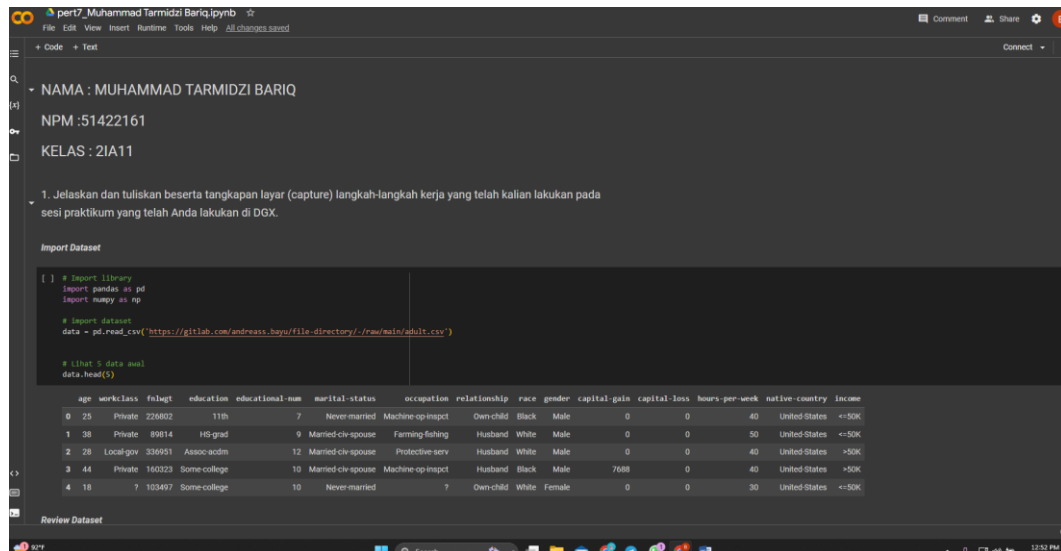


## PERT 7

MUHAMMAD TARMIDZI BARIQ

51422161

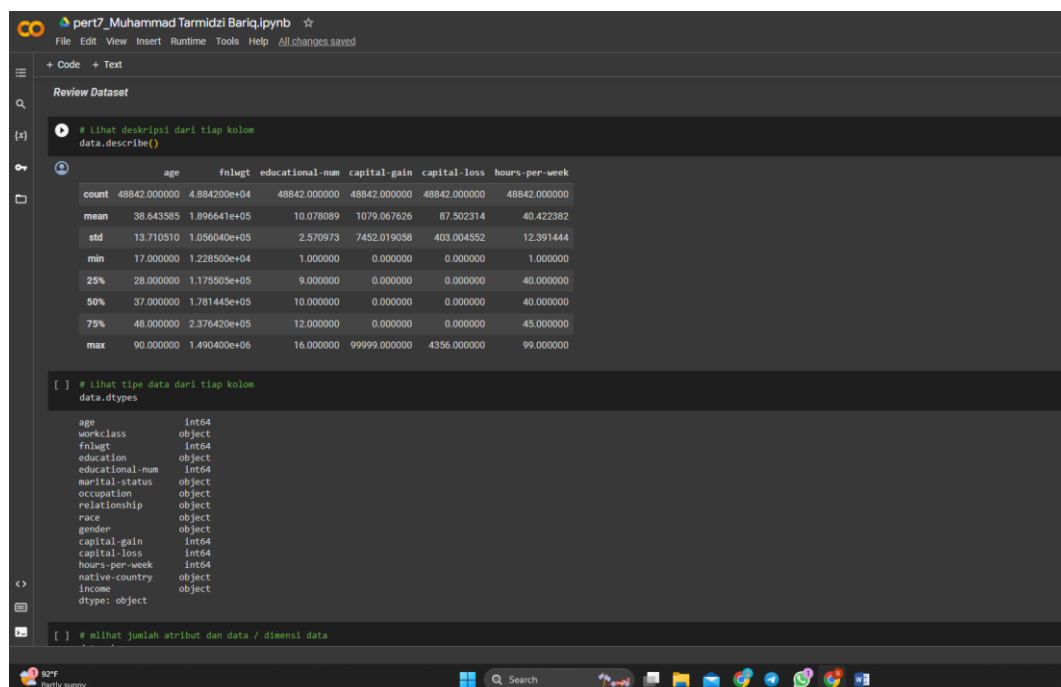


The screenshot shows a Jupyter Notebook interface with the following content:

- File menu: File, Edit, View, Insert, Runtime, Tools, Help. All changes saved.
- Code editor:
  - Cell 1: Text input.
    - NAMA : MUHAMMAD TARMIDZI BARIQ
    - NPM :51422161
    - KELAS : 2IA11
  - Cell 2: Text input.
    - 1. Jelaskan dan tuliskan beserta tangkapan layar (capture) langkah-langkah kerja yang telah kalian lakukan pada sesi praktikum yang telah Anda lakukan di DGX.
  - Cell 3: Code input.
    - ```
# Import library
import pandas as pd
import numpy as np

# Import dataset
data = pd.read_csv('https://gitlab.com/andreas.bayu/file-directory/-/raw/main/adult.csv')

# lihat 5 data awal
data.head(5)
```
  - Output of Cell 3: A preview of the dataset with 5 rows and 14 columns: age, workclass, fnlgt, education, educational-num, marital-status, occupation, relationship, race, gender, capital-gain, capital-loss, hours-per-week, native-country, income.
  - Review Dataset button.



The screenshot shows a Jupyter Notebook interface with the following content:

- File menu: File, Edit, View, Insert, Runtime, Tools, Help. All changes saved.
- Code editor:
  - Cell 1: Text input.
    - Review Dataset
  - Cell 2: Code input.
    - ```
# lihat deskripsi dari tiap kolom
data.describe()
```
  - Output of Cell 2: A summary statistics table for the dataset.
  - Cell 3: Code input.
    - ```
# lihat tipe data dari tiap kolom
data.dtypes
```
  - Output of Cell 3: A table showing the data types for each column.
  - Cell 4: Code input.
    - ```
# lihat jumlah atribut dan data / dimensi data
```
- Review Dataset button.

```
pert7_Muhammad Tarmidzi Bariq.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ] hours-per-week      int64
[ ] native-country      object
[ ] income              object
dtype: object

[ ] # melihat jumlah atribut dan data / dimensi data
data.shape

(48842, 15)

# hitung dan melihat jumlah data per label kelas
for col in data.columns:
    if data[col].dtype == "object":
        print("Attribute name:", col)
        print("-----")
        print(data[col].value_counts())
        print("-----")

Attribute name: workclass
-----
Private          33906
Self-emp-not-inc 3862
Local-gov        3136
?                2799
State-gov        1981
Self-emp-inc     1695
Federal-gov      1432
Without-pay      21
Never-worked     18
Name: workclass, dtype: int64
-----
Attribute name: education
-----
HS-grad          15784
Some-college     10878
Bachelors        8025
Masters          2657
Assoc-voc        2961
11th             1812
Assoc-acdm       1601
10th             1389
7th-8th          955
Prof-school      834
9th              756
12th             657

92°F
Partly sunny
```

```
pert7_Muhammad Tarmidzi Bariq.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

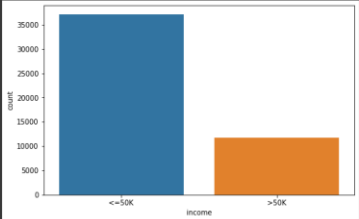
>>>
>50K      11687
Name: income, dtype: int64
-----

# import library seaborn untuk visualisasi
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Plot figure untuk menentukan distribusi kelas
plt.figure(figsize=(8,5))

# menghitung baris setiap kelas
sns.countplot(x="income", data=data)

<matplotlib.axes._subplots.AxesSubplot at 0x7fb74e9f8e80>



Dataset memiliki distribusi kelas yang tidak seimbang/imbalance sehingga secara teknis akan digunakan teknik untuk menangani data yang tidak seimbang

Dataset preparation

[ ] # Buat salinan dataframe
```

co

pert7\_Muhammad Tarmidzi Bariq.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

uang semidang

Dataset preparation

[ ] # Buat salinan dataframe  
df = data.copy(deep = True)  
  
