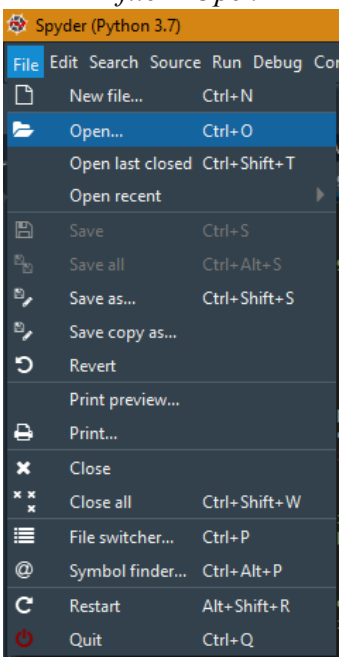
TUTORIAL MENJALANKAN PROGRAM
UAS BIG DATA – KELAS A

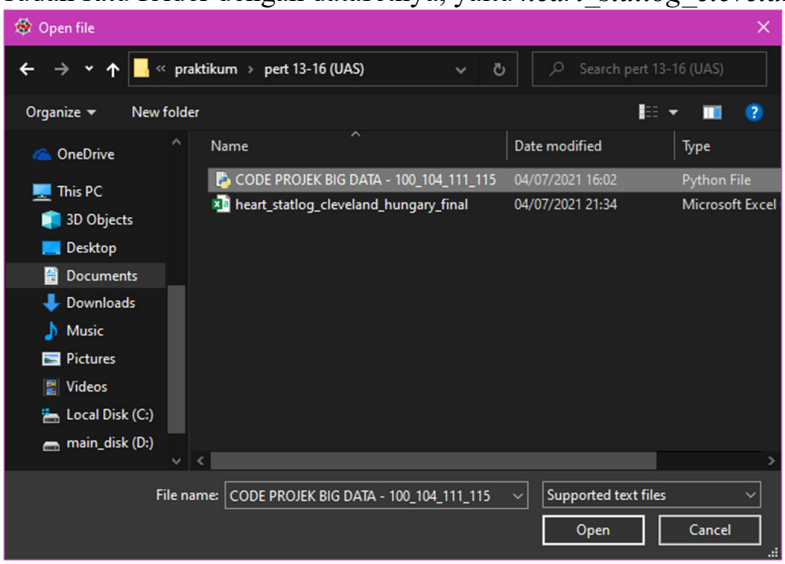
Irza Ramira Putra	1810511100
Deo Haganta Depari	1810511104
Nadhifa Zhafira	1810511111
Quina Alifa	1810511115

