.949950	0.175288	0.107368	1.621593	0.175288
6	NeuralNetFastAI	0.9372	0.942943	0.071053	0.014813	9.562637	0.071053
7	ExtraTreesGini	0.9360	0.946947	0.221845	0.096801	0.972927	0.221845
8	ExtraTreesEntr	0.9358	0.942943	0.238914	0.107383	0.953809	0.238914
9	RandomForestGini	0.9352	0.944945	0.176280	0.107651	1.196594	0.176280
10	LightGBMXT	0.9320	0.945946	0.288275	0.049073	4.083335	0.288275
11	NeuralNetTorch	0.9262	0.940941	0.020746	0.008270	33.750055	0.020746
12	KNeighborsDist	0.2210	0.213213	0.026592	0.014049	0.019487	0.026592
13	KNeighborsUnif	0.2180	0.223223	0.028118	0.015117	0.021147	0.028118

Gambar 3. 6 *Syntax Leaderboard* Model *Machine Learning*

4. Visualisasi Model *Machine Learning*

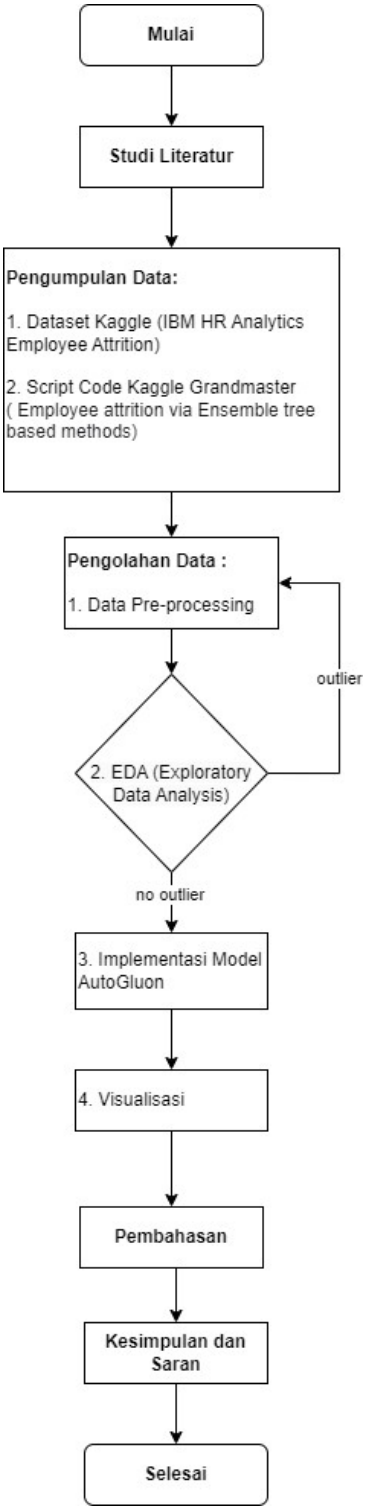
Tahapan visualisasi disini bertujuan untuk mempresentasikan hasil atau *output* dari model *machine learning* yang telah di bangun.

Selain itu, pada tahapan analisis ini juga dilakukan re-*coding* terhadap *script code* dari kaggle *grandmaster* yang menggunakan *dataset* yang sama. Adapun sumber *script code* yang digunakan diperoleh dari link dibawah ini:

https://www.kaggle.com/*code*/arthurtok/*employee-attrition*-via-*ensemble-tree*-based-methods

3.5 Alur Penelitian

Adapun alur dari penelitian ini adalah sebagai berikut.



Gambar 3. 7 *Flowchart* Penelitian

1. Studi Literatur

   Tahapan studi literatur, memuat teori-teori dari metode yang digunakan pada penelitian ini yang diperoleh dari karya-karya ilmiah seperti buku, jurnal, dan dokumentasi yang telah resmi dipublikasikan oleh *platform* yang kredibel. Kajian literatur yang terdapat pada penelitian ini berfokus kepada topik bahasan terkait dashboard *Machine Learning* dan *Data science*.

2. Pengumpulan Data

   Pengumpulan data pada penelitian ini dilakukan dengan cara mengunduh langsung *dataset* pada link berikut https://www.kaggle.com/*datasets*/pavansubhasht/ibm-hr-*analytics-attrition dataset*?resource=download.

3. Pengolahan Data

   Pada tahapan pengolahan data ini, dilakukan dengan menerapkan prinsip proses pengolahan *Data science* yang dimulai dari Data *Pre-Process*ing, *Exploratory Data Analysis (EDA)*, Implementasi Model *Machine Learning*, hingga Visualisasi Model. Pada tahapan ini, jika data yang telah melalui tahapan *pre-processing* masih mempunyai *outlier*, maka tahapan tersebut harus perlu dilakukan kembali. Ketika data tersebut sudah tidak memiliki *outlier*, maka data tersebut dapat masuk tahapan tahapan implementasi model *machine learning*

4. Pembahasan

   Pembahasan memuat penjelasan terkait *output* dari *AutoGluon* yang nantinya akan menghasilkan satu algoritma terbaik diantara 14 algoritma yang di *train* dan *test*. Selain itu, pada tahapan ini juga akan dilakukan perbandingan waktu dari *script* model *machine learning* yang dibangun dengan *script code* yang dirancang oleh salah satu *user* kaggle dengan ranking *grandmaster* (tier/ranking tertinggi di *platform* kaggle.com) yang telah dilakukan re*coding* lagi dengan beberapa penyesuaian untuk menghindari *copyright*. Hal ini bertujuan untuk membandingkan *script* mana yang lebih efisien dari segi waktu eksekusi.

5. Kesimpulan dan Saran

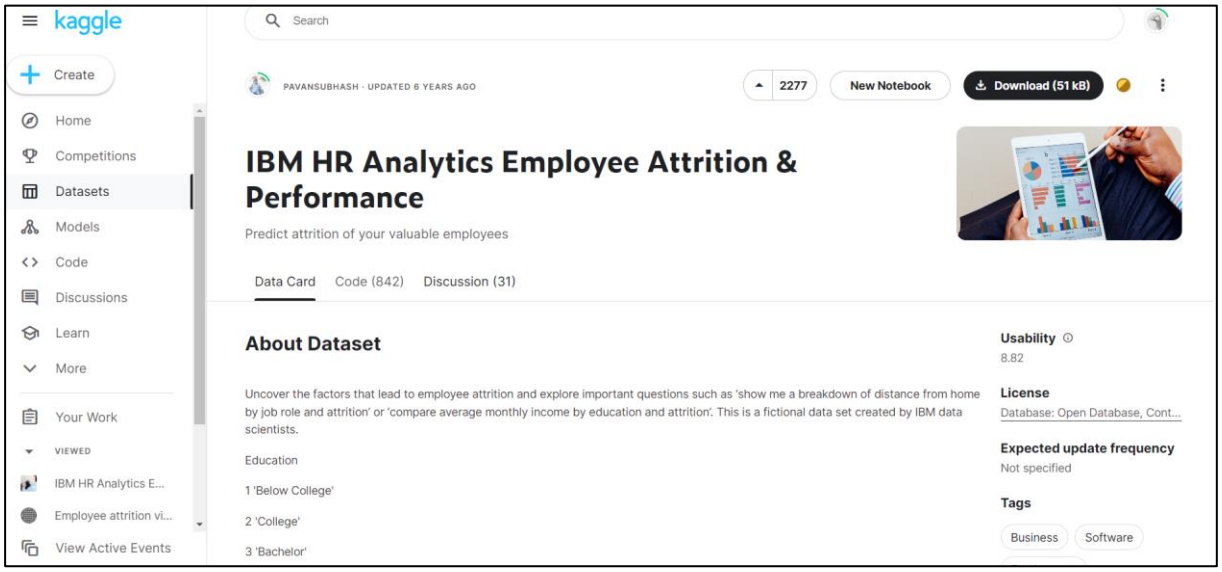
   Kesimpulan ini merupakan tahapan akhir dari penelitian yang memuat jawaban dari masalah dan tujuan penelitian yang telah ditentukan pada bahasan bab sebelumnya. Bagian saran berisikan terkait masukan kepada kepada peneliti lain dalam melakukan penelitian selanjutnya dengan topik terkait *AutoGluon*, ataupun *AutoML*.

BAB IV

PENGUMPULAN DAN PENGOLAHAN DATA

4.1 Pengumpulan Data

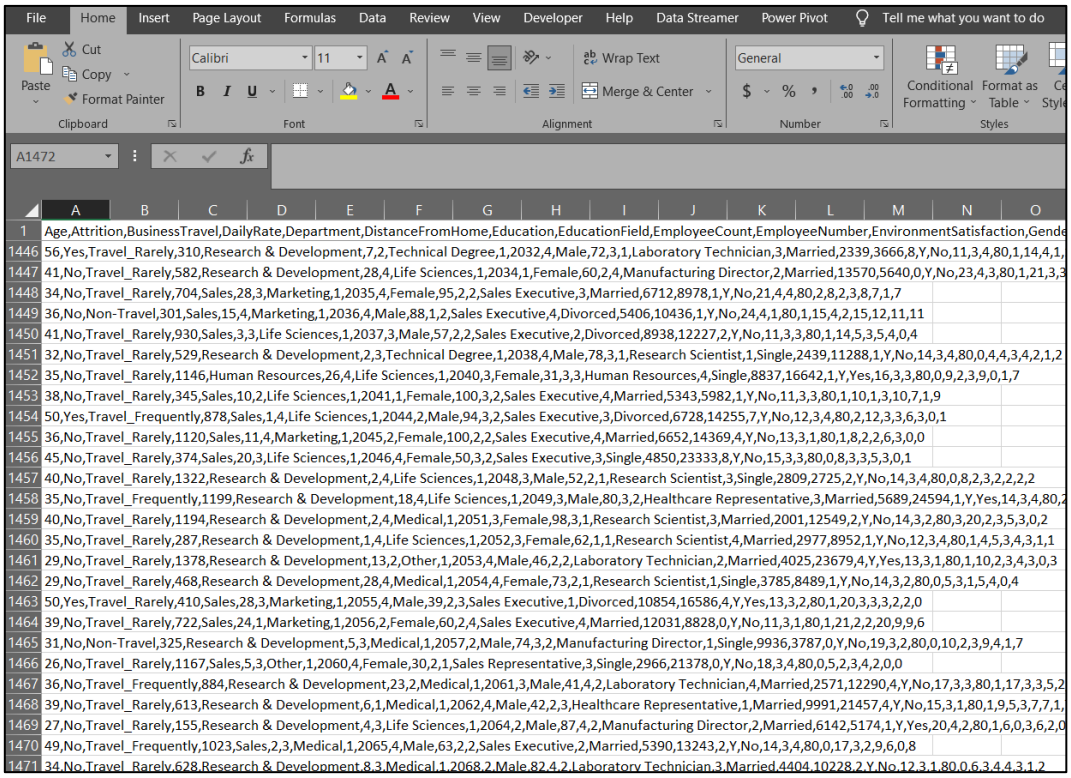
Dalam penelitian ini, pengumpulan data dilakukan dengan cara mengunduh langsung *dataset* IBM HR *Analytics Employee Attrition* pada *platform* kaggle.com. melalui link dibawah ini



Gambar 4. 1 *Dataset* Kaggle

Source : https://www.kaggle.com/*datasets*/pavansubhasht/ibm-hr-*analytics-attrition dataset*?resource=download

Berikut merupakan file *dataset* hasil unduhan melalui link diatas.



Gambar 4. 2 *Dataset* CSV IBM HR *Employee Attrition*

Berikut merupakan rincian *variable* atau *feature* yang terdapat pada *dataset* IBM HR *Employee Attrition* berdasarkan data card kaggle dan pangkalan data ieee-dataport.org

Tabel 4. 1 Rincian Tipe Data

No	***Feature***	Data Type	Scale
1	*Age*	*Numerical Discrete Data*	*None*
2	*Attrition*	*Text Categorical Data*	*None*
3	*Business Travel*	*Text Categorical Data*	*None*
4	*Daily Rate*	*Numerical Discrete Data*	*None*
5	*Department*	*Text Categorical Data*	*None*
6	*Distance From Home*	*Numerical Discrete Data*	*None*
7	*Education*	*Numerical, Categorical Data*	*1: Below college*
			2: College
			3: Bachelor
			4: Master
			5: Doctor
8	*Education Field*	*Text Categorical Data*	*None*
9	*Employee Count*	*Numerical, Categorical Data*	*None*

No	*Feature*	Data Type	Scale
10	*Employee Number*	*Numerical, Categorical Data*	*None*
11	*Environment Satisfaction*	*Numerical, Categorical Data*	*1: Low*
			2: Medium
			3: High
			4: Very High
12	*Gender*	*Text Categorical Data*	*None*
13	*Hourly Rate*	*Numerical Discrete Data*	*None*
14	*Job Involvement*	*Numerical, Categorical Data*	*1: Low*
			2: Medium
			3: High
			4: Very High
15	*JobLevel*	*Numerical, Categorical Data*	*None*
16	*Job Role*	*Text Categorical Data*	*None*
17	*Job Satisfaction*	*Numerical Categorical Data*	*1: Low*
			2: Medium
			3: High
			4: Very High
18	*Marital Status*	*Text Categorical Data*	*None*
19	*Monthly Income*	*Numerical Discrete Data*	*None*
20	*Monthly Rate*	*Numerical Discrete Data*	*None*
21	*Number Companies Worked*	*Numerical Discrete Data*	*None*
22	*Over 18*	*Text Categorical Data*	*None*
23	*Over Time*	*Text Categorical Data*	*None*
24	*Percent Salary Hike*	*Numerical Discrete Data*	*None*
25	*Performance Rating*	*Numerical Categorical Data*	*1: Low*
			2: Good
			3: Excellent
			4: Outstanding
26	*Relationship Satisfaction*	*Numerical Categorical Data*	*1: Low*
			2: Medium
			3: High
			4: Very High
27	*Standard Hours*	*Numerical Discrete Data*	*None*
28	*Stock Option Level*	*Numerical Categorical Data*	*None*
29	*Total Working Years*	*Numerical Discrete Data*	*None*
30	*Training Times Last Year*	*Numerical Discrete Data*	*None*
31	*Work Life Balance*		*1: Bad*

No	*Feature*	Data Type	Scale
		Numerical Categorical Data	*2: Good*
			3: Better
			4: Best
32	*Years At Company*	*Numerical Discrete Data*	*None*
33	*Years In Current Role*	*Numerical Discrete Data*	*None*
34	*Years Since Last Promotion*	*Numerical Discrete Data*	*None*
35	*YearsWithCurrManager*	*Numerical Discrete Data*	*None*

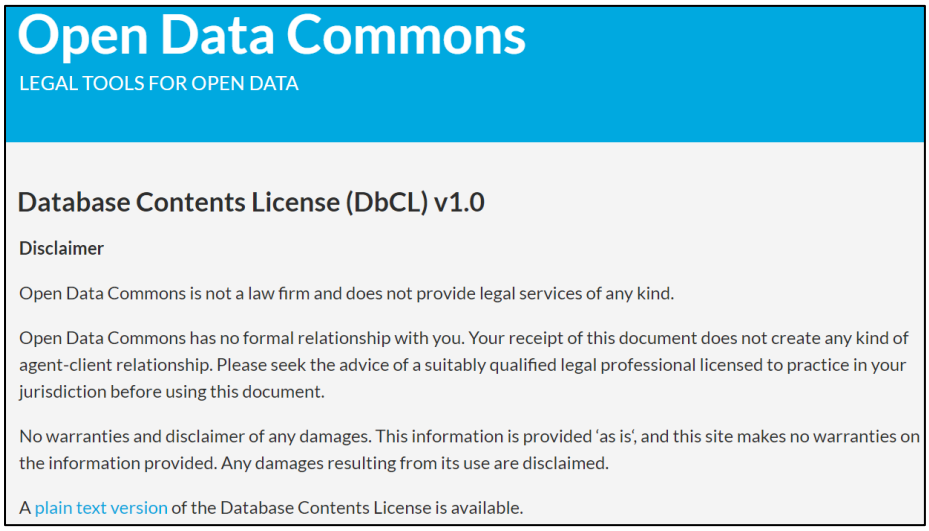
Adapun deskripsi dari tiap *feature* atau variabel pada *dataset* diatas berdasarkan literatur dan penelitian sebelumnya adalah sebagai berikut

Tabel 4. 2 Deskripsi Data

No	*Feature*	Des*cript*ion
1	*Age*	Usia pekerja
2	*Attrition*	Kondisi karyawan yang meninggalkan atau berhenti bekerja untuk perusahaan dengan berbagai alasan seperti pengunduran diri secara sukarela, PHK (Pemutusan Hubungan Kerja), atau bahkan tidak kembali dari cuti.
3	*Business Travel*	Tingkat intensitas atau kecenderungan karyawan dalam melakukan perjalanan bisnis atas mandat perusahaan
4	*Daily Rate*	Tingkat upah harian pekerja yang diperoleh dari monthly rate dibagi jumlah hari kerja
5	*Department*	Nama Departemen yang terdapat pada perusahaan
6	*Distance From Home*	Jarak tempat tinggal pekerja dengan kantor
7	*Education*	Tingkat pendidikan pekerja
8	*Education Field*	Latar belakang pendidikan pekerja
9	*Employee Count*	Keterangan jumlah pekerja
10	*Employee Number*	Nomor ID pekerja
11	*Environment Satisfaction*	Tingkat kepuasan pekerja terhadap lingkungan kerjanya
12	*Gender*	Jenis kelamin pekerja
13	*Hourly Rate*	Tingkat upah perjam pekerja yang diperoleh dari daily rate dibagi total jam kerja
14	*Job Involvement*	Tingkat partisipasi pekerja terhadap role atau jabatan di perusahaan
15	*JobLevel*	Level jabatan pekerja di departemen perusahaan
16	*Job Role*	Jabatan atau tugas spesifik pekerja
17	*Job Satisfaction*	Tingkat kepuasan pekerja terhadap role atau jabatan yang dikerjakan di perusahaan
18	*Marital Status*	Status nikah pekerja

No	*Feature*	Des*cript*ion
19	*Monthly Income*	Upah bulanan pokok pekerja, biasanya bagian dari gaji tahunan (annual salary) yang dibagi dengan 12 bulan kerja
20	*Monthly Rate*	Upah bulanan pekerja yang sudah ditambahkan dengan bonus, seperti penyelesaian project dan sebagainya.
21	*Number Companies Worked*	Total perusahaan yang pernah menjadi tempat bekerja karyawan
22	*Over 18*	Status pekerja dengan usia diatas 18 tahun
23	*Over Time*	Status pekerja yang melakukan kerja dan lembur
24	*Percent Salary Hike*	Persentase kenaikan gaji bulanan
25	*Performance Rating*	Tingkat penilaian atau pengukuran kinerja dari karyawan terhadap standar yang ditentukan perusahaan.
26	*Relationship Satisfaction*	Tingkat kepuasan pekerja dalam menjalin relasi antar sesama pegawai atau keryawan ataupun dengan atasan maupun bawahan di perusahaan
27	*Standard Hours*	Standard waktu kerja yang ditetapkan perusahaan, dapat berupa skala harian, mingguan, atau bulanan.
28	*Stock Option Level*	Opsi pembelian saham dengan harga yang lebih rendah dari harga pasar yang diberikan oleh perusahaan kepada karyawan dengan jabatan eksekutif seperti manajer dan lebih tinggi.
29	*Total Working Years*	Total waktu pegawai secara aktif bekerja selama masa berkarir
30	*Training Times Last Year*	Jarak waktu pegawai terhadap *training* termutakhir yang dilakukan
31	*Work Life Balance*	Tingkat atau ukuran sejauh mana efektivitas dan kepuasan pegawai dalam peran pekerjaan dan keluarga selaras dengan prioritas hidup dari pegawai itu sendiri.
32	*Years At Company*	Total waktu pegawai secara aktif bekerja di perusahaan
33	*Years In Current Role*	Total waktu pegawai secara aktif bekerja pada jabatannya di perusahaan
34	*Years Since Last Promotion*	Jarak waktu pegawai terhadap promosi yang terkahir kali diterima dari perusahaan
35	*YearsWithCurrManager*	Total waktu pegawai secara aktif bekerja dengan manajernya pada kondisi saat ini

Adapun spesifikasi dalam pemilihan *public dataset* ini yaitu dengan memperhatikan lisensi dan kualitas data, yang mana *dataset* yang digunakan pada penelitian ini sudah mempunyai lisensi *Database Contents License* (DbCL) v1.0 seperti pada gambar dibawah ini.



Gambar 4. 3 Lisensi Data

Selain itu, untuk ketentuan (terms and condition) atau *Database Contents Licence* (DbCL) dari penggunaan *dataset* ini adalah sebagai berikut.

Pemberi Lisensi dan Anda menyetujui hal-hal berikut:

1. *Definitions of Capitalised Words*

   Definisi Lisensi Basis Data Terbuka (ODbL) 1.0 dimasukkan melalui referensi ke dalam Lisensi Isi Basis Data.

2. *Rights Granted and Conditions of Use*

   - Hak yang diberikan. Pemberi Lisensi memberikan kepada Anda lisensi hak cipta yang berlaku di seluruh dunia, bebas royalti, non-eksklusif, abadi, dan tidak dapat dibatalkan untuk melakukan tindakan apa pun yang dibatasi oleh hak cipta atas apa pun dalam konten, baik dalam media asli atau media lainnya. Hak-hak ini secara eksplisit mencakup penggunaan komersial, dan tidak mengecualikan bidang usaha apa pun. Hak-hak ini mencakup, tidak terbatas pada, hak untuk mensublisensikan ciptaan.
   - Ketentuan Penggunaan. Anda harus mematuhi ODbL.
   - Hubungan dengan *Database* dan ODbL. Lisensi ini tidak mencakup Hak *database* apapun, hak cipta *database*, atau kontrak apa pun atas konten sebagai bagian dari *database*. Penjelasan lebih detail terkait hal ini dapat dilihat pada cakupan ODbL tentang hak dan kewajiban Anda.

- Tidak adanya penegasan hak cipta atas fakta. Pemberi Lisensi mengambil posisi bahwa informasi faktual tidak dilindungi oleh hak cipta. DbCL memberi Anda izin atas informasi apa pun yang memiliki hak cipta yang terkandung dalam Konten.