# mengubah/convert nilai "?" nilai ke bentuk Na / NaN untuk diproses lebih lanjut  
for col in data.columns:  
 df[[col]] = data[[col]].replace('?', np.NaN)  
  
# seleksi kolom fitur/feature columns dari dataset  
null\_data = df.iloc[:, :-1]  
  
# temukan nilai null untuk semua atribut dan jumlahkan total nilai null  
null\_data.isnull().sum()  
  
age 0  
workclass 2799  
fnlwgt 0  
education 0  
educational-num 0  
marital-status 0  
occupation 2809  
relationship 0  
race 0  
gender 0  
capital-gain 0  
capital-loss 0  
hours-per-week 0  
native-country 857  
dtype: int64  
  
[ ] # jatuhkan/drop semua baris yang memiliki nilai null  
df = df.dropna()  
  
# pilih kolom fitur/feature columns dari dataset  
null\_data = df.iloc[:, :-1]  
  
# cek ulang nilai null  
null\_data.isnull().sum()

co

pert7\_Muhammad Tarmidzi Bariq.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

null\_data.isnull().sum()

age 0  
workclass 0  
fnlwgt 0  
education 0  
educational-num 0  
marital-status 0  
occupation 0  
relationship 0  
race 0  
gender 0  
capital-gain 0  
capital-loss 0  
hours-per-week 0  
native-country 0  
dtype: int64

StandardScaler adalah class dari sklearn untuk melakukan normalisasi data agar data yang digunakan tidak memiliki penyimpangan yang besar.

[ ] # Import library standard scaler  
  
from sklearn.preprocessing import StandardScaler  
  
# Buat dataframe dengan tipe data int64  
colname = []  
for col in df.columns:  
 if df[col].dtype == "int64":  
 colname.append(col)  
  
# Buat salinan dataset untuk keperluan persiapan data / data preparation  
df\_copy = df.copy(deep = True)  
df\_fe = df\_copy()  
  
# Buat kerangka data untuk fitur kategoris / categorical features  
df\_fe.drop('income', axis='columns', inplace=True)  
df\_fe.drop(colname, axis='columns', inplace=True)  
  
