Perbandingan Model pada *Dataset* Kaggle: *Spaceship Titanic* dan *Predict Failures Keep it Dry*

1st I Wayan Ivan Zenatmaja
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
iwayanivanz@student.ub.ac.id

2nd Hugo Alfedo Putra
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
hugoalfedoputra@student.ub.ac.id

3rd Gratia Yudika Morado Silalahi
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
gratiasilalahi_@student.ub.ac.id

4th M. Hasan Fadhlillah
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
hasanfadhlillah@student.ub.ac.id

5th M. Husain Fadhlillah
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
muhammadhusainf@student.ub.ac.id

6th Hilda Tri Fatikasari Hilal
Teknik Informatika
Universitas Brawijaya
Malang, Indonesia
hildatfh@student.ub.ac.id

***Abstract*—Proyek ini merinci alur kerja pemrosesan data dan pemodelan machine learning untuk klasifikasi dataset Kaggle, dengan fokus pada dua studi kasus. Setelah menghapus kolom yang tidak relevan, analisis distribusi kelas dan korelasi antar fitur dilakukan. Fitur kategorikal dienkode menggunakan One Hot Encoding (OHE) dan Target Encoding (TE), pada algoritma model klasifikasi pada dua dataset menggunakan KNeighborsClassifier, SVM, Gaussian Navie Bayes, Decision Tree Classifier, CatBoost Classifier, Gradient Boosting Classifier, XGBoost Classifier, XGBoost Random Forest Classifier, Random Forest Classifier, Balanced Random Forest Classifier, Histogram Gradient Boosting Classifier, RUSBoost Classifier, Logistic Classifier, Easy Ensemble Classifier, Votting Classifier, XGBoost Regressor,CatBoost Regressor,Stacking Classifier sementara fitur numerik diimputasi dengan KNN Imputer dan dinormalisasi menggunakan z-score. Data latih dibagi menjadi set tidak seimbang dan seimbang, dengan balancing dilakukan menggunakan SMOTE. Berbagai model machine learning, termasuk CatBoost, XGBoost, dan Random Forest, dievaluasi menggunakan hyperparameter tuning dan validasi silang. Akurasi model terbaik diuji dengan root mean square error dan tiga model terbaik disubmit ke Kaggle. Dalam studi kasus Spaceship Titanic, fitur diekstraksi dari kolom "Cabin", dan fitur kategorikal dienkode ulang. Proses pemodelan menggunakan meta-ensembling meningkatkan prediksi akhir, menghasilkan solusi klasifikasi yang optimal.**

***Index Terms*—Machine Learning, Klasifikasi, Data Imputation, One Hot Encoding, Target Encoding, SMOTE, Hyperparameter Tuning, Meta-Ensembling, Kaggle, Spaceship Titanic**

I. Pendahuluan

Pada era digital ini, data merupakan salah satu aset penting yang dapat diolah dan dianalisis untuk menghasilkan informasi dan prediksi yang bermanfaat. Machine learning, sebagai salah satu cabang kecerdasan buatan, memainkan peran krusial dalam mengolah data dan menghasilkan prediksi yang akurat. Kaggle, sebuah platform online yang terkenal dengan kompetisi dan kolaborasi di bidang data science, menyediakan berbagai studi kasus yang menantang bagi para praktisi untuk menerapkan dan menguji kemampuan machine learning mereka.

Penelitian ini berfokus pada perbandingan berbagai algoritma machine learning dalam memprediksi hasil pada dua studi kasus Kaggle yang populer, yaitu "Spaceship Titanic" dan "Predict Failures: Keep It Dry". Pada studi kasus Spaceship Titanic, tujuannya adalah untuk memprediksi kelangsungan hidup penumpang kapal Titanic berdasarkan data historis mereka. Sebaliknya, pada Predict Failures: Keep It Dry, tantangannya adalah untuk memprediksi kegagalan perangkat keras berdasarkan data sensor.

Memperoleh prediksi yang paling akurat untuk kedua kasus ini sangatlah penting. Untuk Spaceship Titanic, prediksi yang tepat dapat memberikan wawasan berharga tentang faktor-faktor yang mempengaruhi kelangsungan hidup dalam situasi darurat. Sementara itu, prediksi kegagalan perangkat keras yang akurat pada Predict Failures: Keep It Dry dapat membantu dalam pengembangan sistem pemeliharaan prediktif yang lebih efektif, mengurangi downtime, dan meningkatkan efisiensi operasional.

Dalam penelitian ini, dilakukan analisis mendalam terhadap dataset yang diperoleh dari Kaggle. Dataset terdiri dari train set dan test set, di mana kolom 'id' dan 'product_code' dihapus karena dianggap tidak relevan dalam proses training dan prediksi. Distribusi kelas pada train set menunjukkan ketimpangan yang signifikan, dengan persentase kelas True sebesar 21,3% dan kelas False sebesar 78,7%. Untuk mengatasi ini, berbagai teknik preprocessing seperti One Hot Encoding (OHE) dan Target Encoding (TE) digunakan untuk fitur non-numerik, serta KNN Imputer dan normalisasi z-score untuk fitur numerik.

Analisis korelasi antar fitur menggunakan Pearson Correlation dan Mutual Information menunjukkan bahwa fitur-fitur tersebut tidak berkorelasi atau berkorelasi lemah dan bersifat independen. Histogram plotting juga mengindikasikan distribusi normal pada tiap fitur dengan distribusi kelas yang konsisten. Dataset kemudian dipecah menjadi train set dan validation set untuk evaluasi model menggunakan stratified

train-test split. Selanjutnya, balancing dilakukan pada train set menggunakan SMOTE, namun tidak pada validation set.

Model-model machine learning yang digunakan dalam penelitian ini termasuk KNN, SVM, Decision Tree, dan Gaussian Naive Bayes. Hyperparameter tuning dilakukan menggunakan successive halving grid search dengan 10-fold cross validation. Model dievaluasi berdasarkan akurasi menggunakan root mean square error (RMSE). Tiga model terbaik kemudian di-submit ke Kaggle untuk evaluasi lebih lanjut. Dengan pendekatan ini, diharapkan dapat diperoleh pemahaman yang lebih baik mengenai performa masing-masing algoritma dalam memprediksi hasil pada dua studi kasus Kaggle yang berbeda, serta strategi yang efektif dalam menangani data yang tidak seimbang dan meningkatkan akurasi prediksi.

II. Tinjauan Pustaka

A. Dataset

Kaggle memungkinkan pengguna menemukan kumpulan data yang ingin mereka gunakan dalam membangun model AI, mempublikasikan kumpulan data, bekerja dengan ilmuwan data dan insinyur pembelajaran mesin lainnya, serta mengikuti kompetisi untuk memecahkan tantangan ilmu data. Kaggle memulai usahanya pada tahun 2010 dengan menawarkan kompetisi pembelajaran mesin dan ilmu data serta menawarkan data publik dan platform bisnis berbasis cloud untuk ilmu data dan pendidikan AI [29].

1) Spaceship Titanic:

> ”Bantulah (sic) kru penyelamat dan mengambil penumpang yang hilang, Andalah yang melakukannya ditantang untuk memprediksi penumpang mana yang diangkut anomali menggunakan catatan yang diperoleh dari sistem komputer pesawat ruang angkasa yang rusak [27].”

2) Predict Failures Keep it Dry:

> ”Tabular Playground Series edisi Agustus 2022 adalah kesempatan untuk membantu perusahaan fiksi Keep It Dry meningkatkan produk utamanya, Super Soaker. Produk ini digunakan di pabrik untuk menyerap tumpahan dan kebocoran. Perusahaan baru saja menyelesaikan studi pengujian besar-besaran untuk prototipe produk yang berbeda. Bisakah Anda menggunakan data ini untuk membuat model yang memprediksi kegagalan produk? [28]”

B. Encoding

Aneka encoding tidak dilakukan secara kombinasi. Tetapi dilakukan secara sendiri-sendiri. Didukung oleh tesis Andrei-Iustin Udilă [1] bahwasanya dengan melakukan kombinasi justru akan menurunkan tingkat performansi. Penyebabnya dipetakan menjadi tiga hal: (1) inkonsistensi, skema pengkodean yang tidak konsisten dapat menimbulkan gangguan atau kebingungan pada model terutama untuk model linier dan SVM; (2) kehilangan informasi, pengambilan metode pengkodean yang berbeda dan mewakili informasi kategoris secara berbeda. Penerapan encoder yang berbeda pada kolom yang berbeda memungkinkan kehilangan informasi terjadi secara tidak sengaja ataupun interaksi antara fitur mungkin saja terdistorsi; dan (3) korelasi dan interaksi, di mana penggabungan pengkodean metode lintas kolom dapat menjadikan tantangan bagi model untuk mengidentifikasi korelasi atau interaksi antar fitur yang dikodekan. Beberapa model, seperti decision tree, akan kesulitan mengenali pola secara efektif jika skema pengkodean bervariasi antar fitur.

1) One Hot Encoding: Pada Predict Failures Keep it Dry menggunakan One Hot Encoding. One-Hot Encoding melibatkan pembuatan fitur biner baru untuk setiap kategori dalam variabel kategori. Teknik ini cocok untuk variabel kategori dengan jumlah yang sedikit kategori unik dan banyak digunakan dalam model linier, pohon keputusan, dan jaringan saraf. Kelemahan utama dari satu pengkodean panas adalah dapat menghasilkan fitur ruang berdimensi tinggi, yang dapat berdampak negatif terhadap kinerja beberapa model, dan meningkatkan biaya komputasi. Matriks fitur yang dihasilkan jarang, dimana setiap kolom mewakili kategori dalam variabel kategori, dan setiap baris mewakili sampel dalam kumpulan data [1].

2) Target Encoding: Pada Spaceship Titanic dan Predict Failures Keep it Dry menggunakan Target Encoding.Target Encoding, juga dikenal sebagai pengkodean rata-rata, melibatkan mengganti setiap kategori dengan mean (atau agregasi lainnya) dari variabel target untuk kategori tersebut. Variabel target adalah variabel yang kami coba prediksi dalam model pembelajaran mesin kami. Keuntungan utama dari pengkodean target adalah ini dapat memberikan representasi data yang lebih informatif daripada pengkodean one-hot atau pengkodean ordinal. Dengan mengkodekan kategori berdasarkan hubungannya dengan variabel target, kita berpotensi menangkap lebih banyak struktur yang mendasarinya data. Namun, pengkodean target juga dapat menimbulkan bias jika hubungan antara variabel kategori dan variabel target palsu atau jika terjadi overfitting. Itu penting perlu diperhatikan bahwa saat menggunakan pengkodean target, penting untuk memisahkannya data ke dalam set pelatihan dan validasi sebelum pengkodean dan menghitung sarana hanya menggunakan set pelatihan. Jika tidak, ada risiko kebocoran data, yang dapat menyebabkan overfitting dan kinerja generalisasi yang buruk [1].

3) Label Encoding: Pada Spaceship Titanic menggunakan Label Encoding. Label Encoding adalah metode yang mengubah data kategorikal—yang direpresentasikan sebagai label teks—menjadi format numerik. Dengan kata lain, hal ini melibatkan pemberian nilai numerik yang unik untuk setiap kategori atau label dalam variabel kategorikal. Label Encoding adalah metode yang mengubah data kategorikal yang direpresentasikan sebagai label teks menjadi format numerik. Dengan kata lain, hal ini melibatkan pemberian nilai numerik yang unik untuk setiap kategori atau label dalam variabel kategorikal. Misalnya label ”Warna,” sebagai variabel kategorikal dengan label ”Merah,” ”Hijau,” dan ”Biru.” Label-label ini dapat diberikan nilai numerik berturut-turut 0, 1, dan 2 dalam label encoding [2].

