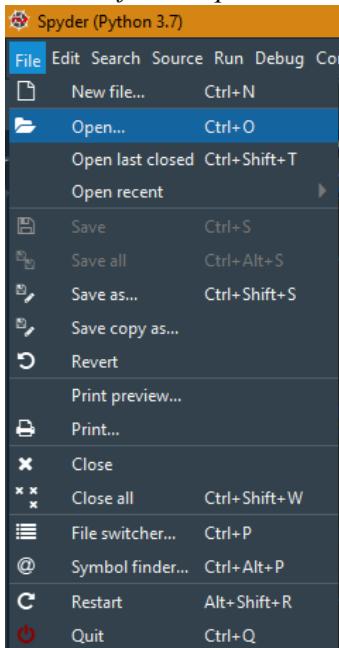


TUTORIAL MENJALANKAN PROGRAM

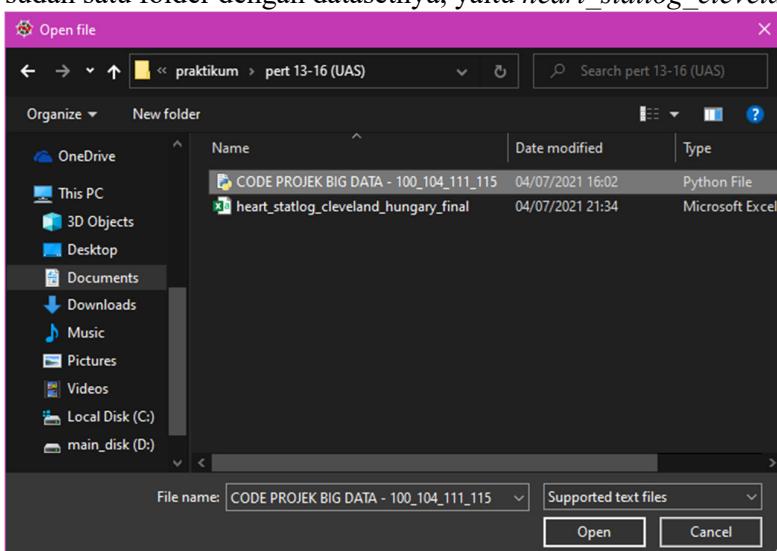
UAS BIG DATA – KELAS A

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.

```

1 # -*- coding: utf-8 -*-
2 #
3 # Created on Sat Jul 3 02:55:34 2021
4 #
5 #Author: Deo Hagenta Depari
6 #
7 #
8 #
9 #Packages
10 #untuk keperluan dataset
11 import pandas as pd
12 import numpy as np
13
14 #untuk visualisasi
15 import matplotlib.pyplot as plt
16
17 #matplotlib inline
18 import seaborn as sns
19
20 #Exploratory data analysis
21 from collections import Counter
22 #import pandas_profiling as pp
23
24 #proses data
25 from sklearn.preprocessing import StandardScaler
26
27 #pembagian data
28 from sklearn.model_selection import train_test_split
29
30 #ensembling
31 #from elxtend.classifier import StackingCVClassifier
32
33 #Packages
34
35 #baca dataset
36 #path = "D:/Deo Hagenta Depari/1.KAMI/PUS/Semester 6/Bio Data/Tugas Akhir/Data Set/Heart Disease"

