Data Analyst Project

Business Decision Research Menggunakan Python

Oleh : Figia Cantikasari Giyana

Pendahuluan dan Rumusan Masalah

Pendahuluan	Rumusan Masalah
DQLab sport center adalah toko yang menjual berbagai kebutuhan olahraga seperti Jaket, Baju, Tas, dan Sepatu. Toko ini mulai berjualan sejak tahun 2013, sehingga sudah memiliki pelanggan tetap sejak lama, dan tetap berusaha untuk mendapatkan pelanggan baru sampai saat ini.	Di awal tahun 2019, terjadi penurunan pelanggan yang membeli kembali ke toko tersebut. Manajer toko mendefinisikan bahwa customer termasuk sudah bukan disebut pelanggan lagi (churn) ketika dia sudah tidak bertransaksi ke tokonya lagi sampai dengan 6 bulan terakhir dari update data terakhir yang tersedia.

Dataset Overview

Data transaksi dari tahun 2013 sampai dengan 2019 dalam bentuk csv dengan nama 'data_retail.csv' dengan jumlah baris 100.000 baris data.

Kolom yang ada pada data tersebut antara lain :

1. no : nomor
2. Row_Num : nomor urut baris
3. Customer_ID : kode unik pelanggan
4. Product : jenis barang yang dibeli
5. First_Transaction : tanggal pertama melakukan transaksi
6. Last_Transaction : tanggal terakhir melakukan transaksi
7. Average_Transaction_Amount : rata-rata jumlah transaksi
8. Count_Transaction : banyak transaksi yang dilakukan

Importing Data

```
import pandas as pd

df = pd.read_csv('data_retail.csv', sep=';')

print(df.head())
print(df.info())
```

```
   no  Row_Num  Customer_ID Product  First_Transaction  Last_Transaction  \
0   1        1        29531   Jaket      1466304274396     1538718482608
1   2        2        29531  Sepatu      1406077331494     1545735761270
2   3        3       141526     Tas      1493349147000     1548322802000
3   4        4       141526   Jaket      1493362372547     1547643603911
4   5        5        37545  Sepatu      1429178498531     1542891221530

   Average_Transaction_Amount  Count_Transaction
0                     1467681                 22
1                     1269337                 41
2                      310915                 30
3                      722632                 27
4                     1775036                 25
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   no                           100000 non-null  int64
 1   Row_Num                      100000 non-null  int64
 2   Customer_ID                  100000 non-null  int64
 3   Product                      100000 non-null  object
 4   First_Transaction            100000 non-null  int64
 5   Last_Transaction             100000 non-null  int64
 6   Average_Transaction_Amount   100000 non-null  int64
 7   Count_Transaction            100000 non-null  int64
dtypes: int64(7), object(1)
memory usage: 6.1+ MB
None
```

Data Cleansing

Setelah melihat data, ternyata :

1. Kolom ‘no’ dan ‘Row_Num’ bisa dihilangkan, karena data pada kolom tersebut tidak diperlukan untuk proses analisa.
2. Kolom ‘First_Transaction’ dan ‘Last_Transaction’ perlu diubah ke tipe data datetime.

```
# Merubah kolom First_Transaction dan Last_Transaction menjadi tipe data datetime
df['First_Transaction'] = pd.to_datetime(df['First_Transaction']/1000, unit='s', origin='1970-01-01')
df['Last_Transaction'] = pd.to_datetime(df['Last_Transaction']/1000, unit='s', origin='1970-01-01')
# Hapus kolom-kolom yang tidak diperlukan
del df['no']
del df['Row_Num']

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   no                          100000 non-null  int64
 1   Row_Num                     100000 non-null  int64
 2   Customer_ID                 100000 non-null  int64
 3   Product                     100000 non-null  object
 4   First_Transaction           100000 non-null  int64
 5   Last_Transaction            100000 non-null  int64
 6   Average_Transaction_Amount  100000 non-null  int64
 7   Count_Transaction           100000 non-null  int64
dtypes: int64(7), object(1)
memory usage: 6.1+ MB
None
```

Sebelum dicleansing

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Customer_ID                 100000 non-null  int64
 1   Product                     100000 non-null  object
 2   First_Transaction           100000 non-null  datetime64[ns]
 3   Last_Transaction            100000 non-null  datetime64[ns]
 4   Average_Transaction_Amount  100000 non-null  int64
 5   Count_Transaction           100000 non-null  int64
dtypes: datetime64[ns](2), int64(3), object(1)
memory usage: 4.6+ MB
None
```

Setelah dicleansing

Churn Customer