4) Ordinal Encoding: Pada Spaceship Titanic menggunakan Ordinal Encoding. Ordinal Encoding memberikan nilai integer unik untuk masing-masingnya kategori dalam variabel kategori berdasarkan urutan atau peringkatnya kategori, dengan nilai bilangan bulat lebih rendah yang ditetapkan ke kategori kucing yang dianggap ”lebih rendah” atau ”lebih awal” dalam urutannya. Setelah pengkodean, kami menormalkan data, seperti beberapa algoritma sensitif terhadap perbedaan skala antar fitur. Salah satu masalah potensial dengan pengkodean ordinal adalah bilangan bulat nilai dapat memperkenalkan urutan numerik ke kategori itu mungkin tidak mencerminkan hubungan sebenarnya antara kategori kucing. Hal ini dapat menjadi masalah khususnya bagi kategorikal variabel yang tidak memiliki urutan bawaan atau yang tidak memiliki urutan terdefinisi dengan baik [1].

C. Pearson Correlation

Pearson Correlation mengukur kekuatan dan arah hubungan linear antara dua variabel. Digunakan dalam penelitian ilmiah untuk mengukur hubungan seperti skor tes dan IQ, atau tinggi badan dan berat badan. Nilai koefisien korelasi menunjukkan kekuatan dan arah hubungan, dengan nilai 0 menunjukkan tidak ada hubungan, positif menunjukkan hubungan positif, dan negatif menunjukkan hubungan negatif. Kelebihannya adalah mudah dihitung dan diinterpretasikan, tahan outlier, dan dapat mengukur hubungan kategorikal dan numerik. Kekurangannya adalah hanya mengukur hubungan linear dan memerlukan asumsi normalitas distribusi [24].

Dimatematiskan:

$$\rho(X,\, Y) = \frac{\text{cov}(X,\, Y)}{\rho_x \cdot \rho_y} \tag{1}$$

D. Mutual Information

Pada Spaceship Titanic dan Predict Failures Keep it Dry menggunakan Mutual Information Score. Mutual Information (MI) adalah ukuran statistik yang mengukur ketergantungan antara dua variabel. MI digunakan dalam Spaceship Titanic dan Predict Failures Keep it Dry untuk memilih fitur yang paling relevan dengan target variabel dan mengukur ketergantungan antara variabel prediktor dan target variabel. MI memiliki kelebihan seperti non-parametrik, dapat mengukur hubungan kategorikal dan numerik, serta non-linear. Kekurangannya adalah tidak tahan outlier, sulit diinterpretasikan, dan tidak mempertimbangkan variabel lain [25].

Dimatematiskan:

$$MI(X,\, Y) = \sum p(x,\, y) \cdot \log(\frac{p(x,\, y)}{p(x) \cdot p(y)}) \tag{2}$$

E. Standard Scaler

Pada Spaceship Titanic menggunakan standar scaller. Standar scaler adalah teknik pra-pemrosesan data yang mentransformasi variabel menjadi skala yang sama dengan distribusi normal. Digunakan dalam Spaceship Titanic untuk mentransformasi variabel numerik sebelum digunakan dalam model prediksi. Kelebihannya adalah meningkatkan performa model prediksi, stabilitas numerik, dan interpretasi fitur. Kekurangannya adalah dapat kehilangan informasi distribusi data asli dan tidak selalu diperlukan untuk semua model prediksi [26].

F. Normalisasi Z-Score

Normalisasi ialah operasi pada data mentah yang menskalakan atau mentransformasikannya sehingga setiap fitur memiliki kontribusi yang seragam. Hal ini mengatasi dua masalah utama data yang menghambat proses pembelajaran model machine learning, yaitu adanya fitur dominan dan outlier (nilai pencilan). Banyak metode telah diusulkan untuk menormalkan data dalam rentang tertentu berdasarkan ukuran statistik dari data mentah (tidak dinormalisasi). Misalkan terdapat suatu himpunan data dengan f fitur dan N sampel:

$$D = x_{i,n}, y_n \mid i \in f \text{ dan } n \in N$$

di mana x mewakili data yang akan dipelajari oleh model pembelajaran dan y adalah label kelas yang sesuai. Salah satunya adalah normalisasi menggunakan Z-score. Pada normalisasi Z-score, Ukuran rata-rata (mean) dan deviasi standar digunakan untuk menskalakan data sehingga fitur yang dihasilkan memiliki mean nol dan varians sebesar satu. Setiap data instan, $x_{i,n}$ ditransformasikan menjadi x'_i dengan:

$$x'_i = \frac{x_i - \mu_i}{\sigma_i} \tag{3}$$

G. Imputasi KNN

Algoritma K-Nearest Neighbor telah banyak digunakan pada implementasi imputasi data hilang atau kosong terutama dalam *dataset* dengan data hilang pada lebih dari satu variabel. Algoritma KNN menggunakan observasi yang mirip atau serupa dengan observasi yang memiliki nilai hilang. Sebuah observasi yang memuat satu atau lebih nilai hilang yang akan diimputasi disebut sebagai observasi target. Berdasarkan pendekatan ini, ukuran jarak dihitung antara observasi target dengan tiap-tiap observasi lainnya. Jika k merepresentasikan jumlah observasi yang mirip atau serupa dengan observasi target, maka kemudian dipilih k observasi yang memiliki jarak minimum dari observasi target. Algoritma imputasi KNN dilakukan dengan (1) menentukan nilai k, yaitu jumlah observasi terdekat yang diinginkan; (2) menghitung jarak euclidian antara observasi target dengan observasi yang tidak memuat nilai kosong, dengan:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{s} (x_j - y_j)^2} \tag{4}$$

di mana $\mathbf{x}$ ialah vektor observasi target dengan variabel sebanyak s variabel, $\mathbf{x}$ ialah $[x_1, x_2, \ldots, x_s]^T\mathbf{y}$ ialah vektor observasi yang tidak memuat nilai *missing* dengan variabel sebanyak s variabel, $\mathbf{y}$ialah$[y_1, y_2, \ldots, y_s]^T d(\mathbf{x}, \mathbf{y})$ ialah jarak antara $\mathbf{x}$ dan $\mathbf{y}x_j$ ialah nilai variabel ke-j pada $\mathbf{x}y_j$ ialah nilai variabel ke-j pada $\mathbf{y}$, dan j ialah $1, 2, \ldots, s$; (3) mencari k observasi yang memiliki nilai $d(\mathbf{x}, \mathbf{y})$ minimum; dan (4) melakukan imputasi *missing data* dengan menggunakan prosedur *weighted mean imputation*, dengan:

$$\hat{x}_j = \frac{1}{W} \sum k = 1^K w_k y_{kj} \quad (5)$$

di mana $\hat{x}_j$ ialah nilai imputasi, y_{kj} ialah nilai variabel ke-j pada observasi ke-k, $k = 1, 2, \ldots, K$, dan $W = \sum_{k=1}^{K} w_k$.

H. Algoritma Klasifikasi

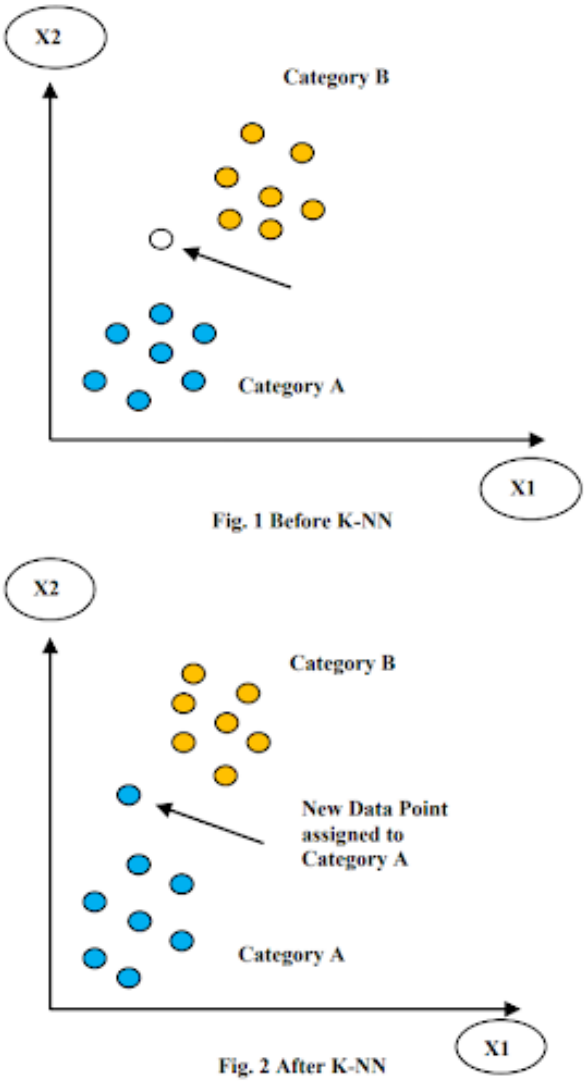
Berdasarkan cara pelatihan, model klasifikasi dapat dibagi menjadi dua macam, yaitu eager learner dan lazy learner. Algoritma eager learner didesain untuk melakukan pembacaan/pelatihan/pembelajaran pada data latih agar dapat memetakan dengan benar setiap vektor masukkan ke label kelas keluarannya, sehingga di akhir proses pelatihan, model sudah dapat memetakan semua vektor data uji ke label kelas keluarannya dengan benar. Algoritma klasifikasi yang masuk kategori ini, di antaranya, adalah Artificial Neural Network (ANN), Support Vektor Machine (SVM), Decision Tree, Naïve Bayes, dan sebagainya. Sementara model yang masuk dalam kategori lazy learner hanya sedikit melakukan pelatihan (atau tidak sama sekali), hanya menyimpan sebagian atau seluruh data latih kemudian menggunakannya dalam proses prediksi. Algoritma klasifikasi yang masuk kategori ini, di antaranya, K-Nearest Neighbor, Fuzzy K-Nearest Neighbor, Regresi Linear, dan sebagainya [3].

Berdasarkan metode pelatihannya, model klasifikasi dapat dibagi menjadi dua jenis yaitu, eager learner dan lazy learner. Eager learner dirancang untuk membaca/melatih/mempelajari data pelatihan sehingga dapat memetakan setiap vektor masukan dengan benar ke label kelas keluarannya, sehingga pada akhir pelatihan, model dapat memetakan semua pengujian vektor data untuk memberi label dengan benar pada kelas keluaran. Algoritma klasifikasi dalam kategori ini meliputi Jaringan Syaraf Tiruan (ANN), Support Vector Machine (SVM), Decision Tree, NaiveBayes, dan lain-lain. Sementara itu, model yang termasuk dalam kategori lazy learner, kelas melatih sedikit (atau tidak sama sekali),hanya menyimpan sebagian atau seluruh data pelatihan, lalu menggunakannya dalam proses prediksi. Algoritma klasifikasi dalam kategori ini adalah K-Nearest neighbour, Fuzzy K-Nearest neighbour, Linear regresi, dan lain-lain [3].

1) KNN: K-Nearest Neighbor adalah pembelajaran mesin paling sederhana. Algoritma KNN sebagian besar digunakan dalam memecahkan masalah klasifikasi. Itu Kebutuhan algoritma klasifikasi KNN dapat dipahami dari contoh ini. Misalkan kita harus diberikan dua kategori, yaitu, kategori $\mathbf{A}$ dan Kategori $\mathbf{B}$, dan memiliki titik data baru $\mathbf{X_1}$ dan ingin melihat titik data mana yang terletak di titik mana kategori-kategori ini. Untuk mengatasi masalah ini diperlukan algoritma KNN. Kategori atau kelas kumpulan data tertentu dapat dengan mudah diselesaikan dengan bantuan algoritma KNN [4].