3. *Warranties, Disclaimer, and Limitation of Liability*

- Konten dilisensikan oleh Pemberi Lisensi "sebagaimana adanya" dan tanpa jaminan apa pun, baik tersurat maupun tersirat, baik judul, keakuratan, adanya tidak adanya kesalahan, kesesuaian untuk tujuan, atau lainnya. Beberapa yurisdiksi tidak mengizinkan pengecualian jaminan tersirat, sehingga pengecualian ini mungkin tidak berlaku untuk Anda.
- Tunduk pada tanggung jawab apa pun yang tidak dapat dikecualikan atau dibatasi oleh hukum, Pemberi Lisensi tidak bertanggung jawab, dan secara tegas mengecualikan, semua tanggung jawab atas kehilangan atau kerusakan, bagaimana pun dan kapan pun, yang ditimbulkan kepada siapa pun oleh penggunaan apa pun berdasarkan Lisensi ini, baik oleh Anda atau oleh orang lain, dan baik disebabkan oleh kesalahan Pemberi Lisensi atau tidak. Pengecualian tanggung jawab ini mencakup, namun tidak terbatas pada, segala kerugian khusus, insidental, konsekuensial, hukuman, atau ganti rugi yang patut dicontoh. Pengecualian ini berlaku meskipun Pemberi Lisensi telah diberitahu tentang kemungkinan kerugian tersebut.
- Jika tanggung jawab tidak dapat dikecualikan berdasarkan hukum, maka tanggung jawab tersebut terbatas pada kerugian finansial aktual dan langsung sepanjang tanggung jawab tersebut disebabkan oleh kelalaian Pemberi Lisensi yang terbukti.

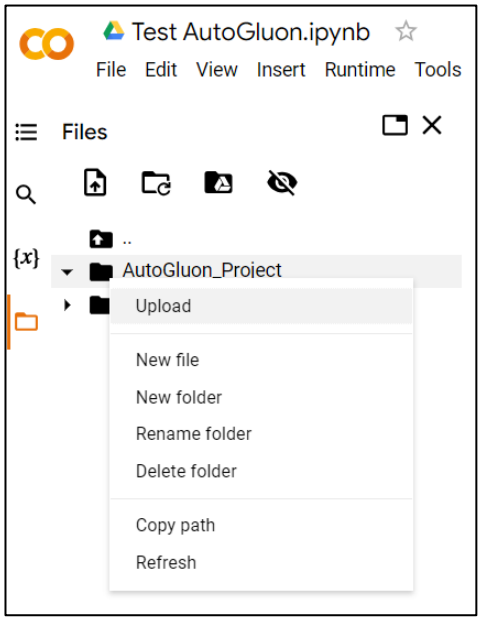
4.2 Pengolahan Data

Tahapan pengolahan data ini mengikuti alur *data science*, seperti yang telah dijelaskan pada bab sebelumnya, yang dimulai dari data pre-*process*ing, EDA, Implementasi Model *Machine* Learnig, hingga Visualisasi model.

4.2.1 Data Pre-*Process*ing

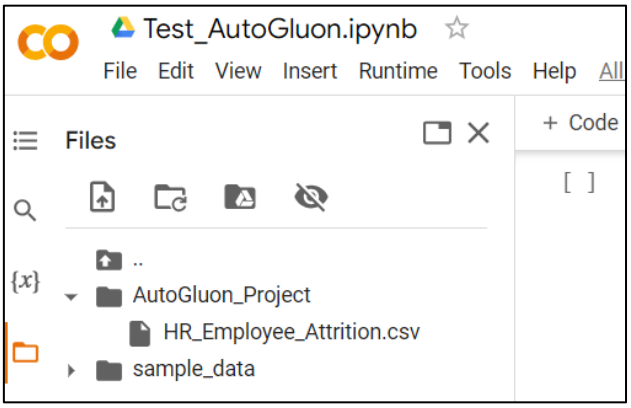
Pada tahapan ini diawali dengan mengakses *google* colab sebagai integrated development environment (IDE) yang digunakan dalam mengolah *dataset* IBM HR *Analytics Employee Attrition*. IDE ini sudah terintegrasi dengan bahasa pemrograman *python* dan berbasis cloud,

sehingga dalam hal akses dan pengoperasiannya dilakukan melalui jaringan internet. Berikut merupakan tampilan dari *google* colab



Gambar 4. 4 Pembuatan *Folde*r Pada *Google* Colab

Gambar diatas merupakan tahapan untuk membuat *folde*r pada *google* colab yang diberi nama *AutoGluon*_Project. Selanjutnya adalah melakukan upload *dataset* CSV yang telah di download pada pembahasan bab sebelumnya.



Gambar 4. 5 *Import Dataset*

Gambar diatas menunjukan bahwa upload file HR_*Employee_Attrition* telah berhasil, selanjutnya melakukan *import library* sebagai alat bantu dalam mengolah data yakni sebagai berikut.

```
#import certain library to access and visualize dataset

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

Gambar 4. 6 *Import Library Python*

Library seaborn dan *matplotlib* digunakan sebagai alat bantu visualisasi *dataset*, sementara *library pandas* digunakan untuk mengoperasikan, analisis, dan manipulasi data. Setelah *import library* berhasil, maka penggunaan fungsi .head() untuk pemanggilan *preview* data dapat dilakukan seperti pada gambar berikut.

```
# read dataset

data = pd.read_csv("/content/AutoGluon_Project/HR_Employee_Attrition.csv")
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relation
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

Gambar 4. 7 *Preview Dataset*

Gambar diatas merupakan *preview* 5 baris *entry* pertama pada *dataset* dengan menggunakan fungsi .head() melalui *library* pandas. Dari *preview* tersebut dapat dilihat bahwa *dataset* IBM HR *analytics employee attrition* yang memiliki 35 *variable* sudah dapat terbaca oleh sistem yang digunakan, sehingga tahapan selanjutnya dapat dilakukan.

4.2.2 Exploratory Data Analysis (EDA)

Tahapan ini dimulai dengan mencari informasi terkait tipe dari 35 *variable* yang terdapat dalam *dataset*. Hal ini dapat diketahui dengan menjalankan *function* .info() seperti gambar dibawah ini

```
# EDA
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

Gambar 4. 8 Data Info

Dari gambar diatas dapat dilihat bahwa semua *variable* mempunyai *entry* data yang sama yaitu 1470 yang artinya tidak terdapat *entry* data yang kosong dalam *dataset* yang sedang diolah. Selain itu, dapat diketahui bahwa pada *dataset* ini terdapat beberapa *variable* yang memiliki tipe data yang berbeda (*object*/*string*) dengan mayoritas *variable* (int64/*numeric*).

```
data.isnull().sum()

Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

Gambar 4. 9 Menghitung *Entry* Data *Null*

Gambar diatas adalah tahapan yang dilakukan dengan tujuan untuk memastikan kembali bahwa tidak terdapat data null atau *entry* kosong pada *dataset*. Dari *output* yang diperoleh pada gambar 4.9, artinya sudah tidak terdapat *entry* kosong pada *dataset*, sehingga dapat dilanjutkan pada tahapan ringkasan statistika deskriptif seperti pada gambar dibawah ini

```
data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	Relat
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	

8 rows × 26 columns

Gambar 4. 10 Data *Describe*

Fungsi .describe() yang di eksekusi pada gambar diatas dilakukan dengan tujuan untuk memperoleh ringkasan statistika deskriptif seperti rerata (mean), standar deviasi (std), Q1 (25%), dan sebagainya. Dari gambar diatas juga dapat diketahui bahwa fungsi yang di eksekusi untuk menampilkan *output* ringkasan statistika deskriptif, hanya dapat dijalankan pada 25 *variable* data numerik (int64), yang artinya terdapat 10 *variable* lainnya merupakan *variable* data kategori (*object*). Oleh karena itu, digunakan fungsi berikut untuk melakukan mengetahui deskripsi dari data kategori tersebut

```
for column in data.select_dtypes(include=['object']):
    print(f"Column: {column}")
    print(data[column].value_counts())
    print()
```

Gambar 4. 11 Pemanggilan Data *Object*

Fungsi *looping* diatas bertujuan untuk menghitung jumlah tiap kategori pada data *object* secara otomatis. Adapun *output* dari *syntax* diatas adalah sebagai berikut

```
Column: Attrition
No     1233
Yes     237
Name: Attrition, dtype: int64

Column: BusinessTravel
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64

Column: Department
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64

Column: EducationField
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64

Column: Gender
Male      882
Female    588
Name: Gender, dtype: int64

Column: JobRole
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64

Column: MaritalStatus
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64

Column: Over18
Y    1470
Name: Over18, dtype: int64

Column: OverTime
No     1054
Yes     416
Name: OverTime, dtype: int64
```

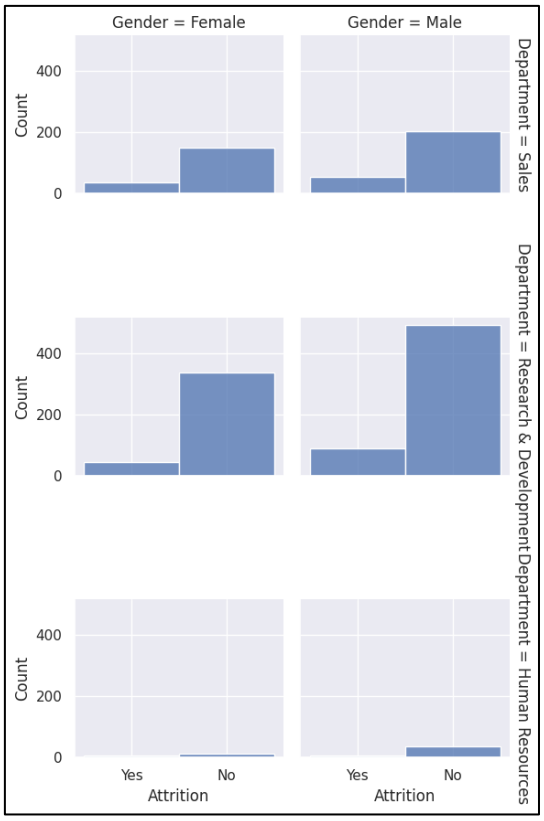
Gambar 4. 12 *Output* Data *Object*

Dari gambar diatas, dapat diketahui bahwa terdapat 3 *variable* data *binnary* yaitu *Gender*, *Overtime*, dan *Attrition*. Sementara *variable* Over18 hanya memiliki satu tipe respon, dengan *variable* lainnya memiliki 3 dan bahkan hingga 9 kategori. Dalam hal ini, *variable* dengan tipe data *binnary* lebih diutamakan pada pembuatan visualisasi karena salah satu dari variabel tersebut akan dijadikan sebagai prediktor pada model *machine learning* yang akan dibangun. Visualisasi ini dilakukan untuk memperoleh gambaran umum terhadap statistik data. Adapun outpu dari visualisasi yang dimaksud dapat diperoleh dengan menjalankan *code* seperti pada gambar dibawah ini.

```
sns.set_theme(style="darkgrid")
df = data
sns.displot(
    df, x="Attrition", col="Gender", row="Department",
    binwidth=3, height=3, facet_kws=dict(margin_titles=True),
)
```

Gambar 4. 13 Pemanggilan Visualisasi Data *Object*

Pemilihan *variable Attrition*, *Gender*, dan Department pada *syntax* diatas dilandaskan atas kecenderungan dalam tampilan report atau dashboard terkait topik HR dimana salah satu task penting yang harus dapat disajikan oleh report adalah ringkasan terkait jumlah departemen berdasarkan *gender*, serta berdasarkan beberapa skenario yang dalam hal ini skenarionya adalah berdasarkan kategori *Attrition*.



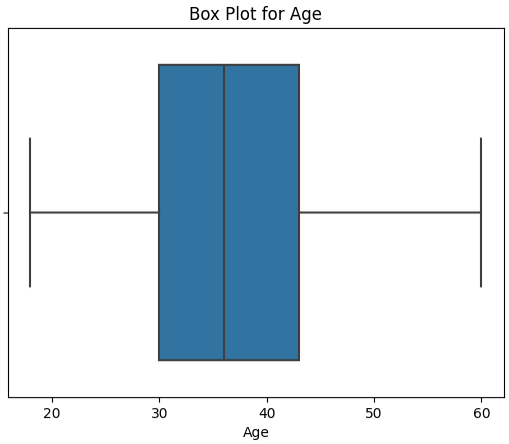
Gambar 4. 14 *Output* Visualisasi Data *Object*

Gambar diatas merupakan bar chart, yang mana dapat diketahui bahwa pada tiap department lebih didominasi oleh karyawan dengan respon "No" untuk kategori *Attrition* yang artinya bahwa *dataset* ini bersifat *imbalance*, sehingga pada evaluator model lebih diutamakan untuk menggunakan *precission*/*recall* dibanding evaluator lainnya. Penggunaan bar chart pada data kategori diatas, didasarkan atas karakteristik dari bar chart yang mudah untuk dipahami dan kebutuhan data kategorikal yang diolah. Namun, hal ini tidak dapat diterapkan pada data numerikal karena tujuan utamanya tidak hanya sekedar mencari gambaran umum terhadap data yang diolah, tetapi juga untuk mencari *outlier*s dari tiap variabel. Adapun untuk *code* yang dijalankan untuk memperoleh visualisasi pada data numerik adalah sebagai berikut.

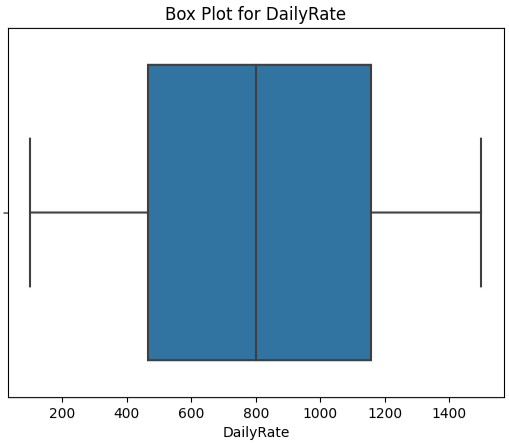
```
int_columns = data.select_dtypes(include=['int'])

