Fraud Detection

Delpita Putri

Model *machine learning* untuk mendeteksi kecurangan dalam *mobile transactions*

Project Overview

Dalam proyek ini, dilakukan beberapa model pelatihan untuk mendeteksi adanya transaksi penipuan. Terdapat 5 model dasar yang digunakan, yaitu **Logistic Regression, KNeighbors Classifier, Random Forest Classifier, XGB Classifier, dan Support Vector Machine Classifier.** Model tersebut terus dioptimalkan sehinggan diperoleh dua model teratas berdasarkan hasil akurasi pelatihan dan pengujian, yaitu model **XGBoost** dan **RandomForest**. Selain itu juga dilakukan pencarian grid pada hiperparameter, menyeimbangkan label dengan SMOTE, dan pengambilan sampel dari dataset asli. Kedua model RandomForest dan XGBoost memiliki akurasi lebih dari 99% pada data acak yang mencakup semua kasus penipuan dan beberapa data yang aman. Hasil akhir yang diperoleh adalah, model XGBoost merupakan model machine learning yang terbaik dan memiliki akurasi 99% pada kedua set pelatihan dan pengujian. Skor akurasi dihitung dengan menghitung *Area Under the Receiver Operating Characteristic Curve* (ROC AUC) dari hasil prediksi.

Data

Dataset yang digunakan adalah Fraud.CSV dari Kaggle. Dataset ini mensimulasikan transaksi uang elektronik berdasarkan sampel transaksi nyata selama 1 bulan dari sebuah perusahaan multinasional yang merupakan penyedia layanan uang elektronik yang saat ini beroperasi di lebih dari 14 negara di seluruh dunia.

Dataset ini memiliki kolom-kolom sebagai berikut:

1. `step`: Menggambarkan satuan waktu. 1 step setara dengan waktu 1 jam. Total step adalah 744 (simulasi selama 30 hari).

2. `type`: Jenis transaksi, seperti CASH-IN, CASH-OUT, DEBIT, PAYMENT, dan TRANSFER.

3. `amount`: Jumlah uang transaksi.

4. `nameOrig`: Nama pelanggan yang melakukan transaksi.

5. `oldbalanceOrg`: Saldo awal sebelum transaksi.

6. `newbalanceOrig`: Saldo baru setelah transaksi.

7. `nameDest`: Nama pelanggan penerima transaksi.

8. `oldbalanceDest`: Saldo awal penerima sebelum transaksi. Perlu dicatat bahwa tidak ada informasi untuk pelanggan yang nama mereka dimulai dengan 'M' (Pedagang).

9. `newbalanceDest`: Saldo baru penerima setelah transaksi. Perlu dicatat bahwa tidak ada informasi untuk pelanggan yang nama mereka dimulai dengan 'M' (Pedagang).

10. `isFraud`: Transaksi yang dilakukan oleh agen-agen penipuan. Dalam dataset ini, penipuan bertujuan untuk mengambil alih akun pelanggan dan mencoba mengosongkan dana dengan mentransfernya ke akun lain, lalu mencairkannya dari sistem.

11. `isFlaggedFraud`: Model bisnis bertujuan untuk mengendalikan dan mendeteksi terjadinya kegiatan transfer dala jumlah sangat besar dari satu akun ke akun lain dan menandai upaya ilegal. Upaya ilegal dalam dataset ini adalah upaya mentransfer lebih dari 200.000 dalam satu kali transaksi.

Project Steps

1. Loading Data dan EDA

```
file_path = '/content/drive/My Drive/Colab Notebooks/Fraud.csv'
data = pd.read_csv(file_path)
data.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

```
#examine the dataset
data.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06	1.290820e-03	2.514687e-06
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06	3.590480e-02	1.585775e-03
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05	0.000000e+00	0.000000e+00
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06	0.000000e+00	0.000000e+00
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08	1.000000e+00	1.000000e+00

```
#Check if there is anu null values
data.isna().sum().sum()

```

```
0
```

```
#check for duplicate values
data.duplicated(keep='first').any()