```
# Pengecekan transaksi terakhir dalam dataset
print(max(df['Last_Transaction']))
```

```
2019-02-01 23:57:57.286000128
```

Terlihat bahwa tanggal transaksi terakhir dari dataset tersebut adalah 2019-02-01.

Klasifikasi customer termasuk sudah bukan disebut pelanggan lagi (churn) ketika customer sudah tidak bertransaksi ke toko tersebut sampai dengan 6 bulan terakhir.
Sehingga untuk batasnya adalah 6 bulan sebelum tanggal transaksi terakhir, yaitu 2018-08-01.

```
# Mengklasifikasikan customer yang berstatus churn atau tidak dengan boolean
df.loc[df['Last_Transaction'] <= '2018-08-01', 'is_churn'] = True
df.loc[df['Last_Transaction'] > '2018-08-01', 'is_churn'] = False

print(df.head())
```

```
   Customer_ID Product            First_Transaction  \
0        29531   Jaket 2016-06-19 02:44:34.396000000
1        29531  Sepatu 2014-07-23 01:02:11.493999872
2       141526     Tas 2017-04-28 03:12:27.000000000
3       141526   Jaket 2017-04-28 06:52:52.546999808
4        37545  Sepatu 2015-04-16 10:01:38.530999808

               Last_Transaction  Average_Transaction_Amount  \
0 2018-10-05 05:48:02.608000000                     1467681
1 2018-12-25 11:02:41.269999872                     1269337
2 2019-01-24 09:40:02.000000000                      310915
3 2019-01-16 13:00:03.911000064                      722632
4 2018-11-22 12:53:41.529999872                     1775036

   Count_Transaction is_churn
0                 22    False
1                 41    False
2                 30    False
3                 27    False
4                 25    False
```

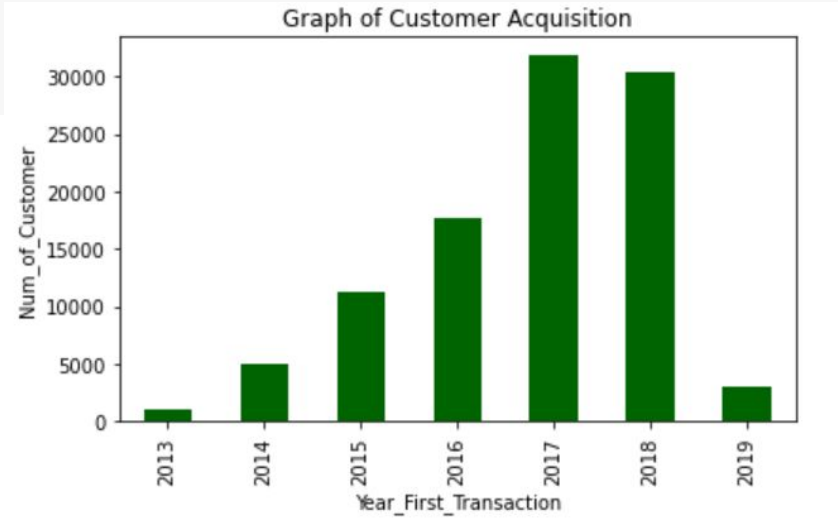
Data Visualization

Customer Acquisition by Year

```
import matplotlib.pyplot as plt

# Membuat kolom tahun transaksi
df['Year_First_Transaction'] = df['First_Transaction'].dt.year
df['Year_Last_Transaction'] = df['Last_Transaction'].dt.year

