Pengembangan Model Kecerdasan Mesin Extreme Gradient Boosting untuk Prediksi Keberhasilan Studi Mahasiswa

Evan Julian Sudarman[#1], Setia Budi[*2]

[#]*S1 Sistem Informasi, Universitas Kristen Maranatha*
Jalan Prof. Drg. Suria Sumantri No. 65, Bandung 40164, Indonesia
[1]2073008@maranatha.ac.id
[2]setia.budi@it.maranatha.edu

Abstract — **The purpose of this study is to investigate the application of the Extreme Gradient Boosting (XGBOOST) model in machine learning for a dataset on student academic achievement. The research employs a quantitative approach, utilizing student retention data and conducting data processing in Jupyter Notebook. The primary outcome of this study is the performance evaluation of the Extreme Gradient Boosting machine learning model. The significance of this research lies in the development of a machine learning model that can aid higher education institutions in forecasting student academic success. The key discoveries of this study encompass predictive factors associated with student academic achievement. The Extreme Gradient Boosting model has demonstrated its effectiveness in predicting the student academic success dataset with an accuracy rate of 76.8%.**

Keywords— **Accuracy, Extreme Gradient Boosting, Machine Learning, Predictive Factors, Student Academic Success.**

I. PENDAHULUAN

Program studi Sistem Informasi merupakan program studi yang menggabungkan beberapa rumpun ilmu menjadi satu program studi. Di dalam program studi Sistem Informasi diajarkan ilmu komputer, ilmu ekonomi manajemen, dan ilmu bisnis. Fokus pembelajaran dari program studi ini adalah perangkat lunak dan bisnis, akan tetapi tidak sekedar mempelajari cara mengembangkan sebuah program, kuliah di program studi ini juga memiliki fokus studi analisis data dan pembelajaran mesin. Sehingga selain mempelajari teknik pemrograman, mahasiswa juga diberi pengetahuan terkait cara mengolah data dan menarik kesimpulan dari sebuah dataset yang dapat dijadikan informasi yang berguna untuk pengembangan sebuah institusi profit dan non-profit.

Pada program studi Sistem Informasi juga diajarkan terkait pengolahan data menggunakan bahasa pemrograman Python dan beberapa library khusus yang mendukung proses eksplorasi data dan pembelajaran mesin [1]. Program studi sistem informasi di Universitas Kristen Maranatha juga mengajarkan beberapa mata kuliah di rumpun data analisis. Pada beberapa mata kuliah tersebut diajarkan menggunakan library milik bahasa pemrograman Python seperti Matplotlib, Pandas, Seaborn, SciPy, dan SKLearn [2].

Sebagai teknologi yang sedang berkembang dengan sangat pesat, teknologi Pembelajaran Mesin atau Machine Learning merupakan bagian dari ilmu Kecerdasan Buatan atau Artificial Intelligence [3]. Teknologi pembelajaran mesin itu sendiri memiliki banyak pemodelan yang tentunya memiliki fungsi atau kegunaan yang berbeda-beda pada setiap modelnya. Masing-masing model memiliki fungsi yang berbeda tergantung dari karakteristik data yang ingin dilakukan eksplorasi [3].

Salah satu fungsi dari teknologi pembelajaran mesin adalah melakukan klasifikasi dan prediksi data ke dalam beberapa kategori atau kelas berdasarkan fitur-fitur yang dimiliki oleh dataset [3]. Hal ini dapat digunakan untuk memprediksi hasil yang mungkin terjadi di masa yang akan datang. Hasil prediksi dari model pembelajaran mesin dapat sangat berguna untuk sebuah instansi profit maupun non-profit seperti perguruan tinggi. Prediksi yang dapat dilakukan untuk membantu perguruan tinggi salah satunya adalah prediksi keberhasilan studi mahasiswa di perguruan tinggi tersebut [4]. Model pembelajaran mesin yang dapat digunakan untuk melakukan prediksi keberhasilan studi mahasiswa di perguruan tinggi adalah Extreme Gradient Boosting. Extreme Gradient Boosting merupakan model pembelajaran mesin yang mampu melakukan pembelajaran mesin dengan teknik klasifikasi maupun regresi.

II. KAJIAN TEORI

A. Pembelajaran Mesin

Pembelajaran mesin merupakan bidang yang berkembang pesat di era digital saat ini. Menurut Alpaydin [5], pembelajaran mesin adalah suatu disiplin ilmu komputer yang mempelajari bagaimana membuat sebuah sistem komputer dapat belajar dari data, tanpa perlu diprogram secara eksplisit. Dalam pembelajaran mesin, terdapat beberapa jenis algoritma, di antaranya adalah algoritma *supervised* dan *unsupervised*. Menurut Mohri et al. [6], algoritma *supervised* adalah algoritma yang membutuhkan label pada *data training* untuk menghasilkan prediksi pada data testing, sedangkan algoritma *unsupervised* adalah algoritma yang tidak membutuhkan label pada *data training*.

Penerapan pembelajaran mesin dapat dilakukan pada berbagai bidang, salah satunya adalah dalam analisis data. Menurut Rajkumar dan Thirunavukarasu [7], pembelajaran mesin dapat membantu dalam mengolah data besar dan kompleks menjadi informasi yang bermanfaat. Selain itu, pembelajaran mesin juga dapat diterapkan pada pengenalan pola, seperti dalam pengenalan wajah atau suara. Menurut Singh dan Gupta [8], pengenalan pola adalah salah satu aplikasi utama dalam pembelajaran mesin yang dapat digunakan untuk membedakan obyek atau karakteristik yang berbeda dari suatu data.

Akan tetapi, penggunaan pembelajaran mesin juga memiliki beberapa tantangan. Salah satunya adalah *overfitting*, yaitu ketika model pembelajaran mesin terlalu kompleks sehingga mampu menghafal *data training* dengan baik, tetapi tidak mampu menggeneralisasi *data testing*. Menurut Goodfellow et al. [9], *overfitting* dapat dihindari dengan menggunakan teknik seperti validasi silang dan regularisasi.

B. Prediksi

Pembelajaran mesin merupakan bidang yang berkembang pesat di era digital saat ini. Menurut Alpaydin [5], pembelajaran mesin adalah suatu disiplin ilmu komputer yang mempelajari bagaimana membuat sebuah sistem komputer dapat belajar dari data, tanpa perlu diprogram secara eksplisit. Dalam pembelajaran mesin, terdapat beberapa jenis algoritma, di antaranya adalah algoritma *supervised* dan *unsupervised*. Menurut Mohri et al. [6], algoritma *supervised* adalah algoritma yang membutuhkan label pada *data training* untuk menghasilkan prediksi pada data testing, sedangkan algoritma *unsupervised* adalah algoritma yang tidak membutuhkan label pada *data training*.

Penerapan pembelajaran mesin dapat dilakukan pada berbagai bidang, salah satunya adalah dalam analisis data. Menurut Rajkumar dan Thirunavukarasu [7], pembelajaran mesin dapat membantu dalam mengolah data besar dan kompleks menjadi informasi yang bermanfaat. Selain itu, pembelajaran mesin juga dapat diterapkan pada pengenalan pola, seperti dalam pengenalan wajah atau suara. Menurut Singh dan Gupta [8], pengenalan pola adalah salah satu aplikasi utama dalam pembelajaran mesin yang dapat digunakan untuk membedakan obyek atau karakteristik yang berbeda dari suatu data.

Akan tetapi, penggunaan pembelajaran mesin juga memiliki beberapa tantangan. Salah satunya adalah *overfitting*, yaitu ketika model pembelajaran mesin terlalu kompleks sehingga mampu menghafal *data training* dengan baik, tetapi tidak mampu menggeneralisasi *data testing*. Menurut Goodfellow et al. [9], *overfitting* dapat dihindari dengan menggunakan teknik seperti validasi silang dan regularisasi.

C. Algoritma Pembelajaran Mesin

Algoritma adalah serangkaian instruksi atau prosedur yang digunakan untuk menyelesaikan suatu masalah atau tugas tertentu. Dalam konteks pembelajaran mesin, algoritma digunakan untuk menemukan pola atau aturan dari data, yang kemudian dapat digunakan untuk membuat prediksi atau keputusan.

D. Supervised Learning

Algoritma supervised learning adalah salah satu jenis algoritma dalam pembelajaran mesin yang membutuhkan label pada data training untuk menghasilkan prediksi pada data testing. Algoritma ini banyak digunakan dalam berbagai aplikasi seperti pengenalan gambar, klasifikasi email, dan prediksi harga saham.

Algoritma supervised learning terdiri dari dua tahap, yaitu tahap training dan tahap testing. Pada tahap training, model pembelajaran mesin akan mempelajari pola pada data training dan membuat hubungan antara fitur (features) dan label (target). Sedangkan pada tahap testing, model akan digunakan untuk memprediksi label pada data testing.

Terdapat beberapa jenis algoritma supervised learning yang sering digunakan, di antaranya adalah decision tree, naive bayes, dan neural network. Decision tree adalah algoritma yang membangun sebuah struktur pohon yang menggambarkan aturan-aturan keputusan berdasarkan fitur-fitur pada data training. Sedangkan naive bayes adalah algoritma yang memprediksi label dengan menghitung probabilitas kemunculan fitur-fitur pada setiap label yang ada pada data training. Dan neural network adalah algoritma yang memodelkan hubungan antara fitur-fitur dan label menggunakan struktur jaringan saraf tiruan.

Dalam pengaplikasiannya, algoritma supervised learning dapat menggunakan beberapa teknik untuk meningkatkan kinerjanya. Salah satunya adalah teknik ensemble, di mana beberapa model dipadukan untuk meningkatkan akurasi prediksi.

Algoritma ini dapat diterapkan dengan menggunakan metode seperti random forest dan gradient boosting. Selain itu, terdapat juga teknik preprocessing seperti feature scaling dan feature selection. Feature scaling dapat meningkatkan kinerja model dengan melakukan normalisasi pada fitur-fitur data, sedangkan feature selection dapat meningkatkan kinerja model dengan memilih fitur-fitur yang paling penting untuk prediksi.