```

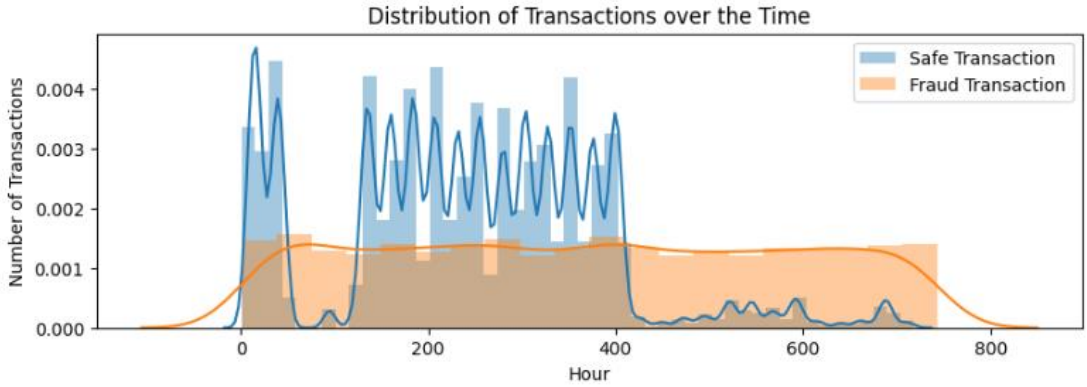
```
False
```

Tidak ada data null ataupun data yang terduplikasi.

- Distribution of All Transactions

```
# Filter data by the labels. Safe and Fraud transaction
safe = data[data['isFraud']==0]
fraud = data[data['isFraud']==1]
```

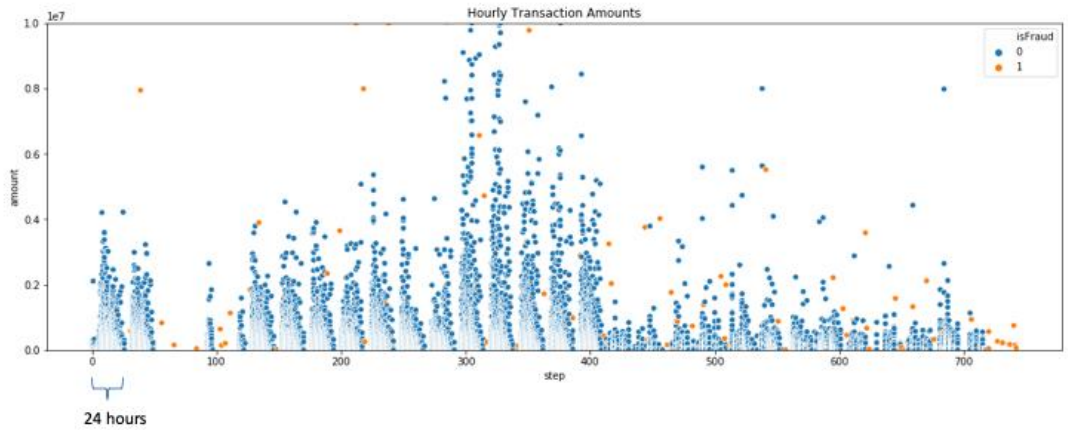
```
#See the frequency of the transactions for each class on the same plot.
plt.figure(figsize=(10, 3))
sns.distplot(safe.step, label="Safe Transaction")
sns.distplot(fraud.step, label='Fraud Transaction')
plt.xlabel('Hour')
plt.ylabel('Number of Transactions')
plt.title('Distribution of Transactions over the Time')
plt.legend()
```



Filter data menjadi 2 kelompok, yaitu safe dan fraud agar mudah untuk dibandingkan. Kemudian visualisasikan Frequency Diagram distribusi yang menunjukkan jumlah transaksi yang terjadi setiap jam (step). Terdapat perubahan drastis dalam jumlah transaksi yang terjadi dari waktu ke waktu. Meskipun transaksi aman mulai melambat pada hari ke-3 dan ke-4 serta setelah hari ke-16 bulan tersebut. Transaksi penipuan terjadi dengan kecepatan yang relatif stabil. Terutama pada pertengahan hingga akhir bulan, terdapat lebih sedikit transaksi aman, tetapi jumlah transaksi penipuan tidak mengalami penurunan sama sekali.