df_year = df.groupby(['Year_First_Transaction'])['Customer_ID'].count()
df_year.plot(x='Year_First_Transaction', y='Customer_ID', kind='bar', color='darkgreen', title='Graph of Customer Acquisition')
plt.xlabel('Year_First_Transaction')
plt.ylabel('Num_of_Customer')
plt.tight_layout()
plt.show()
```



Berdasarkan visualisasi bar chart disamping, dapat disimpulkan bahwa jumlah customer tertinggi diraih pada tahun 2017 yakni sekitar 30000 customer, sedangkan jumlah customer terendah terjadi di tahun 2013.

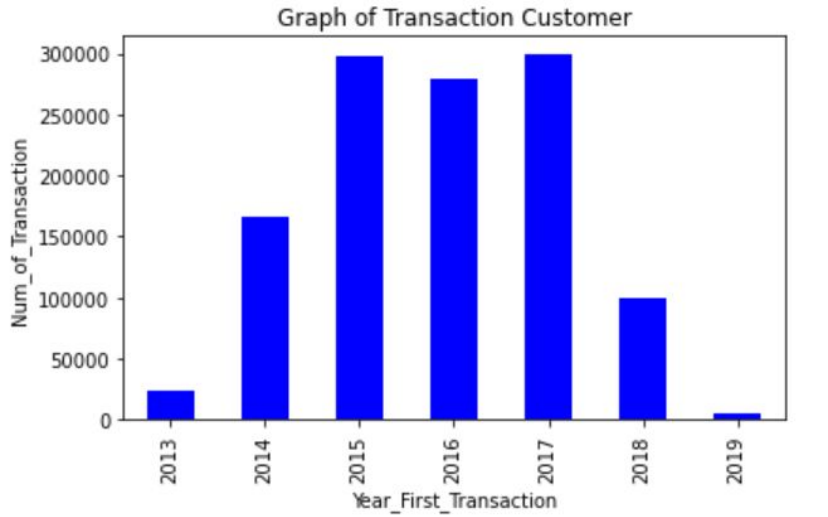
Transaction by Year

```
import matplotlib.pyplot as plt

plt.clf()
df_year = df.groupby(['Year_First_Transaction'])['Count_Transaction'].sum()
df_year.plot(x='Year_First_Transaction', y='Count_Transaction', kind='bar', color='blue', title='Graph of Transaction Customer')
plt.xlabel('Year_First_Transaction')
plt.ylabel('Num_of_Transaction')
plt.tight_layout()
plt.show()
```

Berdasarkan visualisasi bar chart disamping, pada tahun 2017 banyaknya jumlah transaksi berbanding lurus dengan banyaknya jumlah customer.

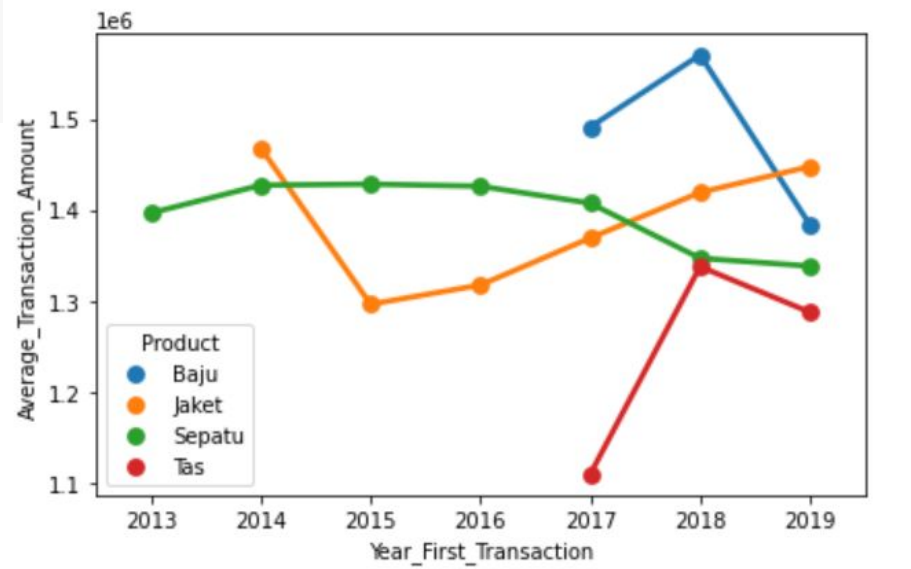
Sedangkan pada tahun 2015, jumlah customer hanya sekitar 12000 tetapi jumlah transaksinya cukup tinggi, hal ini mungkin terjadi karena banyak customer yang melakukan transaksi berkali-kali.



Average Transaction Amount by Year

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.clf()
sns.pointplot(data = df.groupby(['Product', 'Year_First_Transaction']).mean().reset_index(),
              x='Year_First_Transaction',
              y='Average_Transaction_Amount',
              hue='Product')
plt.tight_layout()
plt.show()
```

