

Nama : Yoga Patangga Balapradhana  
NIM : 1103194138

## Penjelasan Classification Task

Data yang digunakan pada Classification Task ini adalah data KlasifikasiUTS.csv yang disediakan oleh Teaching Assistant. Mula-mula file csv tersebut di-copy terlebih dahulu ke dalam Google Drive, untuk kemudian dapat dibaca dengan menggunakan baris perintah:

```
# Menghubungkan Google Colab dengan Google Drive
from google.colab import drive

# Mount Google Drive ke Colab
drive.mount('/content/drive')
file_path = '/content/drive/MyDrive/KlasifikasiUTS.csv'
df = pd.read_csv(file_path)
print(df.head())
```

Setelah itu dilakukan eksplorasi data yang sudah dimuat tersebut menggunakan perintah:

```
print(df.shape)
print(df.info())
X = df.drop("Class", axis=1) # Features
y = df["Class"] # Target

(284807, 31)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
```

Sehingga dapat diketahui data tersebut memiliki 284.807 baris dan 31 kolom dengan tipe data integer pada kolom Class dan tipe data float pada kolom sisanya. Kemudian setelah melihat data statistik dengan menggunakan perintah `df.describe()`, maka kolom Class dipilih sebagai target dan kolom sisanya sebagai features untuk diproses lebih lanjut.

Langkah selanjutnya adalah membagi dataset menjadi data latih dan data uji dengan persentase 70:30 menggunakan perintah `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)`. Setelah itu dilakukan seleksi fitur pada data latih sebanyak 3 langkah, yaitu 1) menghapus quasi-constant features, 2) menghapus fitur duplikat, dan 3) menghapus fitur yang memiliki korelasi tinggi.

Setelah dilakukan seleksi fitur, maka langkah selanjutnya adalah membuat model pipeline dengan perintah:

```
model_pipeline = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression(max_iter=1000))
])

model_pipeline.fit(X_train, y_train)
y_pred = model_pipeline.predict(X_test)
```

Nama : Yoga Patangga Balapradhana  
NIM : 1103194138

Pipeline ini nantinya akan menyesuaikan dengan model yang akan digunakan, misal baris di atas untuk Logistic Regression. Jika model berikutnya akan menggunakan Decision Tree, maka baris (`'classifier', LogisticRegression()`) dapat diganti dengan (`'classifier', DecisionTreeClassifier()`), demikian juga halnya dengan metode yang lainnya. Setelah itu, model dapat dilatih dan dilakukan prediksi pada data uji menggunakan perintah:

```
model_pipeline.fit(X_train, y_train)
y_pred = model_pipeline.predict(X_test)
```

Langkah berikutnya adalah menggambar confusion matrix untuk memvisualisasikan data aktual dan data prediksi serta menghitung matriks evaluasi yaitu accuracy, precision, dan recall, dan F1 score dengan perintah:

```
# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
```

Accuracy adalah metrik yang mengukur seberapa sering model melakukan prediksi yang benar terlepas dari kelas mana yang diprediksi. Accuracy dihitung dengan membandingkan jumlah prediksi yang benar dengan total jumlah data.

Precision mengukur proporsi prediksi positif yang benar dari semua prediksi positif yang dilakukan oleh model. Metrik ini memberikan gambaran tentang seberapa banyak dari semua prediksi positif yang benar-benar positif.

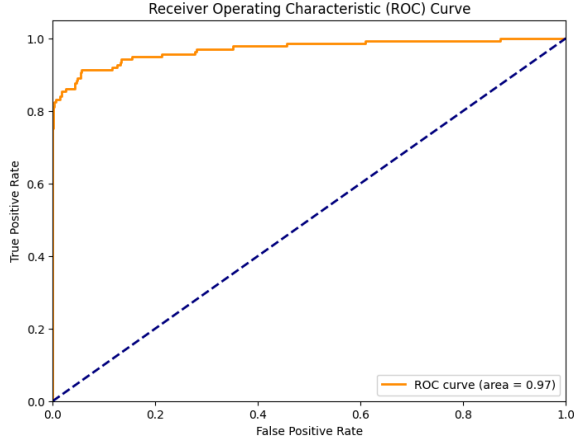
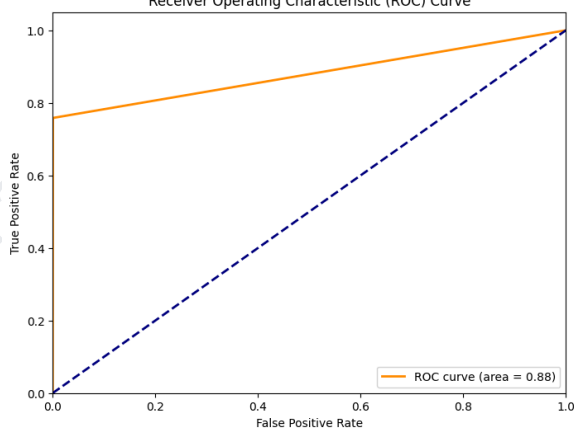
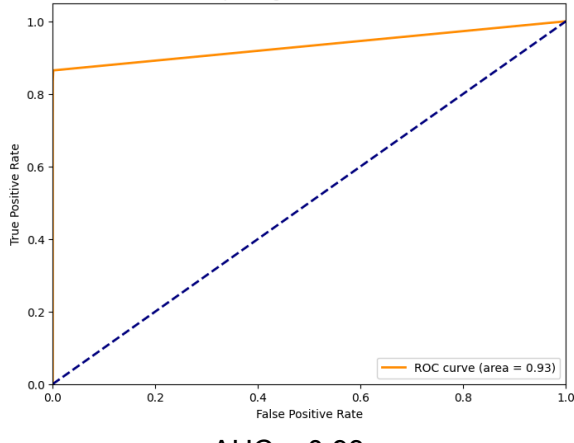
Recall mengukur proporsi prediksi positif yang benar dari seluruh kasus positif yang sebenarnya. Metrik ini memberikan gambaran tentang seberapa banyak dari keseluruhan kasus positif yang berhasil dikenali oleh model.