Akan tetapi, algoritma supervised learning juga memiliki beberapa kelemahan. Salah satunya adalah ketidakmampuan untuk menangani data yang tidak seimbang (imbalanced data). Imbalanced data dapat mengakibatkan model cenderung memprediksi label yang dominan pada data, sehingga performa pada label minoritas menjadi kurang baik. Untuk mengatasi hal ini, dapat digunakan teknik-teknik seperti oversampling dan undersampling pada data training.

E. Student Retention

Student retention merupakan salah satu indikator keberhasilan dalam pendidikan tinggi. Hal ini mengacu pada kemampuan perguruan tinggi untuk mempertahankan mahasiswa hingga berhasil menyelesaikan studi mereka. Masalah retensi mahasiswa merupakan masalah global yang dihadapi oleh banyak perguruan tinggi di seluruh dunia, termasuk di Indonesia. Beberapa faktor yang mempengaruhi tingkat retensi mahasiswa antara lain lingkungan akademik, faktor sosial dan ekonomi, kualitas program studi, dan faktor personal mahasiswa [13].

Sejumlah strategi dapat diterapkan untuk meningkatkan retensi mahasiswa. Salah satu strategi yang dapat dilakukan adalah meningkatkan kualitas lingkungan akademik dan kepuasan mahasiswa terhadap kualitas pengajaran. Hal ini dapat dilakukan melalui peningkatan kualitas pengajaran, penyediaan fasilitas belajar yang memadai, dan pengembangan program mentoring [14].

Selain itu, program pembinaan dan bimbingan akademik juga dapat menjadi salah satu strategi yang efektif dalam meningkatkan retensi mahasiswa. Program ini dapat memberikan bimbingan kepada mahasiswa dalam mengatasi masalah akademik dan non-akademik yang dihadapi selama studi mereka di perguruan tinggi. Melalui program ini, mahasiswa dapat memperoleh dukungan dan bimbingan dalam mengembangkan kemampuan akademik, sosial, dan psikologis mereka [15].

Teknologi juga dapat digunakan untuk meningkatkan retensi mahasiswa. Penggunaan platform e-learning dapat memungkinkan mahasiswa untuk memperoleh akses ke berbagai materi pembelajaran, mengikuti kuliah online, dan berinteraksi dengan dosen dan mahasiswa lainnya. Selain itu, penggunaan aplikasi dan teknologi lainnya seperti mobile learning, virtual reality, dan augmented reality juga dapat memberikan pengalaman belajar yang lebih interaktif dan menarik bagi mahasiswa [16].

Secara keseluruhan, retensi mahasiswa merupakan tantangan yang signifikan dalam pendidikan tinggi. Akan tetapi, dengan menerapkan berbagai strategi seperti peningkatan kualitas lingkungan akademik, program pembinaan dan bimbingan akademik, serta penggunaan teknologi, tingkat retensi mahasiswa dapat ditingkatkan. Hal ini akan membantu perguruan tinggi dalam meningkatkan kualitas pendidikan dan memberikan kontribusi yang lebih besar bagi pembangunan nasional.

F. Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) merupakan salah satu algoritma pembelajaran mesin yang populer digunakan dalam berbagai aplikasi seperti klasifikasi, regresi, dan ranking [17]. Algoritma ini menggunakan konsep ensemble learning, yaitu menggabungkan hasil dari beberapa model untuk menghasilkan prediksi yang lebih akurat.

XGBoost memiliki beberapa keunggulan dibandingkan dengan algoritma pembelajaran mesin lainnya. Salah satu keunggulannya adalah kemampuan untuk menangani data yang besar dan kompleks dengan waktu komputasi yang relatif cepat. Selain itu, XGBoost juga memiliki kemampuan untuk menangani masalah overfitting dan dapat diatur secara fleksibel melalui parameter tuning [18].

Cara kerja XGBoost dapat dijelaskan sebagai berikut. Pertama, XGBoost melakukan inisialisasi model dengan satu nilai prediksi konstan (misalnya rata-rata target variabel) pada setiap titik data. Kemudian, model mulai menghitung residual antara nilai prediksi konstan dan nilai aktual target variabel pada setiap titik data. Setelah itu, XGBoost membangun serangkaian decision tree secara iteratif, di mana setiap pohon bertujuan untuk memperbaiki residual dari pohon sebelumnya. Proses ini dilakukan hingga mencapai kondisi penghentian seperti jumlah pohon yang telah ditentukan atau tidak ada lagi perbaikan yang signifikan dalam residual [19].

XGBoost memiliki beberapa parameter yang dapat diatur untuk meningkatkan performa model. Beberapa parameter tersebut antara lain learning rate, maximum depth, minimum child weight, dan subsample. Learning rate mengontrol seberapa besar perubahan yang dilakukan pada nilai prediksi setelah setiap iterasi pohon. Maximum depth mengontrol kedalaman maksimum setiap decision tree, sedangkan minimum child weight mengatur nilai minimum dari bobot yang dibutuhkan pada setiap cabang pohon. Subsample digunakan untuk mengatur seberapa banyak data yang digunakan pada setiap iterasi pohon [17].

Secara keseluruhan, XGBoost merupakan algoritma pembelajaran mesin yang efektif dan efisien dalam menangani data besar dan kompleks. Dengan kemampuan untuk menangani masalah overfitting dan dapat diatur secara fleksibel melalui parameter tuning, XGBoost menjadi pilihan yang tepat untuk berbagai aplikasi pembelajaran mesin.

XGBoost (Extreme Gradient Boosting) cenderung memberikan bobot lebih besar pada kelas yang jarang dalam konteks ketidakseimbangan kelas (class imbalance) karena algoritma ini berfokus pada peningkatan performa model pada kelas minoritas yang memiliki jumlah sampel yang lebih sedikit.

Ini terjadi karena dalam proses boosting, XGBoost mengoptimalkan fungsi kerugian (loss function) dengan menggunakan gradien dari fungsi tersebut. Dalam kasus ketidakseimbangan kelas, kesalahan pada kelas minoritas cenderung memberikan gradien yang lebih besar, sehingga XGBoost secara alami memberikan bobot yang lebih besar pada kelas minoritas agar dapat meminimalkan kesalahan pada kelas tersebut. Proses boosting dalam XGBoost memperkuat kemampuan model untuk menyesuaikan diri dengan data latihan yang sulit dan mencoba untuk memberikan perhatian lebih pada sampel-sampel yang lebih sulit diprediksi. Dengan memberikan bobot lebih besar pada kelas minoritas, XGBoost dapat memperkuat pengaruhnya pada proses pelatihan dan mengurangi bias yang terjadi akibat ketidakseimbangan kelas. Dalam konteks ketidakseimbangan kelas, memberikan bobot yang lebih besar pada kelas minoritas dapat membantu model untuk lebih baik dalam mengidentifikasi dan memprediksi sampel-sampel dari kelas tersebut. Namun, dalam beberapa kasus, tergantung pada konteks dan masalah yang dihadapi, penggunaan bobot yang lebih besar pada kelas minoritas juga dapat mempengaruhi kinerja pada kelas mayoritas atau menyebabkan overfitting pada kelas minoritas. Oleh karena itu, penting untuk mempertimbangkan dengan hati-hati pendekatan yang tepat dan melakukan evaluasi yang komprehensif terhadap model XGBoost dalam konteks ketidakseimbangan kelas. Berbagai teknik seperti oversampling, undersampling, atau menggunakan metrik evaluasi yang sesuai seperti precision, recall, atau F1-score dapat digunakan untuk memperoleh hasil yang lebih baik pada data yang tidak seimbang.

G. Confusion Matrix

Matriks pengukuran performa yang umum digunakan pada masalah klasifikasi adalah confusion matrix. Confusion matrix terdiri dari empat nilai, yaitu true positive (TP), true negative (TN), false positive (FP), dan false negative (FN). TP dan TN merupakan jumlah data yang diklasifikasikan dengan benar oleh model, sedangkan FP dan FN merupakan jumlah data yang salah diklasifikasikan oleh model. Berdasarkan confusion matrix, terdapat beberapa metrik pengukuran performa yang dapat dihitung, seperti precision, recall, f1-score, dan accuracy.

Precision merupakan rasio antara jumlah data yang diklasifikasikan dengan benar sebagai positif (TP) dibandingkan dengan jumlah data yang diklasifikasikan sebagai positif (TP+FP). Recall merupakan rasio antara jumlah data yang diklasifikasikan dengan benar sebagai positif (TP) dibandingkan dengan jumlah data yang sebenarnya positif (TP+FN). F1-score merupakan harmonic mean dari precision dan recall. Sedangkan accuracy merupakan rasio antara jumlah data yang diklasifikasikan dengan benar (TP+TN) dibandingkan dengan jumlah total data.

III. ANALISIS DAN RANCANGAN

Kegiatan eksplorasi pembelajaran mesin diawali dengan membedah dataset dan memahami karakteristik dataset. Pada proses ini peneliti mencoba untuk memahami bahwa dataset yang akan diolah membahas tentang apa. Lalu pada proses ini juga peneliti melakukan investigasi masing-masing feature yang ada pada dataset ini. Proses pertama ini bertujuan untuk memahami dataset dan juga menjadi landasan untuk tindakan-tindakan dari proses berikutnya. Setelah peneliti mengetahui jenis feature dan target dari dataset ini, peneliti bisa menentukan teknik pembelajaran mesin seperti apa yang tepat untuk feature dan target dari dataset ini. Setelah menentukan teknik pembelajaran mesin yang akan digunakan untuk eksplorasi pembelajaran mesin, selanjutnya peneliti menentukan model yang dapat digunakan untuk melakukan training dataset sesuai dengan teknik pembelajaran mesin yang sudah ditentukan. Selain dapat digunakan sesuai dengan teknik yang ditentukan, peneliti juga memilih model yang memiliki performa terbaik.