Langkah kerja algoritma KNN berdasarkan jarak Euclidean dapat dirincikan sebagai berikut:

1) Pertama-tama, pilih nomor k dari tetangga.



Gambar 1. Hasil pengkategorian dengan algoritma KNN berdasarkan [4]

2) Jarak Euclidean ialah d harus dihitung untuk k sejumlah tetangga dengan:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (6)$$

3) Ambil k tetangga terdekat per perhitungan Jarak Euclidean.
4) Di antara k tetangga ini, hitung jumlahnya titik data di setiap kategori.
5) Poin data baru akan ditetapkan ke kategori yang jumlah tetangganya maksimal.
6) Dengan cara ini, model klasifikasi KNN akan melakukannya bersiap.

Hasil dari implementasi KNN ini digambarkan pada Gambar 1.

2) SVM: SVM pada dasarnya berfungsi sebagai metode untuk menghasilkan garis pemisah (hyperplane) dari himpunan

data ke dalam dua kelas secara linier. Hyperplane di sini ialah istilah yang dibuat secara umum untuk semua dimensi. Sebagai contoh, untuk himpunan data berdimensi satu, maka hyperplane dapat berwujud sebagai sebuah titik, jika himpunan dalam bentuk dua dimensi maka hyperplane berbentuk garis lurus dan jika berdimensi tiga maka berbentuk bidang datar. Pencarian hyperplane paling optimum pada klasifikasi linier dapat dilakukan dengan persamaan. Metode SVM sering digunakan sebagai media klasifikasi yang bersifat otomatis seperti klasifikasi teks, citra, analisis medik ataupun prediksi. Klasifikasi ialah pengumpulan objek yang memiliki ciri khas yang sama ke dalam beberapa kelas. SVM dapat digunakan sebagai parameter yang digunakan dalam pembagian klasifikasi data. Awalnya SVM dibuat hanya digunakan untuk mengklasifikasi dua buah kelas (binary classifier). Penelitian terus berlanjut hingga menjadikan SVM bersifat multi kelas (multi classifier) dengan kemampuan mengklasifikasi lebih dari dua kelas. Untuk klasifikasi data dalam k kelas maka diharuskan membuat model SVM Biner. Sebagai contoh untuk membuat klasifikasi 4 kelas maka, harus membangun enam buah SVM Biner. SVM Biner pertama dilatih dengan data latih dari kelas pertama dan kelas kedua untuk mengklasifikasikan data ke dalam $\mathbf{C1}$ atau $\mathbf{C2}$. SVM Biner kedua dilatih dengan data latih dari kelas $\mathbf{C2}$ atau $\mathbf{C3}$, dan seterusnya. Setiap kelas harus dibandingkan dengan tiga kelas lainya. Cara voting dilakukan untuk menghasilkan kelas keputusan. Kelas yang paling sering menang adalah kelas keputusan [5].

1) SGD (Stochastic Gradien Descent)
   Gradient Descent merupakan salah satu algoritma optimasi iteratif untuk menemukan titik yang meminimumkan suatu fungsi yang dapat diturunkan. Metode ini bekerja dengan memulai dari sebuah tebakan awal dan secara iteratif tebakan ini dapat diperbaiki berdasarkan suatu aturan yang melibatkan gradient/turunan pertama dari fungsi yang ingin diminimumkan. Persamaan di bawah ini digunakan dalam kasus yang secara khusus mengatur langkah-langkah yang diambil untuk menurunkan fungsi yang ingin diminimumkan.

$$\omega_i + 1 = \omega_i - \xi \nabla_{\omega_i} L(\omega_i) \tag{7}$$

   di mana $\omega_i + 1$ ialah parameter model untuk prediksi, ω_i ialah parameter model pada iterasi sebelumnya, ξ ialah tingkat pembelajaran, dan L ialah fungsi *loss cost*.
   Tahap pembelajaran dalam Gradient Descent standar mensyaratkan bahwa turunannya dihitung untuk semua sampel dalam dataset pelatihan di setiap iterasi. Konsekuensinya adalah ketika data latih terlalu besar dapat terjadi komputasi secara intensif. SGD merupakan salah satu varian dari Gradient Descent yang dilatih sebagai sampel pelatihan tunggal yang dipilih secara acak pada suatu waktu. SGD bersifat lebih scalable dan cepat untuk dilatih dengan tanpa adanya batasan waktu dalam pelaksanaan dengan ukuran dataset pelatihan. Tujuan penggunaan SGD yaitu pelaksanaan klasifikasi yang lebih cepat dalam pembelajarannya. Fungsi hinge loss yang digunakan untuk melatih classifier dapat dilihat dengan persamaan:

$$L(x_j, y_j) = \max(0,\ 1 - y_j \cdot (\omega x_j + b)) \tag{8}$$

2) Linier SVC (Support Vector Classification)

3) Gaussian Naive Bayes: Pembelajaran dapat disederhanakan dengan classifier Naïve Bayes dengan menganggap bahwa fitur-fitur diberikan secara independen. Meskipun demikian, asumsi independensinya adalah demikian miskin pada umumnya. Praktisnya, dengan yang lebih canggih pengklasifikasi, Naive Bayes sering kali bersaing secara efektif. fitur vektor menjelaskannya pada contoh yang diberikan, pengklasifikasi Bayesian mengalokasikan kelas yang paling mungkin, yaitu:

$$P(C) = \prod_{i}^{n} = 1p(X_i|C) \tag{9}$$

di mana C adalah pengklasifikasi, dan $X = (X_1, X_2, \ldots, X_n)$ adalah vektor fitur. Dalam praktiknya, pengklasifikasi yang dihasilkan dikenal sebagai Naïve Bayes sangat sukses meskipun ada asumsi yang tidak realistis, dengan terus bersaing dengan teknik yang jauh lebih canggih. Model Naïve Bayes ini adalah versi sederhana dari probabilitas Bayesian. Pada operasi Naïve Bayes, pengklasifikasi dilakukan dengan asumsi independensi yang kuat. Ini berarti bahwa probabilitas satu atribut tidak berpengaruh terhadap probabilitas atribut lainnya [8].

4) Decision Tree Classifier: Pohon keputusan adalah struktur pohon seperti diagram alur di mana simpul internal mewakili fitur (atau atribut), cabang mewakili aturan keputusan, dan setiap simpul daun mewakili hasilnya. Node paling atas dalam pohon keputusan dikenal sebagai node akar. Ia belajar mempartisi berdasarkan nilai atribut. Ini mempartisi pohon secara rekursif yang disebut partisi rekursif. Struktur seperti diagram alur ini membantu Anda dalam pengambilan keputusan. Visualisasinya seperti diagram alur yang dengan mudah meniru pemikiran tingkat manusia. Itulah sebabnya pohon keputusan mudah dipahami dan diinterpretasikan. Pohon keputusan adalah jenis algoritma ML kotak putih. Ini berbagi logika pengambilan keputusan internal, yang tidak tersedia dalam jenis algoritma kotak hitam seperti jaringan saraf. Waktu pelatihannya lebih cepat dibandingkan dengan algoritma jaringan saraf. Kompleksitas waktu dari pohon keputusan merupakan fungsi dari jumlah catatan dan atribut dalam data yang diberikan. Pohon keputusan adalah metode bebas distribusi atau non-parametrik yang tidak bergantung pada asumsi distribusi probabilitas. Pohon keputusan dapat menangani data berdimensi tinggi dengan akurasi yang baik sehingga bisa digunakan dengan baik dalam algoritma klasifikasi [19].

5) CatBoost Classifier: CatBoost adalah metode pembelajaran mesin terawasi yang digunakan oleh alat Latih Menggunakan AutoML dan menggunakan pohon keputusan untuk klasifikasi dan regresi. Seperti namanya, CatBoost memiliki dua fitur utama, ia bekerja dengan data kategorikal (Cat) dan menggunakan peningkatan gradien (Boost) [20].

6) Gradient Boosting Classifier: Gradient Boosting adalah algoritma gradien fungsional yang berulang kali memilih fungsi yang mengarah ke arah hipotesis lemah atau gradien negatif sehingga dapat meminimalkan kerugian fungsi. Pengklasifikasi peningkatan gradien menggabungkan beberapa model pembelajaran yang lemah untuk menghasilkan model prediksi yang kuat [21].

7) Histogram Gradient Boosting Classifier: Histogram Gradient Boosting adalah metode boosting yang menggunakan histogram untuk mempercepat proses fitting. Metode ini mengelompokkan data fitur kontinu ke dalam interval-interval, yang membantu dalam mempercepat perhitungan.

Rumus

Gradient Boosting menggunakan rumus umum untuk boosting, tetapi dengan optimasi histogram:

1) Inisialisasi model $f_0(x)$ dengan konstanta.
2) Untuk $m = 1$ hingga M:
   a) Hitung pseudo-residuals $i_m = \boldsymbol{g}_m - f_{m-1}(x)$.
   b) Fit base learner $h_m(x)$ untuk residuals.
   c) Update model: $f_m(x) = f_{m-1}(x) + \eta h_m(x)$ [10].

8) XGBoost Classifier: XGBoost (eXtreme Gradient Boosting) adalah algoritma peningkatan gradien yang kuat dan banyak digunakan untuk memecahkan berbagai jenis masalah pembelajaran mesin. Ini adalah implementasi peningkatan gradien yang dirancang khusus agar efisien dan terukur, menjadikannya pilihan populer untuk bekerja dengan kumpulan data besar. Secara matematis, XGBoost merupakan metode pembelajaran ansambel yang menggabungkan prediksi beberapa model lemah untuk menghasilkan prediksi yang kuat. Model lemah di XGBoost adalah pohon keputusan, yang dilatih menggunakan peningkatan gradien. Artinya pada setiap iterasi, algoritma mencocokkan pohon keputusan dengan sisa dari iterasi sebelumnya.

Pohon keputusan di XGBoost dilatih menggunakan fungsi tujuan berikut:

$$\min_{\theta}(\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)) \tag{10}$$

dimana l adalah fungsi kerugian, y_i adalah label sebenarnya dari contoh pelatihan ke-i, i adalah label prediksi dari contoh pelatihan ke-i, f_k adalah pohon keputusan ke-k, dan Ω adalah istilah regularisasi yang memberikan penalti pada kompleksitas pohon. Fungsi tujuan ini dioptimalkan menggunakan penurunan gradien.

9) XGBoost Random Forest Classifier: Random Forest dan XGBoost merupakan algoritma pembelajaran mesin canggih yang banyak digunakan untuk tugas klasifikasi dan regresi. Meskipun keduanya memiliki kesamaan dalam pendekatan berbasis ansambel, keduanya berbeda dalam teknik algoritmik, penanganan overfitting, performa, fleksibilitas, dan penyesuaian parameter. Bila digabungkan akan tercipta classifier dengan performa yang lebih baik [22].

10) Random Forest Classifier: Random Forest adalah metode ensemble learning yang menggunakan sejumlah pohon keputusan (decision trees) untuk melakukan klasifikasi. Keputusan final diambil berdasarkan voting mayoritas dari prediksi semua pohon. Algoritmanya adalah: (1) Ambil sampel bootstrap dari dataset asli, (2) Untuk setiap sampel bootstrap, (3) buat pohon keputusan tanpa memangkasnya. Untuk setiap node di pohon, pilih subset fitur secara acak dan pilih fitur terbaik untuk split hingga ulangi sampai pohon selesai. Gabungkan hasil dari semua pohon (voting mayoritas untuk klasifikasi) [9].