F1 Score adalah ukuran yang menggabungkan precision dan recall ke dalam satu angka yang lebih seimbang dan merupakan harmonic mean dari keduanya. F1 Score memberikan kompromi yang baik antara precision dan recall terutama ketika ada ketidakseimbangan antara keduanya.

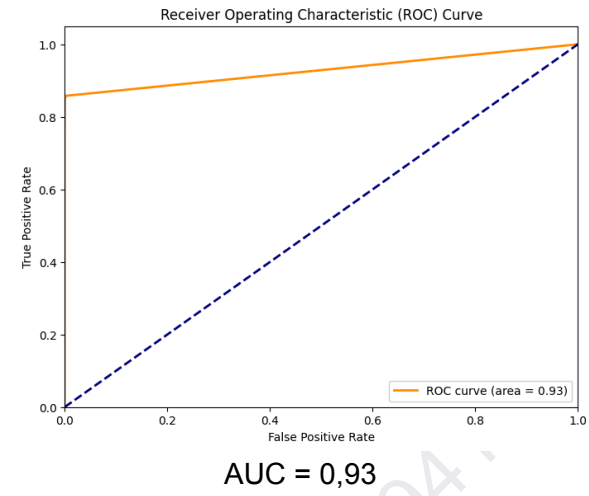
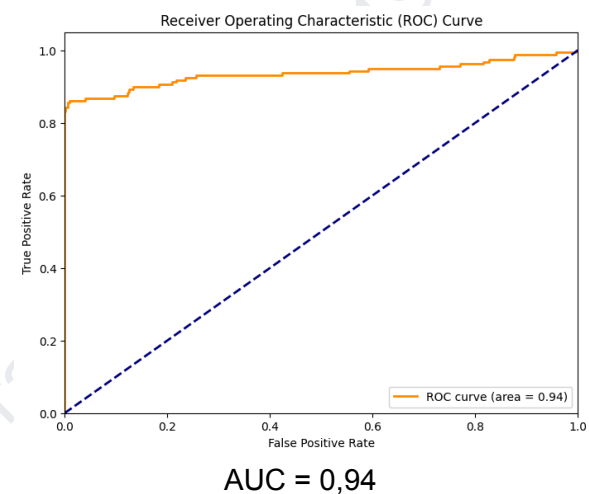
Model yang digunakan dalam Classification Task ini ada 5, yaitu Logistic Regression, Decision Tree Classifier, KNeighbors Classifier, Bagging Classifier, dan SVC. Perbandingan matriks evaluasi dari kelima model tersebut dapat dilihat pada Tabel 1 di bawah ini.

Nama : Yoga Patangga Balapradhana  
NIM : 1103194138

Tabel 1. Perbandingan model berdasarkan matriks evaluasi (Accuracy, Precision, Recall, F1 Score)

Model	Acc	Prec	Recall	F1	ROC Curve & AUC
Logistic	0,99	0,87	0,58	0,69	 <p>AUC = 0,97</p>
Decision Tree	0,99	0,76	0,76	0,76	 <p>AUC = 0,88</p>
KNeighbor	0,99	0,93	0,78	0,85	 <p>AUC = 0,93</p>

Nama : Yoga Patangga Balapradhana  
NIM : 1103194138

Bagging	0,99	0,97	0,75	0,85	 <p>Receiver Operating Characteristic (ROC) Curve</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.93)</p> <p>AUC = 0,93</p>
SVC	0,99	0,95	0,69	0,8	 <p>Receiver Operating Characteristic (ROC) Curve</p> <p>True Positive Rate</p> <p>False Positive Rate</p> <p>ROC curve (area = 0.94)</p> <p>AUC = 0,94</p>

Dari Tabel 1 di atas, dapat disimpulkan bahwa model Bagging Classifier memberikan performa yang paling baik dengan nilai accuracy, precision, dan recall, dan F1 score yang paling tinggi dibandingkan keempat model yang lainnya. Namun jika dilihat dari kurva ROC dan nilai AUC, model Logistic Regression memiliki nilai AUC yang tertinggi dibandingkan keempat model lainnya, meskipun memiliki nilai Recall dan F1 Score yang terkecil.

