Aliza Muslimah

PYTN-07

Classification

Preprocessing

In [1]:

```
# Import Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, normalize, StandardScaler
from sklearn import metrics
import statsmodels.api as sm
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, plot_confusion_matrix

%matplotlib inline
```

In [3]:

```
# Membaca file dataset
df = pd.read_csv('bank.csv', skipinitialspace=True,  sep=';')

#Melihat 10 record pertama
df.head(10)
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0
5	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3
6	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2
7	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0
8	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0
9	43	services	married	primary	no	-88	yes	yes	cellular	17	apr	313	1	147	2

In [4]:

```
#Menampilkan semua nama kolom
df.columns
```

Out[4]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [5]:

```
#melihat informasi dari dataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        4521 non-null   int64
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64
 12  campaign   4521 non-null   int64
 13  pdays      4521 non-null   int64
 14  previous   4521 non-null   int64
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

In [6]:

```
# Melihat dimensi dataFrame
df.shape
```

Out[6]:

```
(4521, 17)
```

In [7]:

```
# Menampilkan jumlah 'unknown' di setiap kolom
df[df[df.columns]=='unknown'].count().sort_values(ascending=False)
```

Out[7]:

```
poutcome     3705
contact      1324
education     187
job            38
month           0
previous        0
pdays           0
campaign        0
duration        0
age             0
day             0
loan            0
housing         0
balance         0
default         0
marital         0
y               0
dtype: int64
```

In [8]:

```
# Mengganti 'unknown' dengan Numpy nan
df[df[df.columns]=='unknown']=np.nan
```

In [9]:

```
# Cek apakah 'unknown' sudah terganti dengan nan
df[df[df.columns]=='unknown'].count().sort_values(ascending=False)
```

Out[9]:

```
age          0
day          0
poutcome     0
previous     0
pdays        0
campaign     0
duration     0
month        0
contact      0
job          0
loan         0
housing      0
balance      0
default      0
education    0
marital      0
y            0
dtype: int64
```

In [10]:

```
# Melihat jumlah np.nan
print("Bank Data:")
print(df.isnull().sum().sort_values(ascending=False), "\n")
```

```
Bank Data:
poutcome     3705
contact      1324
education     187
job            38
month           0
previous        0
pdays           0
campaign        0
duration        0
age             0
day             0
loan            0
housing         0
balance         0
default         0
marital         0
y               0
dtype: int64
```

In [11]:

```
#Mengganti np.nan pada kolom kategorikal dengan modusnya
df.job.fillna(df.job.mode()[0],inplace=True)
df.education.fillna(df.education.mode()[0],inplace=True)
df.contact.fillna(df.contact.mode()[0],inplace=True)
df.poutcome.fillna(df.poutcome.mode()[0],inplace=True)
df.isnull().sum()
```

Out[11]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [12]:

```
df
```

Out[12]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1
3	30	management	married	tertiary	no	1476	yes	yes	cellular	3	jun	199	4	-1	0
4	59	blue-collar	married	secondary	no	0	yes	no	cellular	5	may	226	1	-1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular	30	jul	329	5	-1	0
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	cellular	9	may	153	1	-1	0
4518	57	technician	married	secondary	no	295	no	no	cellular	19	aug	151	11	-1	0
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	6	feb	129	4	211	3
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	3	apr	345	2	249	7

4521 rows × 17 columns

In [13]:

```
#Mengganti nama kolom 'y' menjadi 'subscribed'
df.rename(columns={'y':'subscribed'}, inplace=True)
```

In [14]:

```
# Encoding data kategorikal
df_t = df.copy()
le = preprocessing.LabelEncoder()

def transform(col):
    a = le.fit_transform(df[col])
    pd.DataFrame(a)
    df_t[col] = a

categorical =   ['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
        'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
        'previous', 'poutcome', 'subscribed']

for i in categorical:
    transform(i)

df_t
```

Out[14]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	11	10	1	0	0	1475	0	0	0	18	10	75	0	0	0	0
1	14	7	1	1	0	2030	1	1	0	10	8	216	0	228	4	0
2	16	4	2	2	0	1303	1	0	0	15	0	181	0	219	1	0
3	11	4	1	2	0	1352	1	1	0	2	6	195	3	0	0	0
4	40	1	1	1	0	274	1	0	0	4	8	222	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4516	14	7	1	1	0	119	1	0	0	29	5	325	4	0	0	0
4517	38	6	1	2	1	0	1	1	0	8	8	149	0	0	0	0
4518	38	9	1	1	0	558	0	0	0	18	1	147	10	0	0	0
4519	9	1	1	1	0	1187	0	0	0	5	3	125	3	140	3	1
4520	25	2	2	2	0	1186	1	1	0	2	0	341	1	161	7	1