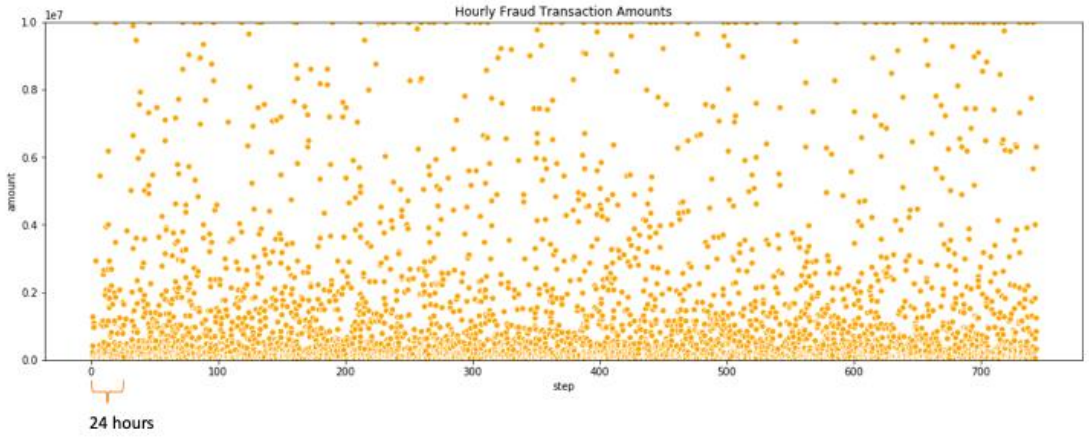
```
smalldata=data.sample(n=100000, random_state=1)
smalldata=smalldata.sort_index()
smalldata=smalldata.reset_index(drop=True)

#plot the small data
plt.figure(figsize=(18,6))
plt.ylim(0, 10000000)
plt.title('Hourly Transaction Amounts')
ax = sns.scatterplot(x="step", y="amount", hue="isFraud",
                     data=smalldata)
```



Dari scatter plot diatas terlihat jumlah transaksi memiliki pola setiap harinya. Grafik plot akan meningkat di pertengahan 24 jam atau bisa disimpulkan jumlah transaksi paling tinggi terjadi pada siang hari.

```
#The hourly amount of al fraud transactions
plt.figure(figsize=(18,6))
plt.ylim(0, 10000000)
plt.title('Hourly Fraud Transaction Amounts')
ax = sns.scatterplot(x="step", y="amount", color='orange',
                     data=fraud)
```



Jumlah terjadinya transaksi penipuan tidak menunjukkan pola yang signifikan. Berdasarkan plot diatas, fraud atau penipuan terjadi hampir setiap jam dengan frekuensi yang sama. Transaksi penipuan paling banyak terjadi dalam transaksi uang yang sedikit, begitu juga sebaliknya. Namun, polanya tidak berubah dari waktu ke waktu.