11) Balanced Random Forest Classifier: Balanced Random Forest adalah variasi dari Random Forest yang menangani ketidakseimbangan kelas dengan menyeimbangkan kelas dalam setiap sampel bootstrap yang dibuat. Algoritma mengikuti langkah-langkah Random Forest, tetapi setiap sampel bootstrap dibuat dengan menyeimbangkan jumlah sampel dari setiap kelas [11].

12) RUSBoost Classifier: RUSBoost (Random Under-Sampling Boost) adalah metode boosting yang menggabungkan teknik under-sampling dengan algoritma AdaBoost untuk mengatasi ketidakseimbangan kelas.

Algoritma Rusboost mengikuti langkah-langkah AdaBoost dengan tambahan under-sampling [12]:

1.) Inisialisasi bobot sampel.
2.) Untuk $t = 1$ hingga T:
2.a.) Lakukan under-sampling pada dataset untuk menyeimbangkan kelas.
2.b.) Train base classifier pada dataset yang di-under-sample.
2.c.) Hitung error dan update bobot sampel.

13) Logistic Classifier: Logistic Classifier atau Logistic Regression adalah model statistik yang digunakan untuk memprediksi probabilitas kejadian kelas biner dengan menggunakan fungsi logistik.

Rumus untuk memprediksi probabilitas kelas biner dengan logistic classifier adalah:

$$P(Y = 1|X) = \frac{1}{1 + e^{-\beta^T X}} \tag{11}$$

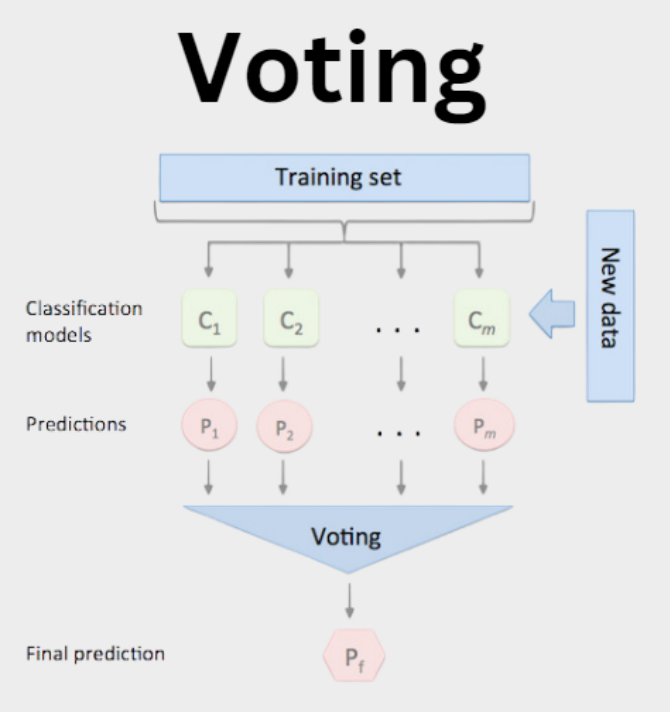
di mana $P(Y = 1|X)$ adalah probabilitas kejadian kelas 1 dengan kondisi X, β adalah vektor koefisien regresi yang diestimasi, dan X adalah vektor fitur [13].

14) Easy Ensemble Classifier: Easy Ensemble adalah metode ensemble yang menggabungkan beberapa under-sampled ensembles untuk mengatasi masalah ketidakseimbangan kelas. Setiap subset data seimbang dilatih menggunakan ADA Boost [14].

Algoritma Easy Ensemble mengikuti langkah-langkah ADA Boost untuk setiap subset under-sampled:

1) Buat beberapa subset under-sampled dari dataset.
2) Train AdaBoost pada masing-masing subset.
3) Kombinasikan hasil dari semua ensemble dengan voting.

15) Voting Classifier: Voting Classifier adalah metode ensemble yang menggabungkan prediksi dari beberapa model



Gambar 2. Cara kerja Voting Classifier

berbeda. Keputusan akhir diambil berdasarkan voting mayoritas (untuk klasifikasi) atau rata- rata (untuk regresi) [15].

Rumus: Jika Y1, Y2,..., Yn adalah prediksi dari n classifier: Untuk klasifikasi: $y = \text{mode}(\hat{y}1, \hat{y}2, ..., \hat{y}n)$ Untuk regresi: $\hat{y} = \frac{\sum_{i=1}^{n} \hat{y}_i}{n}$

16) Stacking Classifier: Stacking adalah teknik ensemble di mana beberapa model (base learners) dilatih dan hasil prediksi mereka digunakan sebagai input untuk model meta-learner yang memberikan prediksi akhir [18].

Jika $M_1, M_2, \ldots, M_n$ adalah base models dan M_{meta} adalah meta-learner:

1) Train base models: M_i pada dataset.
2) Kumpulkan prediksi dari base models sebagai input ke meta-learner.
3) Train meta-learner pada output dari base models.

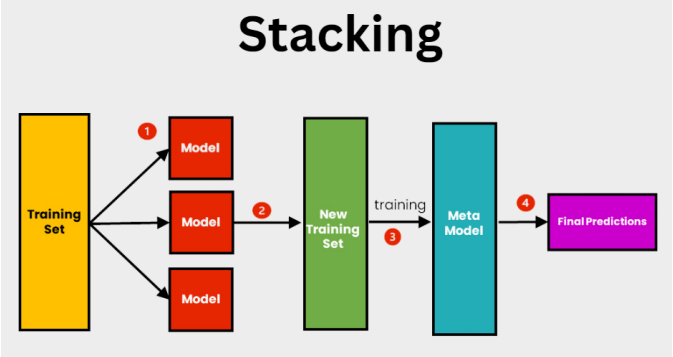
17) XGBoost Regressor: XGBoost (Extreme Gradient Boosting) adalah implementasi dari gradient boosting yang dioptimalkan untuk efisiensi dan performa. Ini sering digunakan untuk tugas regresi dan klasifikasi [16].

Gradient Boosting dengan regularisasi:

$$f_t(x) = f_{t-1}(x) + \eta \sum_{i=1}^{n} g_i h(x_i)$$

di mana g_i adalah gradient dari loss function.

18) CatBoost Regressor: CatBoost (Categorical Boosting) adalah algoritma boosting yang dirancang untuk menangani fitur kategori dengan lebih efisien dan mengurangi overfitting. CatBoost menggunakan kombinasi gradient boosting dan target-based statistics untuk fitur kategori [17].



Gambar 3. Cara kerja Stacking Classifier

19) Balancing: Pada Spaceship Titanic tidak memerlukan balancing karena distribusi targetnya sudah seimbang. Sedangkan, pada Predict Failures Keep it Dry perlu dilakukan balancing karena kolom kelas memiliki ketimpangan antara kelas True dan False dengan persentase masing-masing 21,3% dan 78,7%. Balancing dibagi menjadi 3 macam yaitu, SMOTE, RUS, dan SMOOTEEN.

1) SMOTE (Synthetic Minority Oversampling Technique)
   Teknik Pengambilan Sampel Minoritas Sintetis bekerja untuk kumpulan data ketidakseimbangan kelas. SMOTE mungkin yang paling terkenal untuk menghasilkan contoh Sintetis. SMOTE adalah pendekatan pemeriksaan berlebihan dalam membuat kelas minoritas diuji secara berlebihan .model sintetis alih-alih memeriksa secara berlebihan pengganti.
2) RUS (Random Under-Sampling)
   Undersampling adalah siklus yang berupaya mengurangi jumlah individu kelas mayoritas di perlengkapan latihan. Pengambilan sampel berlebihan secara acak adalah hal yang terkenal untuk metode pengambilan sampel ulang. Kelas mayoritas dokumen dihilangkan secara acak dalam pelatihan ditetapkan sampai minoritas, dan rasio kelas mayoritas berada pada tingkat yang diinginkan. Pada beberapa sampel secara acak diambil dari kelas mayoritas X_{max}, data sampel set $\mathbf{DB}$ dihasilkan dengan menyalin sampel yang dipilih dan kemudian menghapusnya ke X_{max}. Kumpulan data mentahnya adalah dihapus untuk membentuk kelas minoritas baru $X_{new} = X_{max} - \mathbf{DB}$.
3) SMOTEEN (Synthetic Minority Oversampling with Edited Nearest Neighbour)
   Teknik Pengambilan Sampel Minoritas Sintetis dengan Mengedit metode Tetangga Terdekat. ENN adalah metode pembersihan; itu menghilangkan kebisingan contoh dari kelas mayoritas dan minoritas. Menerapkan SMOTE membuat node baru dalam minoritas kelas, dan mencoba memecahkan masalah ketidakseimbangan.

20) Hyperparameter Tuning: Penyetelan hyperparameter melibatkan pencarian melalui ruang hyperparameter berdimensi tinggi yang kompleks untuk menemukan konfigurasi optimal untuk suatu model. Tantangannya tidak hanya terletak pada biaya komputasi tetapi juga pada trade-off antara kom-

pleksitas model, generalisasi, dan overfitting. Untuk memitigasi tantangan ini, algoritma yang efisien sangat penting untuk mengoptimalkan alokasi sumber daya tanpa mengorbankan performa model. Di sinilah halving Berturut-turut mengungguli pendekatan penyetelan hyperparameter tradisional seperti Pencarian Grid, Pencarian Acak, dan Optimasi Bayesian [6] [23].

Salah satu caranya adalah dengan successive halving grid search di mana merupakan sebuah algoritma pengoptimalan berbasis bandit yang dirancang untuk memitigasi berbagai kompleksitas penyetelan hyperparameter. Algoritma bandit termasuk dalam kelas algoritma optimasi yang dirancang untuk pengambilan keputusan di lingkungan dengan informasi yang tidak lengkap atau tidak pasti. Nama "bandit" berasal dari masalah Multi-Armed Bandit, yang merupakan analogi sederhana untuk keputusan kompleks yang dirancang untuk diambil oleh algoritma. Bayangkan seorang penjudi menghadapi deretan mesin slot (atau "bandit bertangan satu"). Setiap mesin memberikan kemungkinan hadiah yang berbeda dan tidak diketahui. Penjudi ingin memaksimalkan total hadiahnya melalui serangkaian penarikan tuas. Tantangannya adalah menemukan strategi optimal untuk mengalokasikan penarikan antar mesin guna memaksimalkan imbalan kumulatif yang diharapkan. Successive Halving mengadopsi kerangka Multi-Armed Bandit untuk memecahkan masalah optimasi hyperparameter. Setiap lengan bandit berhubungan dengan konfigurasi hyperparameter yang berbeda, dan menarik lengan sama dengan mengevaluasi konfigurasi dengan anggaran komputasi tertentu. "Reward" adalah metrik performa yang Anda optimalkan, seperti akurasi atau skor F1 [6] [23].