# Loop through each integer column and create box plots
for column in int_columns.columns:
    sns.boxplot(x=data[column])
    plt.title(f'Box Plot for {column}')
    plt.xlabel(column)
    plt.show()
```

Gambar 4. 15 Pemanggilan Visualisasi Data *Integer*

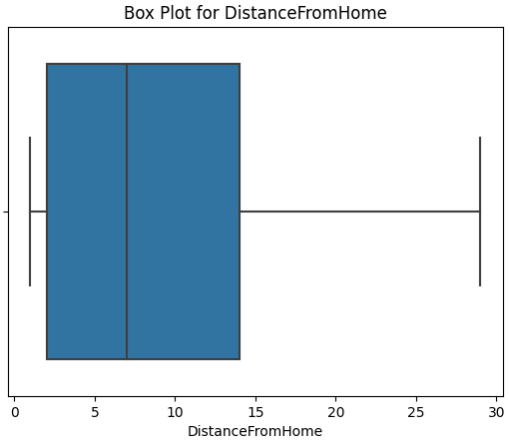
Syntax diatas digunakan untuk membuat visualisasi jenis boxplot pada setiap data *integer* dengan tujuan untuk mencari tahu ada atau tidaknya *outlier*s pada tiap *variable* didalam *dataset*.



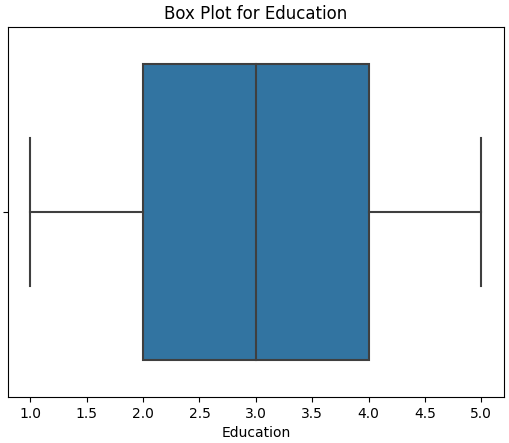
Gambar 4. 16 *Box plot* Variabel *Age*



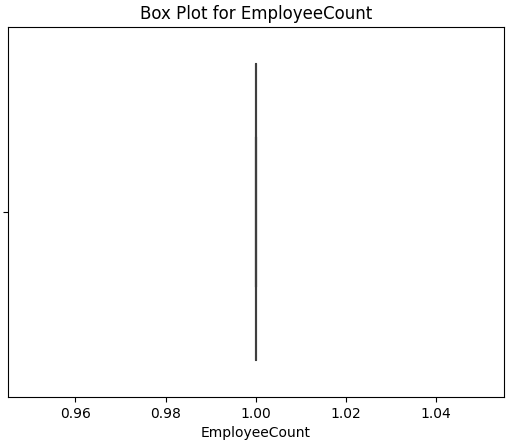
Gambar 4. 17 *Box plot* Variabel *DailyRate*



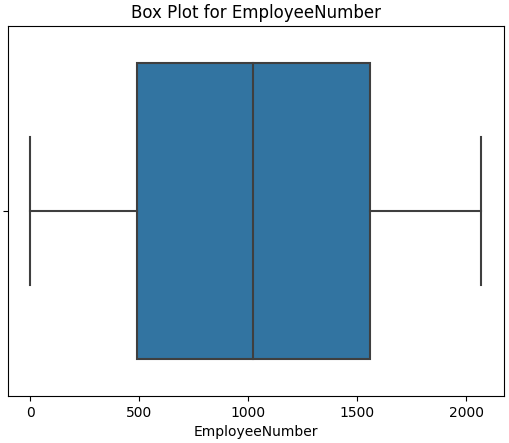
Gambar 4. 18 *Box plot* Variabel *DistanceFromHome*



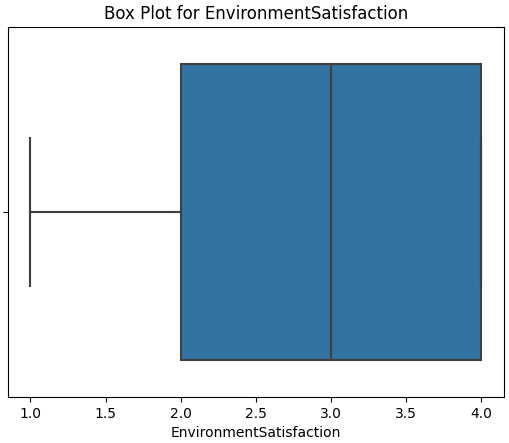
Gambar 4. 19 *Box plot* Variabel *Education*



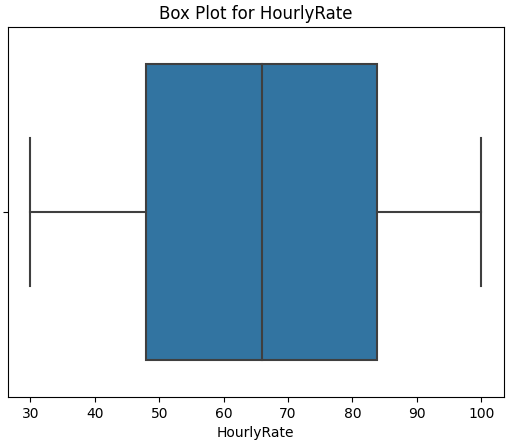
Gambar 4. 20 *Box plot* Variabel *EmployeeCount*



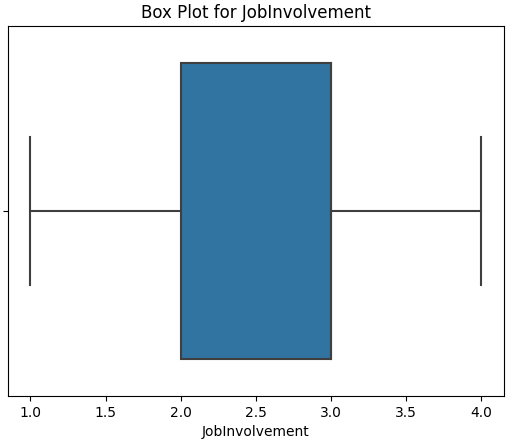
Gambar 4. 21 *Box plot* Variabel *EmployeeNumber*



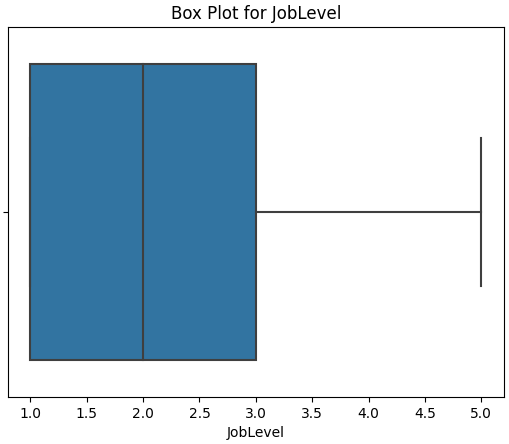
Gambar 4. 22 *Box plot* Variabel *EnvironmentSatisfaction*



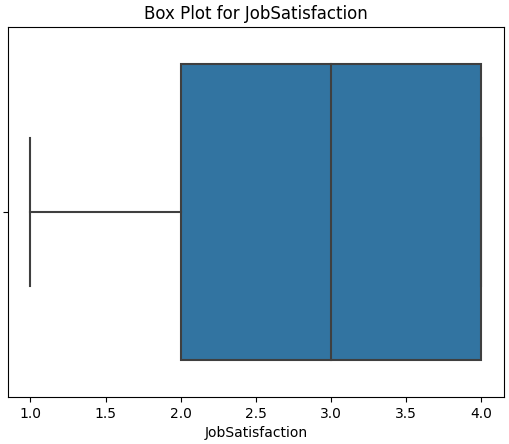
Gambar 4. 23 *Box plot* Variabel *HourlyRate*



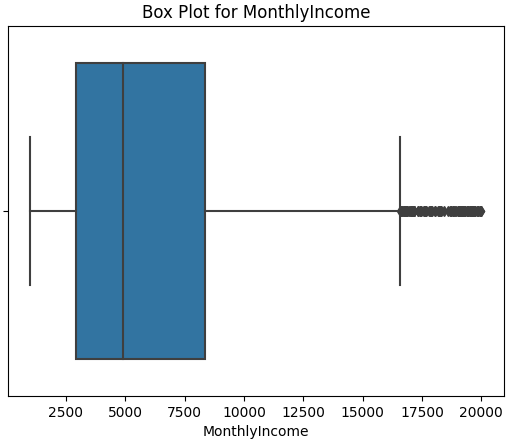
Gambar 4. 24 *Box plot* Variabel *jobinvolvement*



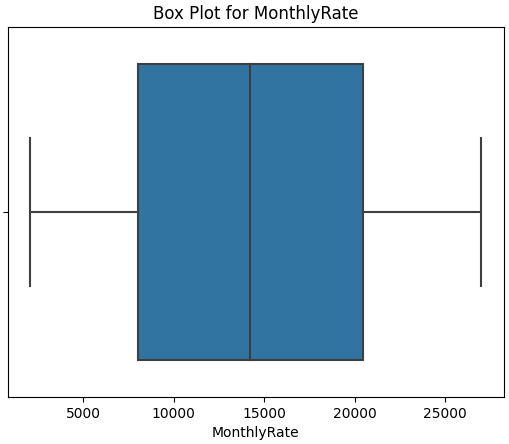
Gambar 4. 25 *Box plot* Variabel *JobLevel*



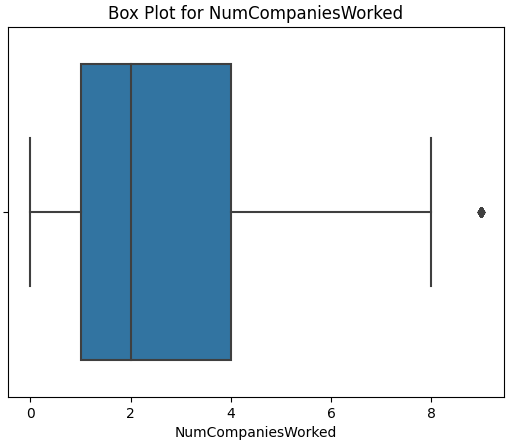
Gambar 4. 26 *Box plot* Variabel *JobSatisfaction*



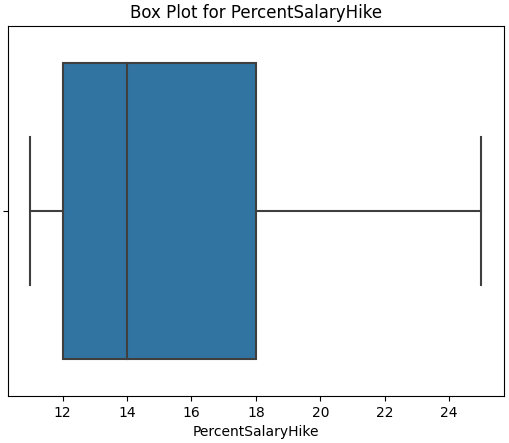
Gambar 4. 27 *Box plot* Variabel *MonthlyIncome*



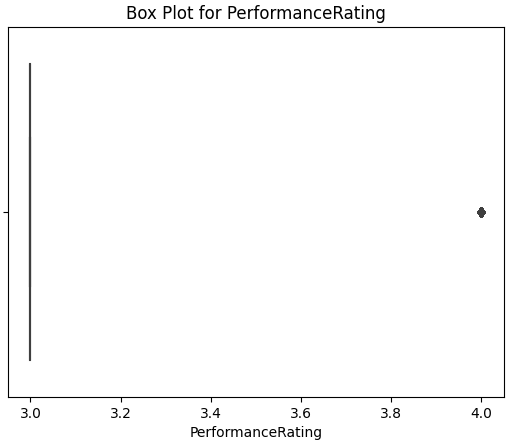
Gambar 4. 28 *Box plot* Variabel *MonthlyRate*



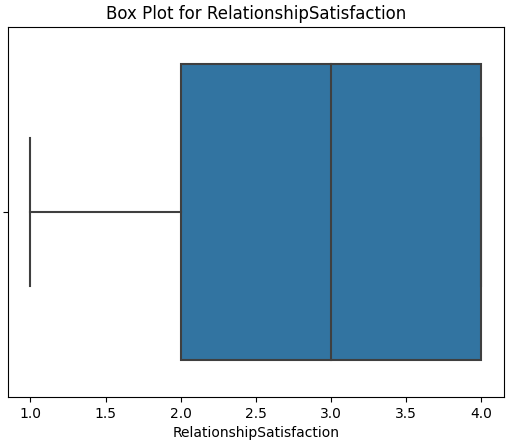
Gambar 4. 29 *Box plot* Variabel *NumCompaniesWorked*



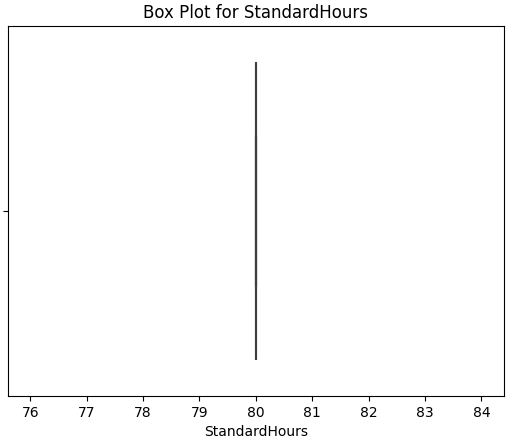
Gambar 4. 30 *Box plot* Variabel *PercentSalaryHike*



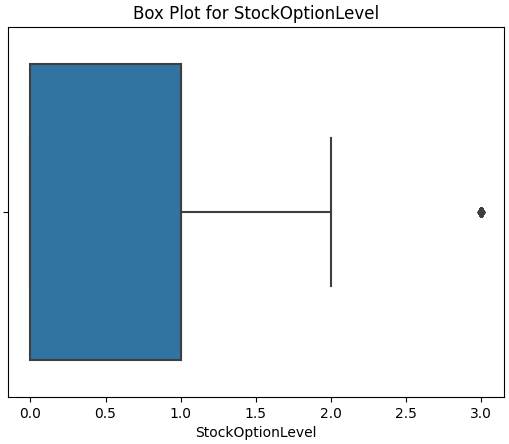
Gambar 4. 31 *Box plot* Variabel *PerformanceRating*



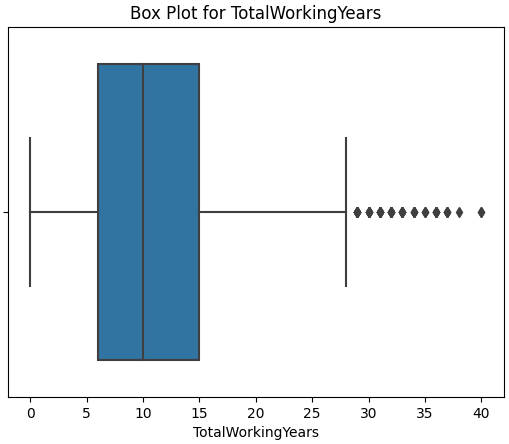
Gambar 4. 32 *Box plot* Variabel *RelationshipSatisfaction*



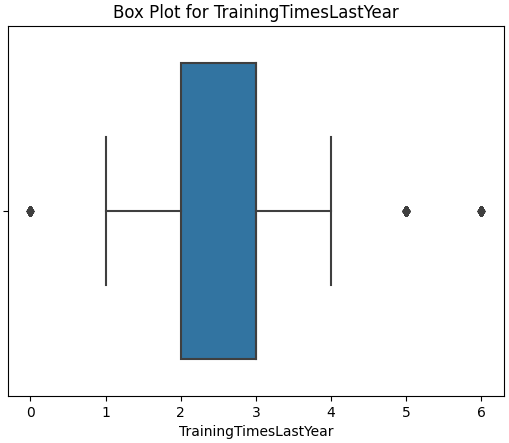
Gambar 4. 33 *Box plot* Variabel *StandardHours*



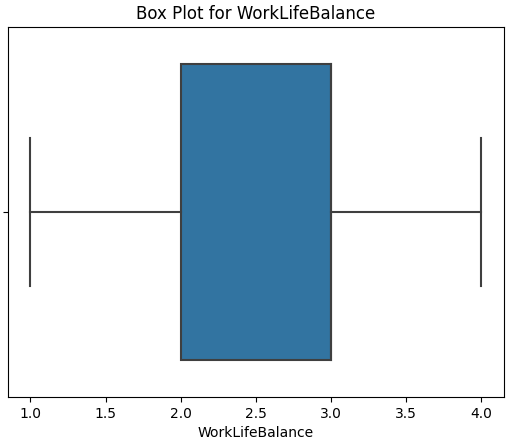
Gambar 4. 34 *Box plot* Variabel *StockOptionLevel*



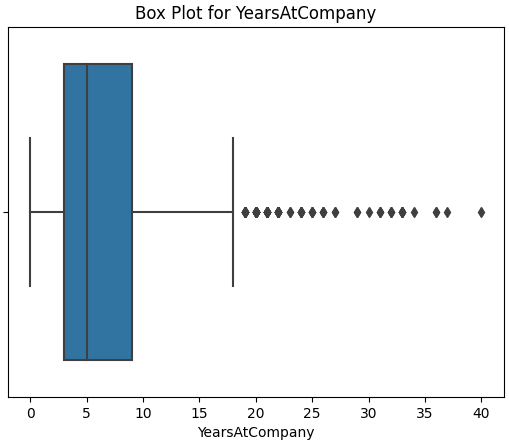
Gambar 4. 35 *Box plot* Variabel *TotalWorkingYears*



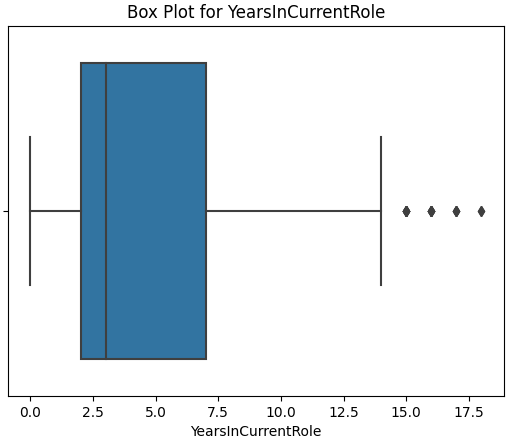
Gambar 4. 36 *Box plot* Variabel *TrainingTimesLastYear*



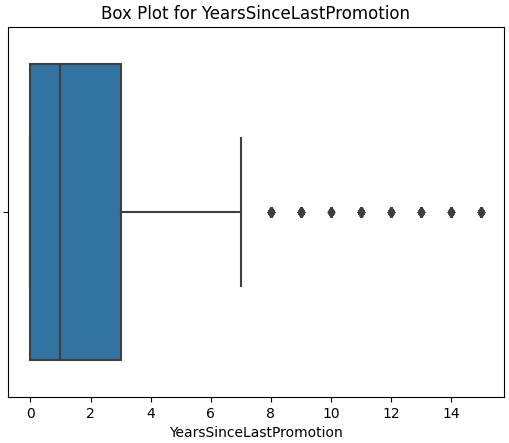
Gambar 4. 37 *Box plot* Variabel *WorkLifeBalance*



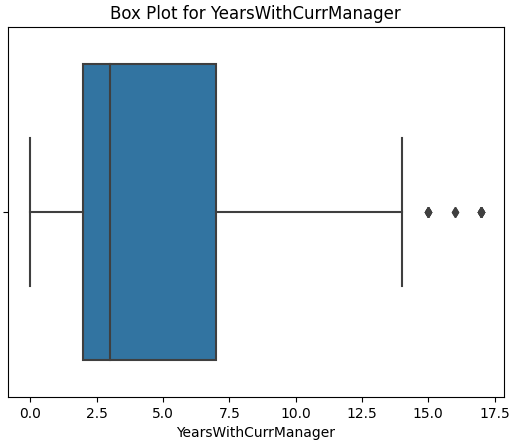
Gambar 4. 38 *Box plot* Variabel *YearsAtCompany*



Gambar 4. 39 *Box plot* Variabel *YearsInCurrentRole*



Gambar 4. 40 *Box plot* Variabel *YearsISinceLastPromotion*



Gambar 4. 41 *Box plot* Variabel *YearsWithCurrentManager*

Berdasarkan hasil visualisasi terhadap data *integer* dan categorical diatas, dapat diketahui bahwa terdapat beberapa *feature* memiliki *outlier* dan *single value entry* (data *entry* yang hanya memiliki satu nilai) yang dapat memberi pengaruh buruk terhadap model *machine learning* yang nantinya dihasilkan. Sehingga dilakukan tahapan *drop* data untuk menghapus data tersebut.

```
# Drop a column
feature_to_drop = [
    'StandardHours', 'EmployeeCount', #integer single value entry
    'YearsWithCurrManager','YearsSinceLastPromotion','YearsInCurrentRole',
    'YearsAtCompany','TrainingTimesLastYear','TotalWorkingYears',
    'StockOptionLevel','PerformanceRating','NumCompaniesWorked',
    'MonthlyIncome', #outliers
    'Over18' #categorical single value entry
    ]

data2 = data.drop(feature_to_drop, axis=1)
```

```
data2.shape
```

```
(1470, 22)
```

Gambar 4. 42 *Drop Column*

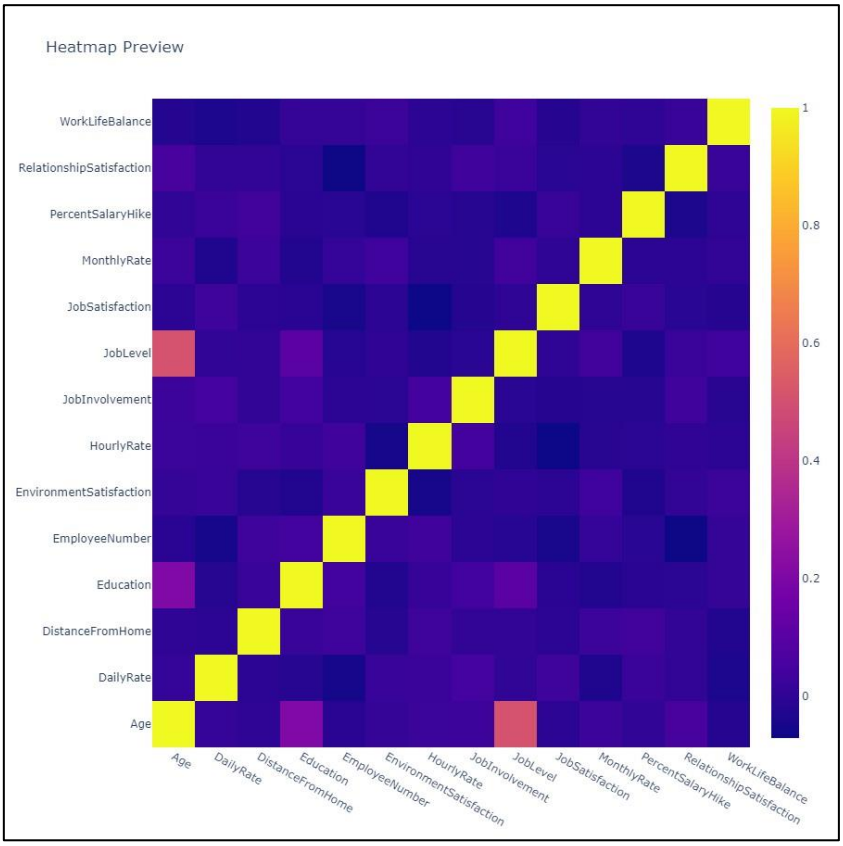
Adapun rincian *variable* atau *feature* yang dieliminasi dan tidak dieliminasi adalah sebagai berikut.

Tabel 4. 3 Rincian *Feature* yang dieliminasi dan tidak

No	***Feature***	Keterangan
1	*Age*	Tidak dieliminasi
2	*Attrition*	Tidak dieliminasi
3	*Business Travel*	Tidak dieliminasi
4	*Daily Rate*	Tidak dieliminasi
5	*Department*	Tidak dieliminasi
6	*Distance From Home*	Tidak dieliminasi
7	*Education*	Tidak dieliminasi
8	*Education Field*	Tidak dieliminasi
9	*Employee Count*	Dieliminasi karena merupakan *single entry value*
10	*Employee Number*	Tidak dieliminasi
11	*Environment Satisfaction*	Tidak dieliminasi
12	*Gender*	Tidak dieliminasi
13	*Hourly Rate*	Tidak dieliminasi
14	*Job Involvement*	Tidak dieliminasi
15	*JobLevel*	Tidak dieliminasi
16	*Job Role*	Tidak dieliminasi
17	*Job Satisfaction*	Tidak dieliminasi
18	*Marital Status*	Tidak dieliminasi
19	*Monthly Income*	Dieliminasi karena memiliki *outliers*
20	*Monthly Rate*	Tidak dieliminasi
21	*Number Companies Worked*	Dieliminasi karena memiliki *outliers*
22	*Over 18*	Dieliminasi karena merupakan *single entry value*
23	*Over Time*	Tidak dieliminasi
24	*Percent Salary Hike*	Tidak dieliminasi
25	*Performance Rating*	Dieliminasi karena memiliki *outliers*

No	*Feature*	Keterangan
26	*Relationship Satisfaction*	Tidak dieliminasi
27	*Standard Hours*	Dieliminasi karena merupakan *single entry value*
28	*Stock Option Level*	Dieliminasi karena memiliki *outlier*s
29	*Total Working Years*	Dieliminasi karena memiliki *outlier*s
30	*Training Times Last Year*	Dieliminasi karena memiliki *outlier*s
31	*Work Life Balance*	Tidak dieliminasi
32	*Years At Company*	Dieliminasi karena memiliki *outlier*s
33	*Years In Current Role*	Dieliminasi karena memiliki *outlier*s
34	*Years Since Last Promotion*	Dieliminasi karena memiliki *outlier*s
35	*YearsWithCurrManager*	Dieliminasi karena memiliki *outlier*s

Berdasarkan tabel diatas maka dapat diketahui bahwa terdapat total 13 *feature* yang dieliminasi dengan rincian 3 *feature* merupakan *single entry value* atau hanya memiliki *entry* 1 nilai yang sama. Sementara 10 *feature* lainnya memiliki *outlier*s, sehingga *feature* yang tersisa untuk digunakan pada tahapan membangun model *machine learning* adalah 22 *feature*

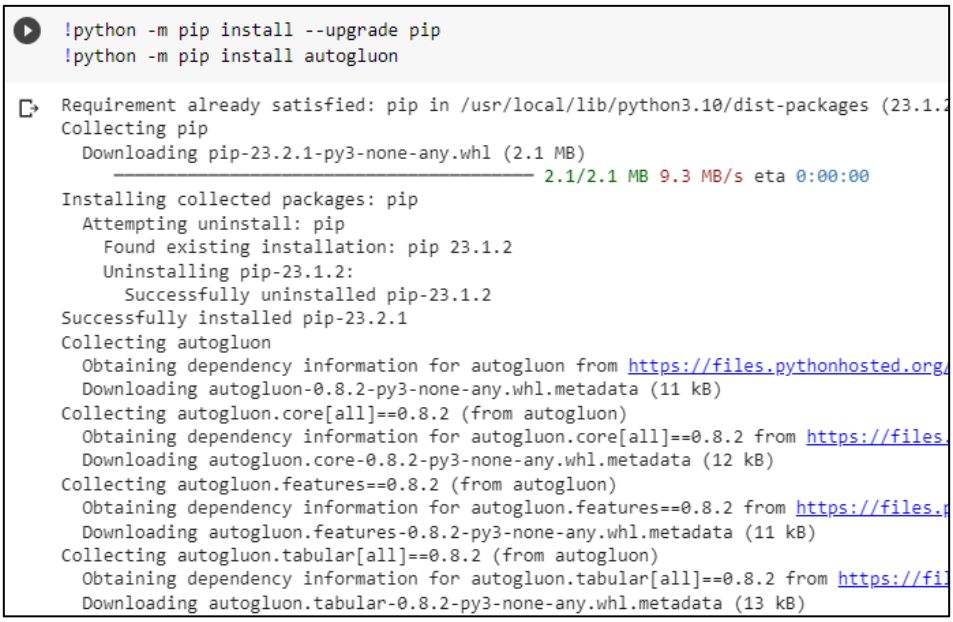


Gambar 4. 43 *Heatmap* Diagram

Dari *heatmap* diagram diatas dapat dilihat bahwa dari 14 variabel yang bertipe *integer* yang tersisa setelah dilakukan eliminasi terhadap beberapa variabel bahwa untuk job level dengan age mempunyai relasi yang tertinggi diantara relasi antar 2 *variable* lainnya. Artinya bahwa, pada dasarya usia pekerja berpengaruh terhadap level jabatan pekerja tersebut diperusahaannya.

4.2.3 Implementasi Model *Machine Learning*

Pada tahapan implementasi model *machine learning* ini, diawali dengan melakukan instalasi *package framework AutoGluon* pada *google* colab seperti pada gambar dibawah ini



Gambar 4. 44 Instalasi *Package AutoGluon*

Setelah proses instalasi berhasil, langkah selanjutnya adalah melakukan *import train* dan *test dataset* sesuai dengan ketentuan yang dijelaskan pada bab sebelumnya. Adapun tahapan dalam *import dataset* tersebut adalah sebagai berikut

Tabel 4. 4 Persentase *Split* Data

Dataset HR	*Entry* Row	Persetase
train	1176	80%
test	294	20%
total/persentase	1470	100%

Berdasarkan tabel diatas, dilakukan pembagian data dengan cutting dataframe dengan menggunakan fungsi .iloc() melalui pandas *library* sesuai dengan jumlah *entry* row pada tabel

diatas yaitu untuk *train*_data diambil dari *entry* baris pertama hingga ke – 1176 dan *test*_data dimulai dari *entry* ke – 1177 hingga 1470 atau baris paling terakhir.

```
train_data = TabularDataset(data2.loc[:1176])
train_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	2	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	2	3	...
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	4	4	...
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	5	4	...
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	7	1	...

