```
import time # menghitung waktu
import pickle # tipe data menyimpan model

import pandas as pd
import numpy as np

from sklearn.neighbors import KNeighborsClassifier # import lib KNN. untuk meberi tahu cara mesin belajar
from sklearn.model_selection import train_test_split # panggil algoritma untuk melatih mesin
from sklearn.metrics import accuracy_score # lalu lihat hasilnya ketika dilatih menggunakan KNN itu hasilnya seperti apa
```

```
data = pd.read_csv('https://raw.githubusercontent.com/krishna0604/homework_machine_learning_DigitalSkola/main/Prediction%20Insurance.csv
data.head()
```

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Char
0	1	Male	44	1	28	0	> 2 Years	Yes	40454	
1	2	Male	76	1	3	0	1-2 Year	No	33536	
2	3	Male	47	1	28	0	> 2 Years	Yes	38294	
3	4	Male	21	1	11	1	< 1 Year	No	28619	
4	5	Female	29	1	41	1	< 1 Year	No	27496	

```
data.shape # punya 12 variance (njumlah kolom)
```

```
(381109, 12)
```

```
data['Region_Code'].unique() # isinya hanya kode wilayah
```

```
array([28,  3, 11, 41, 33,  6, 35, 50, 15, 45,  8, 36, 30, 26, 16, 47, 48,
       19, 39, 23, 37,  5, 17,  2,  7, 29, 46, 27, 25, 13, 18, 20, 49, 22,
       44,  0,  9, 31, 12, 34, 21, 10, 14, 38, 24, 40, 43, 32,  4, 51, 42,
        1, 52])
```

Data Processing

a.) Handle missing data

```
# Handle missing data = data yang hilang
data.info()
data.isna().sum() # menjumlahkan semua data kosong
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381109 entries, 0 to 381108
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Gender                381109 non-null  int64
 1   Age                   381109 non-null  int64
 2   Driving_License       381109 non-null  int64
 3   Region_Code           381109 non-null  int64
 4   Previously_Insured    381109 non-null  int64
 5   Vehicle_Age           381109 non-null  object
 6   Vehicle_Damage        381109 non-null  object
 7   Annual_Premium        381109 non-null  int64
 8   Policy_Sales_Channel  381109 non-null  int64
 9   Vintage               381109 non-null  int64
 10  Response              381109 non-null  int64
dtypes: int64(9), object(2)
memory usage: 32.0+ MB
Gender                  0
Age                     0
Driving_License         0
Region_Code             0
Previously_Insured      0
Vehicle_Age             0
Vehicle_Damage          0
Annual_Premium          0
Policy_Sales_Channel    0
Vintage                 0
Response                0
dtype: int64
```

Report : tidak ada data yang hilang

a.) Handle duplicated data

```
# atau
print('Jumlah baris duplicate: ', data.duplicated().sum())
```

```
Jumlah baris duplicate:  269
```

```
df = data.drop_duplicates()
df.shape
```

```
(380840, 11)
```

```
# atau
print('Jumlah baris duplicate: ', df.duplicated().sum())
```

```
Jumlah baris duplicate:  0
```

Data yang tidak duplikat sudah disimpan dalam df dan menghapus 269 data yang duplikat

b.) Handle Outliers

Cek data outliers pada kolom 'Age', 'Annual_Premium', dan 'Vintage' karena terdapat angka yang memungkinkan adanya outlier