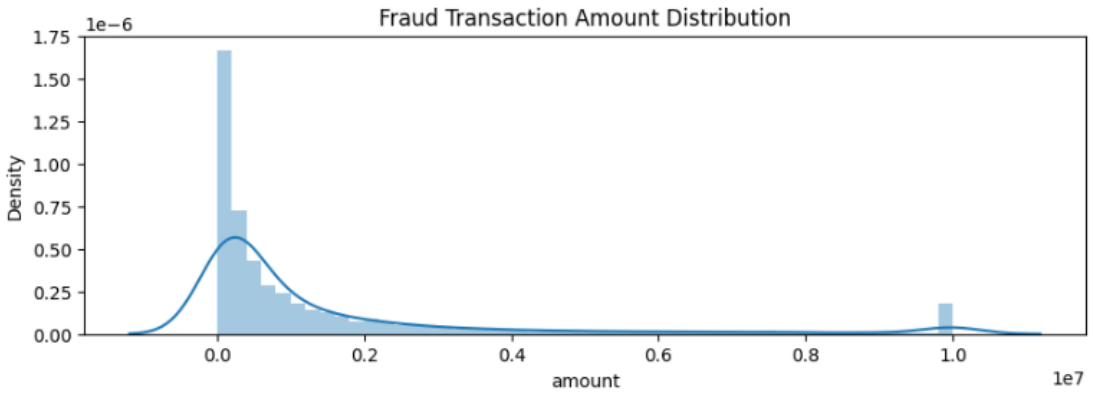
- **Transactions Amount Distributions**

```
# fraud transactions amount value counts
fraud.amount.value_counts()
```

```
10000000.00    287
0.00            16
1165187.89       4
429257.45        4
181.00           2
               ...
149668.66        1
7316255.05       1
222048.71        1
9585040.37       1
234377.29        1
Name: amount, Length: 3977, dtype: int64
```

```
# Fraud transactions amount distribution plot
plt.figure(figsize=(10,3))
plt.title('Fraud Transaction Amount Distribution')
sns.distplot(fraud.amount)
```

```
<Axes: title={'center': 'Fraud Transaction Amount Distribution'}, xlabel='amount', ylabel='Density'>
```



Terdapat 287 transaksi penipuan dengan jumlah $1 juta. Ini adalah jumlah kasus paling sering dari transaksi penipuan. Sebagian besar penipuan terjadi di bawah $400.000.

```
#Fraud transaction boxplot for amount distribution
plt.figure(figsize=(10,3))
plt.title('Fraud Transaction Amount Distribution')
ax = sns.boxplot(x=fraud["amount"])
```


```
sorted_fraud_amounts = fraud['amount'].sort_values(ascending=True)

sorted_fraud_amounts.head(20)

```

```
5996408      0.0
5996410      0.0
6266414      0.0
4965641     63.8
4965642     63.8
277266     119.0
277265     119.0
Name: amount, dtype: float64
```

Transaksi penipuan terjadi dalam rentang yang luas, mulai dari $64 hingga $10 juta dolar. Distribusi frekuensi jumlah uang yang terlibat dalam transaksi penipuan cenderung positif. Ada juga 16 transaksi penipuan dengan jumlah 0 dolar. Hal ini sedikit aneh, mungkin saja para agen penipuan ingin membuat noise dalam transaksi agar bisa mengelabui atau menyembunyikan penipuan yang sebenarnya.

```
#average amount for frauds below 400K
fraud[fraud.amount<400000].amount.mean()
```

```
144912.1682893401
```

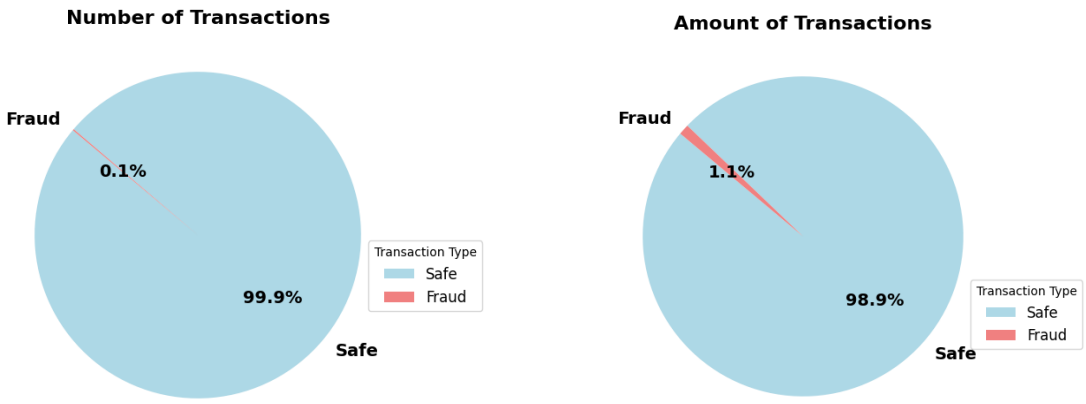
Jumlah uang rata rata dalam transaksi penipuan sekitar 145,000 dolar

