EDA dan Machine Learning untuk Memprediksi Nilai Order

Kelompok 6 Ilmu Komputer Kelas C2

- Naufal Fakhri Al-Najieb (2309648)
- Abdurrahman Rauf Budiman (2301102)
- Haniel Septian Putra Alren (2310978)
- Muhammad Radhi Maulana (2311119)
- Yoga Ilham Prasetio (2304539)

In [2]:
```python
# Import semua Library

import pandas as pd
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import log_loss
from sklearn.preprocessing import StandardScaler
import numpy as np
```

In [3]:
```python
# Praproses EDA
# Load data trainingnya
train_data = pd.read_csv('transact_train.txt', delimiter='|')  # Sesuaikan delimiter jika diperlukan

# Praproses data
train_data.replace('?', np.nan, inplace=True)
train_data.fillna(0, inplace=True)

# mengkodekan target variabel
train_data['order'] = train_data['order'].apply(lambda x: 1 if x == 'y' else 0)

# Pisahkan fitur dan target
X = train_data.drop(columns=['order', 'sessionNo'])
y = train_data['order']

# Konvert non-numeric columns to numeric
X = X.apply(pd.to_numeric, errors='coerce').fillna(0)

# skala fitur
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [4]:
```python
# Kita pake dua klasifikasi, yaitu RandomForest dan XGBoost

# Data di split buat training dan validasi
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=69)

# Latih Klasifikasi RandomForest
model = RandomForestClassifier(n_estimators=100, max_depth=30, random_state=69)
model.fit(X_train, y_train)

# Latih Klasifikasi XGBoost
model2 = XGBClassifier(n_estimators=100, max_depth=30, learning_rate=0.1, random_state=69)
model2.fit(X_train, y_train)
```

Out[4]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=30, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [5]:
```python
# Evaluasi model pada validasi data

# 1. Predict probabilities for log loss
y_val_pred_proba = model.predict_proba(X_val)[:, 1]  # Dapatkan probabilitas untuk kelas positif (order=1)
log_loss_score = log_loss(y_val, y_val_pred_proba)
print("Validation Log Loss untuk model pertama:", log_loss_score)

y_val_pred_proba_2 = model2.predict_proba(X_val)[:, 1]  # Dapatkan probabilitas untuk kelas positif (order=1)
log_loss_score_2 = log_loss(y_val, y_val_pred_proba_2)
print("Validation Log Loss untuk model kedua:", log_loss_score_2)

# Load data klasifikasi (data pengujian)
test_data = pd.read_csv('transact_class.txt', delimiter='|')
session_numbers = test_data['sessionNo']

# Preproses data pengujian serupa dengan data latih
test_data.replace('?', np.nan, inplace=True)
test_data.fillna(0, inplace=True)  # Gantikan NaN dengan nilai 0
X_test = test_data.drop(columns=['sessionNo'])

# Konversi kolom ke tipe numerik dan isi NaN dengan 0
X_test = X_test.apply(pd.to_numeric, errors='coerce').fillna(0)
X_test = scaler.transform(X_test)  # Standarisasi data pengujian

# Prediksi probabilitas untuk data pengujian
predictions_proba = model.predict_proba(X_test)[:, 1]  # Dapatkan probabilitas untuk 'order=1' dari model pertama
predictions_proba_2 = model2.predict_proba(X_test)[:, 1]  # Dapatkan probabilitas untuk 'order=1' dari model kedua
```

```
Validation Log Loss untuk model pertama: 0.19816834736640754
Validation Log Loss untuk model kedua: 0.15194893306762136
```

In [6]:
```python
# Simpan prediksi untuk model pertama
temp = pd.DataFrame({
    'sessionNo': session_numbers,  # Nomor sesi yang ada di data pengujian
    'prediction': predictions_proba  # Probabilitas prediksi dari model pertama
})
temp = temp.groupby('sessionNo', as_index=False).mean()  # Mengelompokkan berdasarkan sessionNo dan menghitung rata-rata prediksi

# Simpan prediksi untuk model kedua
temp2 = pd.DataFrame({
    'sessionNo': session_numbers,  # Nomor sesi yang ada di data pengujian
    'prediction': predictions_proba_2  # Probabilitas prediksi dari model kedua
})
temp2 = temp2.groupby('sessionNo', as_index=False).mean()  # Mengelompokkan berdasarkan sessionNo dan menghitung rata-rata prediksi

# Ambil prediksi hasil rata-rata dari kedua model
predictions_proba = temp['prediction']  # Prediksi dari model pertama
predictions_proba_2 = temp2['prediction']  # Prediksi dari model kedua

# Memuat data final untuk perbandingan
final_prediction = pd.read_csv('DMC 2013_realclass_task1/realclass_t1.txt', delimiter='|')  # Memuat data sebenarnya
final_prediction = final_prediction['prediction']  # Ambil kolom prediksi asli untuk perhitungan log loss

# Menghitung Log Loss untuk model pertama
log_loss_score = log_loss(final_prediction.apply(pd.to_numeric, errors='coerce'), predictions_proba)

# Menghitung Log Loss untuk model kedua
log_loss_score_2 = log_loss(final_prediction.apply(pd.to_numeric, errors='coerce'), predictions_proba_2)

# Membandingkan kedua model dan memilih yang terbaik (berdasarkan nilai log loss lebih kecil)
if(log_loss_score_2 < log_loss_score):
    predictions_proba = predictions_proba_2  # Jika model kedua lebih baik, pilih prediksi dari model kedua

# Menghitung Log Loss setelah memilih model terbaik
log_loss_score_final = log_loss(final_prediction.apply(pd.to_numeric, errors='coerce'), predictions_proba)
print("Final Log Loss : ", log_loss_score_final)  # Menampilkan nilai Log Loss final

# Menyimpan hasil prediksi yang sudah dipilih dalam format CSV
output = pd.DataFrame({
    'sessionNo': session_numbers,  # Nomor sesi
    'prediction': predictions_proba  # Prediksi yang dipilih
})
```

```
# Menyimpan prediksi ke dalam file CSV dengan pemisah '|' dan tanpa menyertakan indeks
output.to_csv('teamname_task1.txt', sep='|', index=False, header=True)
```

```
Final Log Loss :  0.5242076625430745
```