---

1. Buka Spyder.
2. Lalu klik *file* > *Open*

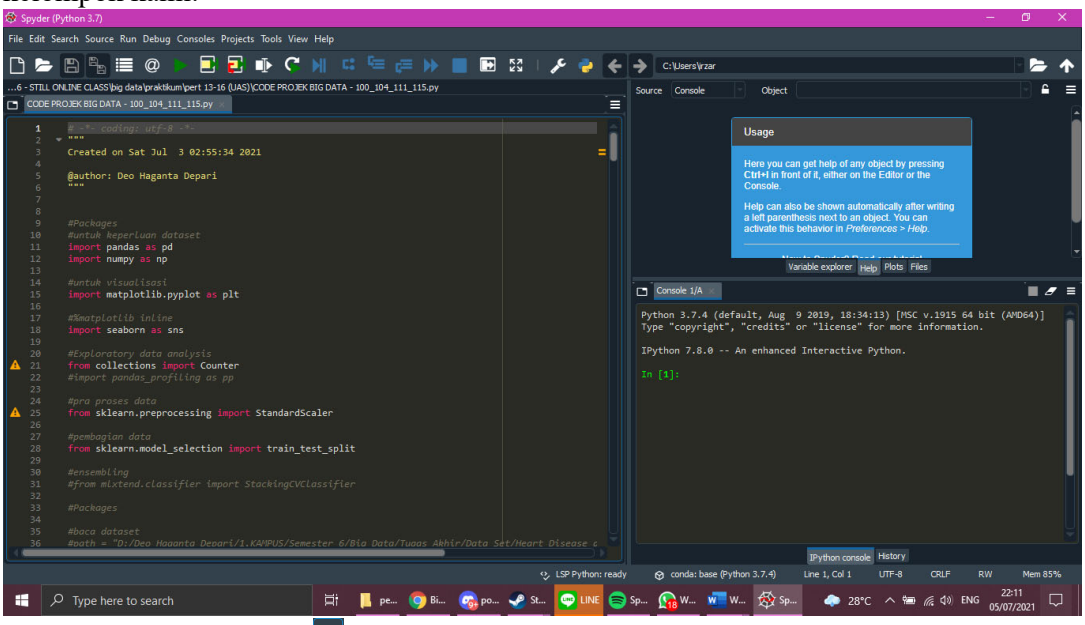


3. Pilih file *CODE PROJEK BIG DATA - 100_104_111_115.py*. Pastikan codingan kami sudah satu folder dengan datasetnya, yaitu *heart_statlog_cleveland_hungary_final.csv*



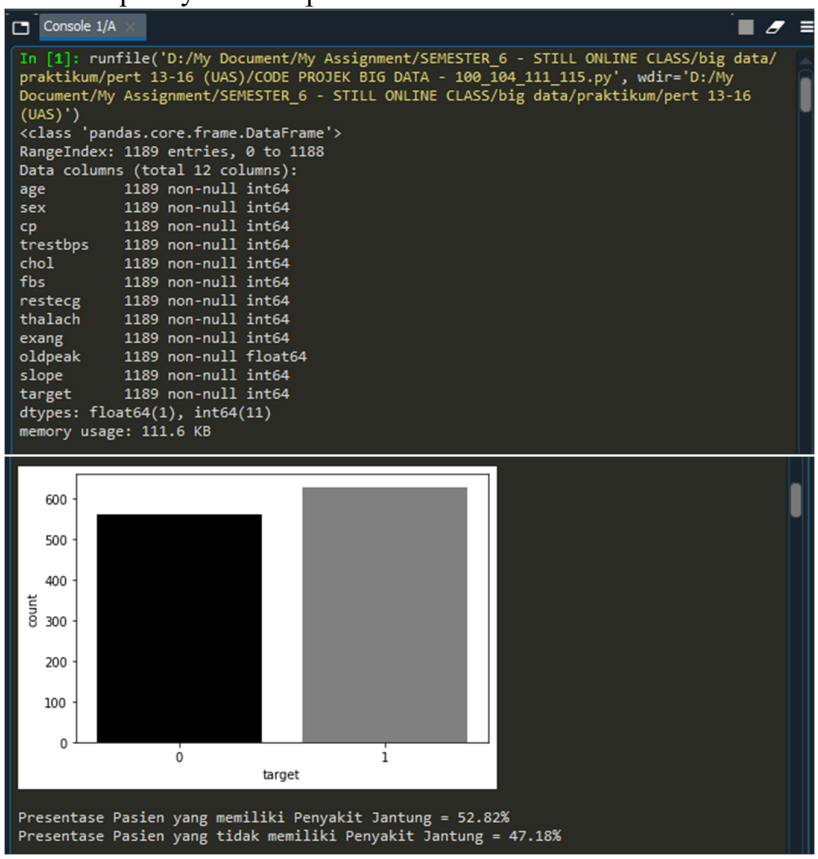
Lalu klik *Open*

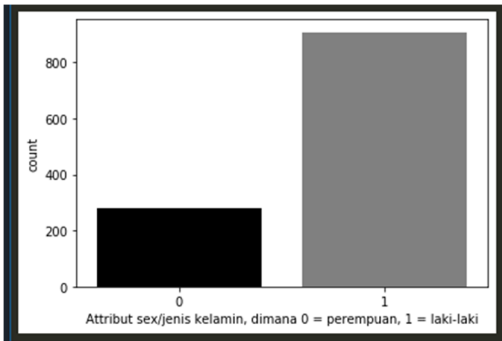
4. Setelah terbuka file pythonnya, tampilan spyder akan terlihat seperti ini dengan codingan kelompok kami.