5 rows × 22 columns

Gambar 4. 45 *Import Train and Test Dataset*

Selanjutnya adalah melakukan *import library AutoGluon tabular* dan *package* pendukungnya untuk melakukan *train* pada *dataset* yang sudah dibagi. Kemudian menentukan variabel yang akan dijadikan sebagai *predictor* dengan menggunakan *syntax* pada gambar berikut.

```
label = 'Attrition'
train_data[label].describe()
```

```
count     1176
unique       2
top         No
freq       986
Name: Attrition, dtype: object
```

Gambar 4. 46 *Variable Predictor*

Pada gambar diatas dapat dilihat bahwa respon "No" untuk *variable attrition* lebih banyak dibandingkan dengan respon "Yes" yaitu sebanyak 986 dari 1176 *dataset* yang dilatih. Dapat dilihat bahwa hasil pada gambar diatas sesuai dengan visualisasi yang dilakukan pada tahapan EDA dimana untuk *variable attrition* ini memang merupakan imbalance *dataset*, sehingga evaluator model *machine learning* yang lebih kompatibel dengan kondisi seperti ini adalah *precission*/*recall*.

```
predictor = TabularPredictor(label=label).fit(train_data)

No path specified. Models will be saved in: "AutogluonModels/ag-20231104_092816/"
Beginning AutoGluon training ...
AutoGluon will save models to "AutogluonModels/ag-20231104_092816/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Linux
Platform Machine:   x86_64
Platform Version:   #1 SMP Wed Aug 30 11:19:59 UTC 2023
Disk Space Avail:   79.82 GB / 115.66 GB (69.0%)
Train Data Rows:    1176
Train Data Columns: 34
Label Column: Attrition
Preprocessing data ...
AutoGluon infers your prediction problem is: 'binary' (because only two unique label-values observed).
        2 unique label values:  ['No', 'Yes']
        If 'binary' is not the correct problem_type, please manually specify the problem_type parameter du
Selected class <--> label mapping:  class 1 = Yes, class 0 = No
        Note: For your binary classification, AutoGluon arbitrarily selected which label-value represents
        To explicitly set the positive_class, either rename classes to 1 and 0, or specify positive_class
Using Feature Generators to preprocess the data ...
Fitting AutoMLPipelineFeatureGenerator...
        Available Memory:                    12311.72 MB
        Train Data (Original)  Memory Usage: 0.87 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set feature_metadata_in to manually sp
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator...
                        Note: Converting 3 features to boolean dtype as they only contain 2 unique values.
```

Gambar 4. 47 *Training* Data

Gambar diatas merupakan eksekusi *syntax* untuk melakukan *training* data, yang dalam hal ini dapat terlihat bahwa *AutoGluon* secara otomatis dapat membaca variabel *attrition* sebagai data *binnary*. Selain itu, *AutoGluon* juga secara otomatis membuat *fold*er baru untuk menyimpan model *machine learning* yang dijalankan yaitu dengan nama "*AutoGluon*Models/ag-20231104_092816".

```
test_data = TabularDataset(data2.loc[1176:1470])

y_pred = predictor.predict(test_data.drop(columns=[label]))
y_pred.head()

1176    No
1177    No
1178    No
1179    No
1180    No
Name: Attrition, dtype: object
```

Gambar 4. 48 Pemanggilan Data *Variable Predictor*

Setelah melakukan *training* pada *dataset train*, selanjutnya melakukan pemanggilan data untuk *variable predictor* sebagai *predictor* pada model *machine learning* yang akan dijalankan.

```
predictor.evaluate(test_data, silent=True)

{'accuracy': 0.8605442176870748,
 'balanced_accuracy': 0.5425275641619568,
 'mcc': 0.21256889923523414,
 'roc_auc': 0.7979245807467803,
 'f1': 0.16326530612244897,
 'precision': 0.6666666666666666,
 'recall': 0.09302325581395349}
```

Gambar 4. 49 Evaluasi *Predictor*

Setelah pemanggilan data *variable predictor* berhasil dilakukan, selanjutnya melakukan evaluasi terhadap model prediksi yang telah dibangun dengan menggunakan *syntax* pada gambar diatas. Dapat dilihat bahwa terdapat 7 parameter evaluasi model yaitu *accuracy*, balanced_*accuracy*, *mcc, roc_auc, f1, precision*, dan *recall*. Terakhir, dengan menggunakan fungsi .*Leaderboard*() untuk memperoleh ranking terbaik pada *test* data dari model yang dibangun seperti pada gambar dibawah ini.

```
predictor.leaderboard(test_data, silent=True)
```

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	XGBoost	0.870748	0.877119	0.018725	0.010430	0.451818	0.018725	0.010430	0.451818	1	True	11
1	CatBoost	0.863946	0.881356	0.010135	0.011950	4.775602	0.010135	0.011950	4.775602	1	True	7
2	LightGBMXT	0.863946	0.872881	0.016283	0.010052	1.609744	0.016283	0.010052	1.609744	1	True	3
3	LightGBMLarge	0.863946	0.860169	0.034495	0.012489	1.225199	0.034495	0.012489	1.225199	1	True	13
4	RandomForestGini	0.863946	0.847458	0.114494	0.069925	1.002258	0.114494	0.069925	1.002258	1	True	5
5	NeuralNetTorch	0.860544	0.860169	0.019192	0.017624	2.655602	0.019192	0.017624	2.655602	1	True	12
6	LightGBM	0.860544	0.881356	0.028380	0.015737	0.711325	0.028380	0.015737	0.711325	1	True	4
7	WeightedEnsemble_L2	0.860544	0.881356	0.031128	0.017029	1.574787	0.002748	0.001292	0.863462	2	True	14
8	ExtraTreesEntr	0.860544	0.855932	0.141055	0.091441	0.816667	0.141055	0.091441	0.816667	1	True	9
9	RandomForestEntr	0.857143	0.855932	0.115191	0.070062	0.963463	0.115191	0.070062	0.963463	1	True	6
10	ExtraTreesGini	0.857143	0.851695	0.132993	0.081382	1.057413	0.132993	0.081382	1.057413	1	True	8
11	NeuralNetFastAI	0.853741	0.868644	0.027517	0.015574	1.848619	0.027517	0.015574	1.848619	1	True	10
12	KNeighborsUnif	0.850340	0.822034	0.017620	0.019632	1.525858	0.017620	0.019632	1.525858	1	True	1
13	KNeighborsDist	0.846939	0.817797	0.015565	0.016148	0.011902	0.015565	0.016148	0.011902	1	True	2

Gambar 4. 50 *Output Predictor Leaderboard*

Dari *output* diatas, dapat diketahui bahwa model terbaik berdasarkan hasil *score*_val adalah *WeightedEnsemble_L2* dengan capaian skor sebesar 0,88 . Adapun dari tampilan gambar diatas merupakan kecenderungan dari *AutoGluon* yang menampilkan urutan model berdasarkan *score_test* sehingga terlihat bahwa model *XGBoost* berada diperingkat pertama namun *score*_val dari model ini lebih rendah jika dibandingkan dengan model *WeightedEnsemble_L2*.

4.2.4 Visualisasi Model *Machine Learning*

Tahapan ini diawali dengan melakukan konversi *output* data *feature importance* menjadi dataframe melalui microsoft excel.

```
#feature importance
predictor.feature_importance(data=test_data)
```

```
Computing feature importance via permutation shuffling for 21 features using 294 rows with 5 shuffle sets...
        6.17s   = Expected runtime (1.23s per shuffle set)
        1.81s   = Actual runtime (Completed 5 of 5 shuffle sets)
```

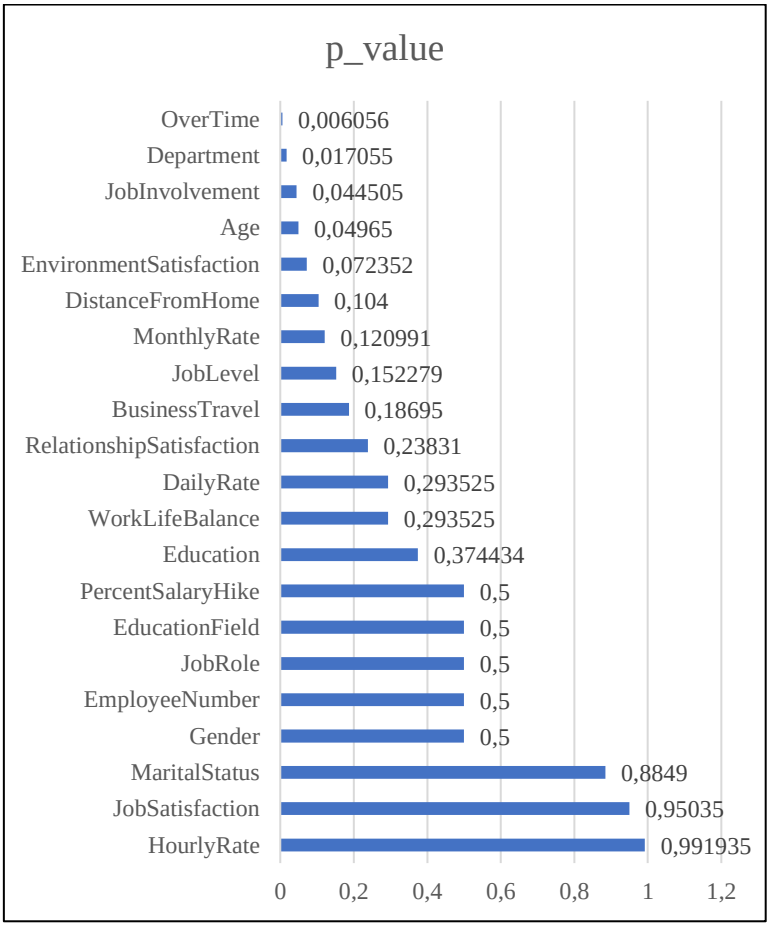
	importance	stddev	p_value	n	p99_high	p99_low
OverTime	0.010884	0.005589	0.006056	5	0.022392	-0.000623
JobInvolvement	0.006803	0.006803	0.044505	5	0.020810	-0.007204
EnvironmentSatisfaction	0.004082	0.005045	0.072352	5	0.014469	-0.006306
Department	0.003401	0.002405	0.017055	5	0.008354	-0.001551
Age	0.002721	0.002846	0.049650	5	0.008581	-0.003138
MonthlyRate	0.002721	0.004435	0.120991	5	0.011852	-0.006410
DistanceFromHome	0.002041	0.003042	0.104000	5	0.008305	-0.004223
JobLevel	0.002041	0.003878	0.152279	5	0.010026	-0.005944
RelationshipSatisfaction	0.001361	0.003878	0.238310	5	0.009346	-0.006625
WorkLifeBalance	0.001361	0.005158	0.293525	5	0.011982	-0.009261
DailyRate	0.001361	0.005158	0.293525	5	0.011982	-0.009261
BusinessTravel	0.000680	0.001521	0.186950	5	0.003812	-0.002452
Education	0.000680	0.004435	0.374434	5	0.009812	-0.008451
Gender	0.000000	0.000000	0.500000	5	0.000000	0.000000
EmployeeNumber	0.000000	0.000000	0.500000	5	0.000000	0.000000
JobRole	0.000000	0.004810	0.500000	5	0.009904	-0.009904
EducationField	0.000000	0.000000	0.500000	5	0.000000	0.000000
JobSatisfaction	-0.002721	0.002846	0.950350	5	0.003138	-0.008581
HourlyRate	-0.002721	0.001521	0.991935	5	0.000411	-0.005853
MaritalStatus	-0.003401	0.005378	0.884900	5	0.007672	-0.014475
PercentSalaryHike	-0.003401	0.000000	0.500000	5	-0.003401	-0.003401

Gambar 4. 51 *Output Feature Importance*

	importance	stddev	p_value	n	p99_high	p99_low
OverTime	0.010884	0.005589	0.006056	5	0.022392	-0.000623
JobInvolvement	0.006803	0.006803	0.044505	5	0.020810	-0.007204
EnvironmentSatisfaction	0.004082	0.005045	0.072352	5	0.014469	-0.006306
Department	0.003401	0.002405	0.017055	5	0.008354	-0.001551
Age	0.002721	0.002846	0.049650	5	0.008581	-0.003138
MonthlyRate	0.002721	0.004435	0.120991	5	0.011852	-0.006410
DistanceFromHome	0.002041	0.003042	0.104000	5	0.008305	-0.004223
JobLevel	0.002041	0.003878	0.152279	5	0.010026	-0.005944
RelationshipSatisfaction	0.001361	0.003878	0.238310	5	0.009346	-0.006625
WorkLifeBalance	0.001361	0.005158	0.293525	5	0.011982	-0.009261
DailyRate	0.001361	0.005158	0.293525	5	0.011982	-0.009261
BusinessTravel	0.000680	0.001521	0.186950	5	0.003812	-0.002452
Education	0.000680	0.004435	0.374434	5	0.009812	-0.008451
Gender	0.000000	0.000000	0.500000	5	0.000000	0.000000
EmployeeNumber	0.000000	0.000000	0.500000	5	0.000000	0.000000
JobRole	0.000000	0.004810	0.500000	5	0.009904	-0.009904
EducationField	0.000000	0.000000	0.500000	5	0.000000	0.000000
JobSatisfaction	-0.002721	0.002846	0.950350	5	0.003138	-0.008581
HourlyRate	-0.002721	0.001521	0.991935	5	0.000411	-0.005853
MaritalStatus	-0.003401	0.005378	0.884900	5	0.007672	-0.014475
PercentSalaryHike	-0.003401	0.000000	0.500000	5	-0.003401	-0.003401

Gambar 4. 52 Konversi Data *Output Feature Importance*

Berdasarkan *output feature importance* diatas, dapat dilihat bahwa selama proses pelatihan data tidak terdapat *variable* yang di eliminasi atau reduksi oleh fungsi *.feature_importance*() , sehingga tetap menyisahkan 22 *variable* seperti sebelum.



Gambar 4. 53 P-*value Feature Importance*

Berdasarkan P-*value* diatas, dapat diketahui bahwa *feature overtime* merupakan fitur dengan nilai terendah yaitu 0,006056. Artinya bahwa kemungkinan fitur tersebut berbahaya terhadap model *machine learning* yang dihasilkan sangat tinggi, atau dengan kata lain kemungkinan fitur tersebut sangat berguna terhadap model. Selain keempat tahapan diatas, terdapat model *summary* yang dibutuhkan untuk melihat proses *setting* parameter didalam setiap model *machine learning* yang dihasilkan. Adapun model *summary* adalah sebagai berikut

```
predictor.fit_summary()

*** Summary of fit() ***
Estimated performance of each model:
                  model  score_val  pred_time_val  fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0              CatBoost   0.881356       0.011950  4.775602                0.011950           4.775602            1       True          7
1              LightGBM   0.881356       0.015737  0.711325                0.015737           0.711325            1       True          4
2   WeightedEnsemble_L2   0.881356       0.017029  1.574787                0.001292           0.863462            2       True         14
3               XGBoost   0.877119       0.010430  0.451818                0.010430           0.451818            1       True         11
4            LightGBMXT   0.872881       0.010052  1.609744                0.010052           1.609744            1       True          3
5       NeuralNetFastAI   0.868644       0.015574  1.848619                0.015574           1.848619            1       True         10
6         LightGBMLarge   0.860169       0.012489  1.225199                0.012489           1.225199            1       True         13
7        NeuralNetTorch   0.860169       0.017624  2.655602                0.017624           2.655602            1       True         12
8      RandomForestEntr   0.855932       0.070062  0.963463                0.070062           0.963463            1       True          6
9        ExtraTreesEntr   0.855932       0.091441  0.816667                0.091441           0.816667            1       True          9
10       ExtraTreesGini   0.851695       0.081382  1.057413                0.081382           1.057413            1       True          8
11     RandomForestGini   0.847458       0.069925  1.002258                0.069925           1.002258            1       True          5
12       KNeighborsUnif   0.822034       0.019632  1.525858                0.019632           1.525858            1       True          1
13       KNeighborsDist   0.817797       0.016148  0.011902                0.016148           0.011902            1       True          2
Number of models trained: 14
Types of models trained:
{'CatBoostModel', 'LGBModel', 'TabularNeuralNetTorchModel', 'NNFastAiTabularModel', 'KNNModel', 'WeightedEnsembleModel', 'RFModel', 'XTModel',
Bagging used: False
Multi-layer stack-ensembling used: False
Feature Metadata (Processed):
(raw dtype, special dtypes):
('category', [])  :  5 | ['BusinessTravel', 'Department', 'EducationField', 'JobRole', 'MaritalStatus']
('int', [])       : 14 | ['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeNumber', ...]
('int', ['bool']) :  2 | ['Gender', 'OverTime']
Plot summary of models saved to file: AutogluonModels/ag-20231129_112109/SummaryOfModels.html
*** End of fit() summary ***
```

Gambar 4. 54 Model *Summary*

Dari gambar diatas, dapat dilihat bahwa fungsi *summary* pada *AutoGluon* berbeda dengan fungsi *Leaderboard* yang cenderung menampilkan urutan ranking berdasarkan *score_test*. Pada fungsi *summary* kecenderungan ranking model diurutkan berdasarkan *score*_val dan alphabet dari model itu sendiri. Adapun untuk memastikan bahwa model terbaik yang diperoleh adalah model yang sama dapat dilakukan dengan pemanggilan fungsi berikut ini.

```
predictor.get_model_best()

'WeightedEnsemble_L2'
```

Gambar 4. 55 Model Terbaik

Dari gambar diatas dapat diketahui bahwa model terbaik tetaplah *WeightedEnsemble_L2*. Selain itu, pada saat melakukan pemanggilan fungsi *summary* diatas dapat diketahui bahwa *AutoGluon* melakukan pengaturan *hyperparameter* secara otomatis yang disesuaikan dengan algoritma yang dijalankan untuk memperoleh hasil terbaik. Adapun pengaturan otomatis yang dimaksud yaitu sebagai berikut.

Pada *WeightedEnsemble_L2*, jumlah bag *fold* adalah 0, dengan maximum stack level dan jumlah kelas sama dengan 2 serta basis per model adalah 25. Pada K*neighbors*Unif, jenis pembobotan adalah uniform, sementara K*neighbors*Dist jenis pembobotan adalah distance. Pada LightGBMXT, dan LightGBM *learning*_rate adalah 0.05, dengan tambahan *extra trees*

hanya pada LightGBMXT. Pada *RandomForest*Gini dan *RandomForest*Entr jumlah estimators adalah 300, dengan maximum leaf_nodes sebanyak 15000. Selain itu, untuk algoritma CatBoost dan *XGBoost* iterasi yang dilakukan sebanyak 10000 dengan *learning* rate berturut-turut sebesar 0.05 dan 0,01

Adapun deskripsi terkait tiap parameter pada fit_*summary*() adalah sebagai berikut.

Tabel 4. 5 *Leaderboard Argument*

Input argument	***Description***	***Note***
model	Nama dari model	-
score_test	Skor model pada 'eval_metric' untuk data yang disediakan	Metrik skor selalu menunjukan nilai tertinggi
score_val	Validasi skor dari model di dalam "eval_metric"	Metrik skor selalu menunjukan nilai tertinggi
pred_time_test	Waktu inferensi jam dinding end-to-end sebenarnya dari model untuk data yang disediakan	Ekuivalen terhadap jumlah semua nilai 'pred_time_*test*_marginal' untuk model dan semua base modelnya.
pred_time_val	Waktu inferensi yang diperlukan untuk menghitung prediksi pada data validasi secara end-to-end	Ekuivalen terhadap jumlah dari nilai "pred_time_val_marginal" untuk model dan semua base model
fit_time	Waktu penyesuaian yang diperlukan untuk melatih model secara end-to-end (Termasuk base model jika modelnya adalah stack *ensemble*)	Ekuivalen terhadap jumlah dari nilai "fit_time_val_marginal" untuk model dan semua base model
pred_time_test_marginal	Waktu inferensi model untuk data yang disediakan, dikurangi waktu inferensi untuk model dasar model, jika ada.	mengabaikan waktu yang diperlukan untuk memuat model ke dalam memori saat *bagging* dinonaktifkan.
pred_time_val_marginal	Waktu inferensi yang diperlukan untuk menghitung prediksi pada data validasi	

Input argument	***Description***	***Note***
	(Mengabaikan waktu inferensi untuk base model)	
fit_time_marginal	Waktu kesesuaian yang diperlukan untuk melatih model (Mengabaikan base model).	Model dengan tingkat tumpukan N dapat mengambil kumpulan model apa pun dengan tingkat tumpukan kurang dari N sebagai masukan, dengan model tumpukan tingkat 1 tidak memiliki masukan model.
stack_level	Tingkat tumpukan dari model	
can_infer	Status jika model mampu melakukan inferensi pada data baru. Jika Salah, maka model tersebut tidak disimpan, dihapus, atau ancestor model tidak dapat disimpulkan.	can_infer sering bernilai *False* ketika save_bag_*fold*s=*False* ditentukan pada di awal fit().
fit_order	Urutan model yang cocok. Model fit pertama memiliki fit_order=1, dan model fit ke-N memiliki fit_order=N	Urutannya sesuai dengan first child model yang cocok bagged *ensemble*s

BAB V

PEMBAHASAN

Berdasarkan pengolahan data yang dilakukan, khususnya pada tahapan EDA dapat diketahui bahwa didalam *dataset* IBM HR *Analytic Employee Attrition* terdapat 10 *feature* yang memiliki *outlier* dan telah di eliminasi. Proses eliminasi variabel atau *feature* ini tentunya bukan merupakan langkah yang terbaik. Dalam prinsip *data science*, untuk mengatasi masalah *outlier* pada data sebelum dilakukannya *training* model *machine learning*, terdapat beberapa alternatif lain yang dapat dilakukan salah satunya adalah dengan melakukan transformasi tipe data numerik menjadi kategorikal, atau yang biasa disebut dengan istilah *ordinal encoding*. Proses *ordinal encoding* ini bekerja dengan cara mengubah data numerik menjadi beberapa kategori kelompok. Oleh karena itu, beberapa 7 dari 10 *feature* yang memiliki *outlier* yaitu *'YearsWithCurrManager','YearsSinceLastPromotion','YearsInCurrentRole','YearsAtCompany', 'TotalWorkingYears', 'StockOptionLevel', 'MonthlyIncome'* cukup kompatibel untuk dilakukan tahapan *ordinal encoding*. Hal ini dikarenakan, secara umum tiap perusahaan memiliki lebih sedikit pekerja yang ada di level jabatan tinggi dibandingkan dengan level jabatan rendah. Sehingga mengubah data numerik seperti *'MonthlyIncome'* kedalam bentuk *range income low, medium, high* (misalnya) akan lebih efektif dalam menghilangkan outliers pada data tersebut. Namun, hal ini tentunya akan mempengaruhi kualitas hasil prediksi karena informasi yang menjadi input pada model machine learning yang akan dilatih berubah. Selain masalah *outlier*, masalah *single entry value* yang terdapat pada 3 *feature* yaitu *'StandardHours', 'EmployeeCount' 'Over18'* juga dilakukan tahapan eliminasi, karena *feature* ini tidak dapat memiliki relasi (*pearson correlation*) dengan *feature* lainnya. Sehingga, tahapan eliminasi yang dilakukan pada dataset sudah dapat menjadi alternatif solusi dalam mengatasi *outliers*.

Pada *output summary*, dapat diketahui bahwa *WeightedEnsemble*_L2 merupakan algoritma terbaik dengan skor validasi sebesar 0.88 atau 88% dan *score test* sebesar 86% akurat dalam melakukan prediksi terhadap *variable attrition*. Dari skor tersebut dapat disimpulkan bahwa dengan menerapkan algoritma yang sama, prediksi terhadap karyawan yang akan atrisi dapat dihitung dan mempunyai tingkat probabilitas sebesar 86% dengan menggunakan variabel yang sama. Selain itu, dalam algoritma *Weighted ensemble* L2, disini L2 artinya bahwa algoritma ini menggunakan regularisasi L2 dengan pendekatan *ensemble learning*. Model ini menghasilkan skor tertinggi karena pada dasarnya model *ensemble* yang di kombinasikan dengan regularisasi L2 adalah algoritma yang mempunyai performa yang bagus dalam melatih data dengan variabel atau fitur yang banyak.

