Kelompok 7

Anggota Kelompok:

1. Faiz Bayu Erlangga (2311231)
2. Marco Henrik Abineno (2301093)
3. Muhammad Alfi Fariz (2311174)
4. Naufal Dzaki Ibrahim (2309815)
5. Qalam Noer Fazrian (2304746)

Kelas: 3KOMC2

Mata Kuliah: IK505 - Data Mining and Warehouse

Dosen Pengampu: Dr. Yudi Wibisono, S.T., M.T.

Pendahuluan

Dalam dataframe ini termuat informasi tentang sesi interaksi pelanggan dengan platform e-commerce, mencatat berbagai detail tentang aktivitas pengguna, produk yang diklik, produk yang dimasukkan ke keranjang, hingga hasil akhir dari sesi tersebut (pembelian atau tidak). Setiap kolom dalam DataFrame mewakili aspek tertentu dari perilaku dan karakteristik pelanggan selama sesi berlangsung, mulai dari waktu dan durasi interaksi, jumlah produk yang diklik, rentang harga produk, hingga data demografis pelanggan seperti usia, skor evaluasi dari sudut pandang toko, status online, dan informasi lainnya.

DataFrame ini memiliki struktur yang beragam, dengan beberapa kolom yang berisi nilai kontinu, nilai diskrit, dan string, serta beberapa kolom yang mungkin berisi nilai kosong atau tidak lengkap. Nilai kosong ini munoui karena beberapa informasi mungkin tidak tersedia dalam setiap sesi pelanggan. Untuk analisis lebih lanjut, diperlukan penanganan yang tepat terhadap nilai-nilai yang hilang agar tidak mempengaruhi kualitas hasil analisis.

Pada tugas kali ini kami akan membuat model machine learning yang cocok untuk memprediksi apakah pelanggan akan melakukan order atau tidak

Import Library

In [884]:
```
# Standar
%matplotlib inline
# numeric
import numpy as np
# analisis
import pandas as pd
# plot
import matplotlib.pyplot as plt
# visualisasi
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from tabulate import tabulate

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from tabulate import tabulate
```

Proses EDA

Proses ini dilakukan untuk memfilter data-data yang tidak lengkap atau noise, agar dalam memprediksi model machine learning menjadi efisien dan mudah.

Import Data dari .txt

In [884]:
```
#import data
df = pd.read_csv("DMC 2013_task and realclass/transact_train.txt", delimiter="|")
```

In [886]:
```
df.duplicated().value_counts()
```

Out[886]:
```
False    429013
Name: count, dtype: int64
```

In [888]:
```
# Menghitung median dari cMinPrice (mengabaikan nilai '?')
median_cMinPrice = df_filtered[df_filtered['cMinPrice'] != '?']['cMinPrice'].astype(float).median()

# Mengganti nilai '?' pada cMinPrice dengan median
df_filtered['cMinPrice'] = df_filtered['cMinPrice'].replace('?', median_cMinPrice)

# Menghitung median untuk setiap kolom yang memiliki nilai '?'
median_cMaxPrice = df_filtered[df_filtered['cMaxPrice'] != '?']['cMaxPrice'].astype(float).median()
median_cMaxPrice = df_filtered[df_filtered['cMaxPrice'] != '?']['cMaxPrice'].astype(float).median()
median_bMinPrice = df_filtered[df_filtered['bMinPrice'] != '?']['bMinPrice'].astype(float).median()
median_bMaxPrice = df_filtered[df_filtered['bMaxPrice'] != '?']['bMaxPrice'].astype(float).median()

# Mengganti nilai '?' pada setiap kolom dengan median yang sesuai
df_filtered['cMinPrice'] = df_filtered['cMinPrice'].replace('?', median_cMinPrice)
df_filtered['cMaxPrice'] = df_filtered['cMaxPrice'].replace('?', median_cMaxPrice)
df_filtered['bMinPrice'] = df_filtered['bMinPrice'].replace('?', median_bMinPrice)
df_filtered['bMaxPrice'] = df_filtered['bMaxPrice'].replace('?', median_bMaxPrice)
```

Kami menghitung median dari kolom cMinPrice, cMaxPrice, bMinPrice, dan bMaxPrice dengan mengabaikan nilai '?'. Setelah itu, setiap nilai '?' pada kolom-kolom tersebut diganti dengan median yang sesuai. Kami melakukan hal ini untuk memastikan semua kolom numerik tersebut tidak lagi memiliki nilai '?', sehingga data lebih stabil untuk di masukkan ke model machine learning kami.