```

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

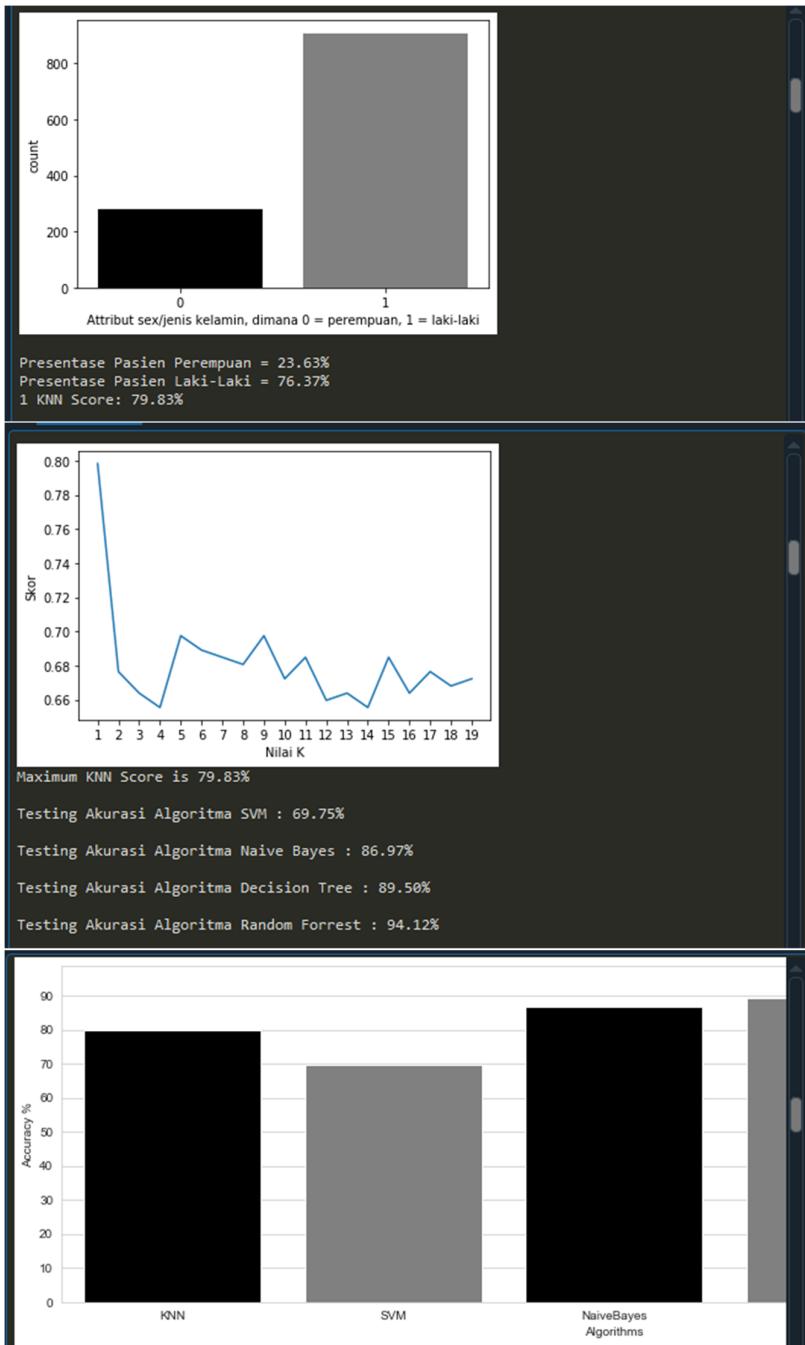
5. Maka output nya akan seperti ini :

```

In [4]: runfile('D:/My Document/My Assignment/SEMESTER_6 - STILL ONLINE CLASS/big data/praktikum/pert 13-16 (UAS)/CODE PROJEK BIG DATA - 100_104_111_115.py', wdir='D:/My Document/My Assignment/SEMESTER_6 - STILL ONLINE CLASS/big data/praktikum/pert 13-16 (UAS)')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1189 entries, 0 to 1188
Data columns (total 12 columns):
age      1189 non-null int64
sex      1189 non-null int64
cp       1189 non-null int64
trestbps 1189 non-null int64
chol     1189 non-null int64
fbs      1189 non-null int64
restecg  1189 non-null int64
thalach  1189 non-null int64
exang    1189 non-null int64
oldpeak  1189 non-null float64
slope    1189 non-null int64
target   1189 non-null int64
dtypes: float64(1), int64(11)
memory usage: 111.6 KB

```

Presentase Pasien yang memiliki Penyakit Jantung = 52.82%
Presentase Pasien yang tidak memiliki Penyakit Jantung = 47.18%



```

[[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

```
9     #Packages
10    #untuk keperluan dataset
11    import pandas as pd
12    import numpy as np
13
14    #untuk visualisasi
15    import matplotlib.pyplot as plt
16
17    #%matplotlib inline
18    import seaborn as sns
19
20    #Exploratory data analysis
21    from collections import Counter
22    #import pandas_profiling as pp
23
24    #pra proses data
25    from sklearn.preprocessing import StandardScaler
26
27    #pembagian data
28    from sklearn.model_selection import train_test_split
```

2. Membaca Data

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

3. Exploratory Data Analysis

