Penting

- Pastikan Anda melakukan Run All sebelum mengirimkan submission untuk memastikan seluruh cell berjalan dengan baik.
- Hapus simbol pagar (#) jika Anda menerapkan kriteria tambahan
- Biarkan simbol pagar (#) jika Anda tidak menerapkan kriteria tambahan

1. Import Library

Pada tahap ini, Anda perlu mengimpor beberapa pustaka (library) Python yang dibutuhkan untuk analisis data dan pembangunan model machine learning.

```
#Type your code here
# Import pustaka untuk pemrosesan data
import pandas as pd #used
import xgboost as xgb #used

# Import pustaka untuk pembagian data
from sklearn.model_selection import train_test_split #used

# Import pustaka untuk model klasifikasi (Skilled dan Advanced)
from sklearn.tree import DecisionTreeClassifier #used
from sklearn.ensemble import RandomForestClassifier #used
from xgboost import XGBClassifier #used

# Import pustaka untuk evaluasi model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score #used
from sklearn.metrics import confusion_matrix, classification_report #used

# Import pustaka untuk hyperparameter tuning (Advanced)
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV #used

# Import pustaka untuk visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# Import pustaka untuk menyimpan model
import joblib #used
```

2. Memuat Dataset dari Hasil Clustering

Memuat dataset hasil clustering dari file CSV ke dalam variabel DataFrame.

```
# Gunakan dataset hasil clustering yang memiliki fitur Target
# Silakan gunakan dataset data_clustering jika tidak menerapkan Interpretasi Hasil Clustering [Advanced]
# Silakan gunakan dataset data_clustering_inverse jika menerapkan Interpretasi Hasil Clustering [Advanced]
df = pd.read_csv('data_clustering.csv')
```

```
# Tampilkan 5 baris pertama dengan function head.
print("5 Baris Pertama Dataset:")
print(df.head())
```

```
5 Baris Pertama Dataset:
   TransactionAmount      TransactionDate  CustomerAge  TransactionDuration  \
0          -0.970546  2023-04-11 16:29:14     1.419862            -0.548393
1           0.268963  2023-06-27 16:44:19     1.307715             0.307960
2          -0.586526  2023-07-10 18:16:08    -1.439874            -0.905207
3          -0.387294  2023-05-05 16:32:11    -1.047361            -1.347656
4          -0.703375  2023-04-03 17:15:01    -1.495947             0.750409

   LoginAttempts  AccountBalance PreviousTransactionDate  \
0      -0.204629       -0.000315     2024-11-04 08:08:08
1      -0.204629        2.218381     2024-11-04 08:09:35
2      -0.204629       -1.024091     2024-11-04 08:07:04
3      -0.204629        0.886694     2024-11-04 08:09:06
4      -0.204629       -1.111505     2024-11-04 08:06:36

   TransactionType_Debit  Location_Atlanta  Location_Austin  ...  \
0                   True             False            False  ...
1                   True             False            False  ...
2                   True             False            False  ...
3                   True             False            False  ...
4                   True             False            False  ...
```

```
   CustomerOccupation_Retired  CustomerOccupation_Student  Channel_Branch  \
0                       False                       False           False
1                       False                       False           False
2                       False                        True           False
3                       False                        True           False
4                       False                        True           False

   Channel_Online  CustomerAge_bin  TransactionAmount_bin  \
0           False            Muda                    NaN
1           False            Muda                  Kecil
2            True             NaN                    NaN
3            True             NaN                    NaN
4           False             NaN                    NaN

   CustomerAge_bin_encoded  TransactionAmount_bin_encoded  Cluster  Target
0                        0                              1        4       5
1                        0                              0        3       5
2                        1                              1        5       6
3                        1                              1        3       4
4                        1                              1        5       1

[5 rows x 61 columns]
```