Setelah memahami dataset hingga menentukan model, kini dilanjutkan dengan proses konfigurasi tools, mulai dari penginstalan jupyter notebook sebagai python environment jika belum tersedia, hingga instalasi modul-modul python yang akan digunakan. Setelah semua tools untuk melakukan eksplorasi siap digunakan, selanjutnya peneliti mulai dengan tahapan awal eksplorasi data. Pada tahap ini akan dilakukan proses pre-processing data. Proses ini bertujuan untuk mempersiapkan dataset yang akan digunakan untuk melakukan pelatihan model. Berlandaskan pada pemahaman yang sudah didapatkan pada saat proses pembedahan karakteristik dataset, pada proses ini peneliti akan melakukan beberapa proses untuk memastikan data sepenuhnya dapat sesuai untuk melakukan pelatihan model.

Setelah data siap sepenuhnya, kini saatnya melakukan pelatihan model. Pelatihan model dilakukan dengan menggunakan beberapa sintaks python. Setelah model selesai dilatih, untuk mengetahui performa dan akurasi dari model, peneliti melakukan proses pengukuran model menggunakan confusion matrix dan classification report. Setelah hasil pengukuran muncul, peneliti tinggal melakukan visualisasi data dan proses intepretasi data.

Pada penelitian ini, dataset yang digunakan untuk proses eksplorasi adalah dataset keberhasilan studi mahasiswa atau student retention. Dataset ini memberikan pandangan mendalam terkait mahasiswa di institusi pendidikan. Mahasiswa yang terdapat pada dataset ini bukan hanya berasal dari satu program studi atau program studi, akan tetapi dataset ini mencakup berbagai jenis mahasiswa dari berbagai program studi. Data yang diambil dari tiap mahasiswa diantaranya:

- Data Informasi Demografis
- Data faktor sosial dan ekonomi:
  - Faktor Sosial:
    - Kebangsaan
    - Status Pernikahan
    - Jenis kebutuhan khusus mahasiswa
  - Faktor Ekonomi:
    - *Debtor*: Pengutang atau bukan.
    - Pekerjaan ayah dan ibu.
    - Tingkat inflasi, Pengangguran, dan GDP di daerah tempat tinggal mahasiswa.
- Data Kinerja Akademik
  - Data mata kuliah yang diampu per-semester.
  - Data nilai per-semester.

Data-data tersebut dapat digunakan untuk membuat analisis dan prediksi potensi seorang mahasiswa mundur dari perkuliahan atau keberhasilannya dalam perkuliahan. Dataset ini merupakan gabungan dari beberapa database terpisah, data yang didapatkan diperoleh ketika mahasiswa melakukan pendaftaran bidang perkuliahan yang diminati.

Dataset ini dapat mempermudah perguruan tinggi dalam memahami faktor ekonomi masing-masing mahasiswa. Kendala yang dihadapi oleh mahasiswa pada saat studinya di tiap semester. Hingga dapat membantu perguruan tinggi dalam memprediksi mahasiswa yang akan berhasil atau tidak dalam menempuh perkuliahannya.

Di dalam dataset ini terdapat 30 Features (X) yang memiliki jenis data berbeda-beda. Akan tetapi pada dataset ini, baik features berjenis kategorial maupun Numerik akan di konversi menjadi jenis data numerikal. Contoh konversi yang peneliti maksudkan adalah, jika data kategorial bernilai yes or no berarti akan dikonversi menjadi bilangan biner berupa 0 atau 1. Semester jika features kategorial berupa kelas-kelas seperti jenis pekerjaan, maka akan dikonversi menjadi ordinal number atau angka terurut sesuai jumlah jenis dari nilai features tersebut. Berikut dapat dilihat penjelas mengenai features-features dari dataset ini:

- Marital Status: Data yang terdapat pada feature ini berbicara tentang status perkawinan dari mahasiswa yang mendaftar kuliah. (Kategorial)
- Application Mode: Data yang terdapat pada feature ini berbicara tentang jenis application atau jalur pendaftaran mahasiswa saat pertama mendaftar. (Kategorial)
- Application Order: Data yang terdapat pada feature ini berbicara tentang gelombang pendaftaran dan penerimaan mahasiswa baru. (Numerik)
- Course: Data yang terdapat pada feature ini berbicara tentang program studi yang diambil oleh mahasiswa. (Kategorial)
- Daytime (Kategorial): Data yang terdapat pada feature ini berbicara tentang jenis waktu perkuliahan yang diambil oleh mahasiswa. Mungkin jika peneliti tarik relevansi dengan kondisi perkuliahan di indonesia, peneliti mengetahui bahwa secara umum terdapat 2 jenis perkuliahan yaitu:
- Kuliah mahasiswa pada umumnya yang dilaksanakan secara tatap muka di siang hari.
- Kuliah pegawai yang dilaksanakan pada malam hari.
- Previous Qualification: Data yang terdapat pada feature ini berbicara tentang jenjang pendidikan sebelumnya yang diampu oleh mahasiswa sebelum masuk perkuliahan. (Kategorial)
- Nacionality: Data yang terdapat pada feature ini berbicara tentang kewarganegaraan dari mahasiswa yang mendaftar. (Kategorial)
- Mother Qualification: Data yang terdapat pada feature ini berbicara tentang pendidikan terakhir dari ibu mahasiswa. (Kategorial)
- Father Qualification: Data yang terdapat pada feature ini berbicara tentang pendidikan terakhir dari ayah mahasiswa. (Kategorial)
- Mother Occupation: Data yang terdapat pada feature ini berbicara tentang pekerjaan dari ibu mahasiswa. (Kategorial)
- Father Occupation: Data yang terdapat pada feature ini berbicara tentang pekerjaan dari ayah mahasiswa. (Kategorial)
- Displaced: Data yang terdapat pada feature ini berbicara tentang, apakah mahasiswa ini merupakan anak terlantar atau tidak. (Kategorial)

- Educational Special Needs: Data yang terdapat pada feature ini berbicara tentang apakah mahasiswa ini merupakan anak berkebutuhan khusus atau tidak.(Kategorial)
- Debtor: Data yang terdapat pada feature ini berbicara tentang, apakah mahasiswa ini merupakan pengutang atau bukan. (Kategorial)
- Tuition fees up to date: Data yang terdapat pada feature ini berbicara tentang status pembayaran perkuliahan mahasiswa saat ini. Status yang dimaksud adalah terkait cekal pembayaran. (Kategorial)
- Gender: Data yang terdapat pada feature ini berbicara tentang jenis kelamin dari mahasiswa. (Kategorial)
- Scholarship Holder: Data yang terdapat pada feature ini berbicara tentang, apakah mahasiswa ini merupakan penerima beasiswa atau bukan. (Kategorial)
- Age at enrollment: Data yang terdapat pada feature ini berbicara tentang umur mahasiswa saat mendaftar perkuliahan. (Numerik)
- International: Data yang terdapat pada feature ini berbicara tentang, apakah mahasiswa ini merupakan mahasiswa international atau bukan.
- Curricular units 1st sem (credited): Data yang terdapat pada feature ini berbicara tentang jumlah sks maksimal yang dapat diambil oleh mahasiswa ini pada semester pertama. (Numerik)
- Curricular units 1st sem (enrolled): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang ingin diambil oleh mahasiswa ini pada semester pertama. (Numerik)
- Curricular units 1st sem (evaluations): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang dievaluasi oleh dosen pada semester pertama. (Numerik)
- Curricular units 1st sem (approved): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang diterima oleh dosen dan diampu oleh mahasiswa ini pada semester pertama. (Numerik)
- Curricular units 2nd sem (credited): Data yang terdapat pada feature ini berbicara tentang jumlah sks maksimal yang dapat diambil oleh mahasiswa ini pada semester kedua. (Numerik)
- Curricular units 2nd sem (enrolled): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang ingin diambil oleh mahasiswa ini pada semester kedua. (Numerik)
- Curricular units 2nd sem (evaluations): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang dievaluasi oleh dosen pada semester kedua. (Numerik)
- Curricular units 2nd sem (approved): Data yang terdapat pada feature ini berbicara tentang jumlah sks yang diterima oleh dosen dan diampu oleh mahasiswa ini pada semester kedua. (Numerik)
- Unemployment rate: Data yang terdapat pada feature ini berbicara tentang angka pengangguran di wilayah atau domisili tempat mahasiswa tinggal. (Numerik)
- Inflation rate: Data yang terdapat pada feature ini berbicara tentang tingkat inflasi dari wilayah atau domisili tempat mahasiswa tinggal. (Numerik)
- GDP: Data yang terdapat pada feature ini berbicara tentang Gross Domestic Product atau nilai barang yang dihasilkan negara pada periode waktu tertentu. GDP dapat diukur juga ditiap daerah. Pada feature ini GDP yang dimaksud adalah GDP di masing-masing daerah tempat peserta tinggal. (Numerik)

Dataset ini akan diolah menggunakan metode klasifikasi, peneliti menggunakan metode klasifikasi, dikarenakan metode ini merupakan metode yang tepat untuk memprediksi sebuah target atau (keluaran) outcome berupa kelas-kelas (Kategorial). Selain itu, jika peneliti lihat dari tipe data yang dimiliki oleh masing-masing feature, tipe data yang dimiliki oleh masing-masing feature adalah nominal dan kategorial. Hal ini tentunya selaras dengan ketentuan penggunaan metode klasifikasi untuk model pembelajaran mesin.

Setelah model pembelajaran mesin dilatih menggunakan metode klasifikasi, akan dilakukan proses pengukuran performa menggunakan Confusion Matrix. Confusion Matrix digunakan untuk klasifikasi karena menyediakan informasi detail tentang kinerja model klasifikasi dalam memprediksi kelas atau label target. Confusion Matrix adalah tabel yang menunjukan jumlah prediksi yang benar atau salah yang dilakukan oleh model klasifikasi pada dataset pengujian. Dengan menghitung confusion matrix, peneliti dapat menghitung berbagai metrik kinerja klasifikasi seperti accuration, precision, recall, F1-Score, dan area dibawah kurva ROC.