In [889]:
```
# Menghitung mean dari cMinPrice dan cMaxPrice, mengabaikan nilai '?'
df_filtered['cMinPrice'] = df_filtered['cMinPrice'].replace('?', float('nan')).astype(float)
df_filtered['cMaxPrice'] = df_filtered['cMaxPrice'].replace('?', float('nan')).astype(float)

# Menghitung rata-rata cMinPrice dan cMaxPrice
df_filtered['mean_cMinMax'] = (df_filtered['cMinPrice'] + df_filtered['cMaxPrice']) / 2

# Mengganti nilai '?' pada cSumPrice dengan mean_cMinMax * cCount
df_filtered['cSumPrice'] = df_filtered.apply(
    lambda row: row['mean_cMinMax'] * row['cCount'] if row['cSumPrice'] == '?' else row['cSumPrice'],
    axis=1
)

# Menghapus kolom sementara 'mean_cMinMax'
df_filtered.drop('mean_cMinMax', axis=1, inplace=True)

# Menghitung mean dari bMinPrice dan bMaxPrice, mengabaikan nilai '?'
df_filtered['bMinPrice'] = df_filtered['bMinPrice'].replace('?', float('nan')).astype(float)
df_filtered['bMaxPrice'] = df_filtered['bMaxPrice'].replace('?', float('nan')).astype(float)

# Menghitung rata-rata bMinPrice dan bMaxPrice
df_filtered['mean_bMinMax'] = (df_filtered['bMinPrice'] + df_filtered['bMaxPrice']) / 2

# Mengganti nilai '?' pada bSumPrice dengan mean_bMinMax * bCount
df_filtered['bSumPrice'] = df_filtered.apply(
    lambda row: row['mean_bMinMax'] * row['bCount'] if row['bSumPrice'] == '?' else row['bSumPrice'],
    axis=1
)

# Menghapus kolom sementara 'mean_bMinMax'
df_filtered.drop('mean_bMinMax', axis=1, inplace=True)
```

Kami menghitung mean dari kolom cMinPrice, cMaxPrice, bMinPrice, dan bMaxPrice dengan mengabaikan nilai '?'. Setelah itu, setiap nilai '?' pada kolom-kolom cSumPrice diganti dengan mean yang sesuai. Kami melakukan hal ini untuk memastikan semua kolom numerik tersebut tidak lagi memiliki nilai '?', sehingga data lebih stabil untuk di masukkan ke model machine learning kami.

In [892]:
```
# Pastikan bStep adalah integer
df_filtered['bStep'] = df_filtered['bStep'].replace('?', float('nan')).astype(float)

# Menghitung modus untuk setiap sesi (mengabaikan NaN)
modus_per_sesi = df_filtered.groupby('sessionNo')['bStep'].agg(lambda x: x.mode()[0] if not x.mode().empty else 0)

# Mengganti nilai NaN pada bStep dengan modus per sesi atau 0 jika semuanya adalah NaN
df_filtered['bStep'] = df_filtered.apply(
    lambda row: modus_per_sesi[row['sessionNo']] if pd.isna(row['bStep']) else row['bStep'],
    axis=1
)

# Jika semua nilai dalam sesi adalah '?' maka ganti dengan 0
df_filtered['bStep'] = df_filtered.apply(
    lambda row: 0 if pd.isna(row['bStep']) else row['bStep'],
    axis=1
)

# Pastikan bStep diubah menjadi integer setelah penggantian
df_filtered['bStep'] = df_filtered['bStep'].astype(int)
```

Kode ini bertujuan untuk menangani nilai yang hilang pada kolom bStep dalam df_filtered dengan cara yang sistematis. Pertama, kolom bStep yang mengandung nilai '?' diganti dengan NaN dan diubah menjadi tipe data float untuk memudahkan pengolahan nilai kosong. Selanjutnya, untuk setiap sesi yang berbeda (sessionNo), kode menghitung modus dari nilai-nilai bStep, yaitu nilai yang paling sering muncul. Jika tidak ada modus yang ditemukan (misalnya, semua nilai bStep pada sesi tersebut adalah NaN), maka modus diisi dengan nilai 0. Setelah itu, kode mengganti semua nilai NaN pada kolom bStep dengan modus yang telah dihitung berdasarkan sesi masing-masing. Sebagai langkah terakhir, kode memastikan bahwa kolom bStep diubah kembali menjadi tipe data integer setelah penggantian nilai, sehingga data menjadi lebih konsisten untuk analisis lebih lanjut.