# buat dataframe untuk kelas target / target class

```
pert7_Muhammad Tarmidzi Bariq.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

# Buat kerangka data untuk fitur kategoris / categorical features
df_fe.drop('income',axis='columns', inplace=True)
df_fe.drop(colname,axis='columns', inplace=True)

# buat dataframe untuk kelas target / target class
df_cl = df.copy()
df_cl.drop(df_copy.iloc[:, :-1],axis='columns', inplace=True)

# membuat objek scaler / scaler object
std_scaler = StandardScaler()
std_scaler

# Normalisasikan atribut numerik dan tetapkan ke dalam dataframe baru
df_norm = pd.DataFrame(std_scaler.fit_transform(df_copy[colname]), columns=colname)

[ ] # import library Ordinal Encoder dari package library sklearn.preprocessing
from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()

# encode fitur kategoris/categorical features menjadi fitur numerik/numerical features
for col in df_fe.columns[:]:
    if df_fe[col].dtype == "object":
        df_fe[col] = ord_enc.fit_transform(df_fe[[col]])

# encode label kategorikal/categorical label menjadi label biner/binary label
df_cl["income"] = np.where(df_cl["income"].str.contains(">50k"), 0, 1)

[ ] # Masukkan kolom id ke datasets yang berbeda
df_norm.insert(0, 'id', range(0, 0 + len(df_norm)))
df_fe.insert(0, 'id', range(0, 0 + len(df_fe)))
df_cl.insert(0, 'id', range(0, 0 + len(df_cl)))

# Lihat shapes datasets yang telah di proses
```

```
pert7_Muhammad Tarmidzi Bariq.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ] # Masukkan kolom id ke datasets yang berbeda
df_norm.insert(0, 'id', range(0, 0 + len(df_norm)))
df_fe.insert(0, 'id', range(0, 0 + len(df_fe)))
df_cl.insert(0, 'id', range(0, 0 + len(df_cl)))

# Lihat shapes datasets yang telah di proses
print(df_norm.shape)
print(df_fe.shape)
print(df_cl.shape)

(45222, 7)
(45222, 9)
(45222, 2)

[ ] # Gabungkan semua datasets
df_feature = pd.merge(df_norm,df_fe, on="id")
df_final = pd.merge(df_feature,df_cl, on="id")

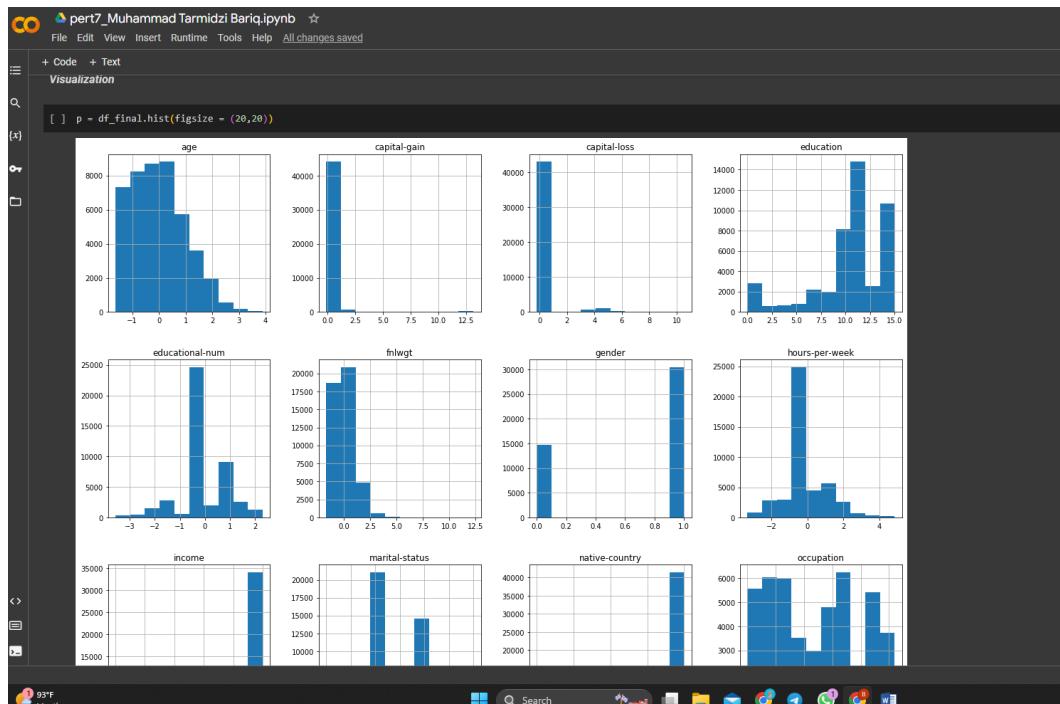
# Drop kolom id dari gabungan dataset
df_final.drop('id',axis='columns', inplace=True)

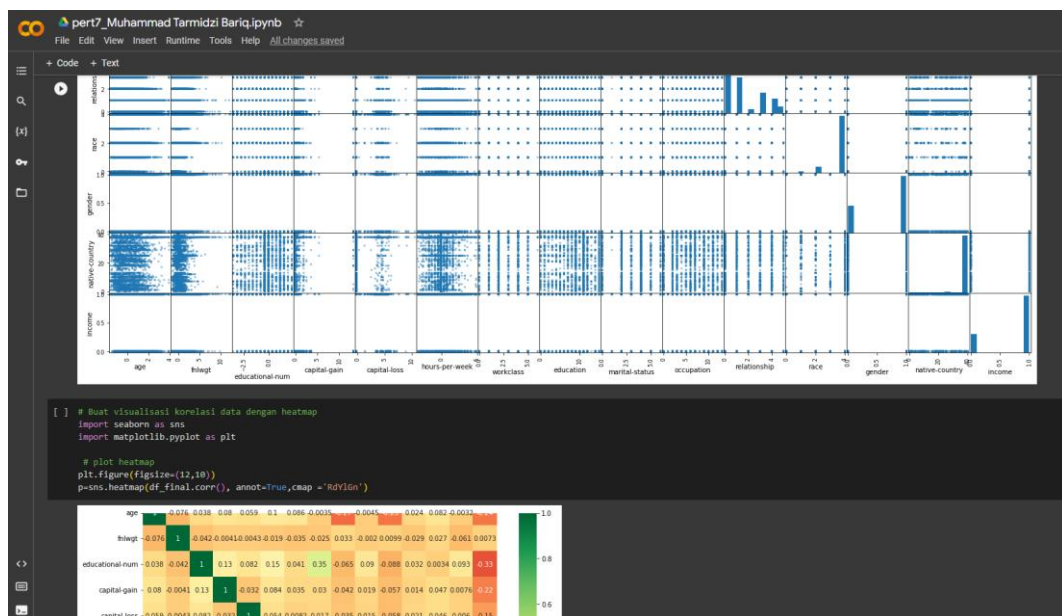
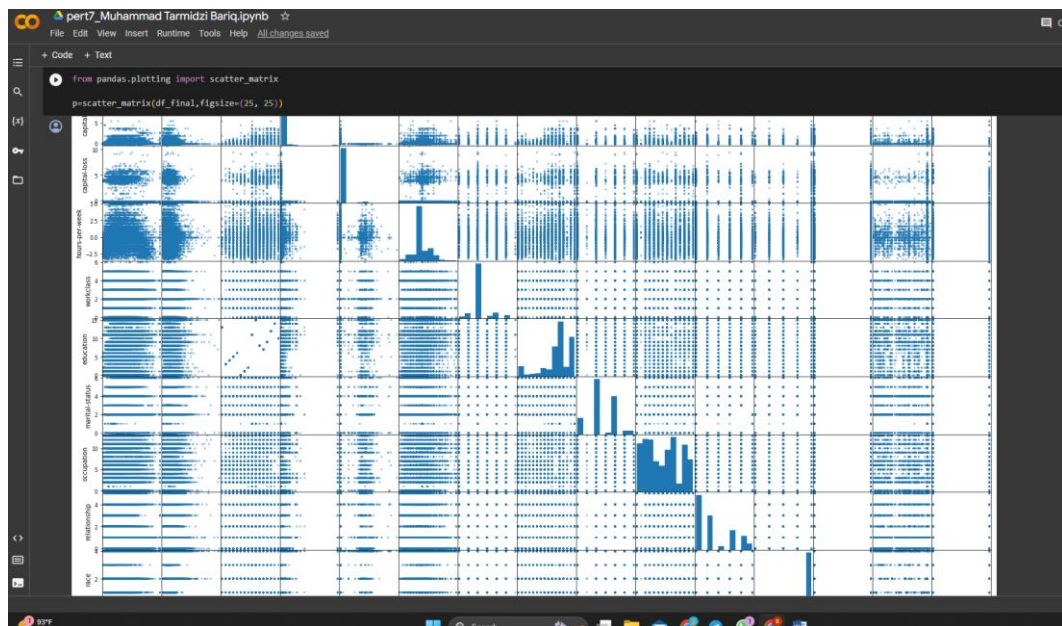
# Lihat 5 data awal dari gabungan dataset
df_final.head(5)
```