```
49      #Descriptive statistics
50      data.describe()
51
52      ## ↓↓ Exploratory data analysis ↓↓
53      data.target.value_counts()
54
55      #deskripsi dan plot target
56      sns.countplot(x = "target", data = data, palette = ['black', 'grey'])
57      plt.show()
58
59      jumlahTidakSakit = len(data[data.target == 0])
60      jumlahSakit = len(data[data.target == 1])
61      print('')
62      ▼ print("Presentase Pasien yang memiliki Penyakit Jantung = {:.2f}%".
63              format((jumlahSakit / (len(data.target))* 100 )))
64
65      ▼ print("Presentase Pasien yang tidak memiliki Penyakit Jantung = {:.2f}%".
66              format((jumlahTidakSakit / (len(data.target))* 100 )))
67
68      #deskripsi dan plot sex
69      sns.countplot(x='sex', data=data , palette = ['black', 'grey'])
70      plt.xlabel ("Attribut sex/jenis kelamin, dimana 0 = perempuan, 1 = laki-laki")
71      plt.show()
72
73
74
75      jumlahperempuan = len(data[data.sex == 0])
76      jumlahlakilaki = len(data[data.sex == 1])
77      print('')
78      ▼ print("Presentase Pasien Perempuan = {:.2f}%".
79              format((jumlahperempuan / (len(data.sex))* 100 )))
80
81      ▼ print("Presentase Pasien Laki-Laki = {:.2f}%".
82              format((jumlahlakilaki / (len(data.sex))* 100 )))
83
84      ## ↑↑ Exploratory data analysis ↑↑
```

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

```

86     #Persiapkan model
87     y = data.target.values
88     x = data.drop(['target'],axis=1)
89
90     #Membuat Variabel Dummy, membantu meningkatkan nilai akurasi
91     #Dikarenakan 'cp', 'thal' dan 'slope' atribut merupakan variable kategori maka bisa
92
93     # dummy_cp = pd.get_dummies(data['cp'], prefix = "cp")
94     # dummy_thal = pd.get_dummies(data['thal'], prefix = "thal")
95     dummy_slope = pd.get_dummies(data['slope'], prefix = "slope")
96
97     #memasukkan frames dummy ke dataframe
98     frames = [data, dummy_slope]
99     data = pd.concat(frames, axis = 1)
100    data.head()
101
102    #drop atribut yang telah di ganti dengan dummy data
103    data = data.drop(columns = ['slope'])
104    data.head()
105
106    #normalisasi data
107    x_normalize = (x - np.min(x)) / (np.max(x) - np.min(x)).values
108
109    #memastikan data sudah di split sebelum di apply ke algoritma
110    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state = 42)
111
112    #transpose matrices
113    x_train = x_train.T
114    y_train = y_train.T
115    x_test = x_test.T
116    y_test = y_test.T

```

5. Membuat array untuk menyimpan akurasi masing-masing model

```

119     #Machine Learning model
120     akurasitotal = {}

```

6. Model K-Nearest Neighbors (KNN)

```

123    #KNN
124    from sklearn.neighbors import KNeighborsClassifier
125    knn = KNeighborsClassifier(n_neighbors = 1) # n_neighbors means k
126    knn.fit(x_train.T, y_train.T)
127    prediksi = knn.predict(x_test.T)
128
129    print("{} KNN Score: {:.2f}%".format(1, knn.score(x_test.T, y_test.T)*100))
130
131
132    # mencari nilai n_terbaik
133    hasilpercobaan = []
134    for i in range(1,20):
135        knn_exp = KNeighborsClassifier(n_neighbors = i) #n_neighbors berarti k = 2
136        knn_exp.fit(x_train.T, y_train.T)
137        hasilpercobaan.append(knn_exp.score(x_test.T, y_test.T))
138
139    plt.plot(range(1,20), hasilpercobaan)
140    plt.xticks(np.arange(1,20,1))
141    plt.xlabel("Nilai K")
142    plt.ylabel("Skor")
143    plt.show()
144
145    akurasi_KNN = max(hasilpercobaan)*100
146    akurasitotal['KNN'] = akurasi_KNN
147    print("Maximum KNN Score is {:.2f}%".format(akurasi_KNN))

```

7. Model Support Vector Machine (SVM)

```
151 #Support Vector Machine  
152 from sklearn.svm import SVC  
153  
154 svm = SVC(random_state = 1)  
155 svm.fit(x_train.T, y_train.T)  
156  
157 akurasi_SVM = svm.score(x_test.T, y_test.T)*100  
158 akurasitotal['SVM'] = akurasi_SVM  
159 print("\nTesting Akurasi Algoritma SVM : {:.2f}%".format(akurasi_SVM))
```

8. Model Naïve Bayes