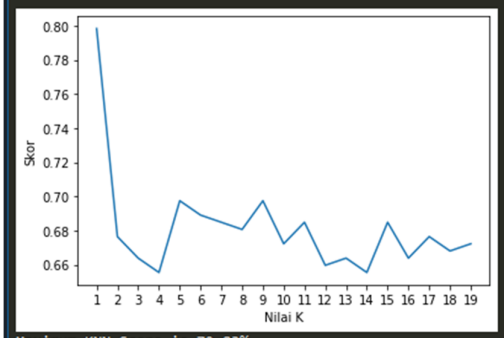
Maka bisa di klik tombol  (*Run File*) atau bisa menekan tombol) atau bisa menekan tombol *F5*.

5. Maka output nya akan seperti ini :






```
Presentase Pasien Perempuan = 23.63%
Presentase Pasien Laki-Laki = 76.37%
1 KNN Score: 79.83%
```



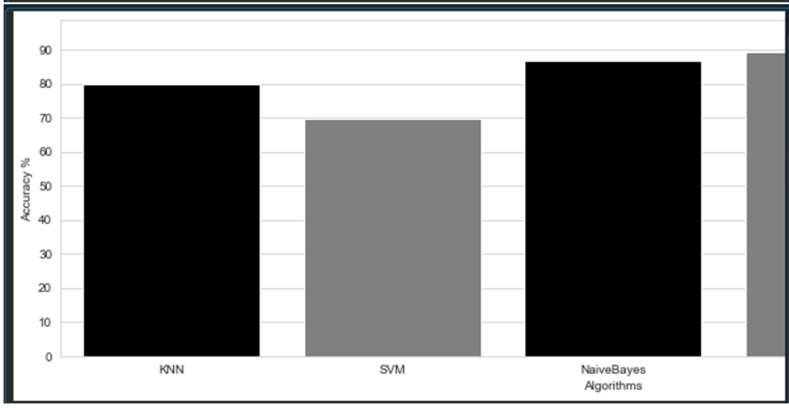

```
Maximum KNN Score is 79.83%

Testing Akurasi Algoritma SVM : 69.75%

Testing Akurasi Algoritma Naive Bayes : 86.97%

Testing Akurasi Algoritma Decision Tree : 89.50%

Testing Akurasi Algoritma Random Forrest : 94.12%
```



```
[[459   0]
 [  0 492]]
KNN CLASSIFICATION REPORT TRAIN DATA:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    459.0  492.0       1.0      951.0         951.0
```

```
[[ 78  24]
 [ 24 112]]
KNN CLASSIFICATION REPORT TEST DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.764706    0.823529  0.798319    0.794118      0.798319
recall       0.764706    0.823529  0.798319    0.794118      0.798319
f1-score     0.764706    0.823529  0.798319    0.794118      0.798319
support    102.000000  136.000000  0.798319  238.000000    238.000000
```

```
[[350 109]
 [160 332]]
SVM CLASSIFICATION REPORT TRAIN DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.686275    0.752834   0.71714    0.719554      0.720709
recall       0.762527    0.674797   0.71714    0.718662      0.717140
f1-score     0.722394    0.711683   0.71714    0.717038      0.716853
support    459.000000  492.000000   0.71714  951.000000    951.000000
```

```
[[78 24]
 [48 88]]
SVM CLASSIFICATION REPORT TEST DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.619048    0.785714  0.697479    0.702381      0.714286
recall       0.764706    0.647059  0.697479    0.705882      0.697479
f1-score     0.684211    0.709677  0.697479    0.696944      0.698763
support    102.000000  136.000000  0.697479  238.000000    238.000000
```

```
[[391  68]
 [ 88 404]]
NaiveBayes CLASSIFICATION REPORT TRAIN DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.816284    0.855932  0.835962    0.836108      0.836796
recall       0.851852    0.821138  0.835962    0.836495      0.835962
f1-score     0.833689    0.838174  0.835962    0.835931      0.836009
support    459.000000  492.000000  0.835962  951.000000    951.000000
```

```
[[ 89  13]
 [ 18 118]]
NaiveBayes CLASSIFICATION REPORT TEST DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.831776    0.900763  0.869748    0.866270      0.871197
recall       0.872549    0.867647  0.869748    0.870098      0.869748
f1-score     0.851675    0.883895  0.869748    0.867785      0.870086
support    102.000000  136.000000  0.869748  238.000000    238.000000
```

```
[[459   0]
 [  0 492]]
Decision Tree CLASSIFICATION REPORT TRAIN DATA:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    459.0  492.0       1.0      951.0         951.0
```

```
[[ 88  14]
 [ 11 125]]
Decision Tree CLASSIFICATION REPORT TEST DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.888889    0.899281  0.894958    0.894085      0.894827
recall       0.862745    0.919118  0.894958    0.890931      0.894958
f1-score     0.875622    0.909091  0.894958    0.892356      0.894747
support    102.000000  136.000000  0.894958  238.000000    238.000000
```

```
[[459   0]
 [  0 492]]
Random Forest CLASSIFICATION REPORT TRAIN DATA:
               0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    459.0  492.0       1.0      951.0         951.0
```

```
[[ 91  11]
 [  3 133]]
Random Forest CLASSIFICATION REPORT Test DATA:
                    0           1  accuracy   macro avg  weighted avg
precision    0.968085    0.923611  0.941176    0.945848      0.942671
recall       0.892157    0.977941  0.941176    0.935049      0.941176
f1-score     0.928571    0.950000  0.941176    0.939286      0.940816
support    102.000000  136.000000  0.941176  238.000000    238.000000
```