4521 rows × 17 columns

In [15]:

```
# Split dataset into features and labels
X=df_t[['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome']].values
y=df_t['subscribed'].values

#splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=5)

# Standarisasi data
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.fit_transform(X_test)
```

In [16]:

```
X
```

Out[16]:

```
array([[  11,   10,    1, ...,    0,    0,    0],
       [  14,    7,    1, ...,  228,    4,    0],
       [  16,    4,    2, ...,  219,    1,    0],
       ...,
       [  38,    9,    1, ...,    0,    0,    0],
       [   9,    1,    1, ...,  140,    3,    1],
       [  25,    2,    2, ...,  161,    7,    1]], dtype=int64)
```

In [17]:

```
y
```

Out[17]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

Logistic Regression

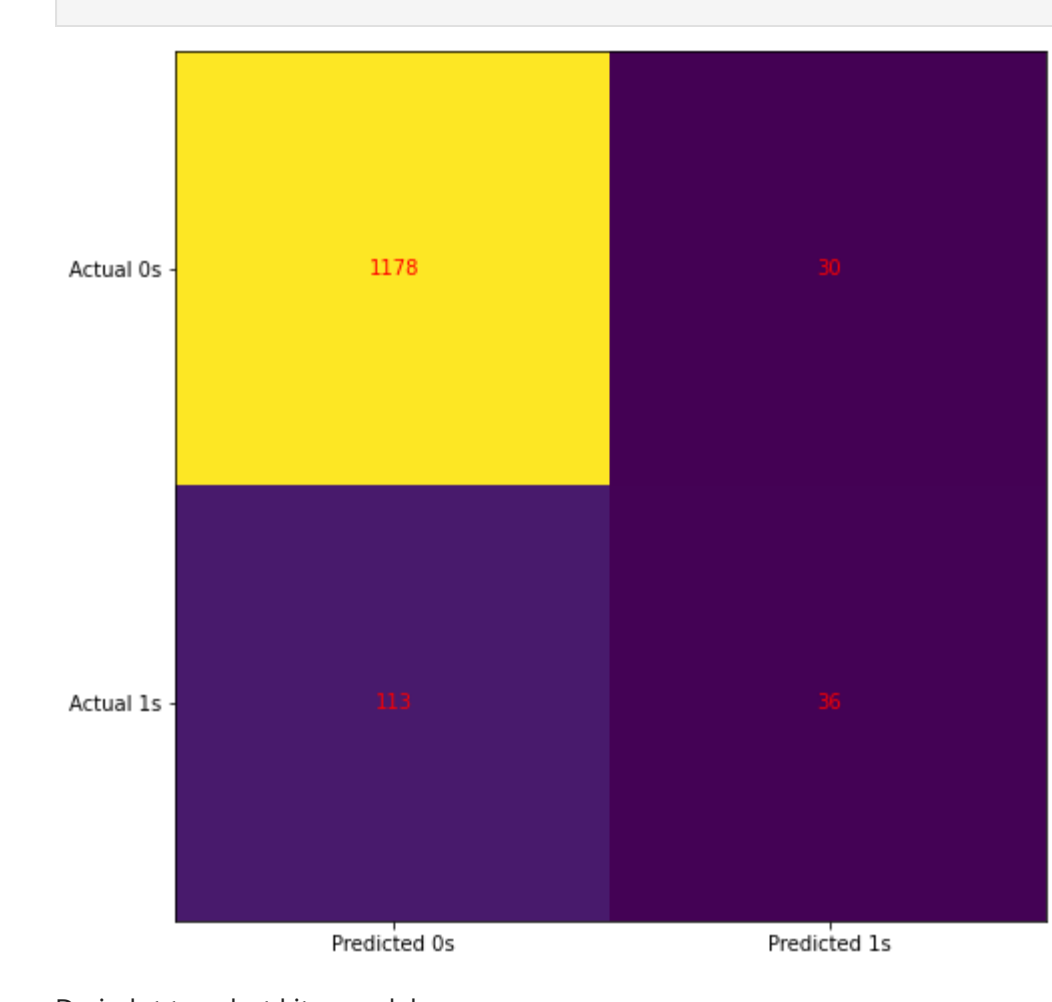
In [18]:

```
# Import package
from sklearn.linear_model import LogisticRegression

# Membuat LogisticRegression Classifier
lr = LogisticRegression()

#Training model
lr.fit(X_train,y_train)

#Predict respon untuk dataset test
pred_lr = lr.predict(X_test)

# Melihat akurasi model
print("Model Score: ", metrics.accuracy_score(y_test,pred_lr))

#Menampilkan confusion matrix
matrix_lr = confusion_matrix(y_test,pred_lr)
print(matrix_lr)

#menampilkan classification report
print(metrics.classification_report(y_test,pred_lr))
```

```
Model Score:  0.894620486366986
[[1178   30]
 [ 113   36]]
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      1208
           1       0.55      0.24      0.33       149

    accuracy                           0.89      1357
   macro avg       0.73      0.61      0.64      1357
weighted avg       0.87      0.89      0.88      1357
```