Age

```
## Daily Time Spent on Site ##

Q1_NumWebPurchases = np.quantile(df['Age'], .25)
Q3_NumWebPurchases = np.quantile(df['Age'], .75)
IQR_NumWebPurchases = Q3_NumWebPurchases - Q1_NumWebPurchases
min_IQR_NumWebPurchases = Q1_NumWebPurchases - 1.5 * IQR_NumWebPurchases
max_IQR_NumWebPurchases = Q3_NumWebPurchases + 1.5 * IQR_NumWebPurchases
nilai_min_NumWebPurchases = np.min(df['Age'])
nilai_max_NumWebPurchases = np.max(df['Age'])

print('')
print('C. Mencari outlier dari kolom NumWebPurchases')
print('1. nilai Q1 dari NumWebPurchases =', Q1_NumWebPurchases)
print('2. nilai Q3 dari NumWebPurchases =', Q3_NumWebPurchases)
print('3. nilai IQR dari NumWebPurchases =', IQR_NumWebPurchases)
print('4. nilai min IQR NumWebPurchases =', min_IQR_NumWebPurchases)
print('5. nilai max IQR NumWebPurchases =', max_IQR_NumWebPurchases)
print('6. nilai min dari NumWebPurchases =', nilai_min_NumWebPurchases)
print('7. nilai max dari NumWebPurchases =', nilai_max_NumWebPurchases)

if (nilai_min_NumWebPurchases < min_IQR_NumWebPurchases):
    print('Ditemukan low outlier!')
else:
    print('Tidak ditemukan low outlier!')

if (nilai_max_NumWebPurchases > max_IQR_NumWebPurchases):
    print('Ditemukan high outlier!')
else:
    print('Tidak ditemukan high outlier!')
```

```

C. Mencari outlier dari kolom NumWebPurchases
1. nilai Q1 dari NumWebPurchases = 25.0
2. nilai Q3 dari NumWebPurchases = 49.0
3. nilai IQR dari NumWebPurchases = 24.0
4. nilai min IQR NumWebPurchases = -11.0
5. nilai max IQR NumWebPurchases = 85.0
6. nilai min dari NumWebPurchases = 20
7. nilai max dari NumWebPurchases = 85
Tidak ditemukan low outlier!
Tidak ditemukan high outlier!
```

Annual_Premium

```
## Daily Time Spent on Site ##

Q1_NumWebPurchases = np.quantile(df['Annual_Premium'], .25)
Q3_NumWebPurchases = np.quantile(df['Annual_Premium'], .75)
IQR_NumWebPurchases = Q3_NumWebPurchases - Q1_NumWebPurchases
min_IQR_NumWebPurchases = Q1_NumWebPurchases - 1.5 * IQR_NumWebPurchases
max_IQR_NumWebPurchases = Q3_NumWebPurchases + 1.5 * IQR_NumWebPurchases
nilai_min_NumWebPurchases = np.min(df['Annual_Premium'])
nilai_max_NumWebPurchases = np.max(df['Annual_Premium'])

print('')
print('C. Mencari outlier dari kolom NumWebPurchases')
print('1. nilai Q1 dari NumWebPurchases =', Q1_NumWebPurchases)
print('2. nilai Q3 dari NumWebPurchases =', Q3_NumWebPurchases)
print('3. nilai IQR dari NumWebPurchases =', IQR_NumWebPurchases)
print('4. nilai min IQR NumWebPurchases =', min_IQR_NumWebPurchases)
print('5. nilai max IQR NumWebPurchases =', max_IQR_NumWebPurchases)
print('6. nilai min dari NumWebPurchases =', nilai_min_NumWebPurchases)
print('7. nilai max dari NumWebPurchases =', nilai_max_NumWebPurchases)

if (nilai_min_NumWebPurchases < min_IQR_NumWebPurchases):
    print('Ditemukan low outlier!')
else:
    print('Tidak ditemukan low outlier!')

if (nilai_max_NumWebPurchases > max_IQR_NumWebPurchases):
    print('Ditemukan high outlier!')
else:
    print('Tidak ditemukan high outlier!')
```

```

C. Mencari outlier dari kolom NumWebPurchases
1. nilai Q1 dari NumWebPurchases = 24426.0
2. nilai Q3 dari NumWebPurchases = 39408.0
3. nilai IQR dari NumWebPurchases = 14982.0
4. nilai min IQR NumWebPurchases = 1953.0
5. nilai max IQR NumWebPurchases = 61881.0
6. nilai min dari NumWebPurchases = 2630
7. nilai max dari NumWebPurchases = 540165
Tidak ditemukan low outlier!
Ditemukan high outlier!
```