Tutorial Program :

1. Menyiapkan library awal yang dibutuhkan

```
#Packages
#untuk keperluan dataset
import pandas as pd
import numpy as np

#untuk visualisasi
import matplotlib.pyplot as plt

#%matplotlib inline
import seaborn as sns

#Exploratory data analysis
from collections import Counter
#import pandas_profiling as pp

#pra proses data
from sklearn.preprocessing import StandardScaler

#pembagian data
from sklearn.model_selection import train_test_split
```

2. Membaca Data

```
path_heart = r"heart_statlog_cleveland_hungary_final.csv"
data = pd.read_csv(path_heart)
```

3. Exploratory Data Analysis

```
#Descriptive statistics
data.describe()

## ↓↓ Exploratory data analysis ↓↓
data.target.value_counts()

#deskripsi dan plot target
sns.countplot(x = "target", data = data, palette = ['black', 'grey'])
plt.show()

jumlahTidakSakit = len(data[data.target == 0])
jumlahSakit = len(data[data.target == 1])
print('')
print("Presentase Pasien yang memiliki Penyakit Jantung = {:.2f}%".
      format((jumlahSakit / (len(data.target))* 100 )))

print("Presentase Pasien yang tidak memiliki Penyakit Jantung = {:.2f}%".
      format((jumlahTidakSakit / (len(data.target))* 100 )))

#deskripsi dan plot sex
sns.countplot(x='sex', data=data , palette = ['black', 'grey'])
plt.xlabel ("Attribut sex/jenis kelamin, dimana 0 = perempuan, 1 = laki-laki")
plt.show()



jumlahperempuan = len(data[data.sex == 0])
jumlahlakilaki = len(data[data.sex == 1])
print('')
print("Presentase Pasien Perempuan = {:.2f}%".
      format((jumlahperempuan / (len(data.sex))* 100 )))

print("Presentase Pasien Laki-Laki = {:.2f}%".
      format((jumlahlakilaki / (len(data.sex))* 100 )))

## ↑↑ Exploratory data analysis ↑↑
```