```
161 #Naive Bayes
162 from sklearn.naive_bayes import GaussianNB
163 naivebayes = GaussianNB()
164 naivebayes.fit(x_train.T, y_train.T)
165
166 akurasi_NaiveBayes = naivebayes.score(x_test.T, y_test.T)*100
167 akurasisitotal['NaiveBayes'] = akurasi_NaiveBayes
168 print("\nTesting Akurasi Algoritma Naive Bayes : {:.2f}%".format(akurasi_NaiveBayes))
```

9. Model Decision Tree

```
170 #Decision Tree Algoritma
171 from sklearn.tree import DecisionTreeClassifier
172 decisiontree = DecisionTreeClassifier()
173 decisiontree.fit(x_train.T, y_train.T)
174
175 akurasi_decisiontree = decisiontree.score(x_test.T, y_test.T)*100
176 akurasitotal['DecisionTree'] = akurasi_decisiontree
177 print("\nTesting Akurasi Algoritma Decision Tree : {:.2f}%".format(akurasi_decisiontree))
```

10. Model Random Forest

```
179 #Random Forest Classification
180 from sklearn.ensemble import RandomForestClassifier
181 randomforest = RandomForestClassifier(n_estimators = 1000, random_state = 1)
182 randomforest.fit(x_train.T, y_train.T)
183
184 akurasi_randomforest = randomforest.score(x_test.T, y_test.T)*100
185 akurasitotal['Random Forrest'] = akurasi_randomforest
186 print("\nTesting Akurasi Algoritma Random Forrest : {:.2f}%".format(akurasi_randomforest))
```

11. Evaluasi Semua Model

```
200 #Confusion Matrix
201 from sklearn.metrics import confusion_matrix, classification_report
202
203 #Nilai predksi
204 #KNN EVALUATION
205 knn_conf = KNeighborsClassifier(n_neighbors = 1)
206 knn_conf.fit(x_train.T, y_train.T)
207 y_head_knn_train = knn_conf.predict(x_train.T)
208 y_head_knn_test = knn_conf.predict(x_test.T)
209
210 confusionmatrix_knn_train = confusion_matrix(y_train, y_head_knn_train)
211 print(confusionmatrix_knn_train)
212 clf_report_knn_train = pd.DataFrame(classification_report(y_train.T,y_head_knn_train, output_dict = True))
213 print("KNN CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_knn_train)
214 print("_____")
215
216 confusionmatrix_knn_test = confusion_matrix(y_test, y_head_knn_test)
217 print(confusionmatrix_knn_test)
218 clf_report_knn_test = pd.DataFrame(classification_report(y_test.T,y_head_knn_test, output_dict = True))
219 print("KNN CLASSIFICATION REPORT TEST DATA:\n",clf_report_knn_test)
220 print("_____")
```