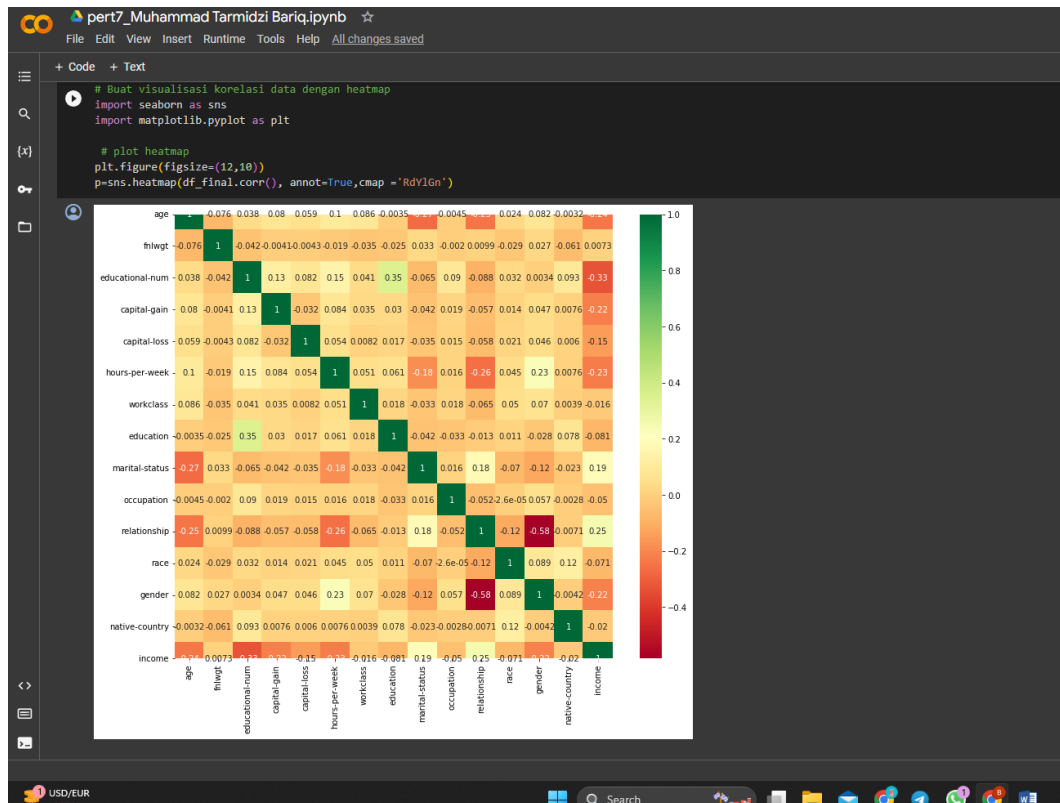
	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week	workclass	education	marital-status	occupation	relationship	race	gender	native-country	income
0	-1.024983	0.350889	-1.221559	-0.146733	-0.21878	-0.078120	2.0	1.0	4.0	6.0	3.0	2.0	1.0	38.0	1
1	-0.041455	-0.945878	-0.438122	-0.146733	-0.21878	0.754701	2.0	11.0	2.0	4.0	0.0	4.0	1.0	38.0	1
2	-0.798015	1.393592	0.737034	-0.146733	-0.21878	-0.078120	1.0	7.0	2.0	10.0	0.0	4.0	1.0	38.0	0
3	0.412481	-0.278420	-0.046403	0.877467	-0.21878	-0.078120	2.0	15.0	2.0	6.0	0.0	2.0	1.0	38.0	0
4	-0.344079	0.084802	-1.613277	-0.146733	-0.21878	-0.910942	2.0	0.0	4.0	7.0	1.0	4.0	1.0	38.0	1