4. Persiapan model, dan Membuat Variabel Dummy, membantu meningkatkan nilai akurasi, serta membagi data menjadi Training dan Target, serta di transpose

```
#Persiapkan model
y = data.target.values
x = data.drop(['target'],axis=1)

#Membuat Variabel Dummy, membantu meningkatkan nilai akurasi
#Dikarenakan 'cp', 'thal' dan 'slope' atribut merupakan variable kategori maka bisa

# dummy_cp = pd.get_dummies(data['cp'], prefix = "cp")
# dummy_thal = pd.get_dummies(data['thal'], prefix = "thal")
dummy_slope = pd.get_dummies(data['slope'], prefix = "slope")

#memasukkan frames dummy ke dataframe
frames = [data, dummy_slope]
data = pd.concat(frames, axis = 1)
data.head()

#drop atribut yang telah di ganti dengan dummy data
data = data.drop(columns = ['slope'])
data.head()

#normalisasi data
x_normalize = (x - np.min(x)) / (np.max(x) - np.min(x)).values

#memastikan data sudah di split sebelum di apply ke algoritma
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_stat

#transpose matrices
x_train = x_train.T
y_train = y_train.T
x_test = x_test.T
y_test = y_test.T
```

5. Membuat array untuk menyimpan akurasi masing-masing model

```
#Machine Learning model
akurasitotal = {}
```

6. Model K-Nearest Neighbors (KNN)

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 1)  # n_neighbors means k
knn.fit(x_train.T, y_train.T)
prediksi = knn.predict(x_test.T)

print("{} KNN Score: {:.2f}%".format(1, knn.score(x_test.T, y_test.T)*100))


    # mencari nilai n_terbaik
hasilpercobaan = []
for i in range(1,20):
    knn_exp = KNeighborsClassifier(n_neighbors = i) #n_neighbors berarti k = 2
    knn_exp.fit(x_train.T, y_train.T)
    hasilpercobaan.append(knn_exp.score(x_test.T, y_test.T))

plt.plot(range(1,20), hasilpercobaan)
plt.xticks(np.arange(1,20,1))
plt.xlabel("Nilai K")
plt.ylabel("Skor")
plt.show()

akurasi_KNN = max(hasilpercobaan)*100
akurasitotal['KNN'] = akurasi_KNN
print("Maximum KNN Score is {:.2f}%".format(akurasi_KNN))
```

7. Model Support Vector Machine (SVM)

```
#Support Vector Machine
from sklearn.svm import SVC

svm = SVC(random_state = 1)
svm.fit(x_train.T, y_train.T)

akurasi_SVM = svm.score(x_test.T, y_test.T)*100
akurasitotal['SVM'] = akurasi_SVM
print("\nTesting Akurasi Algoritma SVM : {:.2f}%".format(akurasi_SVM))
```

8. Model Naïve Bayes

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
naivebayes = GaussianNB()
naivebayes.fit(x_train.T, y_train.T)

akurasi_NaiveBayes = naivebayes.score(x_test.T, y_test.T)*100
akurasitotal['NaiveBayes'] = akurasi_NaiveBayes
print("\nTesting Akurasi Algoritma Naive Bayes : {:.2f}%".format(akurasi_NaiveBayes))
```

9. Model Decision Tree

```
#Decision Tree Algoritma
from sklearn.tree import DecisionTreeClassifier
decisiontree = DecisionTreeClassifier()
decisiontree.fit(x_train.T, y_train.T)

akurasi_decisiontree = decisiontree.score(x_test.T, y_test.T)*100
akurasitotal['DecisionTree'] = akurasi_decisiontree
print("\nTesting Akurasi Algoritma Decision Tree : {:.2f}%".format(akurasi_decisiontree))
```

10. Model Random Forest

```
#Random Forest Classification
from sklearn.ensemble import RandomForestClassifier
randomforest = RandomForestClassifier(n_estimators = 1000, random_state = 1)
randomforest.fit(x_train.T, y_train.T)

akurasi_randomforest = randomforest.score(x_test.T, y_test.T)*100
akurasitotal['Random Forrest'] = akurasi_randomforest
print("\nTesting Akurasi Algoritma Random Forrest : {:.2f}%".format(akurasi_randomforest))
```