Berdasarkan [7] K-FCV dapat digunakan untuk melakukan hyperparameter tuning atau dalam kata lain, memilih nilai k optimal. Langkah-langkah K-FCV dirincikan sebagai berikut:

1.) Acaklah dataset sehingga menghilangkan kemungkinan adanya bias (sehingga membutuhkan underfitting atau overfitting) dalam distribusi data atau suatu urutan arbiter dari sumber dataset.
2.) Pecahlah dataset menjadi fold dengan banyak n dan besar m. Nilai m ini biasanya 10 berdasarkan hasil empiris dengan error rate dengan bias dan variansi yang kecil; nilai m yang semakin besar akan menghasilkan bias yang semakin kecil sedangkan m yang sama dengan besar dataset merupakan bentuk penerapan LOO-CV. Selain itu, nilai m=10 atau lebih besar cocok diterapkan untuk dataset yang kecil. Nilai n menyesuaikan dengan besar m, misalnya dataset dengan besar 500 dan m=10 akan menghasilkan $n = 500 \div 10 = 50$ banyak fold.
3.) Secara iteratif dan independen sebanyak m kali:
3.a.) Tetapkan fold pertama ($m = 1$; iterasi selanjutnya menggunakan fold kedua atau $m = 2, 3, \dots$) sebagai test data dan sisanya (sebanyak $n-1$) sebagai training data.
3.b.) Latihlah model dengan training data.
3.c.) Validasikan hasil, misalnya dengan Classification Accuracy Rate (CAR) atau Root Mean Squared Error (RMSE).
3.d.) Simpanlah nilai hasil validasi.
4.) Lakukan tahap 3 secara iteratif dan independen sebanyak banyak fold dikali dengan m di mana tiap iterasi menggunakan nilai $k = 2m - 1$.
5.) Evaluasi kinerja pada setiap tahap iteratif, misalnya dengan mengambil nilai rata-rata CAR tertinggi dari fold sebanyak m atau dengan mengambil rata-rata RMSE tiap fold.

III. Metodologi

Metodologi yang diuraikan ini mencakup seluruh alur kerja dari persiapan data hingga prediksi akhir. Pendekatan ini dirancang untuk memastikan bahwa data diproses secara menyeluruh dan model yang digunakan mampu menghasilkan prediksi yang akurat dan andal, sehingga memberikan solusi optimal untuk masalah klasifikasi pada dataset Spaceship Titanic.

A. Spaceship Titanic

Penelitian ini bertujuan untuk mengembangkan model prediksi yang akurat untuk menentukan apakah penumpang Spaceship Titanic akan 'Terangkut' atau 'Tidak Terangkut'. Proses ini mencakup persiapan data, pembuatan fitur, prapemrosesan data, pemodelan, dan evaluasi. Metodologi ini dirancang untuk memastikan bahwa data diproses secara cermat dan model yang digunakan dapat memberikan solusi yang optimal untuk masalah klasifikasi ini.

1) Persiapan Data: Proyek ini dimulai dengan tahap persiapan data yang sangat penting untuk memastikan analisis dan pemodelan yang akurat. Data latih dibaca dari file CSV yang disediakan, dan label target, yaitu kolom "Transported", dipisahkan dari fitur-fitur lainnya dalam dataset. Untuk mengurangi kompleksitas dan mempercepat proses analisis, kolom "Name" dihapus karena dianggap tidak relevan untuk tujuan prediksi. Kolom "Cabin", yang berisi informasi tentang tempat tinggal penumpang di kapal, diekstraksi menjadi tiga kolom baru: "deck", "num", dan "side". Ekstraksi ini membantu dalam mendapatkan informasi lebih spesifik tentang lokasi penumpang, yang kemudian dapat membantu dalam model prediksi. Setelah informasi ini diekstraksi, kolom "Cabin" dihapus untuk menghindari redundansi.

2) Pembuatan Fitur Baru: Selanjutnya, fitur baru diciptakan untuk memperkaya dataset dan memberikan lebih banyak informasi kepada model prediksi. Total pengeluaran penumpang dihitung dengan menjumlahkan nilai dari kolom-kolom "RoomService", "FoodCourt", "ShoppingMall", "Spa", dan "VRDeck". Fitur ini memberikan gambaran tentang seberapa banyak penumpang menghabiskan uang mereka selama perjalanan. Selain itu, rasio pengeluaran terhadap usia juga dihitung dengan membagi total pengeluaran dengan usia penumpang. Fitur ini dapat menunjukkan pola pengeluaran yang berbeda berdasarkan kelompok usia yang berbeda. Informasi grup diekstraksi dari "PassengerId" untuk mengidentifikasi kelompok penumpang yang bepergian bersama, yang mungkin memiliki pola perjalanan yang serupa. Setelah

informasi grup ini diekstraksi, kolom ”PassengerId” dihapus karena tidak lagi diperlukan.

3) Analisis Distribusi Target: Setelah melihat distribusi target (Transported) Gambar 5, ditemukan bahwa proporsi 'Tidak Terangkut' sebesar 50.4% dan 'Terangkut' sebesar 49.6%. Dengan distribusi target yang cukup seimbang ini, kita dapat mengasumsikan bahwa model kita tidak akan bias terhadap satu kelas tertentu dan akan memiliki performa yang baik pada kedua kelas tersebut. Hal ini juga berarti bahwa kita tidak perlu menggunakan teknik penyeimbangan kelas seperti oversampling atau undersampling.

4) Pra-Pemrosesan Data:

1) Encoding Kolom Kategorikal
   Pada tahap pra-pemrosesan data, kolom-kolom kategorikal diidentifikasi dan kemudian dienkode menggunakan metode yang sesuai untuk mengubah nilai-nilai kategorikal menjadi nilai numerik yang dapat diproses oleh model machine learning. Berdasarkan temuan dalam jurnal, penggunaan kombinasi encoding tidak efektif dan dapat menyebabkan penurunan kinerja [Sumber : A.-I. Udilă, ”Encoding Methods for Categorical Data: A Comparative Analysis for Linear Models, Decision Trees, and Support Vector Machines,” thesis, Delft Univ. Technol., Delft, Netherlands, 2023.]. Oleh karena itu, hanya satu metode encoding yang dipilih untuk seluruh dataset untuk menjaga konsistensi. Kolom-kolom kategorikal diencode menggunakan Ordinal Encoder, yang mengubah kategori menjadi angka ordinal, sehingga model dapat memahami dan memproses data dengan lebih baik. Langkah ini dipilih karena menghindari ketidakkonsistenan yang mungkin timbul dari kombinasi metode encoding. Selain menggunakan Ordinal Encoder, Label Encoder dan Target Encoder juga diuji untuk perbandingan hasil akurasi.
2) Imputasi Nilai yang Hilang
   Setelah proses encoding selesai, langkah berikutnya adalah mengatasi nilai-nilai yang hilang dalam dataset. Untuk kolom numerik, digunakan Iterative Imputer yang mempertimbangkan hubungan antar fitur untuk mengisi nilai yang hilang secara lebih akurat. Sementara itu, Simple Imputer dengan strategi ”most frequent” digunakan untuk kolom kategorikal, mengisi nilai yang hilang dengan nilai yang paling sering muncul dalam kolom tersebut. Metode-metode imputasi ini membantu dalam memastikan data siap untuk proses pelatihan model machine learning dengan meminimalisir dampak dari data yang hilang.
3) Penskalaan Fitur Numerik
   Setelah nilai yang hilang diimputasi, fitur-fitur numerik diskalakan menggunakan StandardScaler untuk memastikan bahwa semua fitur numerik berada pada skala yang sama. Penskalaan ini penting untuk algoritma yang sensitif terhadap skala fitur, seperti logistic regression dan SVM.
4) Seleksi Fitur Berdasarkan Mutual Information
   Selanjutnya, skor Mutual Information dihitung untuk setiap fitur guna menilai pentingnya fitur tersebut terhadap label target. Fitur-fitur yang memiliki skor Mutual Information rendah, yaitu kurang dari 0.01. Selain Itu, fitur-fitur numerik yang double atau ganda akibat hasil penskalaan juga dihapus. Hal ini bertujuan untuk mengurangi dimensi data dan menghindari overfitting, serta untuk memastikan bahwa hanya fitur-fitur yang paling informatif yang digunakan dalam pemodelan. Hasil dapat dilihat pada Gambar 6 dan 7.

5) Pemodelan dan Evaluasi:

1) Pembagian Dataset
   Tahap berikutnya adalah pemodelan dan evaluasi. Dataset dibagi menjadi data latih dan data validasi dengan proporsi 80% untuk pelatihan dan 20% untuk validasi.
2) Pemodelan Awal
   Berbagai model machine learning diterapkan, termasuk RandomForestClassifier, LogisticRegression, GradientBoostingClassifier, XGBClassifier, CatBoostClassifier, KNeighborsClassifier, SVC, DecisionTreeClassifier, dan GaussianNB. Setiap model dilatih pada data latih dan dievaluasi pada data validasi untuk mengukur kinerja awalnya.
3) Hyperparameter Tuning
   Hyperparameter tuning dilakukan menggunakan GridSearchCV untuk mencari kombinasi hyperparameter terbaik yang memberikan kinerja optimal. Proses tuning ini melibatkan pencarian parameter yang paling sesuai untuk setiap model melalui cross-validation untuk memastikan bahwa model tidak hanya cocok pada data latih tetapi juga mampu melakukan generalisasi pada data yang tidak terlihat sebelumnya.
4) Meta-Ensemble Learning
   Setelah masing-masing model dievaluasi dan dioptimalkan, model terbaik dipilih untuk digunakan dalam tahap prediksi akhir. Untuk meningkatkan kinerja prediksi lebih lanjut, dua model meta-ensembling diterapkan, yaitu Stacking Classifier dan Voting Classifier. Stacking Classifier menggabungkan beberapa model dasar dengan meta model sebagai estimator akhir, menggabungkan kekuatan beberapa algoritma untuk menghasilkan prediksi yang lebih akurat. Voting Classifier menggunakan pendekatan hard voting yang menggabungkan prediksi dari beberapa model dengan memilih hasil mayoritas.
5) Evaluasi dengan Cross-Validation
   Model dilatih pada data latih dan dievaluasi menggunakan cross-validation dengan 3 fold untuk memastikan robustitas dan kemampuan generalisasi model.

6) Prediksi dan Submisi:

1) Persiapan Data Uji
   Setelah model terbaik dipilih melalui evaluasi yang ketat, tahap prediksi dan submisi dilakukan. Data uji dibaca dari file CSV dan diproses dengan cara yang sama seperti data latih, termasuk ekstraksi informasi dari kolom ”Cabin”, pembuatan fitur total pengeluaran dan rasio pengeluaran terhadap usia, serta imputasi dan penskalaan data.

2) Prediksi dan Penyimpanan Hasil
Prediksi dilakukan menggunakan model pilihan terbaik (misalnya Stacking Classifier), dan hasil prediksi disimpan dalam format CSV. File submisi ini mencakup "PassengerId" dan prediksi apakah penumpang tersebut "Transported" atau tidak, siap untuk diserahkan untuk evaluasi akhir.

B. Predict Failures Keep it Dry

Gambar alur kerja dapat dilihat pada Gambar 8 di mana dirincikan di bawah ini.

1) Preprocessing Data: Preprocessing data merupakan langkah paling awal yang diperlukan dalam proses pengolahan data. Preprocessing yang dilakukan adalah pengubahan nama atribut menjadi lebih sederhana dan penghapusan data, yakni kolom 'id' dan 'product_code' karena bukan merupakan data yang diperlukan dalam proses pengolahan.

2) Exploratory Data Analysis (EDA): Merupakan langkah selanjutnya di mana kita menganalisis eksploratori pada data untuk mendapatkan wawasan hal dari data. Untuk EDA sendiri menggunakan 3 tahapan yaitu Pearson Correlation, Mutual Information, dan Plotting Feature.