In [894]:
```
df_filtered.to_csv('data_model.csv', index=False)
```

In [901]:
```
# Ganti nilai '?' dengan 'y' menggunakan apply
df_filtered['onlineStatus_y_True_True'] = df_filtered['onlineStatus_y_True_True'].apply(lambda x: 'y' if x == '?' else x)
```

In [903]:
```
# Ganti nilai '?' dengan NaN pada kolom accountLifetime
df_filtered['accountLifetime'] = df_filtered['accountLifetime'].replace('?', float('nan')).astype(float)

# Hitung median dari kolom accountLifetime, mengabaikan NaN
median_accountLifetime = df_filtered['accountLifetime'].median()

# Ganti NaN dengan median yang dihitung
df_filtered['accountLifetime'] = df_filtered['accountLifetime'].fillna(median_accountLifetime)

# Jika ingin memastikan tipe data tetap numerik (misalnya int atau float), gunakan:
df_filtered['accountLifetime'] = df_filtered['accountLifetime'].astype(int)  # Jika ingin menjadi integer
```

In [905]:
```
# Ganti nilai '?' dengan NaN pada kolom age dan pastikan kolom bertipe numerik (int)
df_filtered['age'] = pd.to_numeric(df_filtered['age'].replace('?', float('nan')), errors='coerce')

# Hitung modus (nilai yang paling sering muncul) dari kolom age
modus_age = df_filtered['age'].mode()[0]

# Ganti NaN dengan modus yang dihitung, dan pastikan hasilnya bertipe integer
df_filtered['age'] = df_filtered['age'].fillna(modus_age).astype(int)
```

In [907]:
```
# Ganti nilai '?' dengan NaN pada kolom address
df_filtered['address'] = df_filtered['address'].replace('?', float('nan'))

# Hitung modus (nilai yang paling sering muncul) dari kolom address
modus_address = df_filtered['address'].mode()[0]

# Ganti NaN dengan modus yang dihitung
df_filtered['address'] = df_filtered['address'].fillna(modus_address)

# Konversi kolom address menjadi integer
df_filtered['address'] = df_filtered['address'].astype(int)
```

In [909]:
```
# Ganti nilai '?' dengan NaN pada kolom lastOrder
df_filtered['lastOrder'] = df_filtered['lastOrder'].replace('?', float('nan'))

# Hitung modus (nilai yang paling sering muncul) dari kolom lastOrder
modus_lastOrder = df_filtered['lastOrder'].mode()[0]

# Ganti NaN dengan modus yang dihitung
df_filtered['lastOrder'] = df_filtered['lastOrder'].fillna(modus_lastOrder)

# Pastikan kolom lastOrder tetap bertipe integer atau sesuai tipe yang diinginkan
df_filtered['lastOrder'] = df_filtered['lastOrder'].astype(int)  # Jika ingin bertipe integer
# Atau jika kolom lastOrder seharusnya string
# df_filtered['lastOrder'] = df_filtered['lastOrder'].astype(str)  # Jika ingin bertipe string
```