3. Data Splitting

Tahap Data Splitting bertujuan untuk memisahkan dataset menjadi dua bagian: data latih (training set) dan data uji (test set).

```
# Menggunakan train_test_split() untuk melakukan pembagian dataset.
# Memisahkan fitur (X) dan label (y)
# Menghapus kolom yang tidak relevan
columns_to_drop = ['Target', 'TransactionDate', 'PreviousTransactionDate',
                   'Cluster', 'CustomerAge_bin', 'TransactionAmount_bin',
                   'CustomerAge_bin_encoded', 'TransactionAmount_bin_encoded']
X = df.drop(columns=columns_to_drop)
y = df['Target']

# Memisahkan dataset menjadi data latih dan uji (80:20, stratify=y)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    stratify=y)

# Menampilkan ukuran data latih dan uji
print("Ukuran Data Latih (X_train):", X_train.shape)
print("Ukuran Data Uji (X_test):", X_test.shape)
print("Ukuran Label Latih (y_train):", y_train.shape)
print("Ukuran Label Uji (y_test):", y_test.shape)

# (Opsional) Menampilkan distribusi kelas di data latih dan uji
print("\nDistribusi Kelas di Data Latih (y_train):")
print(y_train.value_counts())
print("\nDistribusi Kelas di Data Uji (y_test):")
print(y_test.value_counts())
```

```
Ukuran Data Latih (X_train): (1782, 53)
Ukuran Data Uji (X_test): (446, 53)
Ukuran Label Latih (y_train): (1782,)
Ukuran Label Uji (y_test): (446,)

Distribusi Kelas di Data Latih (y_train):
Target
4    415
5    367
2    362
6    256
1    252
3    130
Name: count, dtype: int64

Distribusi Kelas di Data Uji (y_test):
Target
4    104
5     92
2     90
6     64
1     63
3     33
Name: count, dtype: int64
```

4. Membangun Model Klasifikasi

Setelah memilih algoritma klasifikasi yang sesuai, langkah selanjutnya adalah melatih model menggunakan data latih.

Berikut adalah rekomendasi tahapannya.

1. Menggunakan algoritma klasifikasi yaitu Decision Tree.
2. Latih model menggunakan data yang sudah dipisah.

```
# Buatlah model klasifikasi menggunakan Decision Tree
model = DecisionTreeClassifier(random_state=42)

# Melatih model menggunakan data latih
model.fit(X_train, y_train)

print("Model Decision Tree telah selesai dilatih")
```

```
Model Decision Tree telah selesai dilatih
```

```
# Menyimpan Model
# import joblib
# joblib.dump(model, 'decision_tree_model.h5')
joblib.dump(model, 'decision_tree_model.h5')
print("Model Decision Tree telah dilatih dan disimpan sebagai 'decision_tree_model.h5'")
```

```
Model Decision Tree telah dilatih dan disimpan sebagai 'decision_tree_model.h5'
```