Vintage

```
## Daily Time Spent on Site ##

Q1_NumWebPurchases = np.quantile(df['Vintage'], .25)
Q3_NumWebPurchases = np.quantile(df['Vintage'], .75)
IQR_NumWebPurchases = Q3_NumWebPurchases - Q1_NumWebPurchases
min_IQR_NumWebPurchases = Q1_NumWebPurchases - 1.5 * IQR_NumWebPurchases
max_IQR_NumWebPurchases = Q3_NumWebPurchases + 1.5 * IQR_NumWebPurchases
nilai_min_NumWebPurchases = np.min(df['Vintage'])
nilai_max_NumWebPurchases = np.max(df['Vintage'])

print('')
print('C. Mencari outlier dari kolom NumWebPurchases')
print('1. nilai Q1 dari NumWebPurchases =', Q1_NumWebPurchases)
print('2. nilai Q3 dari NumWebPurchases =', Q3_NumWebPurchases)
print('3. nilai IQR dari NumWebPurchases =', IQR_NumWebPurchases)
print('4. nilai min IQR NumWebPurchases =', min_IQR_NumWebPurchases)
print('5. nilai max IQR NumWebPurchases =', max_IQR_NumWebPurchases)
print('6. nilai min dari NumWebPurchases =', nilai_min_NumWebPurchases)
print('7. nilai max dari NumWebPurchases =', nilai_max_NumWebPurchases)

if (nilai_min_NumWebPurchases < min_IQR_NumWebPurchases):
    print('Ditemukan low outlier!')
else:
    print('Tidak ditemukan low outlier!')

if (nilai_max_NumWebPurchases > max_IQR_NumWebPurchases):
    print('Ditemukan high outlier!')
else:
    print('Tidak ditemukan high outlier!')
```

```

C. Mencari outlier dari kolom NumWebPurchases
1. nilai Q1 dari NumWebPurchases = 82.0
2. nilai Q3 dari NumWebPurchases = 227.0
3. nilai IQR dari NumWebPurchases = 145.0
4. nilai min IQR NumWebPurchases = -135.5
5. nilai max IQR NumWebPurchases = 444.5
6. nilai min dari NumWebPurchases = 10
```

```
7. nilai max dari NumWebPurchases = 299
Tidak ditemukan low outlier!
Tidak ditemukan high outlier!
```

Ditemukan high outlier pada kolom Annual_Premium, maka akan di handle

```
Q1 = df['Annual_Premium'].quantile(0.25)
Q3 = df['Annual_Premium'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Annual_Premium'] >= lower_bound) & (df['Annual_Premium'] <= upper_bound)]
```