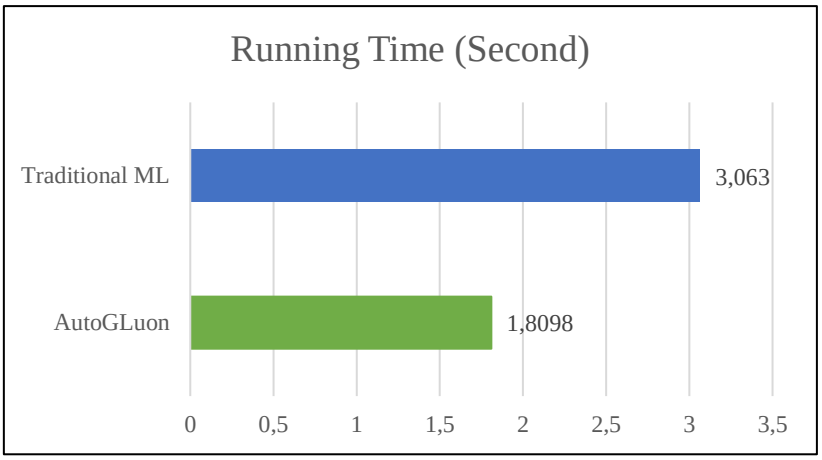
Selain model yang dihasilkan oleh algoritma *WeightedEnsemble*_L2, algoritma gradien *boosting* seperti *XGBoost* dan *CatBoost* juga memiliki hasil model yang cukup baik dalam memprediksi dataset yang dilatih. Kedua algoritma tersebut memperoleh hasil modelnya dengan cara menerapkan iterasi sabanyak 10000 namun dengan parameter learning rate yang berbeda yang secara berturut-turut 0,1 untuk *XGBoost* dan 0,05 untuk *CatBoost.* Algoritma lain yang juga kompatibel untuk digunakan dalam memperoleh prediksi terhadap dataset yang dilatih yaitu seperti *neural network* memperoleh *score_test* yang hampir sama dengan algoritma *weighted ensemble* L2. Karena *dataset* yang dilatih mempunyai 22 *feature* dimana 1 *feature* yang dijadikan sebagai *predictor* adalah data yang bertipe *binnary*. Artinya bahwa *output* pada *neural network* akan menghasilkan prediksi antara yes atau no untuk data yang dilatih. Untuk model *NeuralNetTorch* digunakan fungsi aktifasi ReLu dengan jumlah *layer*s sebanyak 4 dan *epoch* sebanyak 500 serta Optimizer adam.

Selain itu, berdasarkan 14 algoritma atau model *machine learning* yang dihasilkan oleh AutoGluon, dapat diketahui bahwa terdapat kelompok model yang berpotensi *underfitting* atau *overfitting* yaitu kNN (kNN unif dan kNN dist), *Random Forest* (RF gini dan RF entropi), serta *extra tree* (gini dan entropi). Hal ini disebabkan oleh nilai *score_test* lebih tinggi dibandingkan dengan nilai *score_val.* Model lainnya seperti *LightGBMLarge, NeuralNetTorch* juga merupakan model yang berpotensi mengalami hal yang sama, namun kelompok modelnya (*LightGBMXT* dan *NeuralNetFastAI*) tidak mengalami hal demikian. Hasil ini tentunya membuat perlu dilakukannya kajian lebih mendalam terkait *model fitting* yang terdapat pada dokumentasi dari *AutoGluon* versi termutakhir, sekaligus memperjelas bahwa model terbaik *WeightedEnsemble_L2* tidak berpotensi mengalami masalah *overfitting* maupun *underfitting*

Keunggulan dalam menggunakan *AutoGluon* dibandingkan dengan *machine learning* tradisional pada penelitian ini, dapat dilihat pada beberapa tahapan. Pertama pada tahapan one-hot *encoding* yang mana hal ini dapat dilewati, karena sudah otomatis terbaca oleh sistem bahwa label "*Attrition*" merupakan jenis data *binnary*. Berbeda dengan *machine learning* tradisional (pada lampiran) dapat dilihat bahwa sebelum melakukan *train* dan *test* data label yang menjadi *predictor* harus diubah terlebih dahulu menjadi *binnary* number agar memudahkan model *machine learning* tradisional untuk memproses data tersebut. Selain itu, dalam aktivasi dan *setting* parameter maupun *hyperparameter* pada tiap algoritma seperti *random forest*, dapat dilihat bahwa pada *machine learning* tradisional hal ini harus dilakukan secara manual, sementara pada *AutoGluon* sistem sudah dapat menjadi dan memilih *setting* atau penyetelan yang terbaik.

Secara keseluruhan jika dibandingkan antara model *machine learning* yang dibangun dengan menggunakan *AutoGluon* dan model *machine* learnin yang dibangun oleh salah satu kaggle *grandmaster* bahwa tingkat akurasi yang dihasilkan pada masing-masing berturut-turut yaitu 86% atau lebih tepatnya 86,05 untuk tingkat akurasi *AutoGluon* dengan menggunakan *Weighted_Ensemble* L2 dan 85,37 untuk *machine learning* conventional dengan menggunakan *XGBoost*. Artinya bahwa dengan pendekatan *AutoML* pada *Dataset* IBM HR *Analytic Employee Attrition* tingkat akurasi prediksi yang dihasilkan menunjukan skor yang lebih baik dibandingkan dengan *machine learning* tradisional. Oleh karena itu, *AutoGluon* sebagai salah satu teknologi *AutoML* tidak hanya dapat memudahkan penggunanya yang bahkan bukan seorang *expert* untuk dapat bersaing dengan seorang *expert* dalam membangun model *machine learning* dengan tingkat akurasi yang tinggi, tetapi juga lebih efisien dalam hal implementasi model *machine learning*. Namun, jika dilakukan pertimbangan kembali terhadap *feature attrition* yang menjadi target untuk prediksi model *machine learning*, yang mana data tersebut bersifat imbalance atau tidak seimbang dengan respon “No” lebih banyak dari pada respon “Yes”. Sehingga evaluator *precission*/*recall* lebih kompatibel dengan karakteristik data seperti ini. Dimana pada model *AutoGluon* dihasilkan *score* 0,66 untuk *precission* dan 0,93 untuk *recall*.

Selain itu, jika dilihat dari total waktu *fitting* model dan eksekusi pelatihan data hingga evaluasi model, *AutoGluon* cenderung membutuhkan waktu yang cenderung lebih singkat yaitu dengan selisih lebih dari 1253,2 ms jika dibandingkan dengan model *machine learning* tradisional. Seperti pada gambar berikut.



Gambar 5. 1 *Running Time Execution*

Selain itu, ada beberapa hal yang perlu dipertimbangkan kembali dalam menentukan apakah implementasi dari *AutoGluon* memiliki efisiensi dari segi waktu dan tingkat akurasi yang lebih baik dibandingkan dengan *machine learning* tradisional. Salah satu hal yang perlu dipertimbangkan yaitu bahwa saat proses *splitting* data (*train* dan *test*) pada kedua *framework* ini dilakukan dengan cara berbeda. Dimana pada *AutoGluon*, dilakukan dengan cara memisahkan data menggunakan microsoft excel sesuai dengan urutan baris (sequentially), sementara pada *machine learning* tradisional digunakan *library* sklearn.model_selection dimana proses *splitting* datanya tidak secara berurutan (*non-sequentially*). Sehingga hal ini menyebabkan *output* dari *feature importance* dari kedua *framework* ini berbeda walaupun rating *importance* paling tinggi dihasilkan oleh *output* model relatif sama yaitu *variable Overtime* sebagai nilai rating *importance* tertinggi pada model *machine learning* tradisional, dan tertinggi kedua pada *AutoGluon*.

Oleh karena itu, manajemen dari perusahaan dapat memprediksi kemungkinan atrisi karyawan berdasarkan *variable Overtime*. Dengan adanya model *machine learning* yang telah dibangun, pihak manajemen juga dapat menggunakan atau mengimplementasikan teknik yang sama dalam memperoleh prediksi dengan tingkat akurasi lebih dari 80% untuk setiap karakteristik dari *variable-variable* yang terdapat pada *dataset* yang diolah atau dengan kata lain model yang dihasilkan merupakan usulan yang dapat dijadikan sebagai alternatif penyelesaian masalah oleh pihak manajemen dalam mengatasi permasalahan terkait *employee attrition*

BAB VI

PENUTUP

6.1 Kesimpulan

Adapun kesimpulan dari penelitian ini, berdasarkan tujuan yang telah dijelaskan pada bab sebelumnya adalah sebagai berikut

1. Hasil yang diperoleh dari implementasi *AutoGluon* pada *dataset* IBM HR *Data Employee Attrition* bahwa model *WeightedEnsemble_L2* atau algoritma *ensemble* yang dipadukan dengan *regularization* L2 merupakan model terbaik dengan tingkat akurasi prediksi terhadap karyawan yang akan atrisi dengan skor 86%.
2. Implementasi *AutoGluon* dalam melakukan *train* dan *test* data membutuhkan waktu yang lebih singkat sekitar 1253,2 ms lebih cepat dibandingkan dengan pada model *machine learning* tradisional
3. Dengan adanya model *machine learning* yang telah dibangun yang memiliki tingkat akurasi lebih dari 80%, pihak manajemen perusahaan dapat menggunakan atau mengimplementasikan teknik yang sama dalam pengambilan keputusan yang berkaitan dengan kasus *employee attrition.*

6.2 Saran

Berdasarkan dari pengolahan data dan analisis yang telah dilakukan pada bab sebelumnya, saran yang dapat diberikan pada penelitian ini adalah sebagai berikut:

1. Bagi Profesional di Bidang Human Resource

   Diharapkan bagi para profesional di bidang HR agar dapat menggunakan algoritma *ensemble* L2 dalam melakukan prediksi terhadap *employee attrition*. Berdasarkan hasil model *machine learning* yang dibangun algoritma tersebut memperoleh nilai akurasi tertinggi. Hal ini bertujuan untuk memudahkan profesional di bidang HR dalam mengambil keputusan terkait pengurangan karyawan

2. Bagi peneliti selanjutnya

Diharapkan bagi peneliti selanjutnya dapat menggunakan implementasi teknologi *AutoML* lainnya seperti Auto-WEKA, H20, dll. Selain itu, diharapkan untuk implementasi *AutoML* dapat mempunyai akurasi dan waktu *training* yang lebih baik lagi dibandingkan teknologi *AutoML* yang digunakan pada penelitian ini (*AutoGluon*).

DAFTAR PUSTAKA

Al Akasheh, M., Malik, E. F., Hujran, O., & Zaki, N. (2023). A Decade of Re*search* on Data Mining Techniques for Predicting *Employee* Turnover: A Systematic Literature Review. *Journal Pre-Proofs*.

Alduayj, S. S., & Rajpoot, K. (2018). Predicting *Employee Attrition* using *Machine Learning*. *International Conference on Innovations in Information Technology (IIT)*, 3–6.

Anggara, E. D., Widjaja, A., & Suteja, B. R. (2022). Prediksi Kinerja Pegawai sebagai Rekomendasi Kenaikan Golongan dengan Metode *Decision tree* dan Regresi Logistik. *Jurnal Teknik Informatika Dan Sistem Informasi*, *8*(1), 218–234. https://doi.org/10.28932/jutisi.v8i1.4479

AutoML.org. (2023). *AutoML System*. https://www.*AutoML*.org/*AutoML*/

Azevedo, J., Ribeiro, R., Matos, L. M., Sousa, R., Silva, J. P., Pilastri, A., & Cortez, P. (2022). Predicting Yarn Breaks in Textile Fabrics: A *Machine Learning* Approach. *Procedia Computer Science*, *207*(Kes), 2301–2310. https://doi.org/10.1016/j.procs.2022.09.289

Berhil, S., Benlahmar, H., & Labani, N. (2019). A review paper on *artificial intelligence* at the service of human resources management. *Indonesian Journal of Electrical Engineering and Computer Science*, *18*(1), 32–40. https://doi.org/10.11591/ijeecs.v18.i1.pp32-40

Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Barnes & Noble.

Chung, D., Yun, J., Lee, J., & Jeon, Y. (2023). Predictive model of *employee attrition* based on stacking *ensemble learning*. *Expert Systems with Applications*, *215*(November 2022), 119364. https://doi.org/10.1016/j.eswa.2022.119364

Developers.*google*.com. (2022). *Training and Test Sets: Splitting Data*. https://developers.*google*.com/*machine-learning*/crash-course/*training*-and-*test*-sets/*splitting*-data

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). *AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data*. http://arxiv.org/abs/2003.06505

Fallucchi, F., Coladangelo, M., Giuliano, R., & De Luca, E. W. (2020). Predicting *employee attrition* using *machine learning* techniques. *Computers*, *9*(4), 1–17. https://doi.org/10.3390/computers9040086

Felix, A. Y., Vuyyuru, K. R., & Puli, V. (2020). A Proposed Model for Predicting *Employee*s' Performance Using Data Mining Techniques. *Journal of Computational and Theoretical Nanoscience*, *17*(8), 3804–3809. https://doi.org/10.1166/jctn.2020.9274

Frank, H., Lars, K., & Joaquin, V. (2022). *Automated Machine Learning* (Methods, Systems, Challenge). In *Automated Deep learning Using Neural network Intelligence*. https://doi.org/10.1007/978-1-4842-8149-9_2

Haldorai, K., Kim, W. G., Pillai, S. G., Park, T. (Eliot), & Balasubramanian, K. (2019). Factors affecting hotel *employee*s' *attrition* and turnover: Application of pull-push-mooring *framework*. *International Journal of Hospitality Management*, *83*(October 2018), 46–55. https://doi.org/10.1016/j.ijhm.2019.04.003

Hartatik, Bernadetta, K., Titin Agustin, N., Abdillah, B., Robet, I gede Iwa, S., I Putu Susila, H., Iwan, A., Rudi Max Damara, G., & Terttiavini. (2023). *Data science for Business*. PT. Sonpedia Publishing Indonesia. https://books.*google*.co.id/books?hl=en&lr=&id=M6G9EAAAQBAJ&oi=fnd&pg=PA2&dq=implementasi+data+science+dengan+bahasa+*python*&ots=I9MPEnKywo&sig=NpJDrm9IxbiFdou17wHLqZuSZ9o&redir_esc=y#v=onepage&q&f=*false*

Howard, J., & Gugger, S. (2020). Fastai: A *layer*ed api for *deep learning*. *Information (Switzerland)*, *11*(2), 1–27. https://doi.org/10.3390/info11020108

IBM. (2020). *What is Data science?* https://www.ibm.com/id-en/topics/data-science#:~:text=*Data science* combines math and,*decision* making and strategic planning.

Informasi, J., Effendi, M. E., Yuadi, I., & Puspitasari, I. (2023). *Prediksi Guru Kemungkinan Tetap Bekerja di Sekolah Al Uswah Surabaya Menggunakan Machine Learning*. *5*(1), 1–7. https://doi.org/10.37034/jidt.v5i1.361

Kirimi, J. M., & Moturi, C. A. (2016). Application of Data Mining Classification in *Employee* Performance Prediction. *International Journal of Computer Applications*, *146*(7), 975–8887. http://www.ksg.ac.ke/

Lamramot, T. H., Hadiana, A. I., & Santikarama, I. (2022). *Sistem Prediksi Awal Terhadap Atrisi Karyawan*. *1*, 18–24.

Liaw, A., & Wiener, M. (2002). The R Journal: Classification and *regression* by *randomForest*. *R Journal*, *2*(3), 18–22. http://www.stat.berkeley.edu/

Microsoft. (2023a). *LightGBM Documentation*. Lightgbm.readthedocs.io

Microsoft. (2023b). What is *automated machine learning* (*AutoML*)? In *documentation*.

https://learn.microsoft.com/en-us/azure/*machine-learning*/concept-*automated*-ml?view=azureml-api-2

Naga, D., Muster, W., Musvasva, E., & Ecker, G. F. (2022). Off-targetP ML: an open source *machine learning framework* for off-target panel safety assessment of small molecules. *Journal of Cheminformatics*, *14*(1), 1–20. https://doi.org/10.1186/s13321-022-00603-w

Nuraliza, H., Pratiwi, O. N., & Hamami, F. (2022). *Analisis Sentimen IMBd Film Review Dataset Menggunakan Support Vector Machine (SVM) dan Seleksi Feature Importance*. *7*(1), 1–17.

NVIDIA. (2023). *Pytorch*. https://www.nvidia.com/en-us/glossary/data-science/*pytorch*/

Powers, D. M. W., & *Process*ing, N. L. (2019). *What the F-measure doesn ' t measure …*.

Raj, R., Mathew, J., Kannath, S. K., & Rajan, J. (2023). StrokeViT with *AutoML* for brain stroke classification. *Engineering Applications of Artificial Intelligence*, *119*(April 2022), 105772. https://doi.org/10.1016/j.engappai.2022.105772

Siahaan, M. (2021). An Analysis of Contract *Employee* Performance Assessment Using *Machine Learning*. *Journal of Informatics and Telecommunication Engineering*, *5*(1), 121–131. https://doi.org/10.31289/jite.v5i1.5357

Snijders, C., Matzat, U., & Reips, U. (2013). Big Data: Big Gaps of Knowledge in the Field of Internet Science. *International Journal of Internet Science*, *7*(1), 1–5.

Sukhpreet, S. D., Abdullah Al, N., & Robert, A. (2018). Effective Intrusion Detection System Using *XGBoost*. *Journal Information*, *9*.

Vladimir, S., Andy, L., Christopher, T., J. Christopher, C., Robert P., S., & Bradley P., F. (2003). *Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling*. *10*.

Wardhani, F. H., & Lhaksmana, K. M. (2022). Predicting *Employee Attrition* Using *Logistic Regression* With *Feature* Selection. *Sinkron*, *7*(4), 2214–2222. https://doi.org/10.33395/sinkron.v7i4.11783

Wyatt, J. E., & O'Neill, M. (2021). Investigation of early career teacher *attrition* and the impact of induction programs in Western Australia. *International Journal of Educational Research*, *107*(March), 101754. https://doi.org/10.1016/j.ijer.2021.101754

Yandex. (2023). *CatBoost*. https://catboost.ai/

Yang, J., Zeng, B., Ni, Z., Fan, Y., Hang, Z., Wang, Y., Feng, C., & Yang, J. (2023). Comparison of traditional and *automated machine learning* approaches in predicting the

compressive strength of graphene oxide/cement composites. *Construction and Building Materials*, *394*(March), 132179. https://doi.org/10.1016/j.conbuildmat.2023.132179

Yang, S., & Islam, M. T. (2020). *IBM Employee Attrition Analysis*. *10*.

Yedida, R., Reddy, R., Vahi, R., Jana, R. G., & Kulkarni, D. (2018). *Employee attrition prediction*. *10*.

Zhang, X., Zhou, X., Wan, M., Xuan, J., Jin, X., & Li, S. (2022). *PINC: A Tool for Non-Coding RNA Identification in Plants Based on an Automated Machine Learning Framework*. *International Journal of Molecular Sciences*, *23*(19).

LAMPIRAN

A-Script Coding

```
#import certain library to access and visualize dataset

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

```
# read dataset

data = pd.read_csv("/content/AutoGluon_Project/HR_Employee_Attrition.csv")
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relat
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

```
# EDA
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
```

A-*Script Coding* (Lanjutan)

```
data.isnull().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
```

```
data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000

8 rows × 26 columns

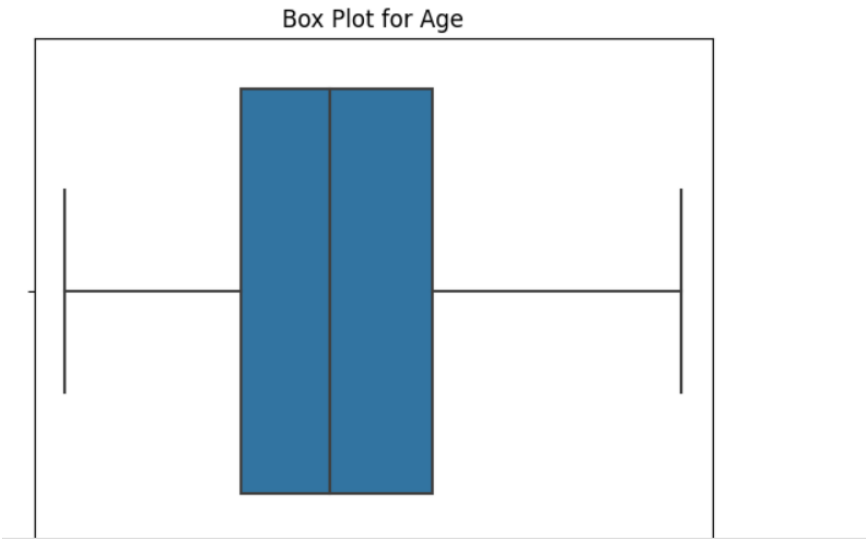
```
for column in data.select_dtypes(include=['object']):
    print(f"Column: {column}")
    print(data[column].value_counts())
    print()
```

```
Column: Attrition
No     1233
Yes     237
Name: Attrition, dtype: int64

Column: BusinessTravel
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64

Column: Department
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64

Column: EducationField
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

A-*Script Coding* (Lanjutan)

```
Column: Gender
Male      882
Female    588
Name: Gender, dtype: int64

Column: JobRole
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64

Column: MaritalStatus
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64

Column: Over18
Y    1470
Name: Over18, dtype: int64

Column: OverTime
No     1054
Yes     416
Name: OverTime, dtype: int64
```

```python
int_columns = data.select_dtypes(include=['int'])

# Loop through each integer column and create box plots
for column in int_columns.columns:
    sns.boxplot(x=data[column])
    plt.title(f'Box Plot for {column}')
    plt.xlabel(column)
    plt.show()
```



A-*Script Coding* (Lanjutan)

TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share Connect

+ Code + Text

```
[ ] # Drop a column
    feature_to_drop = [
        'StandardHours', 'EmployeeCount', #integer single value entry
        'YearsWithCurrManager','YearsSinceLastPromotion','YearsInCurrentRole',
        'YearsAtCompany','TrainingTimesLastYear','TotalWorkingYears',
        'StockOptionLevel','PerformanceRating','NumCompaniesWorked',
        'MonthlyIncome', #outliers
        'Over18' #categorical single value entry
        ]

    data2 = data.drop(feature_to_drop, axis=1)
```

```
[ ] data2.shape

    (1470, 22)
```

```
import plotly.graph_objects as go

int_columns2 = data2.select_dtypes(include=['int'])

fig = go.Figure(data=go.Heatmap(
                    z=int_columns2.astype(float).corr().values,
                    x=int_columns2.columns.values,
                    y=int_columns2.columns.values,
                    hoverongaps = False))

fig.update_layout(
    width=900,
    height=900,
    title_text='Heatmap Preview',)


fig.show()
```

Heatmap Preview

TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share Connect

+ Code + Text

```
import plotly.graph_objects as go

int_columns2 = data2.select_dtypes(include=['int'])

fig = go.Figure(data=go.Heatmap(
                    z=int_columns2.astype(float).corr().values,
                    x=int_columns2.columns.values,
                    y=int_columns2.columns.values,
                    hoverongaps = False))

fig.update_layout(
    width=900,
    height=900,
    title_text='Heatmap Preview',)


fig.show()
```