- **Type of Transactions**

```
#checking type of fraud transactions
fraud.type.value_counts()
```

```
CASH_OUT    4116
TRANSFER    4097
Name: type, dtype: int64
```

Aktivitas penipuan hanya terjadi pada transaksi transfer dan penarikan tunai (cash-out). Penggunaan kartu debit cenderung aman. Jadi untuk pelatihan model hanya menggunakan data transaksi transfer dan cash-out saja karena jenis transaksi lainnya tidak memiliki kasus penipuan. Hal ini akan membantu model fokus pada data yang paling relevan untuk mendeteksi penipuan.



Pada piechart diatas terlihat proporsi perbandingan antara kasus dan jumlah uang pada kasus penipuan VS uang yang aman.

2. Feature Engineering

Kolom 'nameOrig' dan 'nameDest' seharusnya berisi nama-nama orang. Saat ini, kolom-kolom ini tidak dapat digunakan dalam model machine learning karena berisi data teks yang tidak dapat diolah langsung oleh algoritma. Namun, jika ada transaksi yang berulang antara dua orang tertentu, informasi tersebut mungkin berguna bagi model klasifikasi. Setelah dilakukan pengecekan ternyata tidak ada transaksi yang berulang antara dua pihak (nama pengirim dan penerima), setiap transaksi memiliki pasangan yang berbeda. Oleh karena itu, kedua kolom ini dapat dengan aman untuk dihapus, karena informasi di dalamnya tidak memberikan kontribusi yang berguna untuk analisis atau model machine learning yang akan dibuat.

Dataset yang ada terlalu besar untuk digunakan langsung dalam algoritma machine learning. Oleh karena itu, akan diambil sampel acak yang cukup untuk membangun sebuah model machine learning. Dalam proyek ini, data sampel yang digunakan berukuran sebanyak 100.000 data. Dengan kata lain, hanya sebagian kecil dari dataset asli yang akan digunakan dalam analisis dan pemodelan. Kemudian pada semua data yang masih berupa string atau objek, dilakukan encoding agar bisa dimasukkan dalam pemodelan.

3. Machine Learning

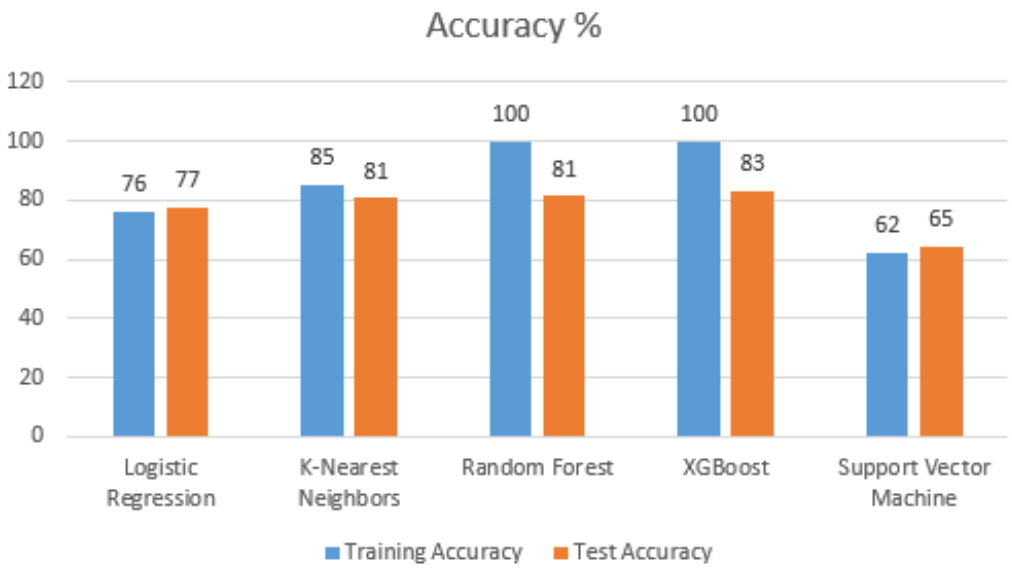
```
from sklearn.model_selection import train_test_split # import train_test_split function
from sklearn.linear_model import LogisticRegression # import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score # import accuracy metrics
from sklearn.ensemble import RandomForestClassifier #import RandomForestClassifier
from sklearn import svm  #import support vector machine classifier
import xgboost as xgb
from xgboost import  XGBClassifier  #import xgboost classifier
from sklearn.neighbors import KNeighborsClassifier #import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV # import GridSearchCV
# suppress all warnings
import warnings
warnings.filterwarnings("ignore")
```

```
#Slice the target and features from the dataset
features=df.drop('isFraud', axis=1)
target =df.isFraud

# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2)
```