IV. IMPLEMENTASI

A. Pembentukan Model Extreme Gradient Boosting

XGBoost (Extreme Gradient Boosting) adalah salah satu algoritma pembelajaran mesin yang populer untuk masalah regresi dan klasifikasi. Strategi pemisahan data yang digunakan dalam validasi model adalah *Holdout, K-Fold Cross Validation*, dan *Stratified K-Fold Cross Validation*.

B. Instalasi Modul XGBoost

Modul *Extreme Gradient Boosting* sendiri tidak terdapat pada library modul *scikit-learn* oleh karena itu penting untuk melakukan instalasi modul XGBoost terlebih dulu pada file *jupyter notebook* yang peneliti gunakan. Untuk melakukan instalasi peneliti dapat menggunakan "!pip install xgboost" seperti pada Gambar 1.

```
In [31]:  !pip install xgboost
```

Gambar 1. Instalasi Modul

C. Memasukan (Impor) Library pada Jupyter Notebook

Pada proses pengembangan model pembelajaran mesin XGBoost ini peneliti akan menggunakan banyak *library* pendukung bahasa pemrograman python. Maka dari itu peneliti perlu untuk memasukan atau melakukan proses impor *library* yang dibutuhkan. Berikut daftar *library* yang akan dibutuhkan pada proses pengembangan model pembelajaran mesin XGBoost:

- Pandas: *Library* ini digunakan untuk melakukan proses pengolahan dataset yang akan digunakan ke dalam bentuk dataframe yang dibuat oleh *pandas.*
- Xgboost: *Library* ini digunakan untuk pembentukan model mesin *extreme gradient boosting* yang nantinya akan menjadi wadah atau otak dari pemrosesan dan pembelajaran data.
- Accuracy Score: *Library* ini digunakan untuk melakukan pengukuran akurasi dari model extreme gradient boosting yang telah dibentuk, dilatih, dan diuji.
- Cross_val_score & Kfold: *Library* ini digunakan untuk melakukan proses *cross validation* menggunakan teknik Kfold.
- Stratified Kfold: *Library* ini digunakan untuk melakukan proses *cross validation* menggunakan teknik Stratified Kfold.

```
In [156]: import pandas as pd
          import xgboost as xgb
          from sklearn.metrics import accuracy_score
          from sklearn.model_selection import cross_val_score, KFold
          from sklearn.model_selection import StratifiedKFold
```

Gambar 2. Importing Modul

D. Mengkonversi dataset menjadi Pandas Dataframe

Pada proses ini akan dilakukan pengolahan dataset menjadi dataframe, dan juga proses pemisahan dataframe menjadi 2 variabel yaitu X (features) dan y (target). Untuk melakukan import dataset menjadi pandas dataframe peneliti dapat menggunakan fungsi "read_csv()" yang terdapat pada modul pandas seperti pada Gambar 3. Modul pandas disini akan peneliti andaikan menjadi "pd" seperti yang dapat peneliti lihat pada langkah sebelumnya pada Gambar 2 mengenai proses *import* modul. Fungsi "read_csv()" sendiri disini bermaksud untuk proses konversi dataset berformat excel *Comma Separated Value* (CSV) menjadi pandas dataframe. Disini perlu peneliti pahami bahwa tujuan dari konversi dataset menjadi pandas dataframe adalah untuk mempermudah pemrosesan kedepannya menggunakan modul *scikit-learn*.

```
# Import dataset menggunakan pandas dataframe.
data = pd.read_csv('dataset.csv')
```

Gambar 3. Konversi Dataset

Setelah melakukan proses pengolahan dataset menjadi pandas dataframe, selanjutnya peneliti akan melakukan proses penentuan feature dan target menjadi 2 buah variable X dan y. Proses ini akan dilakukan dengan menggunakan fungsi "iloc" milik pandas seperti pada Gambar 4 Penggunaan fungsi "iloc" sendiri bertujuan untuk mengindek baris dan kolom berdasarkan lokasi. Fungsi "iloc" disini akan peneliti masukan kedalam variable X dan y. Pada variabel X peneliti akan menggunakan indeks ":,:-1" yang berarti mengambil semua baris dan kolom kecuali kolom terakhir dari obyek dataframe. Titik dua yang pertama memiliki arti mengambil setiap baris dan kolom. Sementara titik dua yang kedua berarti akan mengecualikan 1 kolom terakhir. Indeks yang dibentuk pada fungsi "iloc" ini memiliki arti bahwa peneliti akan menggunakan setiap baris dan kolom, kecuali kolom terakhir sebagai features. Sementara pada variabel y peneliti akan menggunakan indeks ":, -1" yang berarti mengambil semua baris dari kolom terakhir.

```
# Variable X sebagai feature dan y sebagai target.
X, y = data.iloc[:,:-1], data.iloc[:,-1]
```

Gambar 4. Fungsi Iloc

E. Review Dataframe, Variabel Target, Variabel Feature.

Proses review dilakukan peneliti untuk meninjau kesiapan dataframe, variabel target dan variabel feature untuk melakukan proses *training* dan *testing* pembelajaran mesin. Untuk melihat kesiapan variabel dan data yang akan digunakan, peneliti hanya cukup memanggil variabel terkait dan menjalankannya pada jupyter notebook seperti dibawah ini.

```
In [159]: data
Out[159]:
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	...	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)
0	1	8	5	2	1	1	1	13	10	6	...	0	0
1	1	6	1	11	1	1	1	1	3	4	...	0	6
2	1	1	5	5	1	1	1	22	27	10	...	0	6
3	1	8	2	15	1	1	1	23	27	6	...	0	6
4	2	12	1	3	0	1	1	22	28	10	...	0	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4419	1	1	6	15	1	1	1	1	1	6	...	0	6
4420	1	1	2	15	1	1	19	1	1	10	...	0	6
4421	1	1	1	12	1	1	1	22	27	10	...	0	8

Gambar 5. Dataframe *Student Retention*

Pada Gambar peneliti mencoba untuk melihat setiap baris data dan kolom data yang tersedia pada dataframe yang berada di dalam variabel “data”. Setelah itu peneliti juga dapat melakukan hal serupa, dengan memanggil variabel X dan y untuk melihat *features* dan *target* yang telah ditetapkan.

```
In [165]: X
Out[165]:
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	...	Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)	C
0	1	8	5	2	1	1	1	13	10	6	...	0	0	
1	1	6	1	11	1	1	1	1	3	4	...	0	0	
2	1	1	5	5	1	1	1	22	27	10	...	0	0	
3	1	8	2	15	1	1	1	23	27	6	...	0	0	
4	2	12	1	3	0	1	1	22	28	10	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4419	1	1	6	15	1	1	1	1	1	6	...	0	0	
4420	1	1	2	15	1	1	19	1	1	10	...	0	0	
4421	1	1	1	12	1	1	1	22	27	10	...	0	0	
4422	1	1	1	9	1	1	1	22	27	8	...	0	0	
4423	1	5	1	15	1	1	9	23	27	6	...	0	0	

4424 rows × 34 columns

Gambar 5. Pemanggilan Variabel X

Seperti yang dapat peneliti lihat diatas, bahwa disini peneliti memanggil variabel X untuk melihat kolom-kolom apa saja yang akan dijadikan *features*. Selanjutnya peneliti akan memanggil variabel y untuk melihat kesiapan data dan kolom yang akan dijadikan sebagai *target*.

```
In [158]: y

Out[158]: 0        Dropout
          1       Graduate
          2        Dropout
          3       Graduate
          4       Graduate
                    ...
          4419    Graduate
          4420     Dropout
          4421     Dropout
          4422    Graduate
          4423    Graduate
          Name: Target, Length: 4424, dtype: object
```

Gambar 6. Pemanggilan Variabel y

F. Strategi Holdout

Strategi holdout adalah salah satu metode pemisahan data yang digunakan dalam proses validasi model. Metode ini melibatkan pembagian dataset menjadi dua subset yang saling eksklusif: subset pelatihan (training set) dan subset pengujian (test set). Subset pelatihan digunakan untuk melatih model, sedangkan subset pengujian digunakan untuk menguji performa model yang dilatih. Untuk melakukan pemodelan menggunakan strategi *holdout*.

G. Pemisahan Data Menggunakan Fungsi "train_test_split()".

Pertama-tama peneliti harus melakukan pemisahan data menjadi *training set* (X) dan *test set* (y). Untuk melakukan pemisahan data, peneliti bisa menggunakan fungsi train_test_split() dan membantu dalam membagi data sesuai dengan bobot yang tepat.

```
In [44]: # bagi dataset menjadi data latih dan data uji
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Gambar 7. Pemisahan Data

Melalui Gambar 7, dapat lihat bahwa terdapat 4 buah parameter diantaranya:

- Parameter Pertama: Parameter ini digunakan untuk menempatkan variabel X atau variabel yang menampung data feature.
- Parameter Kedua: Parameter ini digunakan untuk menempatkan variabel y atau variabel yang menampung data target.
- Parameter Ketiga: Parameter "test_size" dalam fungsi train_test_split digunakan untuk menentukan proporsi subset pengujian (test set) dalam pemisahan data. Secara khusus, parameter ini mengatur persentase atau fraksi dari dataset awal yang akan dialokasikan sebagai subset pengujian. "test_size=0.3" menunjukkan bahwa 30% dari data akan dialokasikan sebagai subset pengujian, sedangkan 70% sisanya akan digunakan sebagai subset pelatihan.
- Parameter Keempat: Parameter random_state dalam fungsi train_test_split digunakan untuk mengontrol pemilihan data secara acak saat membagi dataset menjadi subset pelatihan dan subset pengujian. Parameter ini memastikan bahwa pembagian data yang dihasilkan konsisten dan dapat direproduksi jika kode dieksekusi lagi dengan nilai random_state yang sama. "random_state=42" menunjukkan bahwa seed (benih) yang digunakan dalam pemilihan data secara acak adalah 42. Dengan menggunakan nilai "random_state" yang sama, pemilihan data secara acak akan selalu menghasilkan pembagian yang sama jika kode dieksekusi kembali dengan nilai "random_state" yang sama. Pemilihan angka mana pun untuk random_state akan menghasilkan hasil yang konsisten jika digunakan secara konsisten dalam eksekusi yang berulang. Tujuannya adalah memastikan bahwa pemilihan data secara acak dapat direproduksi dengan cara yang sama setiap kali kode dieksekusi. Dalam praktiknya, angka apa pun dapat digunakan sebagai nilai "random_state", selama Anda menggunakan nilai yang sama saat menjalankan kode yang sama untuk membagi dataset.

H. Pembuatan Model Pembelajaran Mesin XGB

Untuk membuat model pembelajaran mesin dengan mendeklarasikannya kedalam variabel, peneliti menggunakan fungsi "XGBClassifier()" yang terdapat pada kelas "xgb". Fungsi "XGBClassifier()" digunakan untuk membentuk model pembelajaran mesin yang mampu belajar menggunakan teknik klasifikasi.