Label Encoding

```
# Mengetahui nilai unik pada masing-masing kolom
unique_values = {col: df[col].unique() for col in df.columns}

# Menampilkan hasil
for col, values in unique_values.items():
    print(f"Kolom '{col}' memiliki nilai unik: {values}")
    print(" ")
    print(" ------------- ")
    print(" ")
```

```
 -------------

Kolom 'Region_Code' memiliki nilai unik: [28  3 11 41 33  6 35 50 15 45  8 36 30 26 16 47 48 19 39 23 37  5 17  2
  7 29 46 27 25 13 18 20 49 22 44  0  9 31 12 34 21 10 14 38 24 40 43 32
  4 51 42  1 52]

 -------------

Kolom 'Previously_Insured' memiliki nilai unik: [0 1]

 -------------

Kolom 'Vehicle_Age' memiliki nilai unik: ['> 2 Years' '1-2 Year' '< 1 Year']

 -------------

Kolom 'Vehicle_Damage' memiliki nilai unik: ['Yes' 'No']

 -------------

Kolom 'Annual_Premium' memiliki nilai unik: [40454 33536 38294 ... 20404 13345 20706]

 -------------

Kolom 'Policy_Sales_Channel' memiliki nilai unik: [ 26 152 160 124  14  13  30 156 163 157 122  19  22  15 154  16  52 155
  11 151 125  25  61   1  86  31 150  23  60  21 121 139  12  29  55   7
  47 127 153  78 158  89  32   8  10 120  65   4  42  83 136  24  18  56
  48 106  54  93 116  91  45   9   3 145 147  44 109  37 140 107 128 131
 114 118 159 119 105 135  62 138 129  88  92 111 113  73  36  28  35  59
  53 148 133 108  64  39  94 132  46  81 103  90  51  27 146  63  96  40
  66 100  95 123  98  75  69 130 134  49  97  38  17 110  80  71 117  58
  20  76 104  87  84 137 126  68  67 101 115  57  82  79 112  99  70   2
  34  33  74 102 149  43   6  50 144 143  41]

 -------------

Kolom 'Vintage' memiliki nilai unik: [217 183  27 203  39 176 249  72  28  80  46 289 221  15  58 147 256 299
 158 102 116 177 232  60 180  49  57 223 136 222 149 169  88 253 264 233
  45 184 251 153 186  71  34  83  12 246 141 216 130 282  73 171 283 295
 165  30 218  22  36  79  81 100  63 242 277  61 111 167  74 235 131 243
 248 114 281  62 189 139 138 209 254 291  68  92  52  78 156 247 275  77
 181 229 166  16  23  31 293 219  50 155  66 260  19 258 117 193 204 212
 144 234 206 228 125  29  18  84 230  54 123  86  13 237  85  98  67 128
  95  89  99 208 134 135 268 284 119 226 105 142 207 272 263  64  40 245
 163  24 265 202 259  91 106 190 162  33 194 287 292  69 239 132 255 152
 121 150 143 198 103 127 285 214 151 199  56  59 215 104 238 120  21  32
 270 211 200 197  11 213  93 113 178  10 290  94 231 296  47 122 271 278
 276 107  96 240 172 257 224 173 220 185  90  51 205  70 160 137 168  87
 118 288 126 241  82 227 115 164 236 286 108 274 201  97  25 174 182 154
  48 244  20  53  17 261  41 266  35 140 269 146 101 145  65 298 133 195
  55 188  75  38  43 110  37 129 170 109 267 279 112  76 191  26 280 161
 179 175 252  42 124 187 148 294  44 157 192 262 159 210 250  14 273 297
 225 196]

 -------------

Kolom 'Response' memiliki nilai unik: [1 0]

 -------------
```

Vehicle_Age

```
# Encoding - Education (Tipe ordinal - punya urutan)
df.loc[(df.Vehicle_Age == "> 2 Years"),"Vehicle_Age"] = 2
df.loc[(df.Vehicle_Age == "1-2 Year") ,"Vehicle_Age"] = 1
df.loc[(df.Vehicle_Age == "< 1 Year"),"Vehicle_Age"] = 0
# Adjusting data type
df.Vehicle_Age = df.Vehicle_Age.astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 368983 entries, 0 to 381108
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype 
---  ------                --------------   ----- 
 0   Gender                368983 non-null  int64 
 1   Age                   368983 non-null  int64 
 2   Driving_License       368983 non-null  int64 
 3   Region_Code           368983 non-null  int64 
 4   Previously_Insured    368983 non-null  int64 
 5   Vehicle_Age           368983 non-null  int64 
 6   Vehicle_Damage        368983 non-null  object
 7   Annual_Premium        368983 non-null  int64 
 8   Policy_Sales_Channel  368983 non-null  int64 
 9   Vintage               368983 non-null  int64 
 10  Response              368983 non-null  int64 
dtypes: int64(10), object(1)
memory usage: 33.8+ MB
```

Vehicle_Damage

```
# Encoding - Education (Tipe ordinal - punya urutan)
df.loc[(df.Vehicle_Damage == "Yes"),"Vehicle_Damage"] = 1
df.loc[(df.Vehicle_Damage == "No") ,"Vehicle_Damage"] = 0
# Adjusting data type
df.Vehicle_Damage = df.Vehicle_Damage.astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 368983 entries, 0 to 381108
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Gender                368983 non-null  int64
 1   Age                   368983 non-null  int64
 2   Driving_License       368983 non-null  int64
 3   Region_Code           368983 non-null  int64
 4   Previously_Insured    368983 non-null  int64
 5   Vehicle_Age           368983 non-null  int64
 6   Vehicle_Damage        368983 non-null  int64
 7   Annual_Premium        368983 non-null  int64
 8   Policy_Sales_Channel  368983 non-null  int64
 9   Vintage               368983 non-null  int64
 10  Response              368983 non-null  int64
dtypes: int64(11)
memory usage: 33.8 MB
```