3.1 Baseline Models

Dataset akan dilatih dalam lima model klasifikasi dengan parameter default untuk melihat seberapa baik kinerja masing-masing model. Data yang digunakan sangat tidak seimbang, di mana kelas positif (penipuan) hanya menyumbang 0,01% dari semua transaksi. Oleh karena itu, akan dilakukan pengukuran akurasi menggunakan Area Under the Precision-Recall Curve (AUPRC). Confussion Matrix Accuracy tidak memiliki makna yang signifikan untuk klasifikasi yang tidak seimbang seperti ini.



Dari diagram diatas, kita melihat bahwa akurasi pelatihan terbaik diperoleh dari XGBoost dan Random Forest Classifier. Kedua model ini akan dioptimalkan dengan melakukan pencarian grid terhadap berbagai nilai parameter. Pencarian grid akan bertujuan untuk menemukan parameter terbaik untuk diberikan ke model agar menghasilkan hasil yang paling akurat. Fungsi ini akan mengambil nilai-nilai parameter dan classifier, lalu mencetak kombinasi parameter terbaik.

3.2 Grid Search for Best Hyper-Parameter

```
# Running RandomForestClassifier with best parameters
rf_model=RandomForestClassifier(n_estimators=100,
                                criterion= 'gini',
                                max_depth= 10,
                                min_samples_split= 3)


run_model(rf_model, X_train, y_train,X_test, y_test)
```

```
Model Scores
-----------------------
Training Accuracy: 87.5%
Test Accuracy:     80.65%
-------------------------------------------------------
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19938
           1       1.00      0.61      0.76        62

    accuracy                           1.00     20000
   macro avg       1.00      0.81      0.88     20000
weighted avg       1.00      1.00      1.00     20000
```

Akurasi turun karena model diberikan batasan untuk dapat mempelajari pola dalam data dengan mengatur max_depth menjadi 10. Saat nilai ini lebih tinggi atau pada default, model dapat terus belajar hingga tingkat yang sangat dalam, tetapi ini memerlukan waktu yang lama terutama untuk data besar. Meskipun akurasi menurun, Random Forest Classifier tetap menggunakan batasan ini dan akan ditingkatkan kemudian.

```
# Running XGBClassifier with best parameters
xgb_model=XGBClassifier(colsample_bytree= 1,
                        n_estimators= 100,
                        gamma= 0.1,
                        learning_rate=0.1,
                        max_depth=5
                        )

run_model(xgb_model, X_train, y_train,X_test, y_test)
```

```
Model Scores
------------------------
Training Accuracy: 90.05%
Test Accuracy:     83.06%
-----------------------------------------------------
Classification Report :
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     19938
           1       1.00      0.66      0.80        62

    accuracy                           1.00     20000
   macro avg       1.00      0.83      0.90     20000
weighted avg       1.00      1.00      1.00     20000
```

XGBoost dengan parameter terbaik sepertinya bekerja lebih baik. Random Forest Classifier mungkin dipengaruhi oleh ketidakseimbangan data target. Data cukup tidak seimbang. Masalah ketidakseimbangan ini bisa diatasi dengan meresampling data menggunakan metode SMOTE.