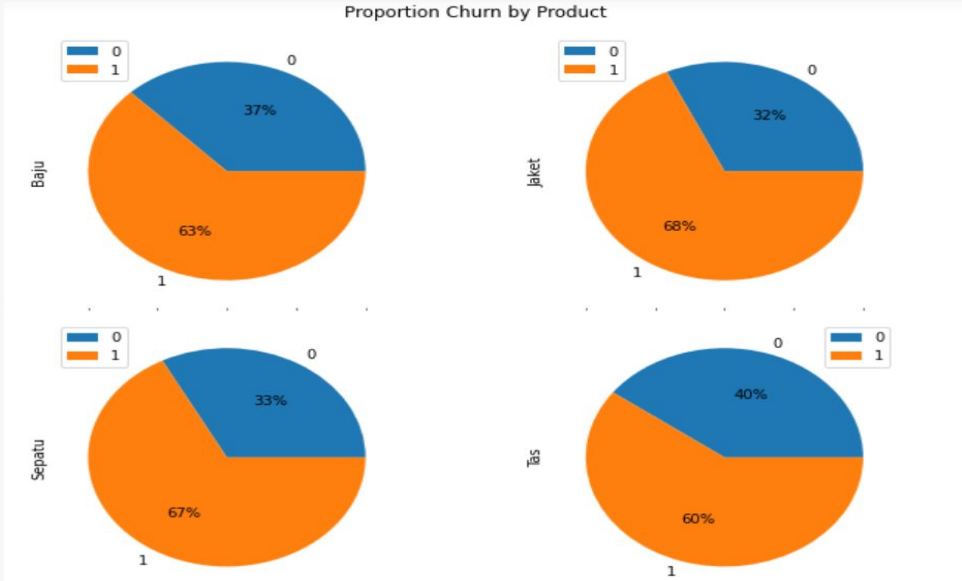


Berdasarkan visualisasi chart disamping, beberapa produk mengalami kecenderungan rata-rata transaksi yang naik-turun, tetapi untuk produk sepatu cenderung lebih konstan.

Proporsi Churned Customer untuk Setiap Produk

```
import matplotlib.pyplot as plt

plt.clf()
# Melakukan pivot data dengan pivot_table
df_piv = df.pivot_table(index='is_churn',
                        columns='Product',
                        values='Customer_ID',
                        aggfunc='count',
                        fill_value=0)
# Mendapatkan Proportion Churn by Product
plot_product = df_piv.count().sort_values(ascending=False).head(5).index
# Plot pie chartnya
df_piv = df_piv.reindex(columns=plot_product)
df_piv.plot.pie(subplots=True,
                figsize=(10, 7),
                layout=(-1, 2),
                autopct='%1.0f%%',
                title='Proportion Churn by Product')
plt.tight_layout()
plt.show()
```



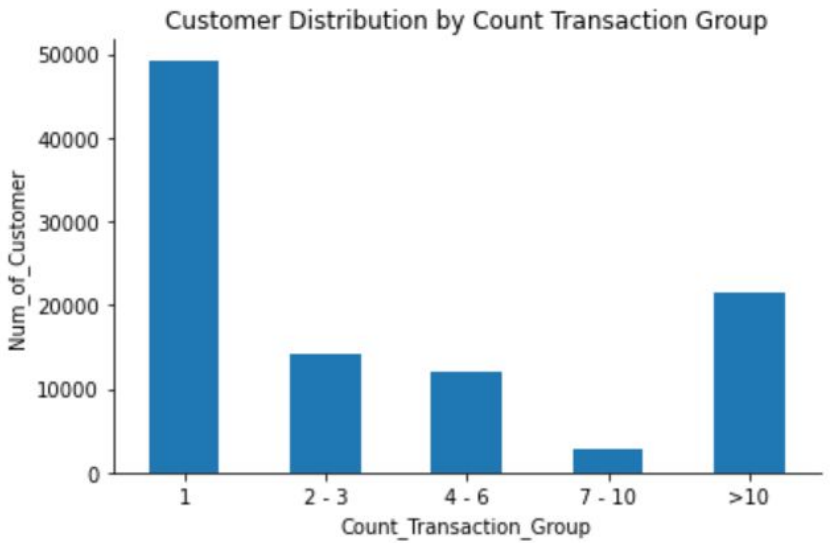
Angka 0 mewakili bukan tergolong customer churn sedangkan angka 1 mewakili tergolong customer churn. Berdasarkan visualisasi pie chat di atas, terlihat bahwa hampir keseluruhan produk memiliki customer churn. Rentang customer churn keseluruhan produk berkisar 60%-68%.