5. Memenuhi Kriteria Skilled dan Advanced dalam Membangun Model Klasifikasi

Biarkan kosong jika tidak menerapkan kriteria skilled atau advanced

```
# Melatih model menggunakan algoritma klasifikasi selain Decision Tree.
# Membuat dan melatih model Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Membuat dan melatih model XGBoost
# Menyesuaikan label y_train dan y_test agar mulai dari 0 (XGBoost memerlukan label 0-5, bukan 1-6)
y_train_xgb = y_train - 1
y_test_xgb = y_test - 1
xgb_model = XGBClassifier(random_state=42, eval_metric='mlogloss')
xgb_model.fit(X_train, y_train_xgb)

print("Model Random Forest telah dilatih")
print("Model XGBoost telah dilatih")
```

```
Model Random Forest telah dilatih
Model XGBoost telah dilatih
```

```
# Menampilkan hasil evaluasi akurasi, presisi, recall, dan F1-Score pada seluruh algoritma yang sudah dibuat.
# Memuat model Decision Tree yang sudah dilatih
dt_model = joblib.load('decision_tree_model.h5')

models = {
    'Decision Tree': dt_model,
    'Random Forest': rf_model,
    'XGBoost': xgb_model
}

results = []
for name, model in models.items():
    # Prediksi pada data uji
    if name == 'XGBoost':
        y_pred = model.predict(X_test) + 1  # Mengembalikan label ke 1-6
    else:
        y_pred = model.predict(X_test)

    # Menghitung metrik evaluasi
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
    precision = report['weighted avg']['precision']
    recall = report['weighted avg']['recall']
    f1 = report['weighted avg']['f1-score']

    # Menyimpan hasil
    results.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
```

```
        'Recall': recall,
        'F1-Score': f1
    })

# Menampilkan hasil evaluasi dalam bentuk tabel
results_df = pd.DataFrame(results)
print("\nHasil Evaluasi Model:")
print(results_df)
```

```
Hasil Evaluasi Model:
           Model  Accuracy  Precision    Recall  F1-Score
0  Decision Tree  0.912556   0.916342  0.912556  0.913125
1  Random Forest  0.926009   0.930245  0.926009  0.925699
2        XGBoost  0.973094   0.973958  0.973094  0.973195
```

```
# Menyimpan Model Selain Decision Tree
# Model ini bisa lebih dari satu
# import joblib
# joblib.dump(___, 'explore_<Nama Algoritma>_classification.h5')
# Menyimpan model tambahan
joblib.dump(rf_model, 'RandomForest_classification.h5')
print("Model Random Forest telah dilatih dan disimpan sebagai 'RandomForest_classification.h5'")
joblib.dump(xgb_model, 'XGBoost_classification.h5')
print("Model XGBoost telah dilatih dan disimpan sebagai 'XGBoost_classification.h5'")
```

```
Model Random Forest telah dilatih dan disimpan sebagai 'RandomForest_classification.h5'
Model XGBoost telah dilatih dan disimpan sebagai 'XGBoost_classification.h5'
```

Hyperparameter Tuning Model

Pilih salah satu algoritma yang ingin Anda tuning

```
# Lakukan Hyperparameter Tuning dan Latih ulang.
# Menentukan grid parameter untuk Random Forest
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

# Membuat model Random Forest untuk tuning
rf_tuned = RandomForestClassifier(random_state=42)

# Melakukan GridSearchCV
grid_search = GridSearchCV(estimator=rf_tuned, param_grid=param_grid,
                           cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Mendapatkan parameter terbaik
best_params = grid_search.best_params_
print("Parameter Terbaik dari GridSearchCV:", best_params)

# Melatih ulang model dengan parameter terbaik
rf_best = RandomForestClassifier(**best_params, random_state=42)
rf_best.fit(X_train, y_train)
```

```
Parameter Terbaik dari GridSearchCV: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 200}
RandomForestClassifier
RandomForestClassifier(max_depth=20, n_estimators=200, random_state=42)
```

```
# Menampilkan hasil evaluasi akurasi, presisi, recall, dan F1-Score pada algoritma yang sudah dituning.
# Evaluasi model yang dituning
y_pred_tuned = rf_best.predict(X_test)
accuracy_tuned = accuracy_score(y_test, y_pred_tuned)
report_tuned = classification_report(y_test, y_pred_tuned, output_dict=True, zero_division=0)
precision_tuned = report_tuned['weighted avg']['precision']
recall_tuned = report_tuned['weighted avg']['recall']
f1_tuned = report_tuned['weighted avg']['f1-score']

# Menampilkan hasil evaluasi
print("\nHasil Evaluasi Model Random Forest (Tuned):")
print(f"Akurasi: {accuracy_tuned:.4f}")
print(f"Presisi: {precision_tuned:.4f}")
print(f"Recall: {recall_tuned:.4f}")
print(f"F1-Score: {f1_tuned:.4f}")
```

```
Hasil Evaluasi Model Random Forest (Tuned):
Akurasi: 0.9350
Presisi: 0.9390
Recall: 0.9350
F1-Score: 0.9346
```

```
# Menyimpan Model hasil tuning
# import joblib
# joblib.dump(model_dt, 'tuning_classification.h5')
joblib.dump(rf_best, 'tuning_classification.h5')
print("\nModel Random Forest yang dituning telah disimpan sebagai 'tuning_classification.h5'")
```

```
Model Random Forest yang dituning telah disimpan sebagai 'tuning_classification.h5'
```