3.3 Dealing with Unbalanced Data

3.3.1. Balancing Data via Oversampling with SMOTE

```
#Running RainForest Model with resampled data
run_model(rf_model, X_train, y_train,X_test, y_test)
```

```
Model Scores
------------------------
Training Accuracy: 98.99%
Test Accuracy:     98.94%
-----------------------------------------------------
```

```
#Running XGBoost Model with resampled data
run_model(xgb_model, X_train, y_train,X_test, y_test)

```

```
Model Scores
------------------------
Training Accuracy: 99.63%
Test Accuracy:     99.62%
--------------------------------------------------------
```

Setelah dilakukan oversampling pada data, kinerja kedua model meningkat secara signifikan memiliki akurasi hampir 100%. Kemungkinan besar itu disebabkan oleh data sintetis yang dihasilkan oleh SMOTE. Karena jumlah instans untuk kelas fraud sangat sedikit, SMOTE menciptakan terlalu banyak data yang sama. Model ini mengingat pola itu dan memberikan hasil yang sempurna pada set uji. Hal ini terjadi karena sangat mungkin bahwa titik data yang sama juga tersedia di set uji.

3.3.2. Subsampling Data with Original Dataset

Dataset yang digunakan selanjutnya tetap disampling namun dengan nilai yang lebih besar yaitu 50.000 data, meskipun belum mencapai 50%, tetapi sudah cukup baik untuk melatih model.

```
# Running RandomForestClassifier with best parameters
run_model(rf_model, X_train2, y_train2,X_test2, y_test2)
```

```
Model Scores
------------------------
Training Accuracy: 93.74%
Test Accuracy:     93.78%
--------------------------------------------------------
```

```
# Running XGBClassifier with best parameters
run_model(xgb_model, X_train2, y_train2,X_test2, y_test2)
```

```
Model Scores
------------------------
Training Accuracy: 99.48%
Test Accuracy:     99.0%
--------------------------------------------------------
```

Hasilnya terlihat jauh lebih realistis meskipun dilakukan oversampling dengan SMOTE. Namun, model XGBoost tampaknya bekerja jauh lebih baik dalam semua dataset yang telah diuji. Meskipun proporsi data kita sudah lebih baik, kita masih memiliki data yang tidak seimbang. Kita dapat melakukan oversampling pada data baru ini untuk mendapatkan lebih banyak data fraud.

3.3.3. Performing SMOTE on the New Data

```
# Running RandomForestClassifier with best parameters
run_model(rf_model, X_train2, y_train2,X_test2, y_test2)
```

```
Model Scores
------------------------
Training Accuracy: 93.77%
Test Accuracy:     93.28%
------------------------------------------------------
```

```
# Running XGBClassifier with best parameters
run_model(xgb_model, X_train2, y_train2,X_test2, y_test2)

```

```
Model Scores
------------------------
Training Accuracy: 99.49%
Test Accuracy:     99.04%
------------------------------------------------------
```