Heatmap Preview

A-*Script Coding* (Lanjutan)

```
!python -m pip install --upgrade pip
!python -m pip install autogluon
```

```
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from accelerate<0.17,>=0.9->autogluon.multimodal==0
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from accelerate<0.17,>=0.9->autogluon.multimodal==0.8.2->aut
Requirement already satisfied: botocore<1.32.0,>=1.31.49 in /usr/local/lib/python3.10/dist-packages (from boto3<2,>=1.10->autogluon.core[all]=
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from boto3<2,>=1.10->autogluon.core[all]==0.
Requirement already satisfied: s3transfer<0.7.0,>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from boto3<2,>=1.10->autogluon.core[all]==
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost<1.3,>=1.1->autogluon.tabular[all]==0.8.2->au
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost<1.3,>=1.1->autogluon.tabular[all]==0.8.2->auto
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost<1.3,>=1.1->autogluon.tabular[all]==0.8.2->autoglu
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multimodal==
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multimodal==0.8.2->auto
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multimodal==0.8.2->au
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multimodal==0.8
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.mult
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multi
Requirement already satisfied: responses<0.19 in /usr/local/lib/python3.10/dist-packages (from evaluate<0.4.0,>=0.2.2->autogluon.multimodal==0
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages (from fastai<2.8,>=2.3.1->autogluon.tabular[all]==0.8.2->autoglu
Requirement already satisfied: fastdownload<2,>=0.0.5 in /usr/local/lib/python3.10/dist-packages (from fastai<2.8,>=2.3.1->autogluon.tabular[a
Requirement already satisfied: fastcore<1.6,>=1.5.29 in /usr/local/lib/python3.10/dist-packages (from fastai<2.8,>=2.3.1->autogluon.tabular[al
Requirement already satisfied: fastprogress>=0.2.4 in /usr/local/lib/python3.10/dist-packages (from fastai<2.8,>=2.3.1->autogluon.tabular[all]
Requirement already satisfied: spacy<4 in /usr/local/lib/python3.10/dist-packages (from fastai<2.8,>=2.3.1->autogluon.tabular[all]==0.8.2->aut
Requirement already satisfied: toolz~=0.10 in /usr/local/lib/python3.10/dist-packages (from gluonts<0.14,>=0.13.1->autogluon.timeseries[all]==
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gluonts<0.14,>=0.13.1->autogluon.timese
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packages (from hyperopt<0.2.8,>=0.2.7->autogluon.core[all]==0.8.2->aut
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from hyperopt<0.2.8,>=0.2.7->autogluon.core[all]==0.8.2
Requirement already satisfied: py4j in /usr/local/lib/python3.10/dist-packages (from hyperopt<0.2.8,>=0.2.7->autogluon.core[all]==0.8.2->autog
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2<3.2,>=3.0.3->autogluon.multimodal==0.8.
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema<4.18,>=4.14->autogluon.multimodal==0.
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema<4.18,
Requirement already satisfied: wheel in /usr/local/lib/python3.10/dist-packages (from lightgbm<3.4,>=3.3->autogluon.tabular[all]==0.8.2->autog
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from mlforecast<0.7.4,>=0.7.0->autogluon.timeseries[all]==0.8
Requirement already satisfied: window-ops in /usr/local/lib/python3.10/dist-packages (from mlforecast<0.7.4,>=0.7.0->autogluon.timeseries[all]
```

TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share

+ Code + Text Connect

```
[ ] from autogluon.tabular import TabularDataset, TabularPredictor
```

```
train_data = TabularDataset(data2.loc[:1176])
train_data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	...	JobInvolvement	JobLevel	JobRole
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	2	...	3	2	Sales Executive
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	2	3	...	2	2	Research Scientist
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	4	4	...	2	1	Laboratory Technician
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	5	4	...	3	1	Research Scientist
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	7	1	...	3	1	Laboratory Technician

5 rows × 22 columns

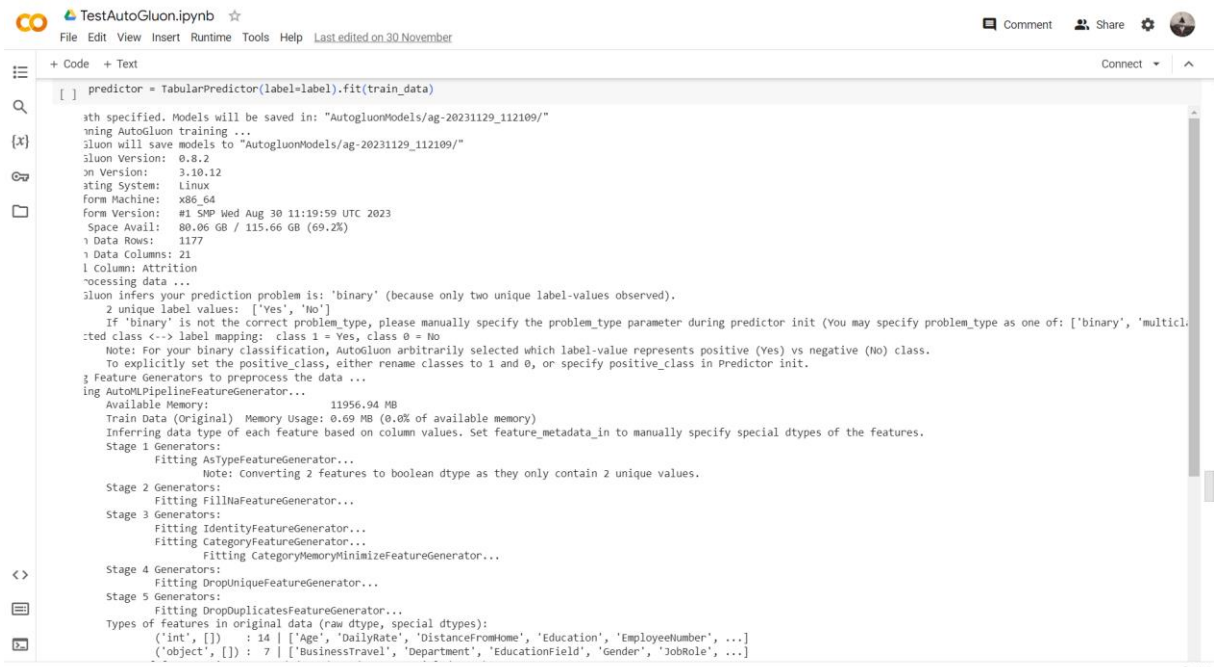
```
[ ] label = 'Attrition'
train_data[label].describe()
```

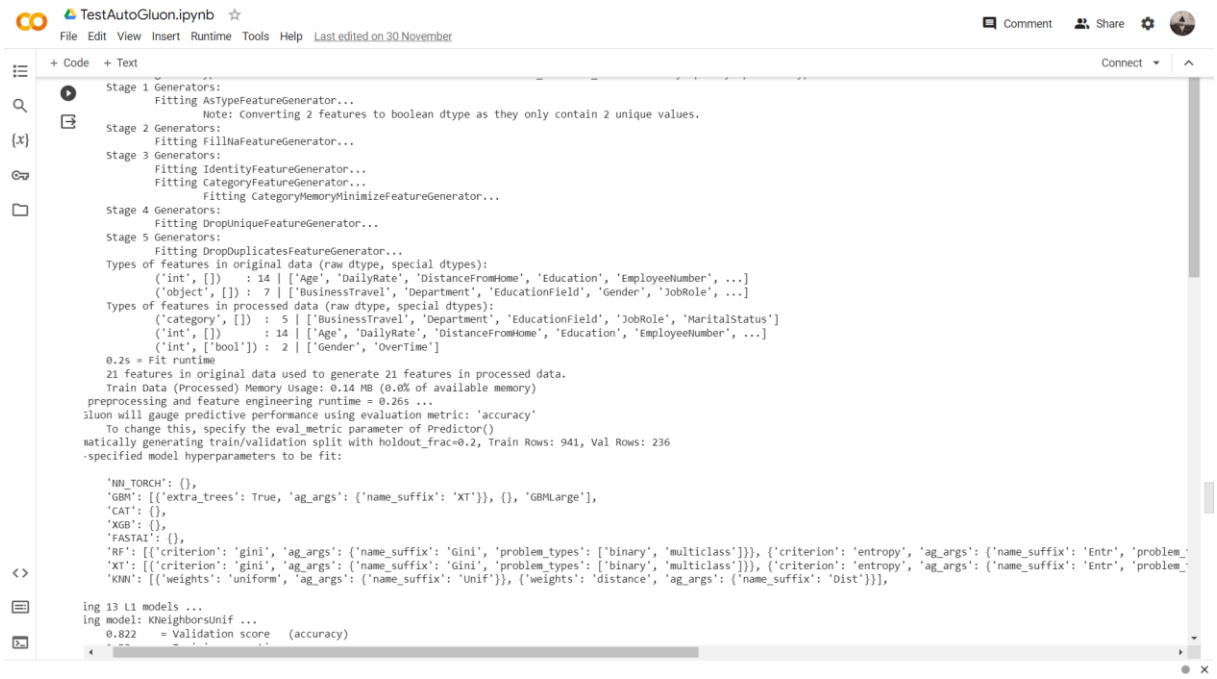
```
count     1177
unique       2
top         No
freq       983
Name: Attrition, dtype: object
```

```
[ ] predictor = TabularPredictor(label=label).fit(train_data)
```

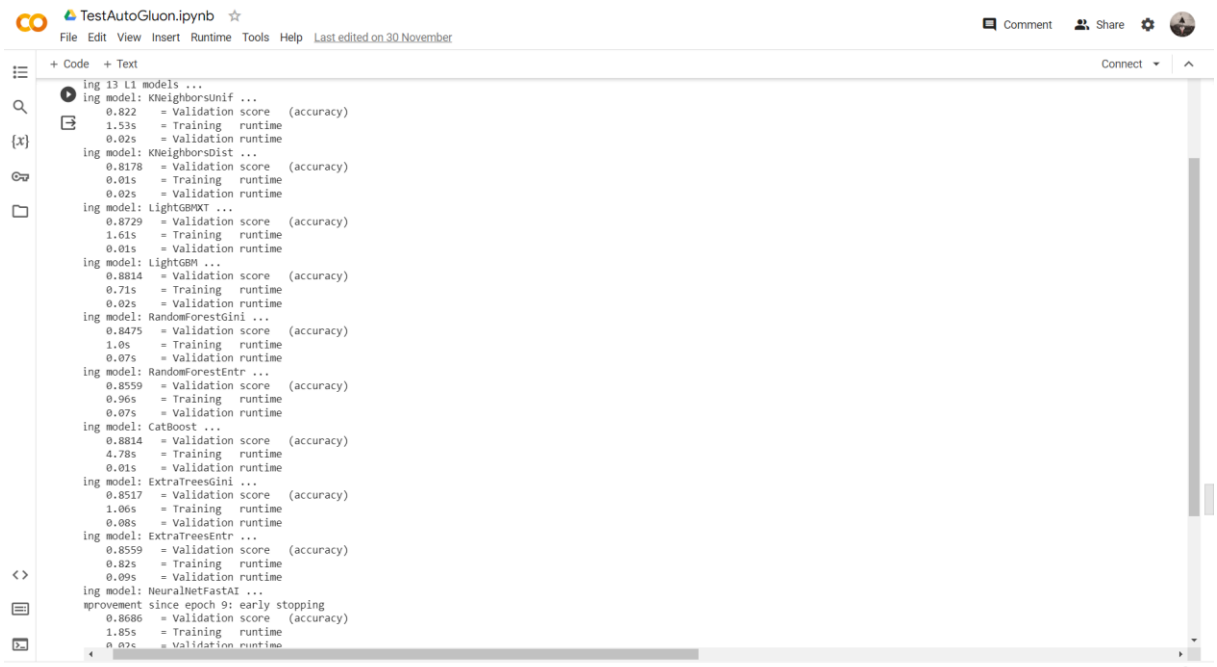
```
ɜth specified. Models will be saved in: "AutogluonModels/ag-20231129_112109/"
ɲning AutoGluon training ...
ɜluon will save models to "AutogluonModels/ag-20231129_112109/"
```

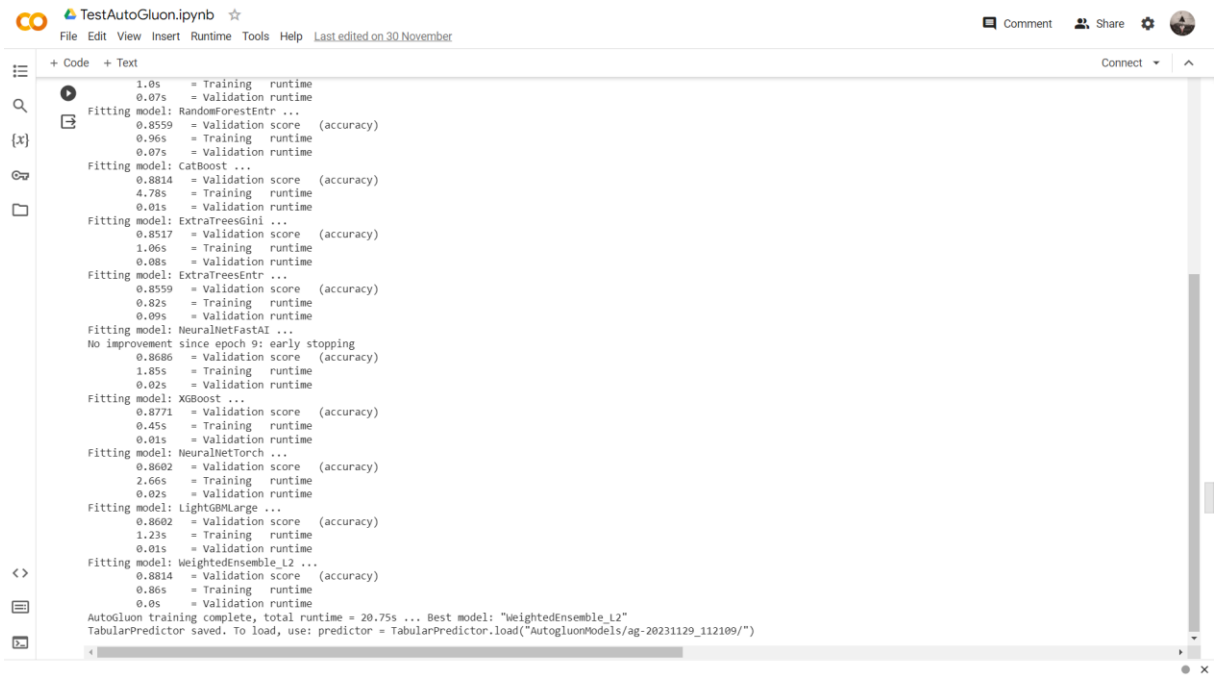
A-*Script Coding* (Lanjutan)



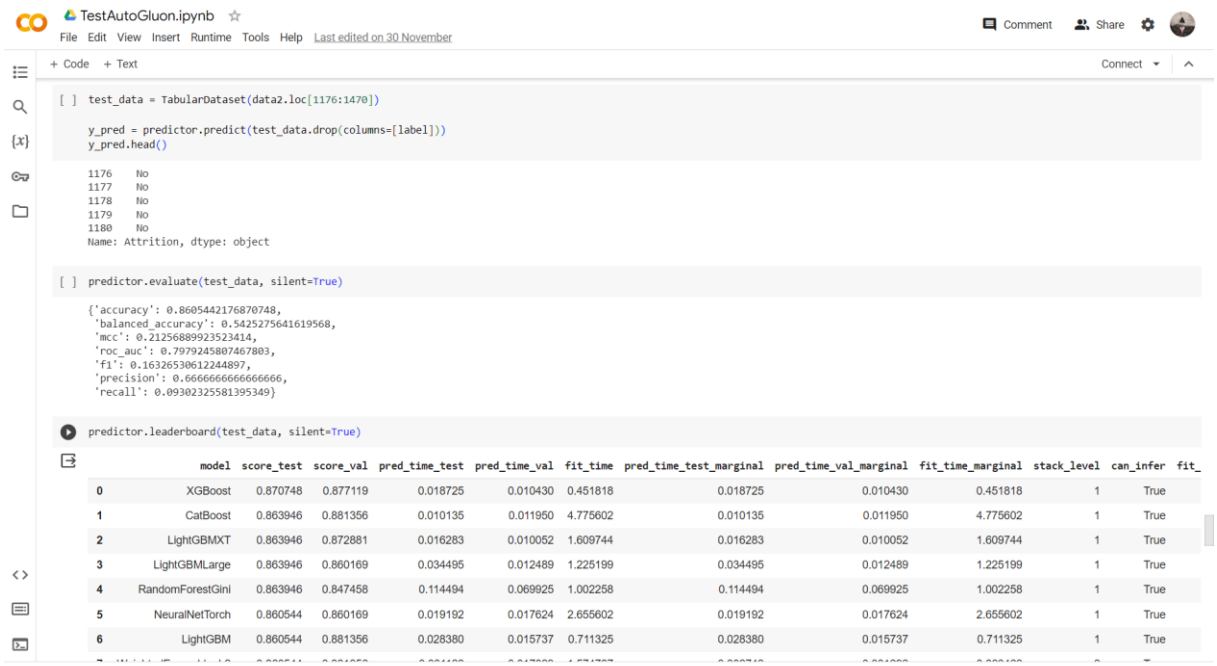


A-*Script Coding* (Lanjutan)





A-*Script Coding* (Lanjutan)



TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share Connect

+ Code + Text

```
[ ] test_data = TabularDataset(data2.loc[1176:1470])

    y_pred = predictor.predict(test_data.drop(columns=[label]))
    y_pred.head()
```

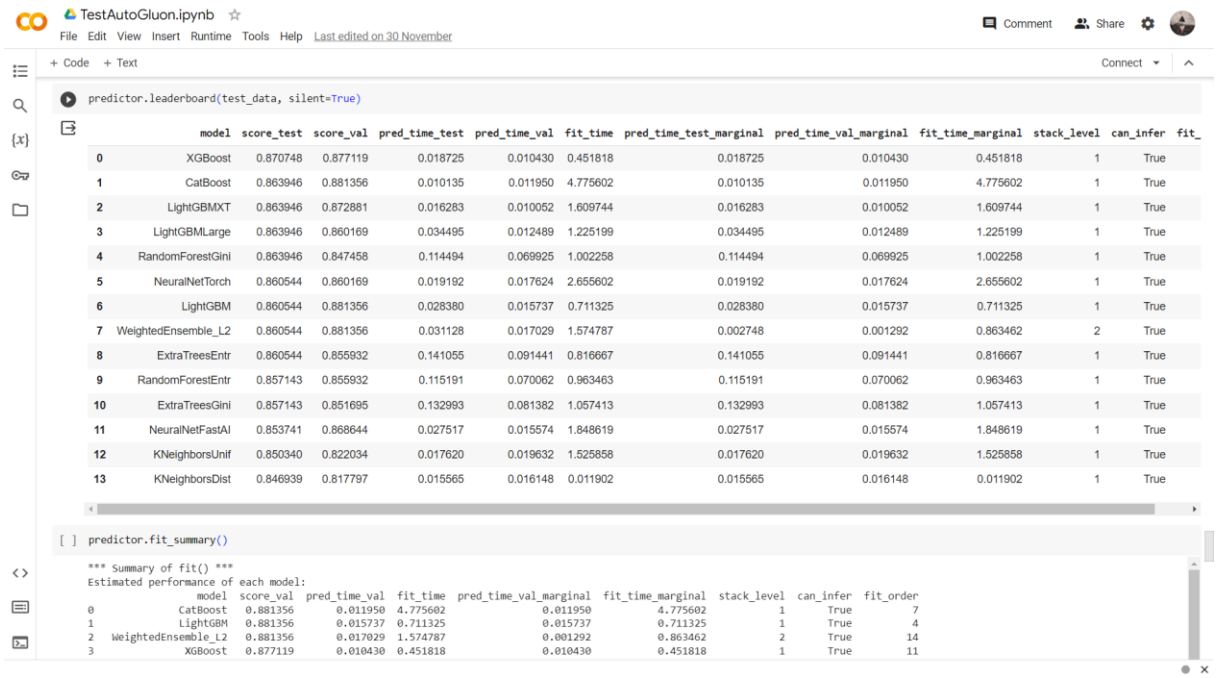
```
1176    No
1177    No
1178    No
1179    No
1180    No
Name: Attrition, dtype: object
```

```
[ ] predictor.evaluate(test_data, silent=True)
```

```
{'accuracy': 0.8605442176870748,
 'balanced_accuracy': 0.5425275641619568,
 'mcc': 0.21256889923523414,
 'roc_auc': 0.7979245807467803,
 'f1': 0.16326530612244897,
 'precision': 0.6666666666666666,
 'recall': 0.09302325581395349}
```