One Hot Encoding

```
df_after = pd.get_dummies(df['Region_Code'])
```

```
df_region = pd.get_dummies(data['Region_Code']) # gimana mesin bsabedain region, maka get dummies. get dummies itu membuat mesin akan m
# ini One-Hot Encoding. bagaimana mesin ketika baca dia tahu pola dan harus dalam bentuk binary

df = data[['Gender', 'Age', 'Driving_License', 'Response']]#.merge(df_region, left_index=True, right_index=True )
df.head(1)
```

```
   Gender  Age  Driving_License  Response
0       1   44                1         1
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381109 entries, 0 to 381108
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Gender           381109 non-null  int64
 1   Age              381109 non-null  int64
 2   Driving_License  381109 non-null  int64
 3   Response         381109 non-null  int64
dtypes: int64(4)
memory usage: 11.6 MB
```

Data Modelling

```
# data modelling
X = df.drop('Response', axis=1)# variabel input
y = df['Response'] # variabel output
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42)

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

def eval_classification(model):
  y_pred = model.predict(X_test)
  y_pred_train = model.predict(X_train)
  y_pred_proba = model.predict_proba(X_test) # output berupa probabilitas
  y_pred_proba_train = model.predict_proba(X_train)

  print('Akurasi (test_set) : ', accuracy_score(y_test, y_pred))
  print('Precision (test_set) : ', precision_score(y_test, y_pred))
  print('Recall (test_set) : ', recall_score(y_test, y_pred))
  print('F1-score (test_set) : ', f1_score(y_test, y_pred))

  print('AUC (test_proba) :', roc_auc_score (y_test, y_pred_proba[:, 1]))
  print('AUC (train_proba) :', roc_auc_score(y_train, y_pred_proba_train[:, 1]))

def show_feature_importance(model):
    feat_importances = pd.Series(model.feature_importances_, index=X.columns)
    ax = feat_importances.nlargest(25).plot(kind='barh', figsize=(10, 8))
    ax.invert_yaxis()

    plt.xlabel('score')
    plt.ylabel('feature')
    plt.title('feature importance score')

def show_best_hyperparameter(model):
    print(model.best_estimator_.get_params())
```

a.) Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
eval_classification(lr)
```

```
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
AUC (test_proba) : 0.6234995712244277
AUC (train_proba) : 0.6238789550767412
```

result

- AUC ROC score gap : 0.0004
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_Logistic_Regression.pkl', 'wb') as file:
    pickle.dump(lr, file)
```

```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_Logistic_Regression.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_Logistic_Regression.pkl' ditemukan di lokasi: /content/model_Logistic_Regression.pkl
Path lengkap file: /content/model_Logistic_Regression.pkl
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_Logistic_Regression.pkl')
```

b.) Decision Tree

```
# decision tree
from sklearn.tree import DecisionTreeClassifier # import decision tree dari sklearn
dt = DecisionTreeClassifier() # inisiasi object dengan nama dt
dt.fit(X_train, y_train) # fit model decision tree dari data train
print("Model Evaluation (Decision Tree)")
eval_classification(dt)
```

```
Model Evaluation (Decision Tree)
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
AUC (test_proba) : 0.700619324923557
AUC (train_proba) : 0.704122599796087
```