```
In [45]: xgb_model = xgb.XGBClassifier(max_depth=5, learning_rate=0.1, n_estimators=100)
```

Gambar 8. Deklarasi Model

Dapat dilihat pada Gambar 8, bahwa terdapat 3 parameter yang digunakan untuk membantu dalam membentuk model pembelajaran mesin dengan performa yang optimal, sehingga tidak terjadi overfitting maupun underfitting pada saat proses validasi. Berikut penjelasan mengenai parameter-parameter yang digunakan:

- Parameter Pertama (max_depth): Parameter ini mengontrol kedalaman maksimum pohon dalam model. Semakin dalam pohon, semakin rumit modelnya. Namun, jika terlalu dalam, model dapat menjadi overfitting pada data training.
- Parameter Kedua (learning rate): Parameter ini mengontrol ukuran langkah yang diambil dalam setiap iterasi selama pelatihan model. Semakin kecil nilai learning rate, semakin lambat konvergensi, tetapi lebih akurat hasilnya. Konvergensi mengacu pada keadaan di mana model secara iteratif mengupdate parameter-nya menggunakan langkah-langkah yang lebih kecil dalam upaya mencapai hasil yang lebih akurat atau mendekati solusi yang optimal.
- Parameter Ketiga (n_estimator): Parameter ini menentukan jumlah pohon dalam model. Semakin banyak pohon, semakin kuat model. Namun, nilai ini juga akan mempengaruhi waktu pelatihan model.

I. Pelatihan Model

Untuk melakukan pelatihan model peneliti terlebih dahulu melakukan Transformasi label atau target menjadi representasi numerik menggunakan obyek ““OrdinalEncoder”()” pada data pelatihan (y_train) dan data pengujian (y_test).

```
In [46]: y_train = y_train.values.reshape(-1, 1)
         ordinal_encoder = OrdinalEncoder()
         y_train = ordinal_encoder.fit_transform(y_train)
         y_test = y_test.values.reshape(-1, 1)
         y_test = ordinal_encoder.fit_transform(y_test)
```

Gambar 10. Transformasi Label

Berikut merupakan penjelasan untuk setiap baris kode pada Gambar :

- y_train = y_train.values.reshape(-1, 1): Pada baris ini, peneliti mengubah bentuk atau dimensi dari y_train menjadi matriks dengan satu kolom. Hal ini dilakukan menggunakan metode values.reshape(-1, 1). Transformasi ini diperlukan karena obyek ““OrdinalEncoder”” dalam scikit-learn mengharapkan input berbentuk array 2D, dengan satu kolom untuk representasi target.
- ordinal_encoder = “OrdinalEncoder”(): Pada baris ini, peneliti membuat obyek ““OrdinalEncoder”” dari scikit-learn. “OrdinalEncoder” digunakan untuk mengubah label kategori menjadi representasi numerik berdasarkan urutan ordinal.
- y_train = ordinal_encoder.fit_transform(y_train): Di baris ini, peneliti menggunakan fit_transform dari “OrdinalEncoder” untuk mengubah nilai label pada y_train menjadi representasi numerik. fit_transform akan menghitung mapping antara nilai kategori dan nilai numerik yang sesuai, dan kemudian mengubah nilai kategori menjadi nilai numerik. Hasil transformasi ini akan disimpan kembali ke y_train.
- y_test = y_test.values.reshape(-1, 1): Pada baris ini, peneliti mengubah bentuk atau dimensi dari y_test menjadi matriks dengan satu kolom, mirip dengan langkah pertama untuk y_train.
- y_test = ordinal_encoder.fit_transform(y_test): Di baris ini, peneliti menggunakan obyek “OrdinalEncoder” yang sama yang telah dibuat sebelumnya untuk mengubah nilai label pada y_test menjadi representasi numerik. Penting untuk dicatat bahwa peneliti menggunakan fit_transform pada y_test dengan obyek “OrdinalEncoder” yang sama yang telah dilatih pada y_train. Ini penting agar mapping antara nilai kategori dan nilai numerik konsisten antara data pelatihan dan data pengujian.

Dengan transformasi ini, label kategori pada y_train dan y_test akan diubah menjadi representasi numerik sesuai dengan urutan ordinalnya. Transformasi semacam ini sering digunakan pada masalah klasifikasi yang membutuhkan representasi numerik untuk variabel target, seperti ketika menggunakan algoritma pembelajaran mesin yang memerlukan input numerik.

Setelah parameter “X_train” (Feature Train) dan “y_train” (Target Train) siap digunakan peneliti hanya perlu memanggil fungsi “fit()” yang dimiliki model pembelajaran mesin *Extreme Gradient Boosting*. Lalu ketika baris kode dijalankan, proses pembelajaran mesin mulai dilakukan seperti pada Gambar 9.

```
In [47]: xgb_model.fit(X_train, y_train)

Out[47]: XGBClassifier(base_score=None, booster=None, callbacks=None,
                       colsample_bylevel=None, colsample_bynode=None,
                       colsample_bytree=None, early_stopping_rounds=None,
                       enable_categorical=False, eval_metric=None, feature_types=None,
                       gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                       interaction_constraints=None, learning_rate=0.1, max_bin=None,
                       max_cat_threshold=None, max_cat_to_onehot=None,
                       max_delta_step=None, max_depth=5, max_leaves=None,
                       min_child_weight=None, missing=nan, monotone_constraints=None,
                       n_estimators=100, n_jobs=None, num_parallel_tree=None,
                       objective='multi:softprob', predictor=None, ...)
```

Gambar 9. Proses Pembelajaran Mesin

J. Prediksi dan Pengukuran Akurasi (Validasi)

Setelah peneliti menjalankan proses *training* model *Extreme Gradient Boosting*. Selanjutnya adalah proses validasi yang meliputi prediksi dan pengukuran akurasi. Prediksi dilakukan dengan menggunakan fungsi "predict()" yang dimiliki model *Extreme Gradient Boosting* seperti pada Gambar 10.

```
In [9]: y_pred = xgb_model.predict(X_test)
```

Gambar 10. Proses Prediksi

Setelah melakukan proses prediksi, selanjutnya akan dilakukan proses pengukuran akurasi. Nilai akurasi sendiri menunjukan ketepatan prediksi, pada saat proses prediksi, model mesin diuji dengan sebuah *testing set*, lalu mesin akan menebak kelas-kelas dari testing set tersebut. Tingkat ketepatan dari tebakan mesin akan meningkatkan nilai akurasi. Hasil dari pengukuran akurasi dari model *Extreme Gradient Boosting* yang dibentuk adalah 0.7703. Proses pengukuran akurasi dapat terlihat seperti pada Gambar 11.

```
In [10]: print("Accuracy Score: ",accuracy_score(y_test,y_pred))
```

Gambar 11. Pengukuran Akurasi

```
Accuracy Score:  0.7703313253012049
```

Gambar 12. Hasil Pengukuran Akurasi

K. Strategi Cross Validation K-Fold & Stratified K-Fold

Strategi *Cross Validation* K-Fold dan *Stratified* K-Fold adalah dua metode validasi silang yang umum digunakan dalam pembelajaran mesin untuk mengevaluasi performa model dengan lebih akurat dan stabil. Keduanya melibatkan pemisahan dataset menjadi subset pelatihan dan pengujian, tetapi dengan pendekatan yang sedikit berbeda.

Dalam K-Fold *Cross Validation*, dataset dibagi menjadi K fold atau subset yang saling eksklusif dengan ukuran yang hampir sama. Selama proses validasi, model dilatih dan dievaluasi K kali, di mana pada setiap iterasi, salah satu fold digunakan sebagai subset pengujian (test set), sementara K-1 Fold lainnya digunakan sebagai subset pelatihan (training set). Setelah K iterasi selesai, hasil evaluasi dari setiap iterasi digabungkan untuk memberikan perkiraan yang lebih stabil tentang performa model secara keseluruhan. Keuntungan K-Fold *Cross Validation* adalah semua data akan digunakan baik sebagai data pelatihan maupun pengujian, sehingga meminimalkan bias dalam evaluasi model. Namun, metode ini bisa menjadi lebih lambat dalam komputasi jika K cukup besar.

Stratified K-Fold *Cross Validation* adalah varian dari K-Fold *Cross Validation* yang mempertahankan proporsi kelas yang seimbang di setiap fold. Biasanya, dalam kasus ketidakseimbangan kelas, *Stratified* K-Fold *Cross Validation* lebih disukai daripada K-Fold *Cross Validation* biasa, karena memastikan bahwa setiap fold memiliki distribusi kelas yang serupa dengan keseluruhan dataset. Dalam *Stratified* K-Fold *Cross Validation* proses pemisahan data dilakukan dengan mempertahankan proporsi kelas yang seimbang di setiap fold, sehingga setiap fold mewakili distribusi kelas secara proporsional. Pendekatan ini penting dalam situasi ketidakseimbangan kelas, di mana ada perbedaan jumlah yang signifikan antara kelas mayoritas dan kelas minoritas.

Keduanya, K-Fold *Cross Validation* dan *Stratified* K-Fold *Cross Validation*, merupakan metode validasi silang yang berguna untuk memberikan perkiraan performa yang lebih akurat dan konsisten dari model yang dilatih. Pilihan antara K-Fold dan *Stratified* K-Fold tergantung pada sifat dataset, terutama apakah dataset memiliki ketidakseimbangan kelas yang signifikan atau tidak. Jika dataset memiliki ketidakseimbangan kelas, *Stratified* K-Fold direkomendasikan untuk memastikan evaluasi yang lebih obyektif.

L. Pembuatan Model Pembelajaran Mesin Extreme Gradient Boosting

Untuk membuat model pembelajaran mesin dengan mendeklarasikannya kedalam variabel, peneliti menggunakan fungsi "XGBClassifier()" yang terdapat pada kelas "xgb". Fungsi "XGBClassifier()" digunakan untuk membentuk model pembelajaran mesin yang mampu belajar menggunakan teknik klasifikasi.