## Soal Analisis

1. Jika model Machine Learning menunjukkan AUC-ROC tinggi (0.92) tetapi Presisi sangat rendah (15%) pada dataset tersebut, jelaskan faktor penyebab utama ketidaksesuaian ini! Bagaimana strategi tuning hyperparameter dapat meningkatkan Presisi tanpa mengorbankan AUC-ROC secara signifikan? Mengapa Recall menjadi pertimbangan kritis dalam konteks ini, dan bagaimana hubungannya dengan cost false negative?

**Penjelasan:** Jika model Machine Learning menunjukkan AUC-ROC tinggi (0.92) tetapi Presisi sangat rendah, maka model tersebut secara umum mampu membedakan antara kelas positif dan negatif dengan baik, tetapi terlalu sering memprediksi positif secara keliru sehingga menghasilkan banyak False Positives (FP).

Nama : Yoga Patangga Balapradhana  
NIM : 1103194138

Penyebab dari ketidaksesuaian ini biasanya akibat nilai threshold default 0,5 yang terlalu rendah untuk prediksi probabilistik, dikarenakan ROC-AUC mengukur kemampuan model dalam mengurutkan kelas positif dan negatif tanpa mempertimbangkan threshold tertentu. Selain itu, pada dataset yang tidak seimbang (misalnya 5% positif, 95% negatif), model bisa mencapai AUC tinggi namun tetap memiliki presisi rendah karena hanya sebagian kecil prediksi positif yang benar.

Strategi tuning hyperparameter yang dapat dilakukan adalah dengan menaikkan nilai threshold yang awalnya 0,5 menjadi lebih tinggi (misal 0,7 atau 0,8) untuk mengurangi False Positives. Namun tradeoff-nya nilai presisi akan meningkat, tapi nilai Recall bisa menurun.

**2. Sebuah fitur kategorikal dengan 1000 nilai unik (high-cardinality) digunakan dalam model machine learning. Jelaskan dampaknya terhadap estimasi koefisien dan stabilitas Presisi! Mengapa target encoding berisiko menyebabkan data leakage dalam kasus dataset tersebut, dan alternatif encoding apa yang lebih aman untuk mempertahankan AUC-ROC.**

**Penjelasan:** Salah satu dampak dari high cardinality feature adalah estimasi koefisien menjadi tidak stabil. Jika model seperti regresi logistik digunakan dengan encoding one-hot untuk fitur kategorikal ber-cardinality tinggi, maka hal tersebut akan mengakibatkan jumlah dimensi input meningkat drastis (sparse features) dan bisa terjadi overfitting. Selain itu, koefisien regresi juga akan menjadi besar, tidak stabil, dan sulit diinterpretasi.

Salah satu cara mengatasinya adalah menggunakan alternatif encoding yang lebih aman untuk mempertahankan AUC-ROC seperti Leave-One-Out Encoding (LOO Encoding), Frequency Encoding, atau Hashing Encoding.

**3. Setelah normalisasi Min-Max, model SVM linear mengalami peningkatan Presisi dari 40% ke 60% tetapi Recall turun 20%. Analisis dampak normalisasi terhadap decision boundary dan margin kelas minoritas! Mengapa scaling yang sama mungkin memiliki efek berlawanan jika diterapkan pada model Gradient Boosting?**

**Penjelasan:** Hal tersebut bisa terjadi karena karakteristik Gradient Boosting adalah berbasis pohon keputusan (decision tree) yang tidak tergantung pada skala fitur karena mereka memilah berdasarkan split nilai fitur, bukan jarak atau bobot numerik absolut. Jika Min-Max Scaling diterapkan maka tidak akan menambah informasi apapun bagi pohon dan bisa mengurangi ketepatan split jika nilai fitur asli lebih informatif (misalnya data ordinal yang telah di-scaling).

Penerapan Scaling seperti Min-Max pada SVM akan meningkatkan stabilitas margin, tetapi model bisa menjadi bias ke kelas mayoritas. Sedangkan jika menggunakan Gradient Boosting, lebih baik hindari scaling atau gunakan original scale untuk hasil yang optimal.