In [19]:

```
# Visualisasi confusion matrix
cm = metrics.confusion_matrix(y_test, pred_lr)

fig, ax = plt.subplots(figsize=(8,8))

ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)

for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')

plt.show()
```



Dari plot tersebut kita peroleh:

- 1178 prediksi negatif yang benar
- 113 prediksi negatif yang salah
- 30 prediksi positif yang salah
- 36 prediksi positif yang benar

Decision Tree

In [20]:

```
#Import packages
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Membuat DecisionTreeClassifier
dt = tree.DecisionTreeClassifier()

#Training model
dt.fit(X_train,y_train)

#Predict the response for test dataset
pred_dt = dt.predict(X_test)

# Melihat akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_dt))
matrix_dt = confusion_matrix(y_test,pred_dt)
print(matrix_dt)
```

```
Accuracy Score:  0.8614591009579956
[[1109   99]
 [  89   60]]
```

Random Forest

In [21]:

```
#Import packages
from sklearn.ensemble import RandomForestClassifier

#Membuat random forest classifier
rf = RandomForestClassifier()

#Training data
rf.fit(X_train,y_train)

#Predict the response for test dataset
pred_rf=rf.predict(X_test)

# Melihat akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_rf))
matrix_rf = confusion_matrix(y_test,pred_rf)
print(matrix_rf)
print(classification_report(y_test,pred_rf))
```

```
Accuracy Score:  0.8960943257184967
[[1175   33]
 [ 108   41]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      1208
           1       0.55      0.28      0.37       149

    accuracy                           0.90      1357
   macro avg       0.73      0.62      0.66      1357
weighted avg       0.88      0.90      0.88      1357
```

Support Vector Machine

In [22]:

```
#import package
from sklearn import svm

#Membuat svm classifier
sv = svm.SVC()

#Training data
sv.fit(X_train,y_train)

#Predict the response for test dataset
pred_sv = sv.predict(X_test)

#melihat akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_sv))
matrix_sv = confusion_matrix(y_test,pred_sv)
print(matrix_sv)
print(classification_report(y_test,pred_sv))
```

```
Accuracy Score:  0.899042004421518
[[1196   12]
 [ 125   24]]
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      1208
           1       0.67      0.16      0.26       149

    accuracy                           0.90      1357
   macro avg       0.79      0.58      0.60      1357
weighted avg       0.88      0.90      0.87      1357
```

Naive Bayes

In [23]:

```
#import package
from sklearn.naive_bayes import GaussianNB

#Membuat svm classifier
nb = GaussianNB()

#training data
nb.fit(X_train,y_train)

#Predict the response for test dataset
pred_nb = nb.predict(X_test)

#melihat akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_nb))
matrix_nb = confusion_matrix(y_test,pred_nb)
print(matrix_nb)
print(classification_report(y_test,pred_nb))
```

```
Accuracy Score:  0.8518791451731761
[[1096  112]
 [  89   60]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.92      1208
           1       0.35      0.40      0.37       149

    accuracy                           0.85      1357
   macro avg       0.64      0.65      0.64      1357
weighted avg       0.86      0.85      0.86      1357
```

K-Nearest Neighbors

In [24]:

```
#import packages
from sklearn.neighbors import KNeighborsClassifier

#membuat KNeighbors Classifier
kn = KNeighborsClassifier()

# training data
kn.fit(X_train,y_train)

#predict respons for test dataset
pred_kn = kn.predict(X_test)

#Melihat Akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_kn))
matrix_kn = confusion_matrix(y_test,pred_kn)
print(matrix_kn)
print(classification_report(y_test,pred_kn))
```

```
Accuracy Score:  0.8968312453942521
[[1185   23]
 [ 117   32]]
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      1208
           1       0.58      0.21      0.31       149

    accuracy                           0.90      1357
   macro avg       0.75      0.60      0.63      1357
weighted avg       0.87      0.90      0.87      1357
```