result

- AUC ROC score gap : 0.006
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_Decision_Tree.pkl', 'wb') as file:
    pickle.dump(dt, file)
```

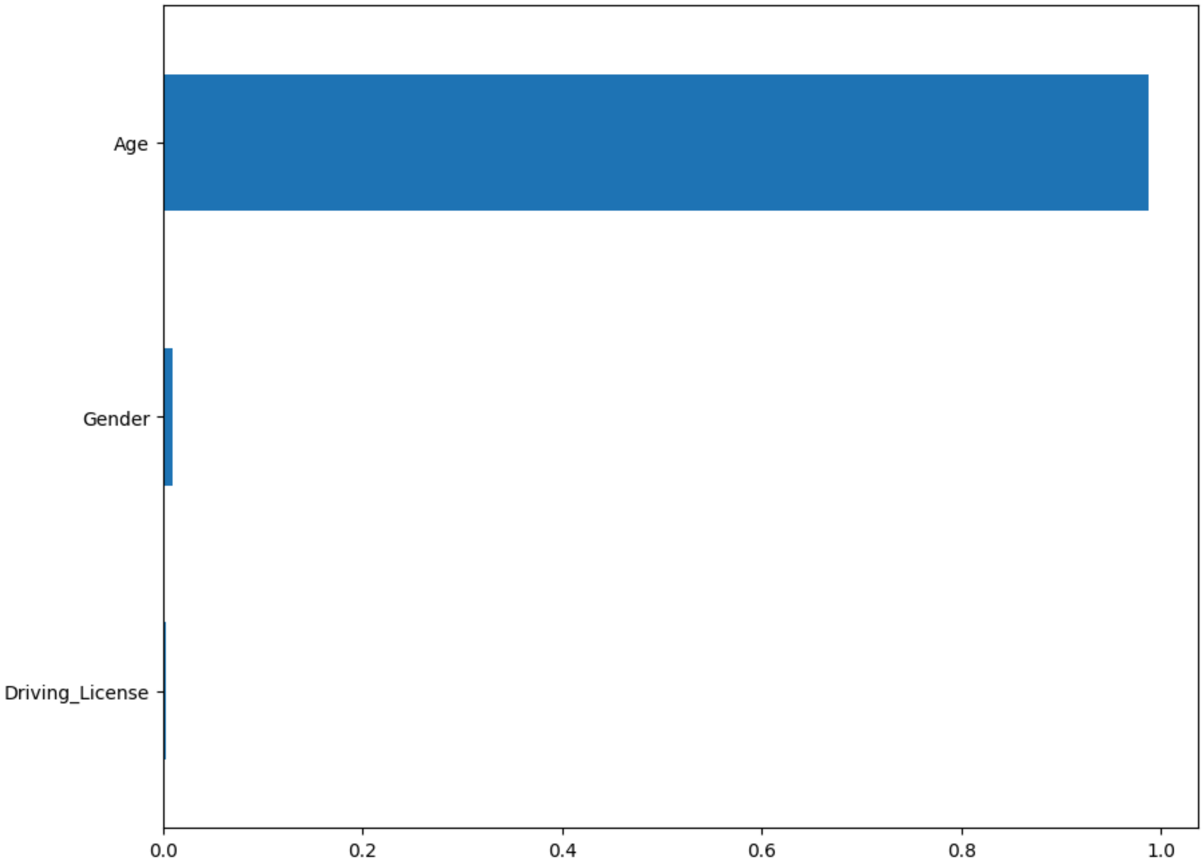
```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_Decision_Tree.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_Decision_Tree.pkl' ditemukan di lokasi: /content/model_Decision_Tree.pkl
Path lengkap file: /content/model_Decision_Tree.pkl
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_Decision_Tree.pkl')
```

```
show_feature_importance(dt)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-51-92c5313d71c1> in <cell line: 1>()
----> 1 show_feature_importance(dt)

<ipython-input-46-f974ca9d0115> in show_feature_importance(model)
     23     ax.invert_yaxis()
     24 
---> 25     plt.xlabel('score')
     26     plt.ylabel('feature')
     27     plt.title('feature importance score')