11. Evaluasi Semua Model

```
#Confusion Matrix
from sklearn.metrics import confusion_matrix, classification_report

#Nilai prediksi
#KNN EVALUATION
knn_conf = KNeighborsClassifier(n_neighbors = 1)
knn_conf.fit(x_train.T, y_train.T)
y_head_knn_train = knn_conf.predict(x_train.T)
y_head_knn_test = knn_conf.predict(x_test.T)

confusionmatrix_knn_train = confusion_matrix(y_train, y_head_knn_train)
print(confusionmatrix_knn_train)
clf_report_knn_train = pd.DataFrame(classification_report(y_train.T,y_head_knn_train, output_dict = True))
print("KNN CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_knn_train)
print("_____")

confusionmatrix_knn_test = confusion_matrix(y_test, y_head_knn_test)
print(confusionmatrix_knn_test)
clf_report_knn_test = pd.DataFrame(classification_report(y_test.T,y_head_knn_test, output_dict = True))
print("KNN CLASSIFICATION REPORT TEST DATA:\n",clf_report_knn_test)
print("_____")
```

```
#SVM EVALUATION
y_head_svm_train = svm.predict(x_train.T)
y_head_svm_test = svm.predict(x_test.T)

confusionmatrix_svm_train = confusion_matrix(y_train, y_head_svm_train)
print(confusionmatrix_svm_train)
clf_report_svm_train = pd.DataFrame(classification_report(y_train.T,y_head_svm_train, output_dict = True))
print("SVM CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_svm_train)
print("_____________________________________________________________")

confusionmatrix_svm_test = confusion_matrix(y_test, y_head_svm_test)
print(confusionmatrix_svm_test)
clf_report_svm_test = pd.DataFrame(classification_report(y_test.T,y_head_svm_test, output_dict = True))
print("SVM CLASSIFICATION REPORT TEST DATA:\n",clf_report_svm_test)
print("_____________________________________________________________")

#NaiveBayes EVALUATION
y_head_naivebayes_train = naivebayes.predict(x_train.T)
y_head_naivebayes_test = naivebayes.predict(x_test.T)

confusionmatrix_naivebayes_train = confusion_matrix(y_train, y_head_naivebayes_train)
print(confusionmatrix_naivebayes_train)
clf_report_naivebayes_train = pd.DataFrame(classification_report(y_train.T,y_head_naivebayes_train, output_dict = True))
print("NaiveBayes CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_naivebayes_train)
print("_____________________________________________________________")

confusionmatrix_naivebayes_test = confusion_matrix(y_test, y_head_naivebayes_test)
print(confusionmatrix_naivebayes_test)
clf_report_naivebayes_test = pd.DataFrame(classification_report(y_test.T,y_head_naivebayes_test, output_dict = True))
print("NaiveBayes CLASSIFICATION REPORT TEST DATA:\n",clf_report_naivebayes_test)
print("_____________________________________________________________")
```

```
#Decision Tree EVALUATION
y_head_decisiontree_train = decisiontree.predict(x_train.T)
y_head_decisiontree_test = decisiontree.predict(x_test.T)

confusionmatrix_decisiontree_train = confusion_matrix(y_train, y_head_decisiontree_train)
print(confusionmatrix_decisiontree_train)
clf_report_decisiontree_train = pd.DataFrame(classification_report(y_train.T,y_head_decisiontree_train, output_dict = True)
print("Decision Tree CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_decisiontree_train)
print("_____________________________________________________________")

confusionmatrix_decisiontree_test = confusion_matrix(y_test, y_head_decisiontree_test)
print(confusionmatrix_decisiontree_test)
clf_report_decisiontree_test = pd.DataFrame(classification_report(y_test.T,y_head_decisiontree_test, output_dict = True))
print("Decision Tree CLASSIFICATION REPORT TEST DATA:\n",clf_report_decisiontree_test)
print("_____________________________________________________________")

#Random Forest EVALUATION
y_head_randomforest_train = randomforest.predict(x_train.T)
y_head_randomforest_test = randomforest.predict(x_test.T)

confusionmatrix_randomforest_train = confusion_matrix(y_train, y_head_randomforest_train)
print(confusionmatrix_randomforest_train)
clf_report_randomforest_train = pd.DataFrame(classification_report(y_train.T,y_head_randomforest_train, output_dict = True))
print("Random Forest CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_randomforest_train)
print("_____________________________________________________________")

confusionmatrix_randomforest_test = confusion_matrix(y_test, y_head_randomforest_test)
print(confusionmatrix_randomforest_test)
clf_report_randomforest_train = pd.DataFrame(classification_report(y_test.T,y_head_randomforest_test, output_dict = True))
print("Random Forest CLASSIFICATION REPORT Test DATA:\n",clf_report_randomforest_train)
print("_____________________________________________________________")
```