In [25]:

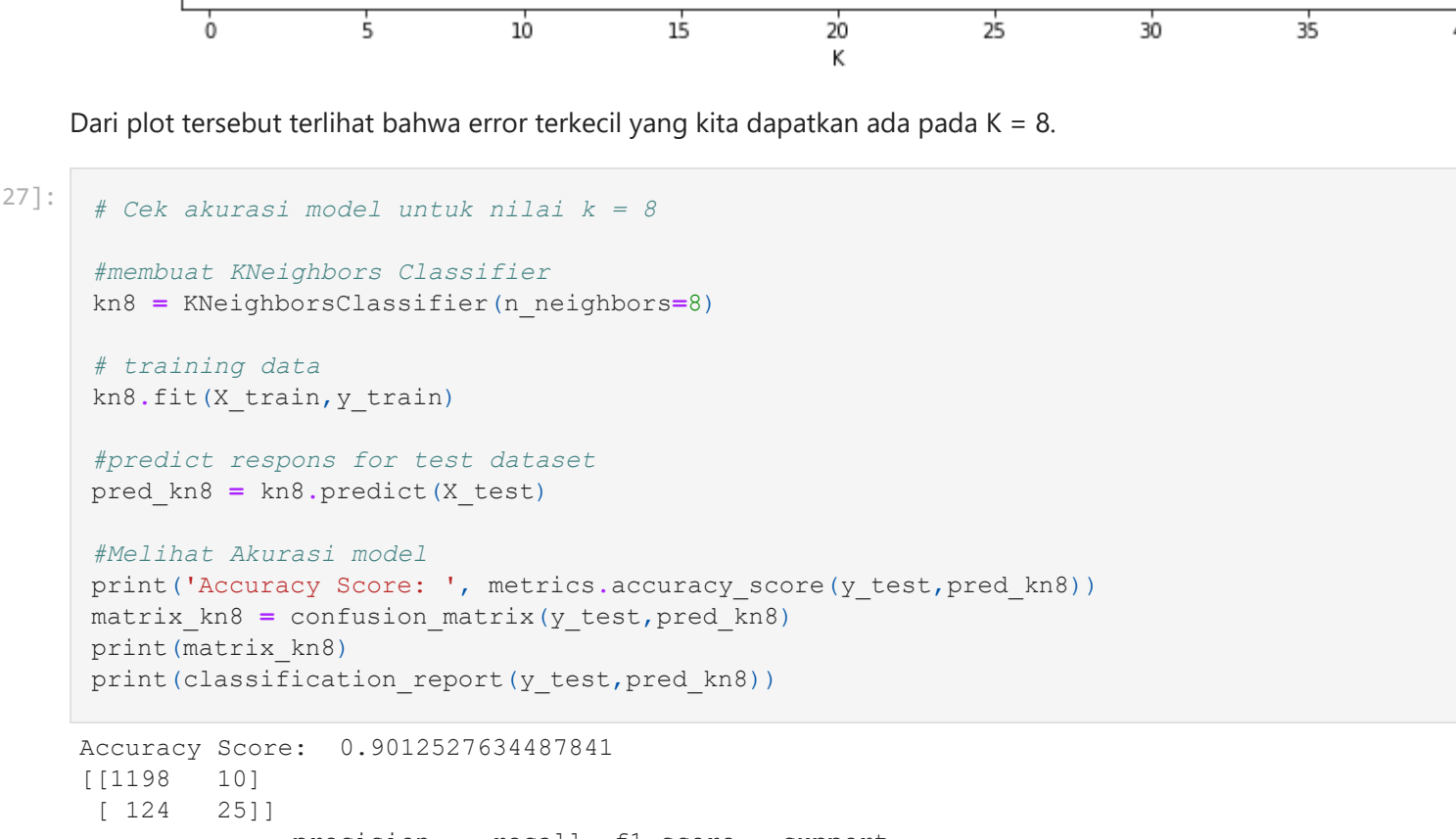
```
# improve model dan mencari tahu nilai k yang optimal

error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    kn = KNeighborsClassifier(n_neighbors=i)
    kn.fit(X_train, y_train)
    pred_i = kn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

In [26]:

```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='brown', linestyle='dashed', marker='o',
         markerfacecolor='black', markersize=10)
plt.title('Error Rate K')
plt.xlabel('K')
plt.ylabel('Error mean')
```

Out[26]:

```
Text(0, 0.5, 'Error mean')
```



Dari plot tersebut terlihat bahwa error terkecil yang kita dapatkan ada pada K = 8.

In [27]:

```
# Cek akurasi model untuk nilai k = 8

#membuat KNeighbors Classifier
kn8 = KNeighborsClassifier(n_neighbors=8)

# training data
kn8.fit(X_train,y_train)

#predict respons for test dataset
pred_kn8 = kn8.predict(X_test)

#Melihat Akurasi model
print('Accuracy Score: ', metrics.accuracy_score(y_test,pred_kn8))
matrix_kn8 = confusion_matrix(y_test,pred_kn8)
print(matrix_kn8)
print(classification_report(y_test,pred_kn8))
```

```
Accuracy Score:  0.9012527634487841
[[1198   10]
 [ 124   25]]
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      1208
           1       0.71      0.17      0.27       149

    accuracy                           0.90      1357
   macro avg       0.81      0.58      0.61      1357
weighted avg       0.89      0.90      0.87      1357
```

Algorithm Analysis

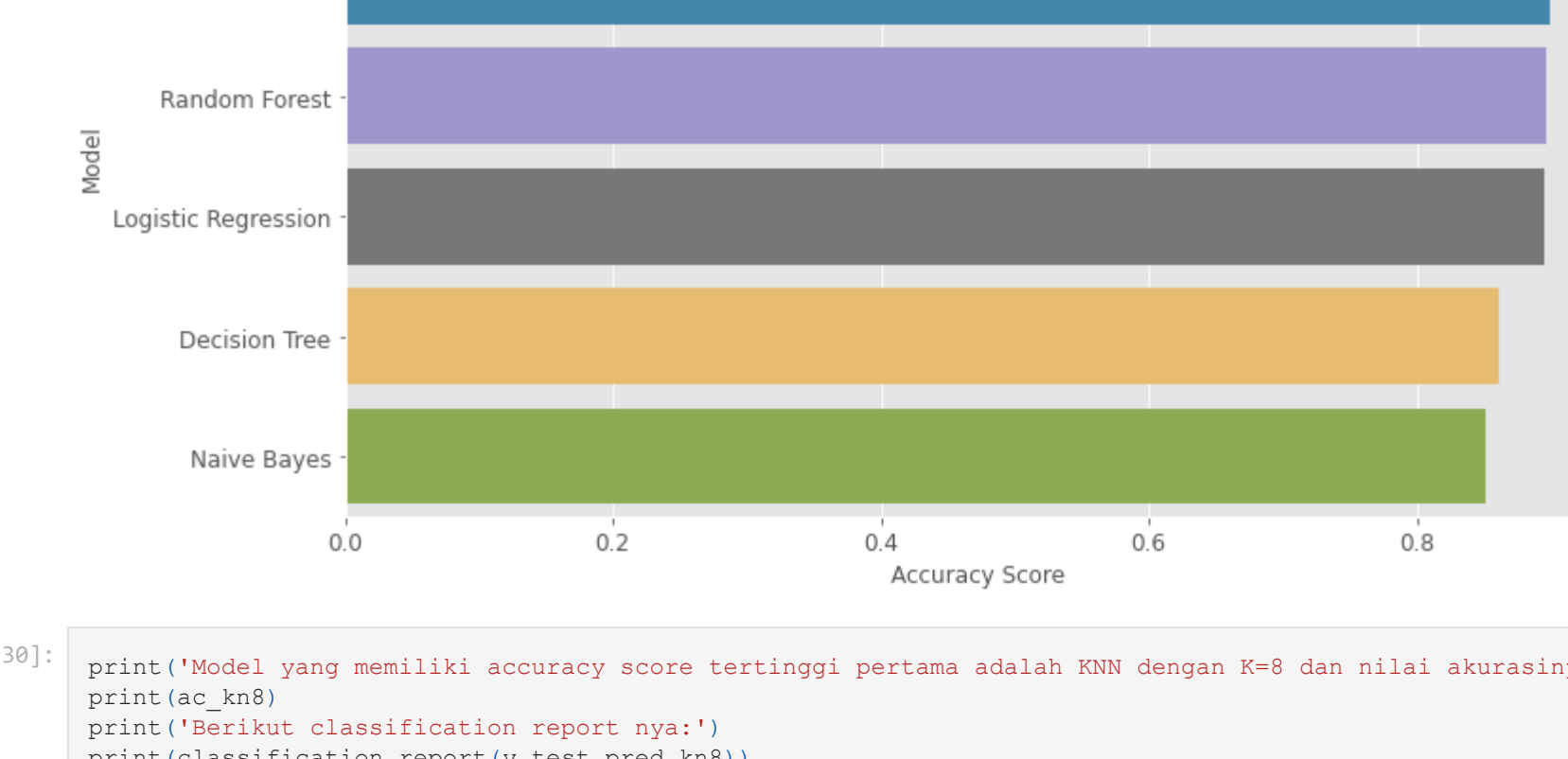
In [28]:

```
# Melihat akurasi masing-masing Model

ac_lr = accuracy_score(y_test, pred_lr)
ac_dt = accuracy_score(y_test, pred_dt)
ac_rf = accuracy_score(y_test, pred_rf)
ac_sv = accuracy_score(y_test, pred_sv)
ac_nb = accuracy_score(y_test, pred_nb)
ac_kn8 = accuracy_score(y_test, pred_kn8)


print('Accuracy Score:')
print('            ')
print("Logistic Regression:", ac_lr)
print("Decision Tree:", ac_dt)
print("Random Forest:", ac_rf)
print("SVM:", ac_sv)
print("Naive Bayes:", ac_nb)
print("KNN:", ac_kn8)
```

```
Accuracy Score:

Logistic Regression: 0.894620486366986
Decision Tree: 0.8614591009579956
Random Forest: 0.8960943257184967
SVM: 0.899042004421518
Naive Bayes: 0.8518791451731761
KNN: 0.9012527634487841
```

In [29]:

```
# Membuat Bar Plot dari nilai akurasi masing-masing model

import seaborn as sns
from matplotlib import style
style.use('ggplot')

warna = ['greenyellow', 'orange', 'lightcoral', 'lightskyblue', 'aqua', 'brown')
model = ['Logistic Regression','Decision Tree','Random Forest','SVM','Naive Bayes','KNN']
accuracy_score = [ac_lr, ac_dt, ac_rf, ac_sv, ac_nb, ac_kn8]
df_acc = pd.DataFrame({'Model':model,
                        'Accuracy Score':accuracy_score
                       })
df_acc = df_acc.sort_values(['Accuracy Score'], ascending=False)

plt.figure(figsize=(12,7))
sns.barplot(x='Accuracy Score',
            y='Model',
            data = df_acc,
            estimator=sum,
            ci=None)


plt.xlabel('Accuracy Score') # add to x-label to the plot
plt.ylabel('Model') # add y-label to the plot
plt.title('Accuracy Score of Each Model') # add title to the plot
plt.xticks(size=12)
plt.yticks(size=12)

plt.show()
```



In [30]:

```
print('Model yang memiliki accuracy score tertinggi pertama adalah KNN dengan K=8 dan nilai akurasinya:')
print(ac_kn8)
print('Berikut classification report nya:')
print(classification_report(y_test,pred_kn8))
print('Dari classification report dapat diketahui bahwa nilai precision dan recall nya:')
print('Precision : ', metrics.precision_score(y_test, pred_kn8))
print('Recall : ', metrics.recall_score(y_test, pred_kn8))
```

```
Model yang memiliki accuracy score tertinggi pertama adalah KNN dengan K=8 dan nilai akurasinya:
0.9012527634487841
Berikut classification report nya:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      1208
           1       0.71      0.17      0.27       149

    accuracy                           0.90      1357
   macro avg       0.81      0.58      0.61      1357
weighted avg       0.89      0.90      0.87      1357

Dari classification report dapat diketahui bahwa nilai precision dan recall nya:
Precision :  0.7142857142857143
Recall :  0.16778523489932887
```

Jadi, model yang dipilih adalah model KNN dengan nilai k=8. Model ini dipilih karena:

1. Memiliki nilai akurasi tertinggi, dimana KNN terkenal dengan tingkat keakuratan yang baik.
2. KNN sangat nonlinear, lebih mudah dipahami dan diimplementasikan
3. Saat nilai k=8, model ini memiliki nilai error yang paling sedikit daripada saat menggunakan nilai k yang lain dalam range(1,40)

In [32]:

```
# Menyimpan model di pickle
import pickle
pickle.dump(kn8, open('./model_classifier.pkl','wb'))
```