Visualization

```
[ ] p = df_final.hist(figsize = (20,20))
```







pert7\_Muhammad Tarmidzi Bariq.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Proses Modelling dengan KNN

```
[ ] import numpy as np
from sklearn.model_selection import KFold

X=df_final.iloc[:,1:].to_numpy()
y=df_final.iloc[:,0].to_numpy()
kf = KFold(n_splits=5)
kf.get_n_splits(X)

print(kf)

for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    x_train, x_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

KFold(n_splits=5, random_state=None, shuffle=False)
TRAIN: [ 9045 9046 9047 ... 45219 45220 45221] TEST: [ 0 1 2 ... 9042 9043 9044]
TRAIN: [ 0 1 2 ... 45219 45220 45221] TEST: [ 9045 9046 9047 ... 18087 18088 18089]
TRAIN: [ 0 1 2 ... 45219 45220 45221] TEST: [18090 18091 18092 ... 27131 27132 27133]
TRAIN: [ 0 1 2 ... 45219 45220 45221] TEST: [27134 27135 27136 ... 36175 36176 36177]
TRAIN: [ 0 1 2 ... 36175 36176 36177] TEST: [36178 36179 36180 ... 45219 45220 45221]

[ ] print('----- x axis test -----')
print(x_test)
print('----- x axis train -----')
print(x_train)
print('----- y axis test -----')
print(y_test)
print('----- y axis train -----')
print(y_train)
print('*****')
```

----- x axis test -----  
[[ 0.63944887 1.45787715 -0.43812161 ... 1. 38.  
1. ]  
[ 1.24469679 -1.49349376 -0.846483 ... 1. 38.

USD/EUR 0.928

```
pert7_Muhammad Tarmidzi Bariq.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ] # import library model KNN dengan alias/as 'KNeighborsClassifier'
from sklearn.neighbors import KNeighborsClassifier
import sklearn.metrics as metrics

----- x axis train -----
[[1.24469679 -1.49349376 -0.046403 ... 1. 38.
1.
[-0.26842301 -0.14603391 -0.046403 ... 1. 38.
1.
...
[ 1.47166476 -0.35805983 -0.43812161 ... 0. 38.
1.
[-1.25195088 0.11127873 -0.43812161 ... 1. 38.
1.
[ 1.01772882 0.92951628 -0.43812161 ... 0. 38.
0.
]]
----- y axis test -----
[[1]
[1]
[1]
...
[1]
[1]
[0]]
----- y axis train -----
[[1]
[1]
[0]
...
[0]
[1]
[1]]
*****

[ ] # import library model KNN dengan alias/as 'KNeighborsClassifier'
from sklearn.neighbors import KNeighborsClassifier
import sklearn.metrics as metrics
```

```
pert7_Muhammad Tarmidzi Bariq.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ] # import library model KNN dengan alias/as 'KNeighborsClassifier'
from sklearn.neighbors import KNeighborsClassifier
import sklearn.metrics as metrics
import matplotlib.pyplot as plt

# buat variabel kosong untuk menyimpan metrik KNN/KNN metrics
scores=[]
# Kita coba nilai k yang berbeda untuk KNN (dari k=1 sampai k=26)
lrange=list(range(1,20))
# loop proses KNN
for k in lrange:
    # masukkan nilai k dan ukuran 'jarak'
    knn=KNeighborsClassifier(n_neighbors=k)
    # masukan data train/ data latih untuk melatih KNN
    knn.fit(x_train,y_train.ravel())
    # lihat prediksi KNN dengan memasukkan data uji/data test
    y_pred=knn.predict(x_test)
    # tambahkan performance metric akurasi
    scores.append(metrics.accuracy_score(y_test,y_pred))
plt.figure(2,figsize=(15,5))

optimal_k = lrange[scores.index(max(scores))]
print("Nilai k KNN yang optimal adalah %d" % optimal_k)
print("Skor optimalnya adalah %.2f" % max(scores))

# plot hasilnya
plt.plot(lrange, scores,ls='dashed')
plt.xlabel('Nilai dari k untuk KNN')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores untuk nilai k dari k-Nearest-Neighbors')
plt.show()

Nilai k KNN yang optimal adalah 3
Skor optimalnya adalah 0.95

Accuracy Scores untuk nilai k dari k-Nearest-Neighbors
```