1.) Pearson Correlation
Pearson Correlation digunakan untuk mengukur hubungan atau korelasi antar tiap fitur pada train set. Pada data train akan ditunjukkan kolom yang memiliki hubungan dengan level hubungannya juga. Korelasi pearson berada di antara -1 hingga 1 dimana jikalau ia bernilai positif maka hubungan itu menunjukkkan searah dan bersifat bertambah, dan sebaliknya jikalau bernilai negatif maka menunjukkan hubungan searah akan dan bersifat berkurang. Untuk tingkat keeratan dapat dideskripsikan, seperti pada Tabel I. Dimatematiskan:

$$r_{xy} = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{\sqrt{\left(n\sum x_i^2 - (\sum x_i)^2\right)\left(n\sum y_i^2 - (\sum y_i)^2\right)}} \tag{12}$$

di mana rxy ialah hubungan antara x dan y, n ialah jumlah sample, x_i ialah nilai x ke-i, dan y_i ialah nilai y ke-i.

Tabel I: Keeratan Korelasi Pearson berdasarkan interval koefisiennya.

Interval Koefisien	**Keeratan Korelasi**
0,00 - 0,20	Sangat Lemah
0,21 - 0,40	Lemah
0,41 - 0,70	Moderate atau Sedang
0,71 - 0,90	Kuat
0,91 - 0,99	Sangat Kuat
1	Korelasi Sempurna

2.) Mutual Information
Untuk Mutual Information sendiri berfungsi untuk mengukur dependensi atau ketergantungan antar 2 variabel. Nilai Mutual Information berkisar antara 0 hingga nilai positif. Nilai 0 menunjukkan ketiadaan ketergantungan, sedangkan nilai yang lebih tinggi menunjukkan ketergantungan yang lebih kuat antara variabel.

3.) Plotting Feature
Plotting Feature melibatkan pembuatan visualisasi (seperti histogram) untuk setiap fitur dalam dataset. Ini membantu dalam pemahaman distribusi data pada setiap fitur, seperti apakah data mengikuti pola tertentu (misalnya, bell curve), ada nilai yang hilang, atau ada outlier. Pada data terdapat plotting features yang dilanjutkan dengan penampilan data plot yang ada.

3) Encoding: Encoding sendiri merupakan proses pengubahan data kategorikal menjadi representasi numerik karena machine learning sendiri bekerja dengan data numerik. Pada encoding sendiri, kode kami menggunakan dua jenis encoding: One Hot Encoding dan Target Encoding.

1.) One-Hot Encoding dilakukan dengan cara merubah nilai setiap kelas menjadi nilai biner, dengan indeks kelas yang bersangkutan diberi nilai 1 dan yang lain diberi nilai 0.
2.) Target Encoding dilakukan dengan cara merubah nilai target dengan merata-ratakan terlebih dahulu dalam kategori kelas yang bersangkutan. Perbedaan utama Target dengan one hot adalah pada representasi nilai numeriknya.

4) Imputing: Imputing sendiri (pada Gambar 10) adalah proses mengganti atau mengisi nilai yang kosong atau hilang pada data dengan nilai yang dapat diterima (sesuai tafsiran). Proses Imputasi Data Train set dilakukan dengan menggunakan KNN Imputer di mana algoritma K-Nearest Neigbors ini memperkirakan nilai hilang namun berdasarkan tetangga terdekat yang berada dalam ruang fitur. Sebenarnya terdapat beberapa imputer lain yang digunakan seperti mean imputing, median imputing, Most Frequent Imputing, dan Normalise Imputing, namun yang berpengaruh hanyalah KNN Imputing.

5) Hyperparameter Tuning: Sebelum dilakukan tuning, alur kerja akan dibagi menjadi 2 bagian yakni, training tanpa balancing dan training pada set yang balanced. Train set yang balanced dibagi menjadi dua bagian, yaitu train set dan validation set dengan train test split di mana perbandingan kolom kelas masih dipertahankan (stratify). Train set hasil pembagian itu dilakukan hyperparameter tuning dengan successive halving grid search dengan 10-fold cross validation. Hasil hyperparameter dilakukan pada model KNN, SVM dengan SGD (Stochastic Gradient Descent), SVM Linear SVC (Support Vector Classification) dengan transformasi Nystroem untuk aproksimasi kernel pada dataset yang besar, Decision Tree, dan Gaussian Naive Bayes. Parameter yang sama digunakan pula sebagai parameter training model pada train set yang balanced nantinya. Model-model yang digunakan antara lain CatBoost Classifier, Stacking Classifier, SVM SVC, XGBoost Regressor, CatBoost Regressor, Ensemble XGBoost Regression dan CatBoost Regression, Voting Classifier, Random Forest Classifier, XGBoost Classifier, Histogram Gradient Boosting Classifier, Easy Ensemble Classifier, Logistic Regression, Gaussian Naive Bayes, Decision Tree, KNN, XGBoost Random Forest Classifier, SVM SGD, Gradient Boosting Classifier, RUSBoost

Classifier, dan Balanced Random Forest Classifier.

6) Balancing: Tahap selanjutnya ialah pelatihan train set yang balanced dengan model-model yang telah disebutkan sebelumnya. Train set dibagi pula menjadi train set dan validation set di mana balancing hanya dilakukan pada train set; validation set tidak dilakukan balancing. Balancing dilakukan dengan SMOTE. Pelatihan model-model ini tidak disertakan dengan hyperparameter tuning karena keterbatasan waktu. Hasil pelatihan model divalidasi akurasinya dengan root mean square error. Kemudian model yang terbaik dari hasil training akan dikumpulkan dengan train set.

7) Prediksi dan Submisi: Tiga model terbaik hasil pelatihan dengan train set OHE dan TE (di mana keduanya diimputasikan dengan KNN Imputer) untuk data yang unbalanced dan unbalanced dipilih dan di-submit ke Kaggle.

IV. Hasil

Hasil evaluasi dari tiap dataset, informasinya tersedia pada Tabel II hingga VII. Model terbaik dapat ditentukan berdasarkan akurasi model dan waktu eksekusinya. Pada Spaceship Titanic, proses imputasi dilakukan dengan menggunakan Simple Imputer dan Iterative Imputer, serta menerapkan tiga jenis encoding: Ordinal, Label, dan Target. Proses pengerjaan dimulai dari tahap penggunaan model dasar, dilanjutkan dengan Hyperparameter Tuning, dan terakhir dilakukan Meta Ensemble Learning pada model yang telah disiapkan. Detail hasil akurasi Spaceship Titanic dapat ditemukan di Tabel II, III, dan IV. Selain itu, tiga submission terbaik dari Spaceship Titanic juga terdokumentasi pada Tabel V.

Selanjutnya, untuk dataset Predict Failures Keep It Dry, proses imputasi menggunakan KNN Imputer dan menerapkan dua jenis encoding, yaitu One-Hot Encoding dan Target Encoding. Informasi mengenai akurasi model yang tidak seimbang (unbalanced) dan seimbang (balanced) dari kedua jenis encoding tersebut dapat ditemukan di Tabel VI dan CII. Proses untuk mencapai keseimbangan (balanced) menggunakan teknik seperti RUS, SMOTE, dan SMOTEEN.

V. Diskusi

A. Spaceship Titanic

Dalam penelitian ini, model telah diimputasi menggunakan Simple Imputer dan Iterative Imputer, dan hasil pelatihan dilakukan sesuai dengan encoding pada train set. Untuk model dengan Ordinal Encoding, empat model terbaik yang dihasilkan adalah Gradient Boosting Classifier dengan akurasi 0.802, Logistic Regression dengan akurasi 0.796, serta CatBoost Classifier dan Support Vector Classification (SVC) yang keduanya mencapai akurasi 0.792. Meski akurasi SVC dan CatBoost Classifier sama, CatBoost Classifier lebih unggul dari segi waktu eksekusi dengan waktu 1 menit 38,5 detik.

Model dengan Label Encoding menunjukkan tiga model terbaik yaitu Custom XGB Classifier dengan akurasi 0.797, CatBoost Classifier dengan akurasi 0.789, dan Gradient Boosting Classifier dengan akurasi 0.788. Sedangkan untuk Target Encoding, tiga model terbaik adalah Logistic Regression dengan akurasi 0.808, Custom XGB Classifier dengan akurasi 0.792, dan SVC dengan akurasi 0.788.

Setelah dilakukan Hyperparameter Tuning, akurasi model meningkat. Untuk model dengan Ordinal Encoding, tiga model terbaik adalah CatBoost Classifier dengan akurasi 0.802, Custom XGB Classifier dengan akurasi 0.800, dan Gradient Boosting Classifier dengan akurasi 0.798. Model dengan Label Encoding menunjukkan CatBoost Classifier dengan akurasi 0.799, Gradient Boosting Classifier dengan akurasi 0.798, dan Custom XGB Classifier dengan akurasi 0.797. Untuk Target Encoding, tiga model terbaik adalah CatBoost Classifier dengan akurasi 0.804, Custom XGB Classifier dengan akurasi 0.802, dan Random Forest Classifier dengan akurasi 0.802. Meski akurasi Custom XGB dan Random Forest Classifier sama, Custom XGB Classifier lebih unggul dari segi waktu eksekusi dengan waktu 1 menit 1,8 detik.

Ensemble Learning Model juga diterapkan menggunakan dua model dalam tiap encoding, yaitu Stacking Classifier dan Voting Classifier. Pada Stacking Classifier, akurasi terbaik didapatkan dengan menggunakan Ordinal Encoder yaitu sebesar 0.819. Sedangkan pada Voting Classifier, akurasi terbaik juga didapatkan dengan menggunakan Ordinal Encoder dengan skor 0.805.

Submission terbaik untuk Spaceship Titanic ini menggunakan Stacking Classifier dengan skor akurasi 0.79962. Stacking Classifier ini menggunakan Ordinal Encoder, model dasar dengan Best Hyperparameter Tuning Model, dan Meta Model menggunakan Logistic Regression dengan Best Hyperparameter Tuning. Hasil ini menunjukkan bahwa pendekatan Stacking Classifier dengan Ordinal Encoder memberikan performa terbaik dalam prediksi apakah penumpang Spaceship Titanic akan 'Terangkut' atau 'Tidak Terangkut'.

B. Predict Failures Keep it Dry

Tiga model terbaik hasil pelatihan dengan train set One-Hot Encoding (OHE) yang diimputasi dengan KNN Imputer adalah CatBoost Classifier, Stacking Classifier, dan kombinasi XGB Regressor serta CatBoost Regressor. Sedangkan untuk train set dengan Target Encoding (TE) yang diimputasi dengan KNN Imputer, tiga model terbaik adalah CatBoost Classifier, Stacking Classifier, dan Voting Classifier.

Kemudian, untuk hasil yang dikumpulkan melalui Kaggle, tiga model terbaik pada hasil pelatihan train set OHE yang diimputasi dengan KNN Imputer adalah XGBoost Regressor, Easy Ensemble Classifier, dan RUSBoost Classifier. Sementara itu, pada hasil pelatihan train set TE yang diimputasi dengan KNN Imputer, model yang dicoba hanya satu yaitu SVM SVC.