Distribusi Kategorisasi Count Transaction

```
import matplotlib.pyplot as plt

plt.clf()
# Kategorisasi jumlah transaksi
def func(row):
    if row['Count_Transaction'] == 1:
        val = '1'
    elif (row['Count_Transaction'] > 1 and row['Count_Transaction'] <= 3):
        val ='2 - 3'
    elif (row['Count_Transaction'] > 3 and row['Count_Transaction'] <= 6):
        val ='4 - 6'
    elif (row['Count_Transaction'] > 6 and row['Count_Transaction'] <= 10):
        val ='7 - 10'
    else:
        val ='>10'
    return val
# Tambahkan kolom baru
df['Count_Transaction_Group'] = df.apply(func, axis=1)

df_year = df.groupby(['Count_Transaction_Group'])['Customer_ID'].count()
df_year.plot(x='Count_Transaction_Group', y='Customer_ID', kind='bar', title='Customer Distribution by Count Transaction Group')
plt.xlabel('Count_Transaction_Group')
plt.ylabel('Num_of_Customer')
plt.xticks(rotation=360)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()
```



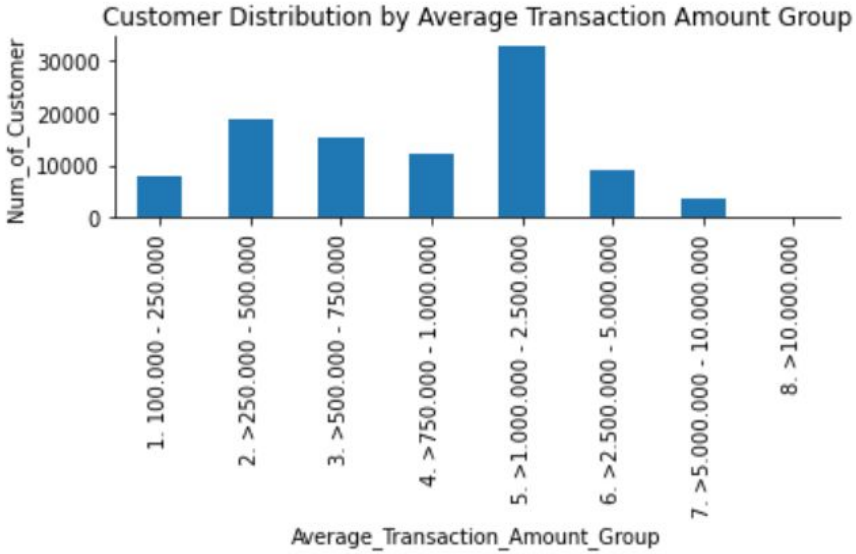
Berdasarkan visualisasi bar chart diatas, ternyata jumlah transaksi mereka hanya terjadi sekali (paling dominan).

Distribusi Kategorisasi Average Transaction Amount

```
import matplotlib.pyplot as plt

plt.clf()
# Kategorisasi rata-rata besar transaksi
def f(row):
    if (row['Average_Transaction_Amount'] >= 100000 and row['Average_Transaction_Amount'] <= 250000):
        val ='1. 100.000 - 250.000'
    elif (row['Average_Transaction_Amount'] > 250000 and row['Average_Transaction_Amount'] <= 500000):
        val ='2. >250.000 - 500.000'
    elif (row['Average_Transaction_Amount'] > 500000 and row['Average_Transaction_Amount'] <= 750000):
        val ='3. >500.000 - 750.000'
    elif (row['Average_Transaction_Amount'] > 750000 and row['Average_Transaction_Amount'] <= 1000000):
        val ='4. >750.000 - 1.000.000'
    elif (row['Average_Transaction_Amount'] > 1000000 and row['Average_Transaction_Amount'] <= 2500000):
        val ='5. >1.000.000 - 2.500.000'
    elif (row['Average_Transaction_Amount'] > 2500000 and row['Average_Transaction_Amount'] <= 5000000):
        val ='6. >2.500.000 - 5.000.000'
    elif (row['Average_Transaction_Amount'] > 5000000 and row['Average_Transaction_Amount'] <= 10000000):
        val ='7. >5.000.000 - 10.000.000'
    else:
        val ='8. >10.000.000'
    return val
# Tambahkan kolom baru
df['Average_Transaction_Amount_Group'] = df.apply(f, axis=1)
```

```
df_year = df.groupby(['Average_Transaction_Amount_Group'])['Customer_ID'].count()
df_year.plot(x='Average_Transaction_Amount_Group', y='Customer_ID', kind='bar',
             title='Customer Distribution by Average Transaction Amount Group')
plt.xlabel('Average_Transaction_Amount_Group')
plt.ylabel('Num_of_Customer')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()
```