```
predictor.leaderboard(test_data, silent=True)
```

	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_
0	XGBoost	0.870748	0.877119	0.018725	0.010430	0.451818	0.018725	0.010430	0.451818	1	True	
1	CatBoost	0.863946	0.881356	0.010135	0.011950	4.775602	0.010135	0.011950	4.775602	1	True	
2	LightGBMXT	0.863946	0.872881	0.016283	0.010052	1.609744	0.016283	0.010052	1.609744	1	True	
3	LightGBMLarge	0.863946	0.860169	0.034495	0.012489	1.225199	0.034495	0.012489	1.225199	1	True	
4	RandomForestGini	0.863946	0.847458	0.114494	0.069925	1.002258	0.114494	0.069925	1.002258	1	True	
5	NeuralNetTorch	0.860544	0.860169	0.019192	0.017624	2.655602	0.019192	0.017624	2.655602	1	True	
6	LightGBM	0.860544	0.881356	0.028380	0.015737	0.711325	0.028380	0.015737	0.711325	1	True	



TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share Connect

+ Code + Text

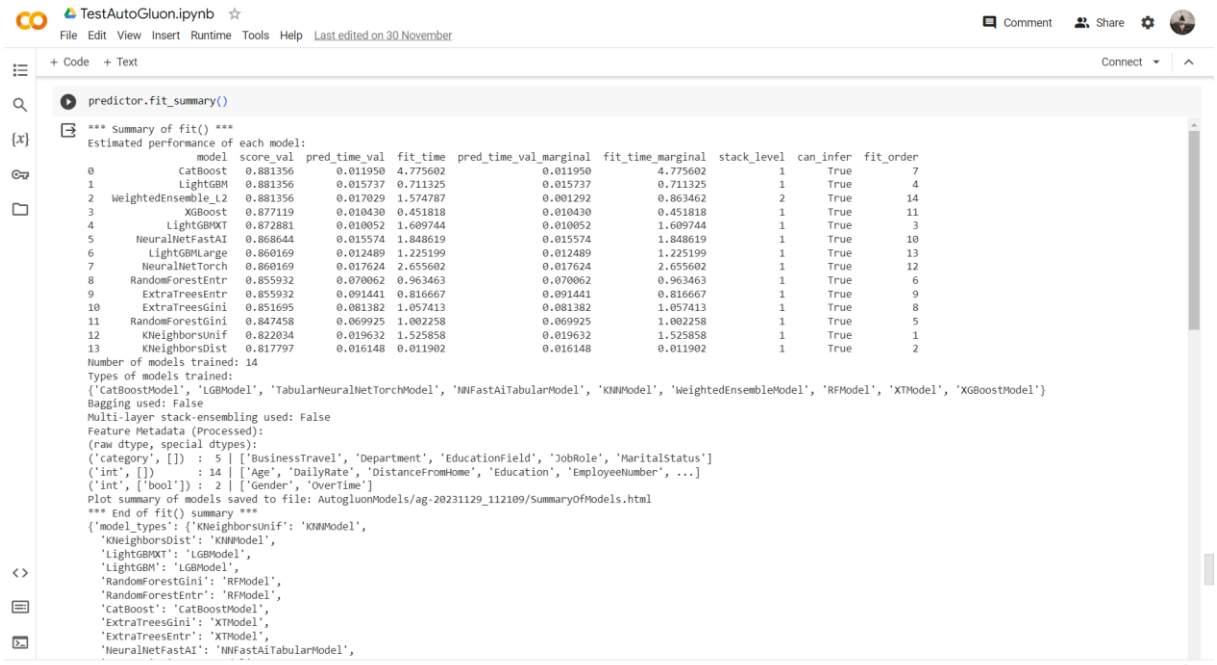
```
predictor.leaderboard(test_data, silent=True)
```

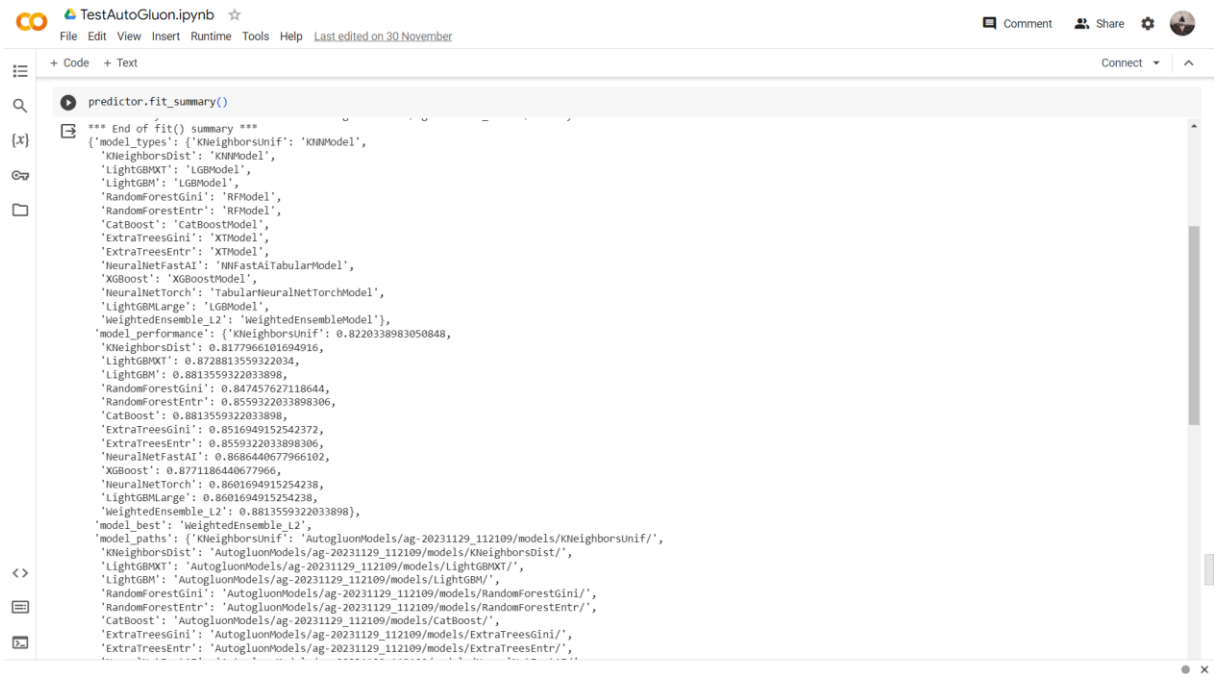
	model	score_test	score_val	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_
0	XGBoost	0.870748	0.877119	0.018725	0.010430	0.451818	0.018725	0.010430	0.451818	1	True	
1	CatBoost	0.863946	0.881356	0.010135	0.011950	4.775602	0.010135	0.011950	4.775602	1	True	
2	LightGBMXT	0.863946	0.872881	0.016283	0.010052	1.609744	0.016283	0.010052	1.609744	1	True	
3	LightGBMLarge	0.863946	0.860169	0.034495	0.012489	1.225199	0.034495	0.012489	1.225199	1	True	
4	RandomForestGini	0.863946	0.847458	0.114494	0.069925	1.002258	0.114494	0.069925	1.002258	1	True	
5	NeuralNetTorch	0.860544	0.860169	0.019192	0.017624	2.655602	0.019192	0.017624	2.655602	1	True	
6	LightGBM	0.860544	0.881356	0.028380	0.015737	0.711325	0.028380	0.015737	0.711325	1	True	
7	WeightedEnsemble_L2	0.860544	0.881356	0.031128	0.017029	1.574787	0.002748	0.001292	0.863462	2	True	
8	ExtraTreesEntr	0.860544	0.855932	0.141055	0.091441	0.816667	0.141055	0.091441	0.816667	1	True	
9	RandomForestEntr	0.857143	0.855932	0.115191	0.070062	0.963463	0.115191	0.070062	0.963463	1	True	
10	ExtraTreesGini	0.857143	0.851695	0.132993	0.081382	1.057413	0.132993	0.081382	1.057413	1	True	
11	NeuralNetFastAI	0.853741	0.868644	0.027517	0.015574	1.848619	0.027517	0.015574	1.848619	1	True	
12	KNeighborsUnif	0.850340	0.822034	0.017620	0.019632	1.525858	0.017620	0.019632	1.525858	1	True	
13	KNeighborsDist	0.846939	0.817797	0.015565	0.016148	0.011902	0.015565	0.016148	0.011902	1	True	

```
[ ] predictor.fit_summary()
```

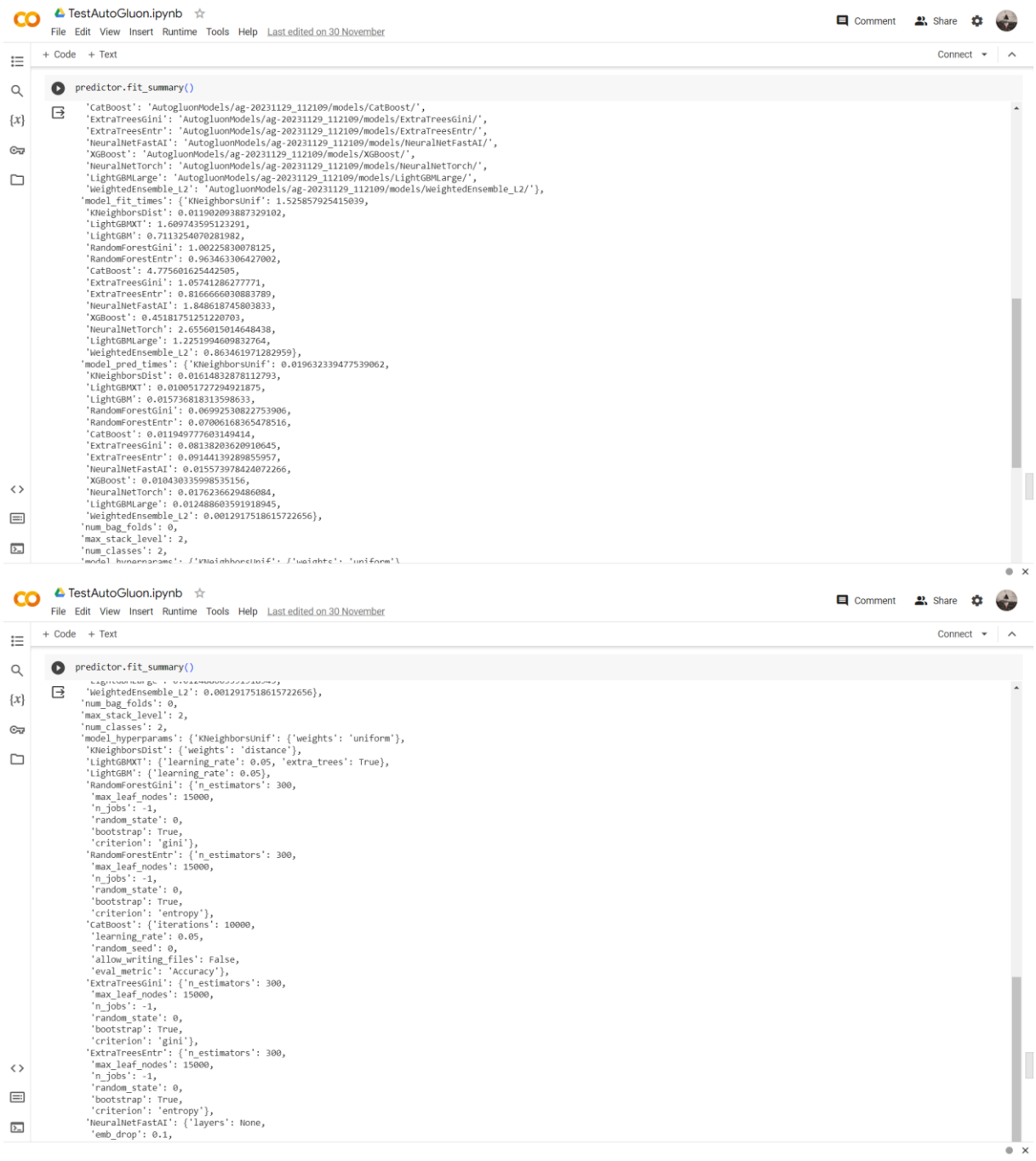
```
*** Summary of fit() ***
Estimated performance of each model:
                 model  score_val  pred_time_val  fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0             CatBoost   0.881356       0.011950  4.775602                0.011950           4.775602            1       True          7
1             LightGBM   0.881356       0.015737  0.711325                0.015737           0.711325            1       True          4
2  WeightedEnsemble_L2   0.881356       0.017029  1.574787                0.001292           0.863462            2       True         14
3              XGBoost   0.877119       0.010430  0.451818                0.010430           0.451818            1       True         11
```

A-*Script Coding* (Lanjutan)

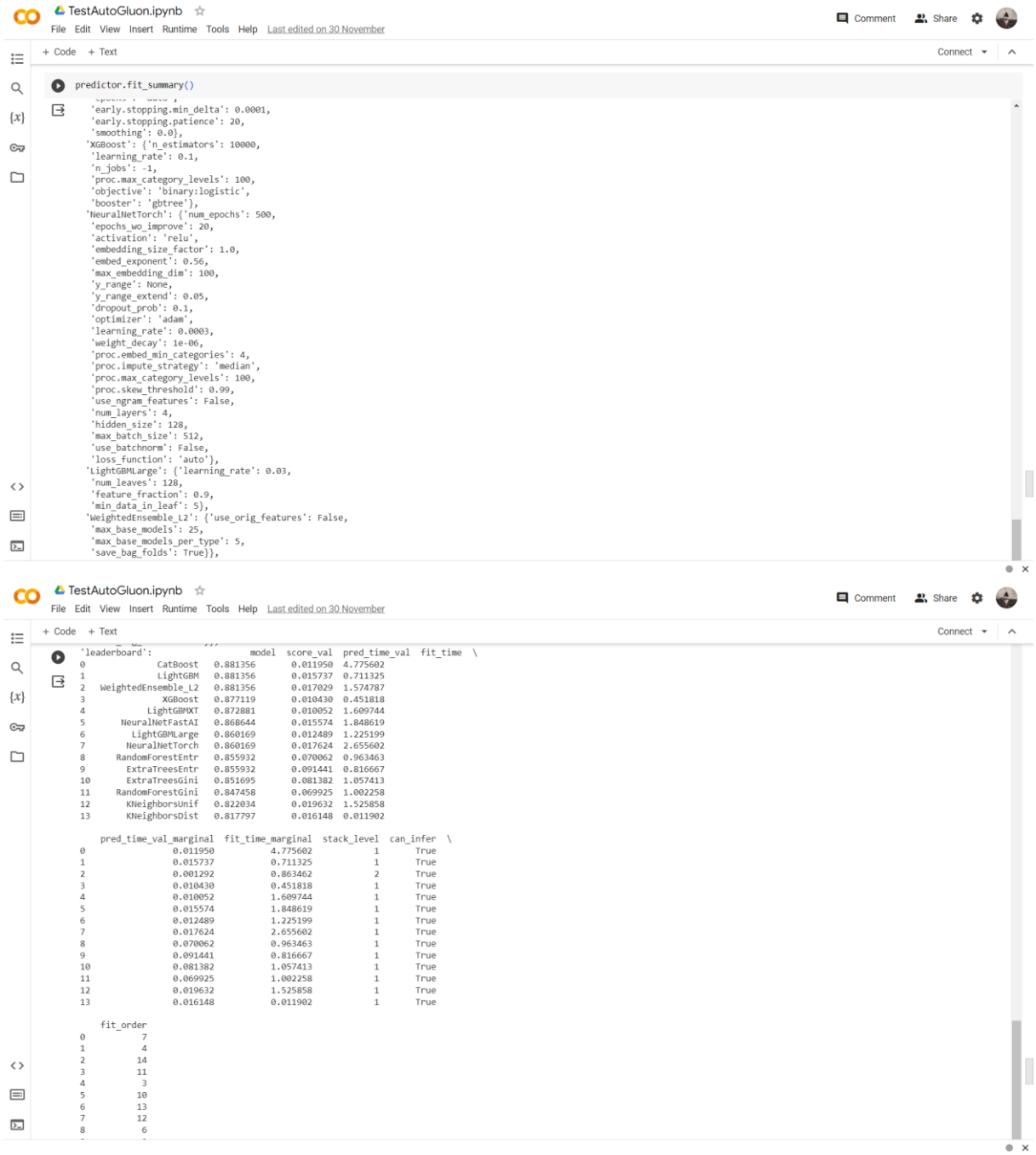




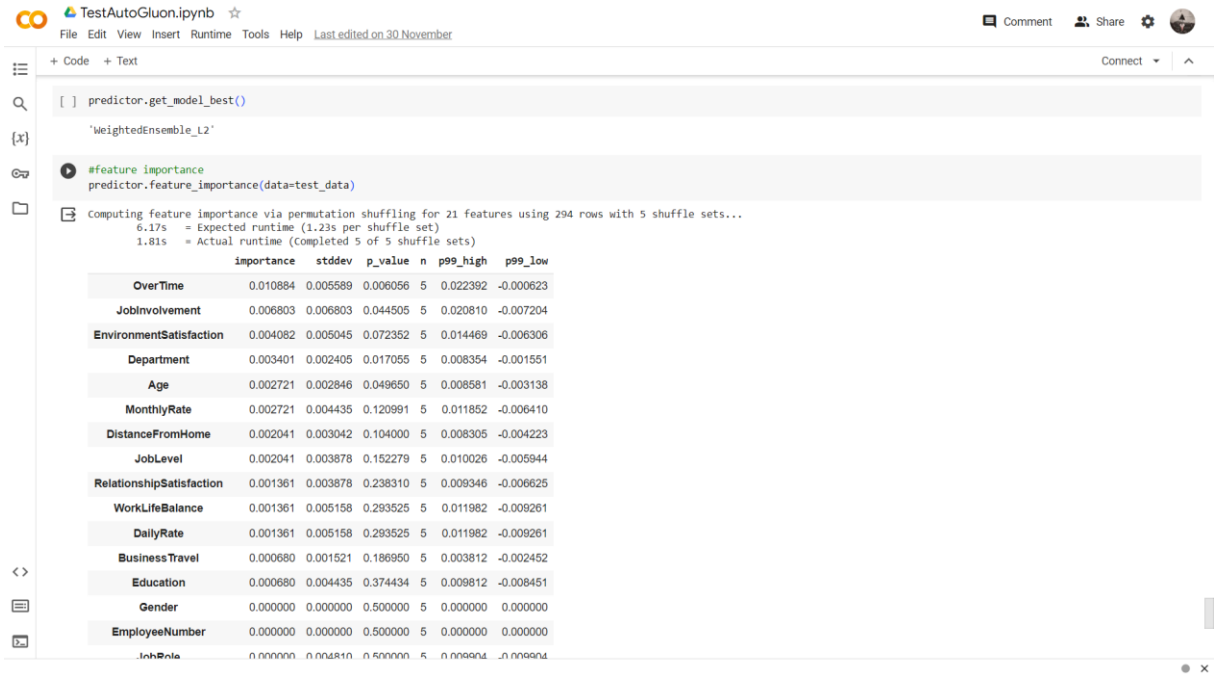
A-*Script Coding* (Lanjutan)



A-*Script Coding* (Lanjutan)



A-*Script Coding* (Lanjutan)



TestAutoGluon.ipynb

File Edit View Insert Runtime Tools Help Last edited on 30 November

Comment Share

+ Code + Text

Connect

```
[ ] predictor.get_model_best()
```

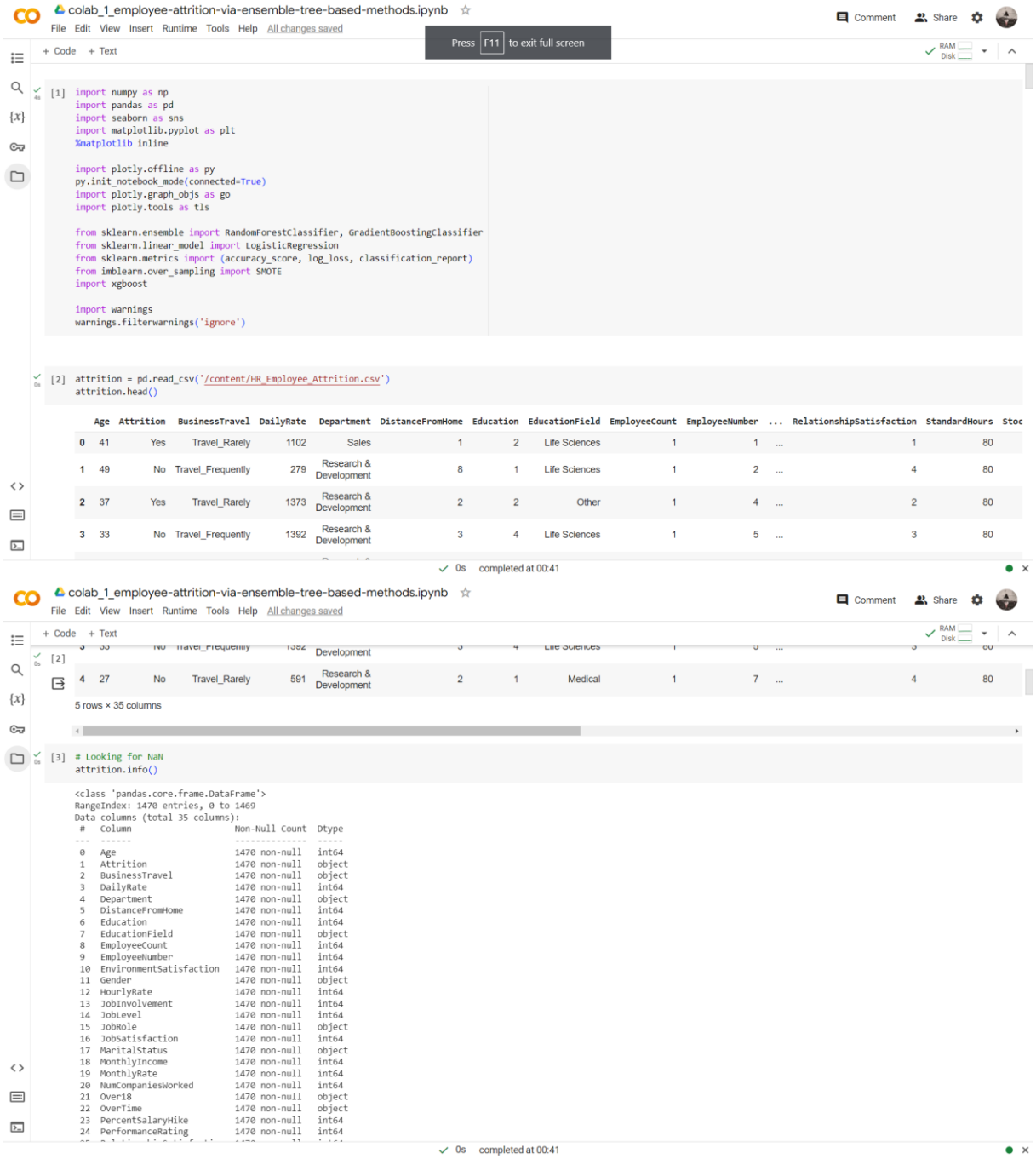
```
'WeightedEnsemble_L2'
```

```
#feature importance
predictor.feature_importance(data=test_data)
```

```
Computing feature importance via permutation shuffling for 21 features using 294 rows with 5 shuffle sets...
        6.17s   = Expected runtime (1.23s per shuffle set)
        1.81s   = Actual runtime (Completed 5 of 5 shuffle sets)
```

	importance	stddev	p_value	n	p99_high	p99_low
OverTime	0.010884	0.005589	0.006056	5	0.022392	-0.000623
JobInvolvement	0.006803	0.006803	0.044505	5	0.020810	-0.007204
EnvironmentSatisfaction	0.004082	0.005045	0.072352	5	0.014469	-0.006306
Department	0.003401	0.002405	0.017055	5	0.008354	-0.001551
Age	0.002721	0.002846	0.049650	5	0.008581	-0.003138
MonthlyRate	0.002721	0.004435	0.120991	5	0.011852	-0.006410
DistanceFromHome	0.002041	0.003042	0.104000	5	0.008305	-0.004223
JobLevel	0.002041	0.003878	0.152279	5	0.010026	-0.005944
RelationshipSatisfaction	0.001361	0.003878	0.238310	5	0.009346	-0.006625
WorkLifeBalance	0.001361	0.005158	0.293525	5	0.011982	-0.009261
DailyRate	0.001361	0.005158	0.293525	5	0.011982	-0.009261
BusinessTravel	0.000680	0.001521	0.186950	5	0.003812	-0.002452
Education	0.000680	0.004435	0.374434	5	0.009812	-0.008451
Gender	0.000000	0.000000	0.500000	5	0.000000	0.000000
EmployeeNumber	0.000000	0.000000	0.500000	5	0.000000	0.000000
JobRole	0.000000	0.004810	0.500000	5	0.009904	-0.009904

B-*Script Re-Coding Kaggle Grandmaster*



colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Press F11 to exit full screen

+ Code + Text

RAM Disk

```
[1] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    %matplotlib inline

    import plotly.offline as py
    py.init_notebook_mode(connected=True)
    import plotly.graph_objs as go
    import plotly.tools as tls

    from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import (accuracy_score, log_loss, classification_report)
    from imblearn.over_sampling import SMOTE
    import xgboost

    import warnings
    warnings.filterwarnings('ignore')
```

```
[2] attrition = pd.read_csv('/content/HR_Employee_Attrition.csv')
    attrition.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours	Stoc
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80	