VI. Kesimpulan

Dibutuhkan algoritma klasifikasi yang mumpuni pada peningkatan kinerja prediksi data set yang berfokus meningkatkan alur kerja pemrosesan data dan model yang efektif untuk meningkatkan akurasi prediksi dalam dataset klasifikasi. Pada evaluasi dua dataset kaggle ini hasil metode yang dihasilkan yaitu:

Tabel II: Tabel Accuracy Spaceship Titanic menggunakan Base Model

Model	Encoding Method					
	Ordinal		Label		Target	
	Accuracy	Runtime	Accuracy	Runtime	Accuracy	Runtime
Random Forest	0.788	52.0s	0.774	1m 40.1s	0.782	43.2s
Logistic Regression	0.796	0.5s	0.764	0.3s	0.808	0.2s
Gradient Boosting	0.802	1m 5.1s	0.788	2m 8.0s	0.787	50.1s
Custom XGB	0.789	4.3s	0.797	3.6s	0.792	3.1s
CatBoost	0.792	1m 38.5s	0.789	1m 23.1s	0.778	1m 34.1s
KNeighbors	0.548	0.2s	0.541	0.4s	0.541	0.2s
Support Vector	0.792	3m 37.6s	0.771	3m 20.9s	0.788	4m 3.5s
Decision Tree	0.711	0.1s	0.736	0.0s	0.729	0.1s
Gaussian Naive Bayes	0.683	0.0s	0.656	0.0s	0.709	0.0s

Tabel III: Tabel Accuracy Spaceship Titanic menggunakan Hyperparameter Tuning Model

Model	Encoding Method					
	Ordinal		Label		Target	
	Accuracy	Runtime	Accuracy	Runtime	Accuracy	Runtime
Random Forest	0.795	2m 3.1s	0.794	3m 8.0s	0.802	2m 1.1s
Logistic Regression	0.783	5.9s	0.783	2.9s	0.784	1.6s
Gradient Boosting	0.798	2m 55.8s	0.798	6m 22.3s	0.801	2m 29.4s
Custom XGB	0.8	57.1s	0.797	50.6s	0.802	1m 1.8s
CatBoost	0.802	15m 37.4s	0.799	22m 26.3s	0.804	13m 1.1s
KNeighbors	0.554	1.8s	0.56	1.8s	0.559	0.9s
Support Vector	0.782	7m 25.9s	0.782	6m 20.5s	0.78	6m 49.1s
Decision Tree	0.763	1.6s	0.763	1.7s	0.767	1.8s
Gaussian Naive Bayes	-	-	-	-	-	-

Tabel IV: Tabel Meta-Ensemble Learning Model menggunakan Encoding Method

Meta-Ensemble Learning Model	Encoding Method								
	Ordinal			Label			Target		
	Accuracy	Runtime	ACVS	Accuracy	Runtime	ACVS	Accuracy	Runtime	ACVS
Stacking Classifier	0.819	18 2.4s	0.8	0.785	14m 1.5s	0.795	0.798	14m 52.0s	0.797
Voting Classifier	0.805	14.9s	0.792	0.783	16.3s	0.793	0.793	12.7s	0.791

Note: ACVS berarti Average Cross-Validation Score.

Tabel V: Tiga Submission Kaggle Spaceship Titanic Terbaik

3 Submission Kaggle Terbaik	Kaggle		
	Public Score (30%) Accuracy	*Private Score (100%) Accuracy*	*Catatan*
Stacking Classifier v7	0.79962	0.79962	• Menggunakan Ordinal Encoder • Base Model menggunakan Best Hyperparameter Tuning Model • Meta Model menggunakan Best Hyperparameter Tuning Logistic Regression Model
Stacking Classifier	0.79915	0.79915	• Menggunakan Ordinal Encoder • Base Model menggunakan Model tanpa Hyperparameter Tuning • Meta Model menggunakan Logistic Regression Model tanpa Hyperparameter Tuning
Best Tuning Catboost	0.79915	0.79915	• Menggunakan Ordinal Encoder • Menggunakan Best Hyperparameter Tuning CatBoost Classifier Model

1) Integrasi teknik pemrosesan data yang komprehensif terbukti meningkatkan kualitas data dan performa model. Hal ini termasuk penghapusan kolom yang tidak relevan, analisis distribusi kelas dan korelasi antar fitur, encoding fitur kategorikal dengan OHE dan TE, imputasi fitur numerik dengan KNN Imputer, dan normalisasi fitur numerik dengan z-score.
2) Penyeimbangan data dengan SMOTE terbukti efektif dalam menangani dataset yang tidak seimbang dan meningkatkan akurasi prediksi, terutama pada dataset Titanic yang memiliki proporsi kelas yang tidak seimbang (50.4% "Tidak Terangkut" dan 49.6% "Terangkut").
3) Evaluasi berbagai model machine learning dengan hyperparameter tuning dan validasi silang menghasilkan model terbaik untuk masing-masing dataset. Model yang berbeda menunjukkan performa yang bervariasi pada dataset yang berbeda, menunjukkan pentingnya pemilihan model yang tepat untuk tugas tertentu.
4) Penggunaan meta-ensembling dengan StackingClassifier dan VotingClassifier menghasilkan akurasi prediksi yang lebih tinggi dibandingkan dengan model tunggal. Ensemble learning memungkinkan kombinasi kekuatan dari beberapa model untuk menghasilkan prediksi yang lebih akurat dan robust.

VII. Daftar Rujukan

References

[1] A.-I. Udilă, "Encoding Methods for Categorical Data: A Comparative Analysis for Linear Models, Decision Trees, and Support Vector Ma-

Tabel VI: Tabel Accuracy Predict Failures Keep it Dry menggunakan One Hot Encoded

Unbalanced		Balanced			Notes
Model	**Accuracy**	**RUS Accuracy**	**SMOTE Accuracy**	**SMOTEENN Accuracy**	**Kaggle (20%)**	
CatBoost Clf	0.787	0.663	0.839	0.614	0.501	SMOTE
Stacking Classifier	0.785	0.548	0.836	0.669		
SVM SVC	0.787	0.637	0.789	0.534	0.499	SMOTE
XGB Reg & CatB Reg	0.538	0.647	0.725	0.590		
Voting Classifier	0.787	0.641	0.720	0.273		
XGBoost Regressor	0.537	0.646	0.715	0.585	0.554	RUS
CatBoost Regressor	0.532	0.650	0.715	0.589		
Random Forest Clf	0.787	0.637	0.685	0.277		
XGBoost Clf	0.787	0.610	0.664	0.219		
Hist Gradient Boosting Clf	0.787	0.598	0.616	0.230		
Easy Ensemble Clf	0.580	0.574	0.584	0.519	0.552	Unbalanced
Logistic Regression	0.787	0.566	0.565	0.213		
NB Gaussian	0.780	0.614	0.558	0.350		
Decision Tree	0.787	0.573	0.474	0.212		
KNN	0.785	0.559	0.455	0.226	0.500	Unbalanced
XGBoost Random Forest Clf	0.787	0.788	0.430	0.212		
SVM SGD	0.787	0.787	0.212	0.212		
Gradient Boosting Clf	0.787					
RUSBoost Clf	0.581				0.513	Unbalanced
Balanced Random Forest Clf	0.787					

Tabel VII: Tabel Accuracy Predict Failures Keep it Dry menggunakan Target Encoded

Unbalanced		Balanced			Notes
Model	**Accuracy**	**RUS Accuracy**	**SMOTE Accuracy**	**SMOTEENN Accuracy**	**Kaggle (20%)**	
CatBoost Clf	0.787	0.663	0.838	0.628		
Stacking Classifier	0.785	0.563	0.838	0.661		
SVM SVC	0.787	0.639	0.790	0.537	0.499	SMOTE
Voting Classifier	0.787	0.645	0.751	0.289		
XGBoost Clf	0.787	0.606	0.731	0.217		
XGB Reg & CatB Reg	0.531	0.651	0.725	0.596		
CatBoost Regressor	0.536	0.650	0.715	0.593		
Hist Gradient Boosting Clf	0.787	0.597	0.713	0.226		
Random Forest Clf	0.787	0.648	0.712	0.290		
XGBoost Regressor	0.535	0.643	0.711	0.597		
Easy Ensemble Clf	0.582	0.573	0.648	0.582		
Logistic Regression	0.787	0.568	0.576	0.214		
NB Gaussian	0.780	0.618	0.560	0.351		
Decision Tree	0.787	0.573	0.485	0.212		
XGBoost Rand Forest Clf	0.787	0.787	0.480	0.212		
KNN	0.786	0.558	0.456	0.226		
SVM SGD	0.787	0.787	0.212	0.212		
Gradient Boosting Clf	0.787					
RUSBoost Clf	0.581					
Balanced Rand Forest Clf	0.787					

chines," thesis, Delft Univ. Technol., Delft, Netherlands, 2023.

[2] Herdian, C., Kamila, A., and Budidarma, I. G. A. M., "Studi Kasus Feature Engineering Untuk Data Teks: Perbandingan Label Encoding dan One-Hot Encoding Pada Metode Linear Regresi," Technologia, vol. 15, no. 1, pp. 1-10, 2023.

[3] E. Prasetyo, "Data Mining: Konsep Dan Aplikasi Menggunakan Matlab," Yogyakarta: ANDI, 2014.

[4] Suyal, M. and Goyal, P., "A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms based on Supervised Learning," Intell. J. Eng. Trends Technol. (IJETT), vol. 70, no. 7, pp. 43-48, 2022.

[5] R. Umar, I. Riadi, and Purwono, "Perbandingan Metode SVM, RF dan SGD untuk Penentuan Model Klasifikasi Kinerja Programmer pada Aktivitas Media Sosial," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 4, no. 2, pp. 329-335, 2020.

[6] Viresh. "Efficient Hyperparameter Tuning with Successive Halving," Medium, Sep. 5, 2023. Available: https://medium.com/@vireshj/efficient-hyperparameter-tuning-with-successive-halving-7f50a57bb160

[7] Melek N, Kayikcioglu T, Maleki M, et al. "A Novel Simple Method to Select Optimal k in k-Nearest Neighbor Classifier," International Journal of Computer Science and Information Security (IJCSIS), vol. 15, no. 2, 2017.

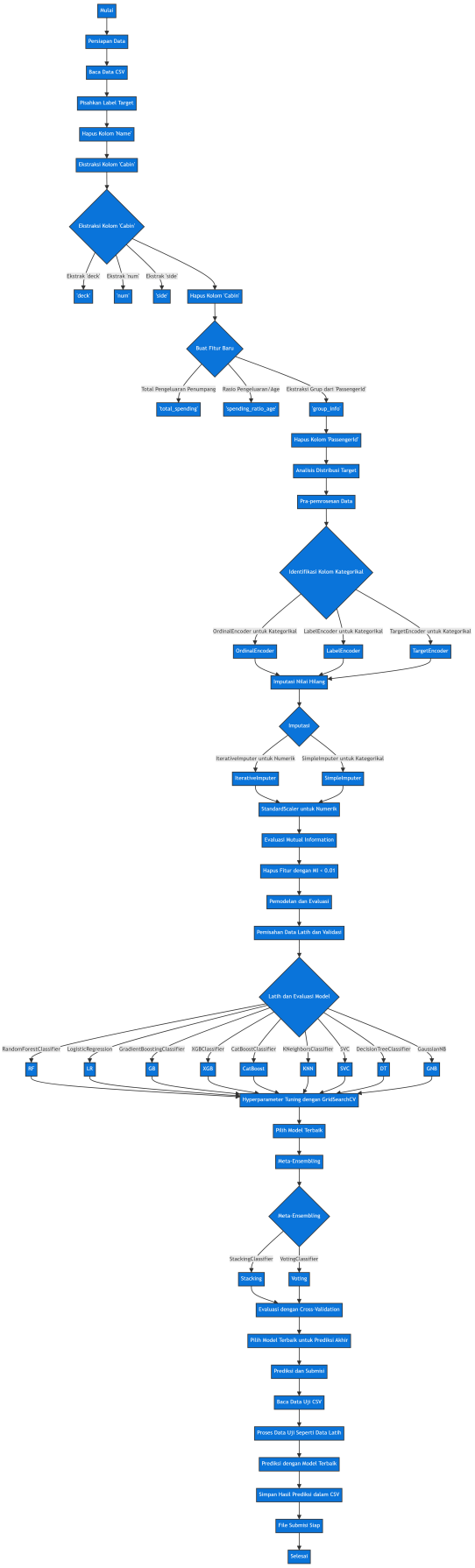
[8] V. Anand, B. Kiran, S. R. Srividhya, Md. H. Rahman, and A. A. Khan, "Gaussian Naive Bayes Algorithm: A Reliable Technique Involved in the Assortment of the Segregation in Cancer," Mobile Information Systems, vol. 2022, no. 2, pp. 1-7, Jun. 2022, doi: 10.1155/2022/2436946.