12. Plot Hasil Evaluasi

```
#plotting
#KNN Plot
plt.figure(figsize=(6,6))

plt.suptitle("KNN Confusion Matrixes",fontsize=18)
plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

plt.subplot(1,2,1)
plt.title("Train Data",fontsize=12)
sns.heatmap(confusionmatrix_knn_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

plt.subplot(1,2,2)
plt.title("Test Data",fontsize=12)
sns.heatmap(confusionmatrix_knn_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

#SVM Plot
plt.figure(figsize=(6,6))

plt.suptitle("SVM Confusion Matrixes",fontsize=18)
plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

plt.subplot(1,2,1)
plt.title("Train Data",fontsize=12)
sns.heatmap(confusionmatrix_svm_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

plt.subplot(1,2,2)
plt.title("Test Data",fontsize=12)
sns.heatmap(confusionmatrix_svm_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
```

```
    #NaiveBayes Plot
    plt.figure(figsize=(6,6))

    plt.suptitle("Naive Bayes Confusion Matrixes",fontsize=18)
    plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

    plt.subplot(1,2,1)
    plt.title("Train Data",fontsize=12)
    sns.heatmap(confusionmatrix_naivebayes_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(1,2,2)
    plt.title("Test Data",fontsize=12)
    sns.heatmap(confusionmatrix_naivebayes_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    #Decision Tree Plot
    plt.figure(figsize=(6,6))

    plt.suptitle("Decision Tree Confusion Matrixes",fontsize=18)
    plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

    plt.subplot(1,2,1)
    plt.title("Train Data",fontsize=12)
    sns.heatmap(confusionmatrix_decisiontree_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(1,2,2)
    plt.title("Test Data",fontsize=12)
    sns.heatmap(confusionmatrix_decisiontree_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
```

```
    #Random Forest Plot
    plt.figure(figsize=(6,6))

    plt.suptitle("Random Forest Confusion Matrixes",fontsize=18)
    plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

    plt.subplot(1,2,1)
    plt.title("Train Data",fontsize=12)
    sns.heatmap(confusionmatrix_randomforest_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(1,2,2)
    plt.title("Test Data",fontsize=12)
    sns.heatmap(confusionmatrix_randomforest_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
```

```
    #Comparison Train Data Plot
    plt.figure(figsize=(24,12))

    plt.suptitle("Confusion Matrixes",fontsize=24)
    plt.subplots_adjust(wspace = 0.4, hspace= 0.4)

    plt.subplot(3,2,1)
    plt.title("K Nearest Neighbors Confusion Matrix")
    sns.heatmap(confusionmatrix_knn_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(3,2,2)
    plt.title("Support Vector Machine Confusion Matrix")
    sns.heatmap(confusionmatrix_svm_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(3,2,3)
    plt.title("Naive Bayes Confusion Matrix")
    sns.heatmap(confusionmatrix_naivebayes_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(3,2,4)
    plt.title("Decision Tree Confusion Matrix")
    sns.heatmap(confusionmatrix_decisiontree_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

    plt.subplot(3,2,5)
    plt.title("Random Forest Confusion Matrix")
    sns.heatmap(confusionmatrix_randomforest_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
```