```
In [19]: xgb_model = xgb.XGBClassifier(colsample_bytree=0.848007252062556,
                                       learning_rate= 0.5682837569370375,
                                       max_depth= 5,
                                       n_estimators= 100,
                                       subsample= 0.8100050604323482)
```

Gambar 13. Pembuatan Model Pembelajaran Mesin

M. Hypertuning Parameter

Proses *Hypertuning Parameter* dilakukan dengan menggunakan metode *Grid Search*. Tujuan dari proses ini adalah mengetahui parameter paling optimal dari sebuah model. Sehingga dapat menjadi acuan parameter mana yang akan digunakan pada fase pengembangan model. Untuk mencari parameter paling optimal dari sebuah model, peneliti menggunakan metode *Grid Search*, metode ini bekerja dengan membandingkan beberapa nilai parameter, Grid Search juga memiliki output berupa nilai akurasi. Nilai akurasi ini yang akan peneliti gunakan untuk menentukan *hyper parameter* yang akan digunakan pada model. Untuk menemukan parameter terbaik. Sebelum memulai proses *hyperparameter tuning*, sebelumnya peneliti perlu membagi data menjadi *validation* dan *testing*. Tujuannya adalah untuk membandingkan nilai akurasi dari *validation set* dan *testing set*. Sehingga peneliti dapat melihat overfitting dari nilai akurasi testing set yang peneliti miliki. Untuk membagi menjadi *validation set* dan *testing set* peneliti cukup menggunakan train_test_split seperti pada Gambar 14.

```
# bagi dataset menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

Gambar 14. *Validation and Testing Set*

Dapat peneliti lihat pada Gambar 15. *Grid Search Parameter Set*, peneliti membuat sebuah struktur data dict yang digunakan untuk membuat set atau kumpulan data yang akan digunakan untuk proses *Grid Search*.

```
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 500, 1000]
}
```

Gambar 15. *Grid Search Parameter Set*

```
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5)
```

Gambar 16. *Grid Search Obyek*

```
grid_search.fit(X_train_split, y_train_split_encoded)
print("Hyperparameter terbaik: ", grid_search.best_params_)
```

```
Hyperparameter terbaik:  {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 1000}
```

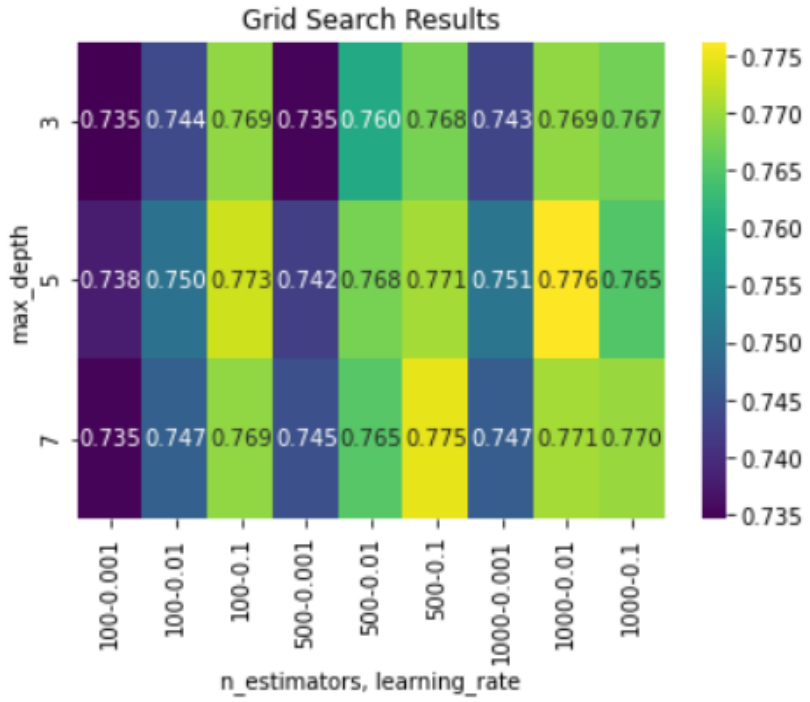
Gambar 17. *Grid Search Fitting*

Setelah parameter terbaik dari muncul dari proses *grid search*, selanjutnya peneliti menampilkan hasil nilai akurasi dari proses pelatihan yang telah dilakukan menggunakan *dictionary* param_grid. Untuk memunculkan hasil dari nilai akurasi, dapat dilihat pada Gambar 18. Dan untuk mempermudah mengevaluasi nilai akurasi dari grid_results, peneliti membentuk sebuah visualisasi data dengan model *heatmap*. Untuk membentuk heatmap dari grid_results, peneliti menggunakan modul seaborn. Hasil dari visualisasi grid_results dapat dilihat pada Gambar 19.

```
grid_results = grid_search.cv_results_
print(grid_results)
```

```
{'mean_fit_time': array([ 0.3620028 ,  1.43189573,  3.31589131,  0.45464907,  2.39024734,
        4.23356194,  0.73122463,  3.25142198,  8.54847002,  0.33757658,
        2.36830568,  4.12447562,  0.63155637,  3.39853845,  7.34364748,
        0.70415707,  4.90810757, 11.30673032,  0.41183028,  2.48591528,
        5.04277458,  0.42267156,  3.42762914,  8.11369295,  0.87329016,
        5.59994359,  9.17880082]), 'std_fit_time': array([0.08106975, 0.06002714, 0.57549421, 0.05838437, 0.16834807,
       0.09219341, 0.06048231, 0.24588795, 2.07919837, 0.01645123,
       0.63486878, 0.98233993, 0.15407797, 1.581915  , 1.71405548,
       0.11980542, 1.4092692 , 2.33088625, 0.12543517, 1.30212083,
       1.86468897, 0.03289774, 1.33914869, 1.54424441, 0.24647861,
       1.58723349, 1.29476844]), 'mean_score_time': array([0.00840073, 0.00960035, 0.0162015 , 0.00780025, 0.01260133,
       0.01700201, 0.00880098, 0.01540155, 0.01963091, 0.00760093,
       0.0172029 , 0.02500501, 0.00820031, 0.01556239, 0.01900082,
       0.0110064 , 0.02644968, 0.033002  , 0.00913305, 0.01199808,
       0.01500125, 0.00824003, 0.01920156, 0.02180142, 0.01118279,
       0.0163662 , 0.03160238]), 'std_score_time': array([0.00195954, 0.00101963, 0.00515482, 0.00213397, 0.00265474,
       0.00200059, 0.00074839, 0.00326232, 0.00298409, 0.00080038,
       0.0098685 , 0.01028112, 0.00299337, 0.00689173, 0.00399987,
       0.00402952, 0.0094246 , 0.02850517, 0.00185196, 0.00368786,
       0.00460517, 0.00299173, 0.00930471, 0.00646305, 0.00316808,
       0.006014  , 0.01048125]), 'param_learning_rate': masked_array(data=[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.01,
                   0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.001,
                   0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False],
       fill_value='?',
            dtype=object), 'param_max_depth': masked_array(data=[3, 3, 3, 5, 5, 5, 7, 7, 7, 3, 3, 3, 5, 5, 5, 7, 7, 7,
                   3, 3, 3, 5, 5, 5, 7, 7, 7],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False],
       fill_value='?',
            dtype=object), 'param_n_estimators': masked_array(data=[100, 500, 1000, 100, 500, 1000, 100, 500, 1000, 100,
```

Gambar 18. *Grid Result*



Gambar 19. *Heatmap Grid Results*

Selanjutnya peneliti membandingkan nilai akurasi dari *validation set* dan *testing set* yang ada. Tujuan dari perbandingan ini adalah memastikan tidak adanya *overfitting* dari nilai akurasi milik *testing set*. Karena jika nilai akurasi dari keduanya jauh berbeda, maka jelas terjadi *overfitting*. Untuk mendapatkan kedua nilai akurasi, dapat dilihat seperti pada Gambar 20.

```
In [14]: # Mengevaluasi kinerja model pada validation set
         y_val_pred = best_model.predict(X_val)
         validation_accuracy = accuracy_score(y_val_encoded, y_val_pred)
         print("Akurasi model pada validation set: ", validation_accuracy)

         # Mengevaluasi kinerja model pada testing set
         y_test_pred = best_model.predict(X_test)
         testing_accuracy = accuracy_score(y_test_encoded, y_test_pred)
         print("Akurasi model pada testing set: ", testing_accuracy)

Akurasi model pada validation set:  0.7838709677419354
Akurasi model pada testing set:  0.7718373493975904
```

Gambar 20. Nilai Akurasi *Validation Set* dan *Testing Set*

N. Feature Importance

Proses *feature importance* adalah metode untuk mengukur seberapa signifikan atau berpengaruhnya setiap fitur terhadap prediksi atau output model. Ini membantu dalam pemahaman tentang fitur mana yang memiliki kontribusi yang paling besar dalam mempengaruhi hasil prediksi model. *Feature importance* dapat membantu mengidentifikasi fitur-fitur yang paling informatif atau memberikan kontribusi yang signifikan terhadap prediksi model. Ini memungkinkan penulis untuk melakukan seleksi fitur dan mengurangi dimensi data dengan mempertimbangkan hanya fitur-fitur yang penting.