In [911]:
```
# Mengubah kolom target ('order') menjadi numerik (1 untuk 'y' dan 0 untuk 'n')
df_filtered['order'] = df_filtered['order'].apply(lambda x: 1 if x == 'y' else 0)

# Menyeleksi hanya atribut yang akan digunakan dalam model
selected_features = [
    'sessionNo', 'startHour', 'startWeekday', 'duration', 'cCount', 'cMinPrice', 'cMaxPrice', 'cSumPrice',
    'bCount', 'bMinPrice', 'bMaxPrice', 'bSumPrice', 'bStep', 'onlineStatus_y_True_True', 'customerNo', 'accountLifetime',
    'paymentA', 'age', 'address', 'lastOrder'
]
df_filtered = df_filtered[selected_features + ['order']]

# Label Encoding untuk kolom customerID dengan banyak nilai unik
label_encoder = LabelEncoder()
df_filtered['customerNo'] = label_encoder.fit_transform(df_filtered['customerNo'])

# One-hot encoding pada kolom onlineStatus dengan jumlah kategori terbatas
df_filtered = pd.get_dummies(df_filtered, columns=['onlineStatus_y_True_True'], drop_first=True)

# Menyiapkan fitur dan target
X = df_filtered.drop(columns=['sessionNo', 'order'])  # Menghilangkan 'sessionNo' agar tidak mempengaruhi model
y = df_filtered['order']

# Membagi data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Menyimpan kolom 'sessionNo' dari data pengujian untuk hasil akhir
session_no_test = df_filtered.loc[X_test.index, 'sessionNo']

# Standarisasi fitur
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Inisialisasi dan pelatihan model Random Forest
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Prediksi dengan model
y_pred = model.predict(X_test)

# Membuat DataFrame hasil dengan kolom 'sessionNo' dan prediksi 'order'
df_result = pd.DataFrame({
    'sessionNo': session_no_test.values,
    'order': ['y' if pred == 1 else 'n' for pred in y_pred]
})

# Tampilkan hasil dalam bentuk tabel
print("\nHasil Prediksi dalam Bentuk Tabel:")
print(tabulate(df_result.head(20), headers='keys', tablefmt='pretty'))

# Evaluasi performa model
print("\nAkurasi:", accuracy_score(y_test, y_pred))
print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))
print("\nMatriks Kebingungan:\n", confusion_matrix(y_test, y_pred))
```

```
Hasil Prediksi dalam Bentuk Tabel:
+----+------------+---------+
|    | sessionNo  | order   |
+====+============+=========+
|  0 |   17084    |  n   |
|  1 |   41035    |  n   |
|  2 |    6809    |  n   |
|  3 |    5811    |  n   |
|  4 |   33030    |  n   |
|  5 |   46729    |  n   |
|  6 |   43121    |  n   |
|  7 |   27342    |  n   |
|  8 |   12585    |  n   |
|  9 |   37573    |  n   |
| 10 |   11292    |  n   |
| 11 |   18257    |  n   |
| 12 |   49979    |  n   |
| 13 |   15869    |  n   |
| 14 |   20898    |  n   |
| 15 |   25787    |  n   |
| 16 |   12819    |  n   |
| 17 |   40012    |  n   |
| 18 |   26155    |  n   |
| 19 |   47735    |  n   |
+----+------------+---------+

Akurasi: 1.0

Laporan Klasifikasi:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    122599

    accuracy                           1.00    122599
   macro avg       1.00      1.00      1.00    122599
weighted avg       1.00      1.00      1.00    122599


Matriks Kebingungan:
 [[122599]]
```

Dalam kode ini, pertama-tama kolom target `order` diubah menjadi format numerik, di mana 'y' dikodekan sebagai 1 dan 'n' sebagai 0, untuk mempermudah pemrosesan oleh model machine learning. Selanjutnya, atribut yang akan digunakan dalam model dipilih, termasuk fitur-fitur seperti `sessionNo`, `startHour`, `startWeekday`, `duration`, dan lain-lain, sementara kolom `order` tetap dipertahankan sebagai target. Kolom `customerID` yang memiliki banyak nilai unik kemudian dilakukan label encoding agar dapat digunakan dalam model, dan kolom `onlineStatus` yang memiliki kategori terbatas diproses dengan one-hot encoding.

Setelah itu, fitur dan target dipisahkan, dengan `sessionNo` yang tidak dimasukkan ke dalam fitur untuk menghindari pengaruhnya terhadap model. Data kemudian dibagi menjadi data latih dan uji dengan proporsi 70% untuk latih dan 30% untuk uji. Kolom `sessionNo` dari data uji disimpan untuk digunakan dalam hasil prediksi akhir. Fitur-fitur selanjutnya distandarisasi menggunakan `StandardScaler` untuk memastikan setiap fitur memiliki skala yang seragam.

Model Random Forest kemudian diinisialisasi dan dilatih dengan data latih. Setelah model terlatih, prediksi dilakukan pada data uji, dan hasil prediksi disusun dalam sebuah DataFrame dengan kolom `sessionNo` dan hasil prediksi `order` yang berisi 'y' untuk order dan 'n' untuk tidak order. Hasil ini ditampilkan dalam bentuk tabel yang rapi menggunakan `tabulate`.

Untuk mengevaluasi performa model, beberapa metrik digunakan, seperti akurasi, laporan klasifikasi, dan matriks kebingungannya, yang membantu untuk mengetahui seberapa baik model dalam memprediksi apakah seorang pelanggan akan melakukan order atau tidak berdasarkan fitur yang diberikan.