0s completed at 00:41

colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[2]
```

					Development									
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80	

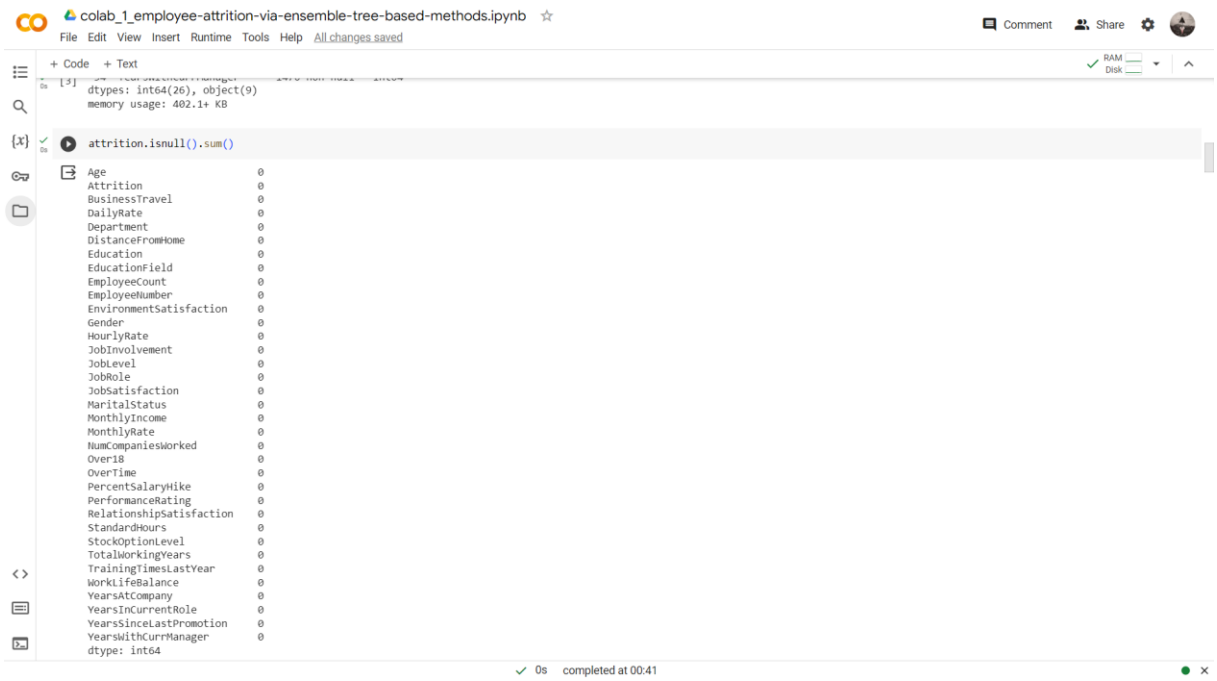
5 rows × 35 columns

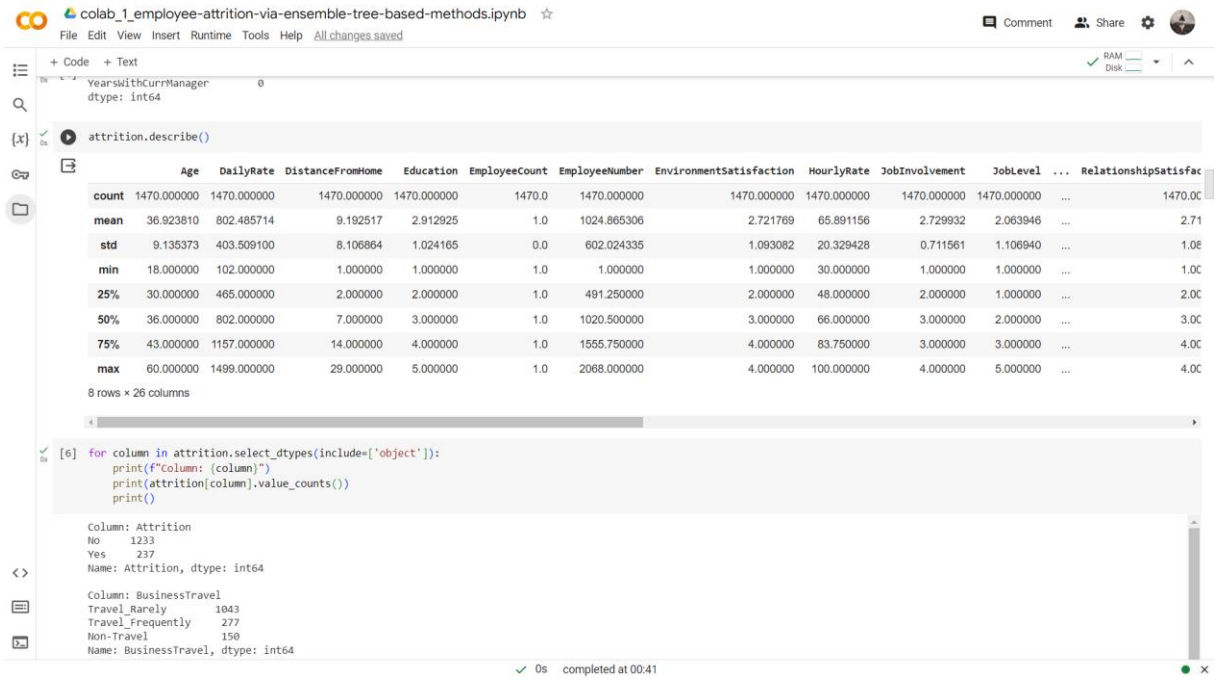
```
[3] # Looking for NaN
    attrition.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
```

0s completed at 00:41

Script Re-Coding Kaggle Grandmaster (Lanjutan)





Script Re-*Coding* Kaggle *Grandmaster* (Lanjutan)

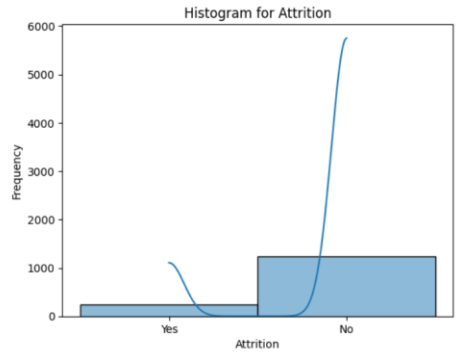
colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

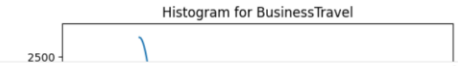
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
object_columns = attrition.select_dtypes(include=['object'])

# Loop through each integer column and create histograms
for column in object_columns.columns:
    sns.histplot(attrition[column], kde=True)  # Create a histogram with a KDE plot
    plt.title(f'Histogram for {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```





0s completed at 00:41

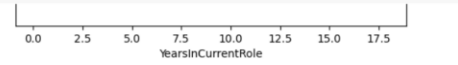
colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

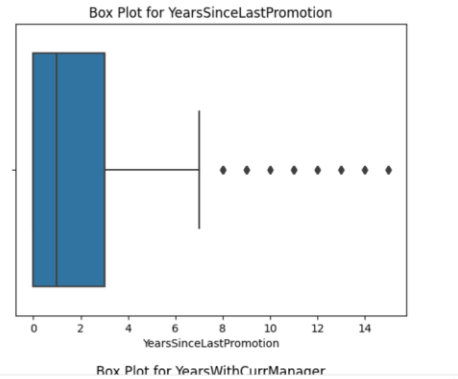
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

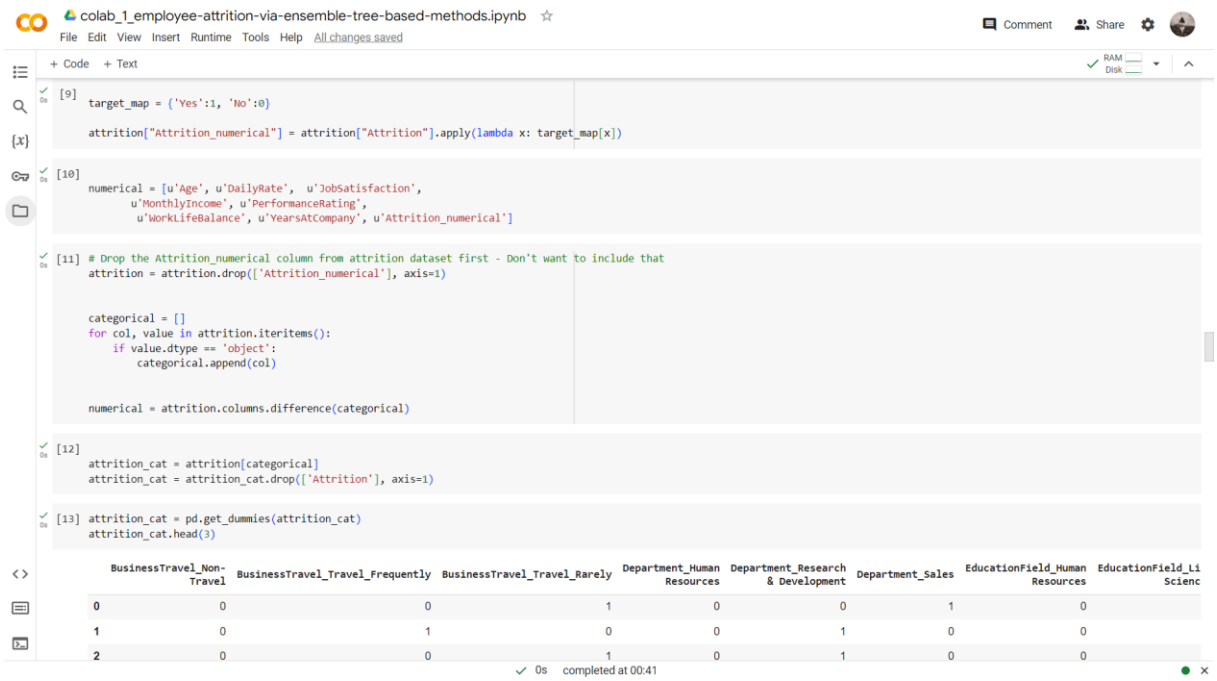
```
int_columns = attrition.select_dtypes(include=['int'])

# Loop through each integer column and create box plots
for column in int_columns.columns:
    sns.boxplot(x=attrition[column])
    plt.title(f'Box Plot for {column}')
    plt.xlabel(column)
    plt.show()
```





0s completed at 00:41

Script Re-*Coding* Kaggle *Grandmaster* (Lanjutan)



colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

[9]
```
target_map = {'Yes':1, 'No':0}

attrition["Attrition_numerical"] = attrition["Attrition"].apply(lambda x: target_map[x])
```

[10]
```
numerical = [u'Age', u'DailyRate',  u'JobSatisfaction',
       u'MonthlyIncome', u'PerformanceRating',
        u'WorkLifeBalance', u'YearsAtCompany', u'Attrition_numerical']
```

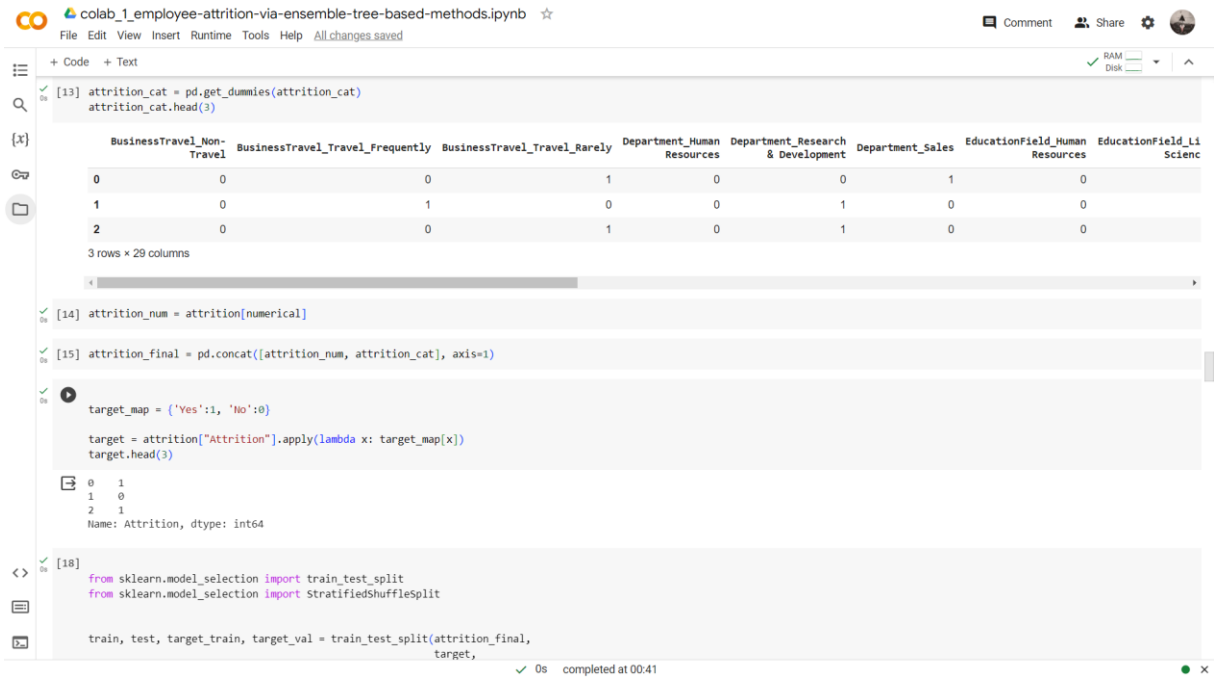
[11]
```
# Drop the Attrition_numerical column from attrition dataset first - Don't want to include that
attrition = attrition.drop(['Attrition_numerical'], axis=1)


categorical = []
for col, value in attrition.iteritems():
    if value.dtype == 'object':
        categorical.append(col)


numerical = attrition.columns.difference(categorical)
```

[12]
```
attrition_cat = attrition[categorical]
attrition_cat = attrition_cat.drop(['Attrition'], axis=1)
```

[13]
```
attrition_cat = pd.get_dummies(attrition_cat)
attrition_cat.head(3)
```

	BusinessTravel_Non-Travel	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_Human Resources	Department_Research & Development	Department_Sales	EducationField_Human Resources	EducationField_Li Scienc
0	0	0	1	0	0	1	0	
1	0	1	0	0	1	0	0	
2	0	0	1	0	1	0	0	

0s completed at 00:41



colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

[13]
```
attrition_cat = pd.get_dummies(attrition_cat)
attrition_cat.head(3)
```

	BusinessTravel_Non-Travel	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_Human Resources	Department_Research & Development	Department_Sales	EducationField_Human Resources	EducationField_Li Scienc
0	0	0	1	0	0	1	0	
1	0	1	0	0	1	0	0	
2	0	0	1	0	1	0	0	

3 rows × 29 columns

[14]
```
attrition_num = attrition[numerical]
```

[15]
```
attrition_final = pd.concat([attrition_num, attrition_cat], axis=1)
```

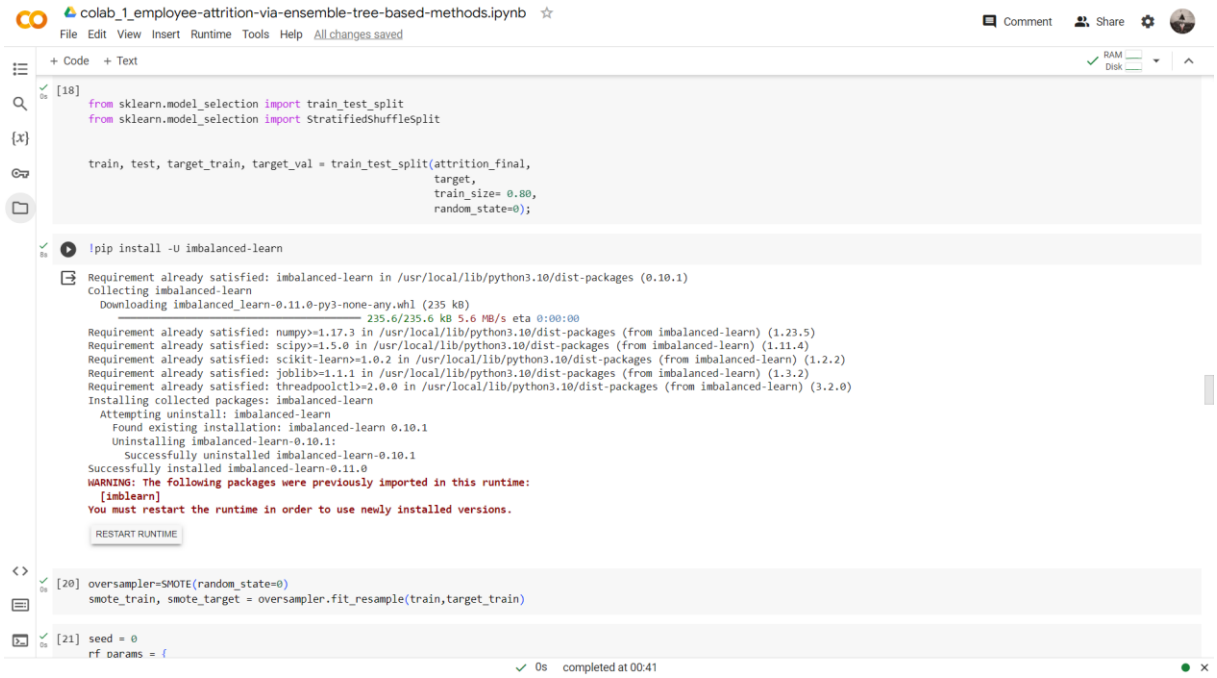
```
target_map = {'Yes':1, 'No':0}

target = attrition["Attrition"].apply(lambda x: target_map[x])
target.head(3)
```

```
0    1
1    0
2    1
Name: Attrition, dtype: int64
```

[18]
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit


train, test, target_train, target_val = train_test_split(attrition_final,
                                                         target,
```

0s completed at 00:41

Script Re-*Coding* Kaggle *Grandmaster* (Lanjutan)



colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[18] from sklearn.model_selection import train_test_split
     from sklearn.model_selection import StratifiedShuffleSplit


     train, test, target_train, target_val = train_test_split(attrition_final,
                                                              target,
                                                              train_size= 0.80,
                                                              random_state=0);
```

```
!pip install -U imbalanced-learn
```

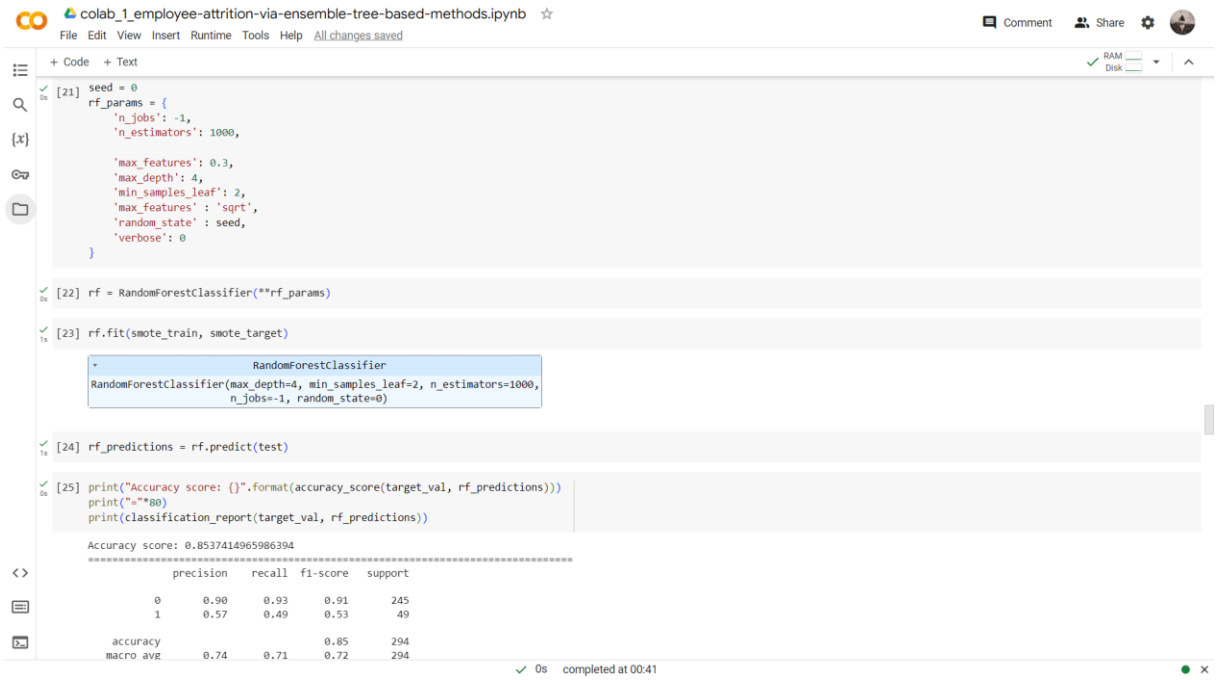
```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.10.1)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.11.0-py3-none-any.whl (235 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.6/235.6 kB 5.6 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.2.0)
Installing collected packages: imbalanced-learn
  Attempting uninstall: imbalanced-learn
    Found existing installation: imbalanced-learn 0.10.1
    Uninstalling imbalanced-learn-0.10.1:
      Successfully uninstalled imbalanced-learn-0.10.1
Successfully installed imbalanced-learn-0.11.0
WARNING: The following packages were previously imported in this runtime:
  [imblearn]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
[20] oversampler=SMOTE(random_state=0)
     smote_train, smote_target = oversampler.fit_resample(train,target_train)
```

```
[21] seed = 0
     rf_params = {
```

0s completed at 00:41



colab_1_employee-attrition-via-ensemble-tree-based-methods.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[21] seed = 0
     rf_params = {
         'n_jobs': -1,
         'n_estimators': 1000,

         'max_features': 0.3,
         'max_depth': 4,
         'min_samples_leaf': 2,
         'max_features' : 'sqrt',
         'random_state' : seed,
         'verbose': 0
     }
```

```
[22] rf = RandomForestClassifier(**rf_params)
```

```
[23] rf.fit(smote_train, smote_target)
```

RandomForestClassifier

RandomForestClassifier(max_depth=4, min_samples_leaf=2, n_estimators=1000, n_jobs=-1, random_state=0)

```
[24] rf_predictions = rf.predict(test)
```

```
[25] print("Accuracy score: {}".format(accuracy_score(target_val, rf_predictions)))
     print("="*80)
     print(classification_report(target_val, rf_predictions))
```

```
Accuracy score: 0.8537414965986394
================================================================================
              precision    recall  f1-score   support

           0       0.90      0.93      0.91       245
           1       0.57      0.49      0.53        49

    accuracy                           0.85       294
   macro avg       0.74      0.71      0.72       294
```

0s completed at 00:41

Script Re-*Coding* Kaggle *Grandmaster* (Lanjutan)