Berdasarkan visualisasi bar chart diatas, ternyata rata-rata besar transaksi terjadi pada rentang angka >1.000.000 - 2.500.000.

Modelling

Feature columns dan target

```
# Feature column: Year_Diff
df['Year_Diff'] = df['Year_Last_Transaction'] - df['Year_First_Transaction']

# Nama-nama feature columns
feature_columns = ['Average_Transaction_Amount', 'Count_Transaction', 'Year_Diff']

# Features variable
X = df[feature_columns]

# Target variable
y = df['is_churn']
```

Hal yang akan diprediksi dalam machine learning disebut **target**.

Split X dan y ke dalam bagian training dan testing

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Training dataset digunakan untuk membuat atau melatih model machine learning.
Test dataset digunakan untuk menguji performa atau akurasi dari model yang telah di training.

Train, Predict, dan Evaluate

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Inisiasi model logreg
logreg = LogisticRegression()
# fit the model with data
logreg.fit(X_train, y_train)
# Predict model
y_pred = logreg.predict(X_test)
# Evaluasi model menggunakan confusion matrix
cnf_matrix =  confusion_matrix(y_test, y_pred)
print('Confusion Matrix:\n', cnf_matrix)
```

```
Confusion Matrix:
 [[    1  8330]
 [    3 16666]]
```

Confusion Matrix adalah pengukuran performa untuk masalah klasifikasi machine learning dimana keluaran dapat berupa dua kelas atau lebih. Confusion Matrix adalah tabel dengan 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. Ada empat istilah yang merupakan representasi hasil proses klasifikasi pada confusion matrix yaitu **True Positif**, **True Negatif**, **False Positif**, dan **False Negatif**.

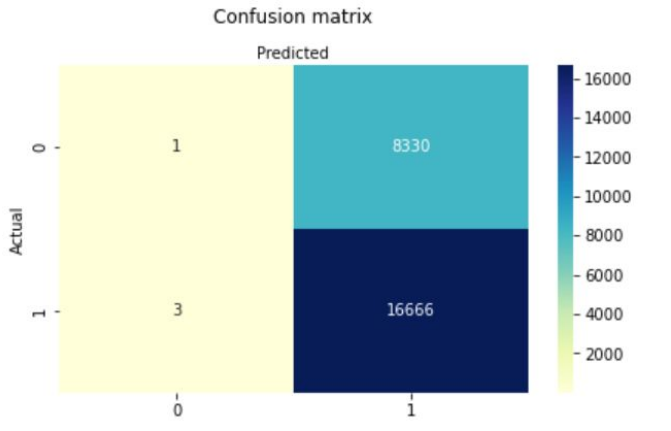
Visualisasi Confusion Matrix

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.clf()
# name  of classes
class_names = [0, 1]
fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='YlGnBu', fmt='g')
ax.xaxis.set_label_position('top')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.tight_layout()
plt.show()
```



Accuracy, Precision, dan Recall

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

#Menghitung Accuracy, Precision, dan Recall
print('Accuracy :', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred, average='micro'))
print('Recall   :',  recall_score(y_test, y_pred, average='micro'))
```

```
Accuracy : 0.66668
Precision: 0.66668
Recall   : 0.66668
```

Dengan demikian, nilai accuracy, precision, dan recall memiliki nilai yang sama sebesar 0.66668 atau 67%.

Accuracy menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar.
Precision adalah perbandingan antara True Positive (TP) dengan banyaknya data yang diprediksi positif.
Recall adalah perbandingan antara True Positive (TP) dengan banyaknya data yang sebenarnya positif.

Thank You

http://www.linkedin.com/in/figiacg