Feature importance membantu penulis memahami dan menjelaskan model yang telah dibangun. Dengan mengetahui fitur-fitur yang paling berpengaruh, penulis dapat menginterpretasikan alasan di balik prediksi model dan mendapatkan wawasan tentang faktor-faktor yang mempengaruhi output. Jika terdapat perbedaan antara ekspektasi dan hasil dari *feature importance*, itu dapat menunjukkan masalah atau kesalahan dalam data atau model. Fitur yang diharapkan memiliki pengaruh tinggi tetapi memiliki *feature importance* yang rendah mungkin perlu ditinjau lebih lanjut. *Feature importance* dapat memberikan wawasan tentang fitur-fitur yang perlu ditingkatkan atau diubah dalam proses *feature engineering*. Misalnya, jika fitur tertentu memiliki pengaruh rendah, penulis dapat mencoba untuk menggabungkannya dengan fitur lain atau mengubahnya menjadi representasi yang lebih informatif. Dengan pemahaman tentang feature importance, penulis dapat membuat model yang lebih baik, mengurangi dimensi data, dan mendapatkan wawasan yang lebih baik tentang faktor-faktor yang mempengaruhi prediksi model.

Untuk menginisialisasi proses *feature importance*, pertama penulis mendeklarasikan sebuah variabel yang berisikan pemanggilan fungsi feature_importances_ yang sudah dimiliki oleh model XGB seperti pada Gambar 21.

```
# Mengakses feature importance
feature_importances = xgb_model.feature_importances_
```

Gambar 21. *Feature Importances*

Setelah melakukan inisialisasi, selanjutnya penulis mengeliminasi feature yang terdapat pada variabel feature_importances sehingga hanya tersisa 5 fitur paling berpengaruh pada dataset. Proses eliminasi dapat dilihat melalui Gambar 22.

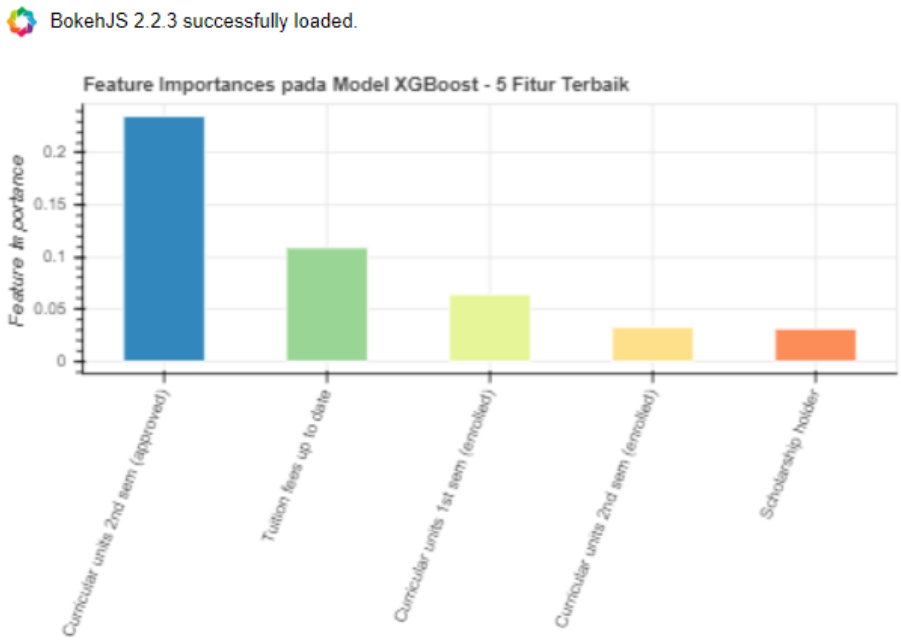
```
feature_names = X.columns
# Mengurutkan fitur berdasarkan kepentingan
sorted_indices = np.argsort(feature_importances)[::-1]
sorted_feature_importances = feature_importances[sorted_indices]
# Mengambil 5 fitur teratas
top_features_indices = sorted_indices[:5]
top_feature_importances = sorted_feature_importances[:5]
top_feature_names = list(feature_names[top_features_indices]) # Mengubah menjadi list
```

Gambar 22. 5 Fitur Teratas

Langkah terakhir dari proses *features importance* adalah proses visualisasi data berbentuk *bar chart*. Visualisasi dilakukan untuk mempermudah dalam melihat fitur mana yang paling berpengaruh didalam dataset. Kode untuk melakukan visualisasi dapat dilihat melalui Gambar 23 dan hasil visualisasi dapat dilihat seperti pada Gambar 24.

```
# Mengatur output untuk ditampilkan di Jupyter Notebook
output_notebook()

# Membuat data source dari fitur-fitur teratas
source = ColumnDataSource(data=dict(features=top_feature_names, importance=top_feature_importances))

# Membuat plot
p = figure(x_range=top_feature_names, plot_height=400, plot_width=600,
           title='Feature Importances pada Model XGBoost - 5 Fitur Terbaik',
           toolbar_location=None, tools="")

# Menggambar batang vertikal
p.vbar(x='features', top='importance', width=0.5, source=source,
       line_color='white', fill_color=factor_cmap('features', palette=Spectral6, factors=top_feature_names))

# Mengatur label pada sumbu x
p.xaxis.major_label_orientation = 1.2

# Mengatur label pada sumbu y
p.yaxis.axis_label = 'Feature Importance'

# Menampilkan plot
show(p)
```

Gambar 23. *Feature Importance Visualization*



Gambar 24. *Feature Importance Bar Chart*

O. Pelatihan, Cross Validation, & Pengukuran Akurasi

Untuk melakukan rangkaian proses pelatihan hingga pengukuran akurasi pada strategi silang. Pertama-tama peneliti perlu untuk melakukan proses *mapping*. Proses ini dilakukan untuk mengubah variabel y (target) menjadi nilai numerik, karena secara default pada dataset *student retention*, target yang tersedia berbentuk kategorial. Untuk melakukan proses *mapping* peneliti membentuk sebuah struktur data "dict" atau *dictionary*. *Dictionary* yang dibentuk memiliki *key* berupa data y (target) yang berbentuk kategorial. Sedangkan untuk *value* dari *dictionary* peneliti isi dengan nilai numerik terurut. Setelah itu dilakukan iterasi untuk mengubah nilai y (target) secara keseluruhan menjadi nilai numerik. Proses mapping yang dilakukan peneliti dapat dilihat seperti pada Gambar 25.

```
class_mapping = {'Dropout': 0, 'Enrolled': 1, 'Graduate': 2}
y = [class_mapping[cls] for cls in y]
```

Gambar 25. *Mapping* Variabel Target

Setelah melakukan proses mapping, selanjutnya peneliti mendeklarasikan obyek KFold. Pada obyek KFold yang telah dideklarasikan juga ditentukan nilai jumlah fold dengan menggunakan parameter "n_splits" dan parameter "shuffle. Berbeda dengan strategi holdout sebelumnya, saat menggunakan strategi Cross Validation peneliti tidak melakukan pelatihan model mesin secara terpisah menggunakan fungsi "fit()", karena Cross Validation sudah secara otomatis melakukan pelatihan sebanyak jumlah fold. Sehingga peneliti cukup mendeklarasikan obyek cross_val_score kedalam sebuah variabel, lalu mengisi setiap parameter yang dibutuhkan pada obyek cross_val_score.

```
scores = cross_val_score(xgb_model,
                         X,
                         y,
                         cv=kfold,
                         scoring='accuracy')
```

Gambar 26. Obyek Cross_Val_Score

Seperti yang dapat dilihat melalui Gambar 26, terdapat 5 parameter yang digunakan pada obyek cross_val_score diantaranya sebagai berikut:

- Parameter Pertama (xgb_model): Parameter ini merujuk pada variabel model yang akan digunakan untuk divalidasi menggunakan strategi *cross validation*.
- Parameter Kedua (X): Parameter ini merujuk pada variabel data *features* (X).
- Parameter Ketiga (y): Parameter ini merujuk pada variabel data *target* (y).
- Parameter Keempat (cv= kfold): Parameter ini merujuk pada jenis *cross validation* yang ingin digunakan untuk strategi *cross validation*.
- Parameter Kelima (Scoring = 'Accuracy'): Parameter ini merujuk pada jenis pengukuran yang ingin dilakukan, seperti *accuracy*, *recall*, dan *F1-Score* yang biasa digunakan untuk klasifikasi.