NameError: name 'plt' is not defined
```



Next steps: Explain error

c.) Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
print('Model Evaluation (Random Forest)')
eval_classification(rf)
```

```
Model Evaluation (Random Forest)
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
AUC (test_proba) : 0.7006337115675783
AUC (train_proba) : 0.7041127730841708
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
```

result

- AUC ROC score gap : 0.006
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_Random_Forest.pkl', 'wb') as file:
    pickle.dump(rf, file)
```

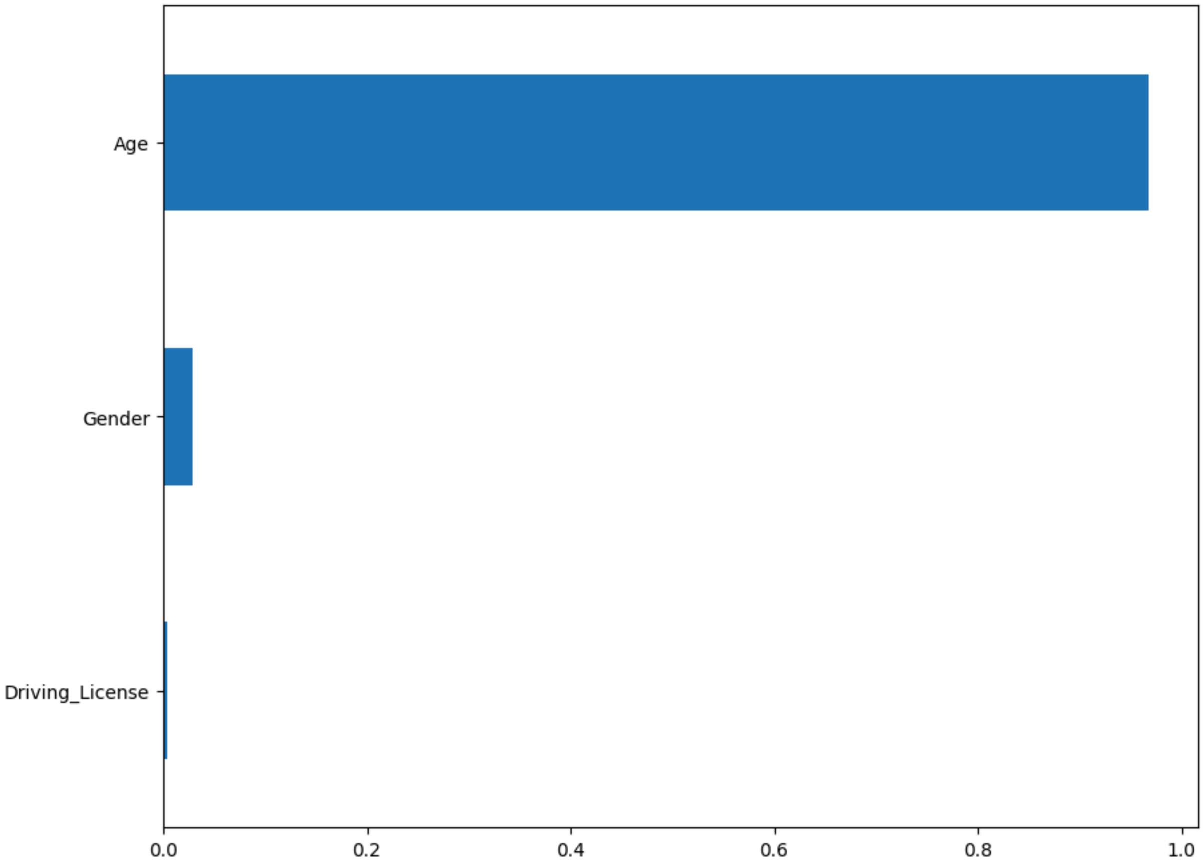
```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_Random_Forest.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_Random_Forest.pkl' ditemukan di lokasi: /content/model_Random_Forest.pkl
Path lengkap file: /content/model_Random_Forest.pkl
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_Random_Forest.pkl')
```

```
show_feature_importance(rf)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-53-536d324188c8> in <cell line: 1>()
----> 1 show_feature_importance(rf)

<ipython-input-46-f974ca9d0115> in show_feature_importance(model)
     23     ax.invert_yaxis()
     24 
---> 25     plt.xlabel('score')
     26     plt.ylabel('feature')
     27     plt.title('feature importance score')