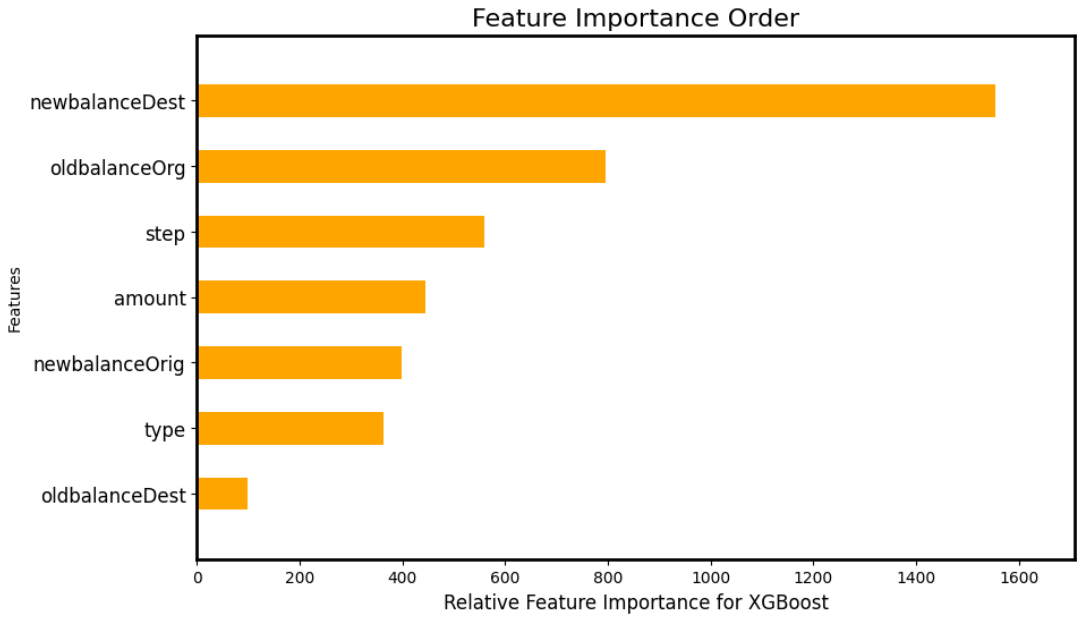
XGBoost mengalami peningkatan sedikit lebih lanjut, tetapi akurasi Random Forest mengalami penurunan dengan data baru ini. Dapat disimpulkan bahwa Random Forest tidak dapat menangani terlalu banyak data yang berulang-ulang demi keseimbangan.

4. Machine Learning Pipeline

Pipeline adalah alat yang sangat berguna untuk menulis kode yang bersih dan mudah dikelola dalam machine learning. Membuat model melibatkan banyak langkah seperti membersihkan data, mentransformasinya, seleksi fitur, dan kemudian menjalankan algoritma machine learning. Dengan menggunakan pipeline, kita dapat melakukan semua langkah ini dalam satu langkah.

Setelah dilakukan pipeline, diperoleh bahwa XGBoost Classifier merupakan model pengujian yang terbaik dengan akurasi hingga 99%.

5. Feature Importance



Setiap model memberikan tingkat fitur yang berbeda. Namun, newbalanceDest dan oldbalanceOrg adalah indikator utama yang merupakan fitur paling berpengaruh.

6. Conclusion

Kinerja model telah meningkat setelah lima iterasi dan akhirnya mencapai:
- Akurasi 99% dengan XGBoost Classifier dan Data Seimbang.

- Fitur yang paling berpengaruh adalah saldo pengirim sebelum transaksi (oldBalanceOrig) dan saldo penerima setelah transaksi (newBalanceDest).

EDA (Exploratory Data Analysis):
- Meskipun transaksi yang aman mulai melambat pada hari ke-3, ke-4 dan setelah hari ke-16 dalam sebulan, transaksi penipuan tetap berlangsung dengan kecepatan yang stabil. Terutama setelah minggu ketiga sampai akhir bulan, jumlah transaksi yang aman jauh lebih sedikit, tetapi jumlah transaksi penipuan tidak berkurang sama sekali.

- Proporsi penipuan terhadap semua transaksi adalah 0,01%, sementara proporsi jumlah penipuan terhadap semua jumlah adalah 0,1%.

- Ada jenis pola dalam jumlah transaksi setiap 24 jam. Namun, transaksi penipuan tidak menunjukkan pola yang signifikan. Penipuan terjadi hampir setiap jam dengan frekuensi yang sama.

- Ada lebih banyak transaksi penipuan dengan jumlah rendah dan lebih sedikit dengan jumlah tinggi. Distribusi ini tidak berubah banyak.
- Transaksi penipuan terjadi dalam rentang yang cukup luas, mulai dari 64 dolar hingga 10 juta dolar. Sebagian besar transaksi penipuan memiliki jumlah yang lebih rendah. Namun, pada 1 juta dolar, ada peningkatan menarik yang mirip dengan transaksi aman.

- Terdapat 16 kasus penipuan palsu dengan jumlah '0'.

- Aktivitas penipuan hanya terjadi pada transaksi TRANSFER dan CASH_OUT. Penggunaan DEBIT sangat aman.