Setelah pemanggilan obyek cross_val_score, selanjutnya dilakukan pengukuran akurasi setiap fold dan perhitungan rata-rata akurasi setiap fold. Proses ini dilakukan guna mengetahui stabilitas akurasi dari setiap fold. Bilamana terdapat nilai akurasi yang jauh berbeda dari fold lainnya, kemungkinan besar terjadi overfitting ataupun underfitting. Pada proses ini, untuk mengetahui akurasi dari setiap fold peneliti melakukan iterasi menggunakan teknik *for-each looping*. Hasil pengukurang akurasi didapatkan nilai 0.768 untuk rata-rata dari setiap fold yang berjumlah total 10 fold.

```
In [21]: # Menampilkan skor akurasi dari setiap fold
         for i, score in enumerate(scores):
             print(f"Fold {i+1} Accuracy: {score}")

         # Menampilkan rata-rata skor akurasi dari cross-validation
         print("Mean Accuracy: ", scores.mean())
```

```
Fold 1 Accuracy: 0.7471783295711061
Fold 2 Accuracy: 0.7381489841986456
Fold 3 Accuracy: 0.7900677200902935
Fold 4 Accuracy: 0.781038374717833
Fold 5 Accuracy: 0.7714932126696833
Fold 6 Accuracy: 0.7149321266968326
Fold 7 Accuracy: 0.7692307692307693
Fold 8 Accuracy: 0.7669683257918553
Fold 9 Accuracy: 0.8031674208144797
Fold 10 Accuracy: 0.7579185520361991
Mean Accuracy:  0.7640143815817697
```

Gambar 27. K-Fold Cross_Val_Score

Terakhir, untuk menerapkan strategi *Stratified* K-Fold *Cross Validation*, langkah yang perlu ditempuh cukup sederhana dan tidak berbeda jauh dari langkah-langkah K-Fold *Cross Validation*.

```
In [22]: # SK-Fold
         skfold=StratifiedKFold(n_splits=10)
         scores=cross_val_score(xgb_model,X,y,cv=skfold)
         print(np.mean(scores))

0.7680918868676139
```

Gambar 28. SK-Fold Cross_Val_Score

Sama seperti pada saat peneliti menerapkan strategi K-Fold *Cross Validation*. Pada langkah awal penerapan strategi ini, peneliti mendeklarasikan obyek StratifiedKFold terlebih dahulu dan mengisi parameter n_splits seperti pada Gambar 28. Parameter ini digunakan untuk menentukan jumlah fold dari strategi SKfold. Pada penerapan SKfold, peneliti juga tidak perlu melakukan proses *training* dan *testing* model secara terpisah, sama seperti yang dilakukan pada proses penerapan Kfold. Parameter yang digunakan untuk obyek cross_val_scores juga seluruhnya sama dengan yang digunakan pada saat penerapan KFold. Dan hasil akhir akurasi dari penerapan strategi SKFold adalah 0.768.

V. KESIMPULAN

Dari hasil implementasi eksplorasi pembelajaran mesin menggunakan model Extreme Gradient Boosting. Peneliti menarik kesimpulan sebagai berikut:

- Pada studi kasus data keberhasilan studi mahasiswa, faktor prediktif yang berkaitan dengan keberlanjutan studi mahasiswa adalah fitur-fitur berikut:
  - Curricular Units 2nd Sem (Approved): Fitur ini memiliki nilai 0.235, nilai 0.235 merupakan nilai tertinggi yang dimiliki oleh fitur dari dataset keberhasilan studi mahasiswa. Fitur ini memiliki pengaruh paling besar terhadap dataset.
  - Tuition Fees Up To Date: Fitur ini memiliki nilai 0.109, nilai 0.109 merupakan nilai tertinggi kedua yang dimiliki oleh fitur dari dataset keberhasilan studi mahasiswa. Fitur ini memiliki pengaruh terbesar kedua terhadap dataset.
  - Curricular Units 1st Sem (Enrolled): Fitur ini memiliki nilai 0.064, nilai 0.064 merupakan nilai tertinggi ketigas yang dimiliki oleh fitur dari dataset keberhasilan studi mahasiswa. Fitur ini memiliki pengaruh terbesar kedua terhadap dataset.
- Untuk memastikan performa yang baik dari suatu model pembelajaran mesin, nilai akurasi yang didapatkan dari strategi pengembangan holdout, KFold cross validation, dan Stratified KFold cross validation dapat menjadi acuan. Ketiga strategi tersebut digunakan untuk memastikan bahwa nilai akurasi yang muncul merupakan nilai akurasi yang valid.
- Model Extreme Gradient Boosting terbukti efektif dalam memprediksi studi kasus dataset keberhasilan studi mahasiswa. Efektifitas dari model Extreme Gradient Boosting terlihat dari hasil pengukuran nilai akurasi yang tercipta dari proses prediksi dan validasi. Nilai akhir yang didapatkan untuk akurasi dari model Extreme Gradient Boosting adalah 76.8%. Nilai tersebut juga cukup dapat dipercaya, karena setelah pengujian dari ketiga strategi yang berbeda, nilai akurasi yang muncul tetap sama. Selain telah melalui penerapan ketiga strategi yang berbeda, nilai akurasi yang muncul juga telah divalidasi dengan menggunakan teknik Hyper-parameter Tuning. Dari hasil proses Hyper-Parameter dapat disimpulkan bahwa nilai tersebut paling optimal dengan menggunakan parameter yang tentunya paling optimal ketimbang parameter lainnya. Nilai tersebut juga telah melalui proses validasi antara nilai akurasi testing set dan validation set sehingga tidak mungkin terjadi overfitting pada model XGB yang dikembangkan. Dari nilai akurasi yang didapat bisa peneliti simpulkan bahwa dari dataset yang berjumlah 4.424 data, model Extreme Gradient Boosting berhasil mengklasifikasikan 3.397 data dengan benar. Dan sebanyak 1.027 data diklasifikasikan dengan tidak tepat.

UCAPAN TERIMA KASIH

Puji syukur peneliti panjatkan kehadirat Tuhan Yang Maha Esa. Karena atas berkat dan karunia-Nya lah, Tugas Akhir ini dapat berjalan dengan lancar dan laporan ini dapat terselesaikan. Penelitian laporan ini bertujuan sebagai persyaratan kelulusan studi program S1 saya di program studi Sistem Informasi Universitas Kristen Maranatha.

Topik yang dipilih adalah Eksplorasi Pembelajaran Mesin Menggunakan Model Extreme Gradient Boosting Untuk Dataset Keberhasilan Studi Mahasiswa. Topik ini diberikan oleh bapak Setia Budi, S.Kom, M.Comp., Ph. D. selaku dosen program studi Sistem Informasi. Peneliti mengucapkan terima kasih kepada bapak Setia Budi karena berkat beliau, bimbingan berlangsung dengan intens dan penjadwalan yang dilakukan sangat terstruktur.

Terima kasih juga peneliti ucapkan kepada Ibu Adelia, S.Kom., M.T. selaku koordinator KP/TA yang sudah dengan sabar membimbing para mahasiswa terkait prosedur KP mulai dari awal pengajuan sampai dengan kelulusan KP. Peneliti harap kerja praktek kompentensi yang akan datang bisa memiliki topik yang lebih banyak dan lebih variatif untuk dipilih oleh mahasiswa.

Akhir kata sebagai penutup, peneliti ingin memberikan pernyataan penutup. Sebuah bangunan tidak akan mempengaruhi perkembangan anda dengan begitu signifikan, tetapi cara anda memaknai proses dan mengeksplorasi diri yang akan menjadikan diri anda tumbuh secara signifikan.

DAFTAR PUSTAKA

[1] N. A. Alshahrani, "Machine Learning in Information Systems Research: A Systematic Literature Review," *Journal of Management Information Systems*, vol. 36, no. 1, pp. 11-54, 2019.

[2] P. S. S. I. U. K. Maranatha, Buku Panduan Program Studi S-1 Sistem Informasi Tahun Akademik 2021-2022, Bandung: Fakultas Teknologi Informasi Universitas Kristen Maranatha, 2021.

[3] E. Alpaydin, Introduction to machine learning, Cambridge: MIT Press, 2010.

[4] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of machine learning., Cambridge: MITPress, 2021.

[5] A. R. Rajkumar and P. Thirunavukarasu, "Machine learning applications in big data analytics: a review," *International Journal of Advanced Intelligence Paradigms*, vol. 1, no. 15, pp. 1-20, 2021.

[6] R. Singh and R. K. Gupta, "Pattern recognition techniques for machine learning: a review," *Soft Computing*, vol. 13, no. 24, pp. 9651-9672, 2020.

[7] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge: MIT press, 2016.

[8] S. S. Ali and S. S. Wahid, "An Overview of Decision Tree Algorithm with Ensemble Learning," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 23, pp. 4354-4364, 2020.

[9] S. S. Ali and S. S. Wahid, "Decision Tree Algorithm: A Comprehensive Survey," *Journal of Applied Sciences*, vol. 20, no. 3, pp. 120-127, 2020.

[10] S. S. Ali and S. S. Wahid, "Overfitting Problem in Decision Tree Algorithm: A Review," *International Journal of Computer Applications*, vol. 179, no. 31, pp. 7-11, 2020.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.

[12] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system.," in *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

[14] M. Fauzi, "aktor-Faktor yang Mempengaruhi Retensi Mahasiswa Perguruan Tinggi," *Jurnal Kajian Pendidikan dan Pengajaran*, vol. 1, no. 4, pp. 1-8, 2021.

[15] J. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 3, no. 9, pp. 90-95, 2007.

[16] M. Kumar and D. Toshniwal, "Extreme Gradient Boosting (XGBoost)," in *Encyclopedia of Machine Learning and Data Mining*, Springer, 2021.

[17] R. Kumar and P. Aggarwal, "A Survey on Decision Tree Algorithm with Pruning Techniques," *Journal of Emerging Technologies and Innovative Research*, vol. 7, no. 5, pp. 362-367, 2020.

[18] A. Ma'ruf and A. S. Siregar, "Implementasi Faktor-faktor yang Mempengaruhi Retensi Mahasiswa: Studi Kasus Universitas Pembangunan Jaya.," *Jurnal Pendidikan Bisnis dan Manajemen*, vol. 6, no. 1, pp. 1-10, 2020.

[19] W. McKinney, "Data structures for statistical computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010.

[20] F. V. G. Pedregosa, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, no. 12, pp. 2825-2830, 2011.

[21] V. Realinho, J. Machado, L. Baptista and M. V. Martins, "Predict students' dropout and academic success (1.0)," 13 December 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5777340. [Accessed 26 March 2023].

[22] V. Realinho, J. Machado, L. Baptista and M. V. Martins, "Zenodo," 13 December 2021. [Online]. Available: https://zenodo.org/record/5777340#.ZE_wu_xBzIV. [Accessed 02 May 2023].

[23] S. R. Shubham and G. Singh, "Decision Tree Algorithm for Classification in Machine Learning: An Overview," *International Journal of Advanced Research in Computer Science*, vol. 11, no. 4, pp. 23-28, 2020.

[24] C. Tianqi, "Understanding XGBoost," arXiv preprint, 2018.

[25] Z. H. Zhou, Ensemble methods: foundations and algorithms, Shanghai: CRC press, 2012.