[9] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001, doi: 10.1023/A:1010933404324.

[10] S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin, "Parallel boosted regression trees for web search ranking," in Proceedings of the 20th International Conference on World Wide Web, 2011, pp. 387-396, doi: 10.1145/1963405.1963450

[11] H. Chen, Z. Guo, B. Chen, and J. Shi, "Improving Random Forests With

Balanced Forests," in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, doi: 10.1109/IJCNN.2018.8489211.
[12] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 40, no. 1, pp. 185-197, Jan. 2010, doi: 10.1109/TSMCA.20092029559.
[13] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, Applied Logistic Regression, 3rd ed., New York, NY, USA: Wiley, 2013.
[14] X. Liu, J. Wu, and Z. Zhou, "Exploratory Undersampling for Class-Imbalance Learning," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 2, pp. 539-550, Apr. 2009, doi: 10.1109/TSMCB.2008.2007853.
[15] T.K. Ho, "The Random Subspace Method for Constructing Decision Forests," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 832-844, Aug. 1998, doi:10.1109/34.709601.
[16] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794, doi:10.1145/2939672.2939785.
[17] L. Prokhorenkova, G. Gusev, A. Vorobev, A. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in Advances in Neural Information Processing Systems 31 (NIPS 2018), 2018, pp. 6638-6648.
[18] L.S. Breiman, "Stacked Regressions," Machine Learning, vol. 24, no. 1, pp. 49-64, Jul. 1996, doi: 10.1023/A:1018046112853.
[19] K. Bekhouche, "Decision Tree Classification in Python," DataCamp, Nov. 29, 2022. [Online]. Available: https://www.datacamp.com/tutorial/decision-tree-classification-python.
[20] "How CatBoost algorithm works," ArcGIS Pro Documentation, [Online]. Available: https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-catboost-works.htm. [Accessed: 20-May-2024].
[21] Simplilearn. 'Gradient Boosting Algorithm: A Complete Guide for Beginners.' [Online]. Available: https://www.simplilearn.com/gradient-boosting-algorithm-in-python-article. [Accessed: 20-May-2024].
[22] GeeksforGeeks. 'Difference Between Random Forest and XGBoost.' [Online]. Available: https://www.geeksforgeeks.org/difference-between-random-forest-vs-xgboost/. [Accessed: 20-May-2024].
[23] V. Jain, "Efficient Hyperparameter Tuning with Successive Halving," Medium, Nov. 18, 2022. [Online]. Available: https://medium.com/@vireshj/efficient-hyperparameter-tuning-with-successive-halving-7f50a57bb160. [Accessed: May 20, 2024].
[24] J. F. Hair, W. C. Black, B. J. Babin, and R. E. Anderson, "Multivariate data analysis," Cengage Learning, 2010.
[25] T. M. Cover and J. A. Thomas, "Elements of information theory," John Wiley & Sons, 2006.
[26] A. Géron, "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow," O'Reilly Media, 2019.
[27] Kaggle, "Spaceship Titanic: Predict which passengers are transported to an alternate dimension," [Online]. Available: https://www.kaggle.com/competitions/ppm-spaceship-titanic. [Accessed: May 21, 2024].
[28] Kaggle (2021). "Tabular Playground Series - Mar 2021," Kompetisi Kaggle, Tersedia online: https://www.kaggle.com/competitions/ppm-tabular-playground-series/, Diakses pada: 21 Mei 2024.
[29] Kaggle. "Kaggle," Tersedia online: https://www.kaggle.com/, Diakses pada: 20 May 2024.



Gambar 4. Alur kerja pada *dataset* Spaceship Titanic. Gambar lebih jelas dapat diakses melalui link ini: https://drive.google.com/file/d/1wOY6bIO1ZnfX1VdaDFZ13UwRpwp0jGDS/view?usp=sharing

Gambar 5. Distribusi target (kolom kelas ”Transported”)

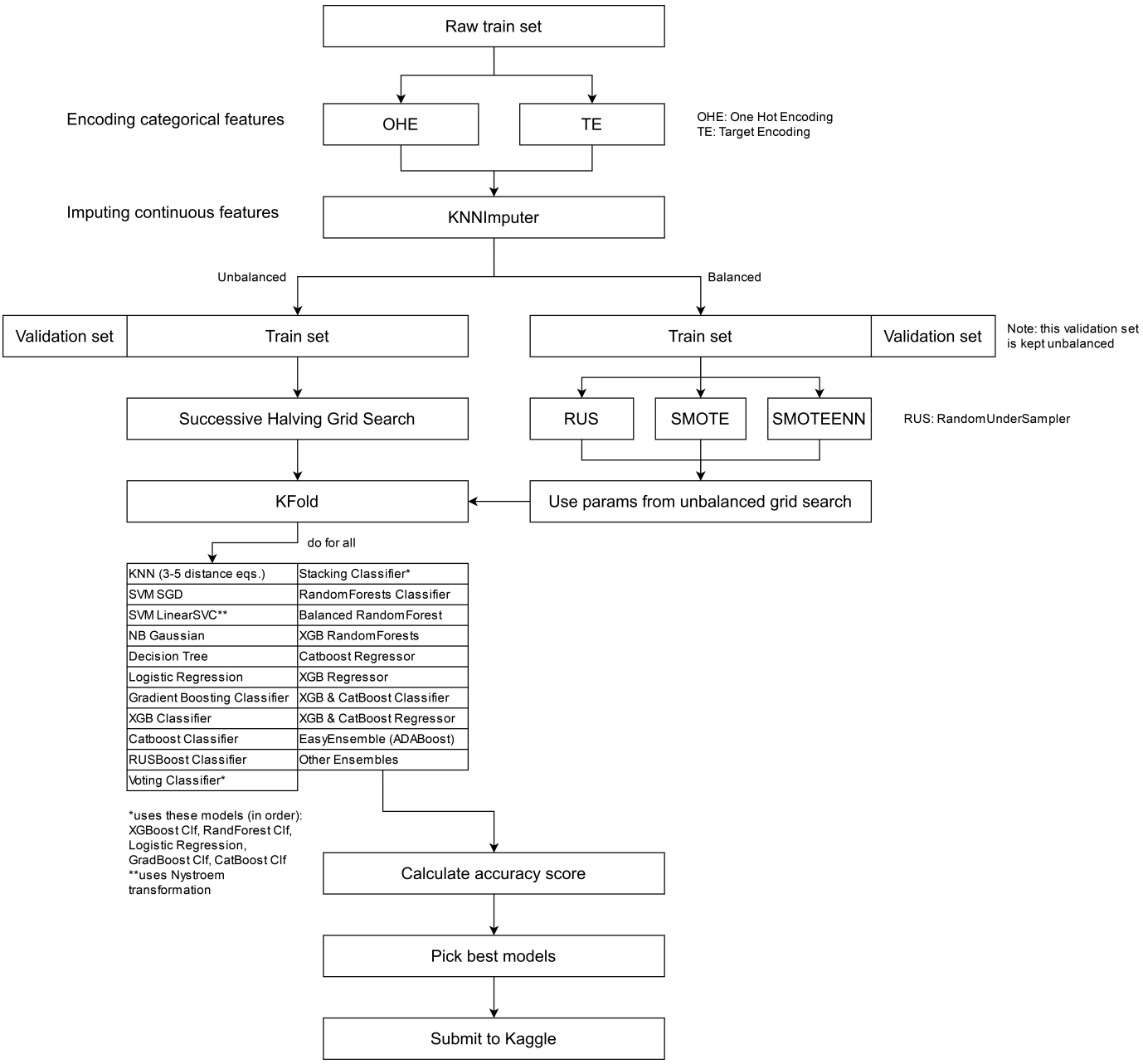
TotalSpend	0.135769
TotalSpend_scaled	0.131117
SpendAgeRatio	0.112989
CryoSleep	0.112680
SpendAgeRatio_scaled	0.112418
Spa_scaled	0.075916
Spa	0.073786
RoomService	0.073186
RoomService_scaled	0.065981
VRDeck_scaled	0.062338
VRDeck	0.060381
ShoppingMall_scaled	0.051886
FoodCourt	0.050537
FoodCourt_scaled	0.044224
ShoppingMall	0.041630
group	0.024067
group_scaled	0.018509
Age_scaled	0.017394
HomePlanet	0.017131
Age	0.013503
deck	0.008448
Destination	0.004865
num	0.001865
VIP	0.000000
side	0.000000

Name: MI Scores, dtype: float64

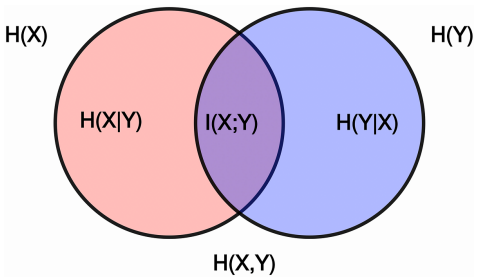
Gambar 6. Fitur-fitur sebelum dilakukan seleksi fitur

```
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   HomePlanet            8693 non-null   float64
 1   CryoSleep             8693 non-null   float64
 2   deck                  8693 non-null   float64
 3   num                   8693 non-null   float64
 4   side                  8693 non-null   float64
 5   group                 8693 non-null   float64
 6   Age_scaled            8693 non-null   float64
 7   RoomService_scaled    8693 non-null   float64
 8   FoodCourt_scaled      8693 non-null   float64
 9   ShoppingMall_scaled   8693 non-null   float64
 10  Spa_scaled            8693 non-null   float64
 11  VRDeck_scaled         8693 non-null   float64
 12  TotalSpend_scaled     8693 non-null   float64
 13  SpendAgeRatio_scaled  8693 non-null   float64
 14  group_scaled          8693 non-null   float64
dtypes: float64(15)
memory usage: 1018.8 KB
```

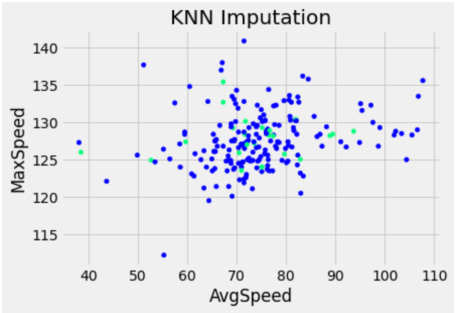
Gambar 7. Fitur-fitur setelah dilakukan seleksi fitur



Gambar 8. Alur kerja pada *dataset* Keep it Dry



Gambar 9. Diagram Venn untuk menggambarkan mutual information



Gambar 10. Fitur-fitur setelah dilakukan seleksi fitur

Pembagian Tugas Kelompok

Link Repositori

- Repositori Intro to ML
- Repositori Spaceship Titanic

Pembagian Tugas Kelompok

- Hasan dan Husain: Buat kode Spaceship Titanic
- Hugo: Buat kode Predict Failures Keep it Dry
- Wayan, Hilda, dan Gratia: Buat Makalah