NameError: name 'plt' is not defined
```



Next steps: Explain error

d.) KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Model Evaluation (KNN)')
eval_classification(knn)
```

```
Model Evaluation (KNN)
Akurasi (test_set) :  0.8579062912719863
Precision (test_set) :  0.2227774855339295
Recall (test_set) :  0.0599094638562738740
F1-score (test_set) :  0.09442586399108138
AUC (test_proba) : 0.5737514749760388
AUC (train_proba) : 0.57888548467073
```

result

- AUC ROC score gap : 0.005
- Accuration : 0.857

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_KNN.pkl', 'wb') as file:
    pickle.dump(knn, file)
```

```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_KNN.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_KNN.pkl' ditemukan di lokasi: /content/model_KNN.pkl
Path lengkap file: /content/model_KNN.pkl
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_KNN.pkl')
```

e.) Naive Bayes

```
# Applying Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Model Evaluation (Naive Bayes)')
eval_classification(gnb)
```

```
Model Evaluation (Naive Bayes)
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
AUC (test_proba) : 0.6780016046090234
AUC (train_proba) : 0.6800404886224097
```

result

- AUC ROC score gap : 0.01
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_Naive_Bayes.pkl', 'wb') as file:
    pickle.dump(gnb, file)
```

```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_Naive_Bayes.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_Naive_Bayes.pkl' ditemukan di lokasi: /content/model_Naive_Bayes.pkl
Path lengkap file: /content/model_Naive_Bayes.pkl
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_Naive_Bayes.pkl')
```

f.) XGBoost

```
from xgboost import XGBClassifier

xg = XGBClassifier()
xg.fit(X_train, y_train)
print('Model Evaluation (XGBoost)')
eval_classification(xg)
```

```
Model Evaluation (XGBoost)
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
AUC (test_proba) : 0.7007947940474908
AUC (train_proba) : 0.7040300568090417
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
```

result

- AUC ROC score gap : 0.006
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_XGBoost.pkl', 'wb') as file:
    pickle.dump(xg, file)
```

```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_XGBoost.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
```

```
from google.colab import files

# Download file pickle dari Google Colab
files.download('/content/model_XGBoost.pkl')
```

g.) AdaBoost

```
File 'model_XGBoost.pkl' ditemukan di lokasi: /content/model_XGBoost.pkl
```

```
from sklearn.ensemble import AdaBoostClassifier
clf = AdaBoostClassifier()
clf.fit(X_train, y_train)
print('Model Evaluation (Adaboost)')
eval_classification(clf)
```

```
Model Evaluation (Adaboost)
Akurasi (test_set) :  0.8763436628095126
Precision (test_set) :  0.0
Recall (test_set) :  0.0
F1-score (test_set) :  0.0
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined a
  _warn_prf(average, modifier, msg_start, len(result))
AUC (test_proba) : 0.7000226891696149
AUC (train_proba) : 0.7009862209623449
```

result

- AUC ROC score gap : 0.0001
- Accuration : 0.876

pickle

```
# Menyimpan model ke dalam file pickle
with open('model_AdaBoost.pkl', 'wb') as file:
    pickle.dump(clf, file)
```

```
import os

# Nama file pickle yang telah disimpan
file_name = 'model_AdaBoost.pkl'

# Mendapatkan direktori kerja saat ini
current_directory = os.getcwd()

# Mendapatkan path lengkap file
file_path = os.path.join(current_directory, file_name)

# Cek apakah file ada dan tampilkan lokasi file
if os.path.exists(file_path):
    print(f"File '{file_name}' ditemukan di lokasi: {file_path}")
else:
    print(f"File '{file_name}' tidak ditemukan di direktori: {current_directory}")

# Tampilkan path lengkap file
print("Path lengkap file:", file_path)
```

```
File 'model_AdaBoost.pkl' ditemukan di lokasi: /content/model_AdaBoost.pkl
Path lengkap file: /content/model_AdaBoost.pkl
```