```

223 #SVM EVALUATION
224 y_head_svm_train = svm.predict(x_train.T)
225 y_head_svm_test = svm.predict(x_test.T)
226
227 confusionmatrix_svm_train = confusion_matrix(y_train, y_head_svm_train)
228 print(confusionmatrix_svm_train)
229 clf_report_svm_train = pd.DataFrame(classification_report(y_train.T,y_head_svm_train, output_dict = True))
230 print("SVM CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_svm_train)
231 print("_____")
232
233 confusionmatrix_svm_test = confusion_matrix(y_test, y_head_svm_test)
234 print(confusionmatrix_svm_test)
235 clf_report_svm_test = pd.DataFrame(classification_report(y_test.T,y_head_svm_test, output_dict = True))
236 print("SVM CLASSIFICATION REPORT TEST DATA:\n",clf_report_svm_test)
237 print("_____")
238
239 #NaiveBayes EVALUATION
240 y_head_naivebayes_train = naivebayes.predict(x_train.T)
241 y_head_naivebayes_test = naivebayes.predict(x_test.T)
242
243 confusionmatrix_naivebayes_train = confusion_matrix(y_train, y_head_naivebayes_train)
244 print(confusionmatrix_naivebayes_train)
245 clf_report_naivebayes_train = pd.DataFrame(classification_report(y_train.T,y_head_naivebayes_train, output_dict = True))
246 print("NaiveBayes CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_naivebayes_train)
247 print("_____")
248
249 confusionmatrix_naivebayes_test = confusion_matrix(y_test, y_head_naivebayes_test)
250 print(confusionmatrix_naivebayes_test)
251 clf_report_naivebayes_test = pd.DataFrame(classification_report(y_test.T,y_head_naivebayes_test, output_dict = True))
252 print("NaiveBayes CLASSIFICATION REPORT TEST DATA:\n",clf_report_naivebayes_test)
253 print("_____")
254
255 #Decision Tree EVALUATION
256 y_head_decisiontree_train = decisiontree.predict(x_train.T)
257 y_head_decisiontree_test = decisiontree.predict(x_test.T)
258
259 confusionmatrix_decisiontree_train = confusion_matrix(y_train, y_head_decisiontree_train)
260 print(confusionmatrix_decisiontree_train)
261 clf_report_decisiontree_train = pd.DataFrame(classification_report(y_train.T,y_head_decisiontree_train, output_dict = True))
262 print("Decision Tree CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_decisiontree_train)
263 print("_____")
264
265 confusionmatrix_decisiontree_test = confusion_matrix(y_test, y_head_decisiontree_test)
266 print(confusionmatrix_decisiontree_test)
267 clf_report_decisiontree_test = pd.DataFrame(classification_report(y_test.T,y_head_decisiontree_test, output_dict = True))
268 print("Decision Tree CLASSIFICATION REPORT TEST DATA:\n",clf_report_decisiontree_test)
269 print("_____")
270
271
272 #Random Forest EVALUATION
273 y_head_randomforest_train = randomforest.predict(x_train.T)
274 y_head_randomforest_test = randomforest.predict(x_test.T)
275
276 confusionmatrix_randomforest_train = confusion_matrix(y_train, y_head_randomforest_train)
277 print(confusionmatrix_randomforest_train)
278 clf_report_randomforest_train = pd.DataFrame(classification_report(y_train.T,y_head_randomforest_train, output_dict = True))
279 print("Random Forest CLASSIFICATION REPORT TRAIN DATA:\n",clf_report_randomforest_train)
280 print("_____")
281
282 confusionmatrix_randomforest_test = confusion_matrix(y_test, y_head_randomforest_test)
283 print(confusionmatrix_randomforest_test)
284 clf_report_randomforest_train = pd.DataFrame(classification_report(y_test.T,y_head_randomforest_test, output_dict = True))
285 print("Random Forest CLASSIFICATION REPORT Test DATA:\n",clf_report_randomforest_train)
286 print("_____")

```

12. Plot Hasil Evaluasi

```

289 #plotting
290 #KNN Plot
291 plt.figure(figsize=(6,6))
292
293 plt.suptitle("KNN Confusion Matrixes",fontsize=18)
294 plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
295
296 plt.subplot(1,2,1)
297 plt.title("Train Data",fontsize=12)
298 sns.heatmap(confusionmatrix_knn_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
299
300 plt.subplot(1,2,2)
301 plt.title("Test Data",fontsize=12)
302 sns.heatmap(confusionmatrix_knn_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
303
304 #SVM Plot
305 plt.figure(figsize=(6,6))
306
307 plt.suptitle("SVM Confusion Matrixes",fontsize=18)
308 plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
309
310 plt.subplot(1,2,1)
311 plt.title("Train Data",fontsize=12)
312 sns.heatmap(confusionmatrix_svm_train,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})
313
314 plt.subplot(1,2,2)
315 plt.title("Test Data",fontsize=12)
316 sns.heatmap(confusionmatrix_svm_test,annot=True,cmap="gray",fmt="d",cbar=False, annot_kws={"size": 24})

```

```

318     #NaiveBayes Plot
319     plt.figure(figsize=(6,6))
320
321     plt.suptitle("Naive Bayes Confusion Matrixes", fontsize=18)
322     plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
323
324     plt.subplot(1,2,1)
325     plt.title("Train Data", fontsize=12)
326     sns.heatmap(confusionmatrix_naivebayes_train, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
327
328     plt.subplot(1,2,2)
329     plt.title("Test Data", fontsize=12)
330     sns.heatmap(confusionmatrix_naivebayes_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
331
332     #Decision Tree Plot
333     plt.figure(figsize=(6,6))
334
335     plt.suptitle("Decision Tree Confusion Matrixes", fontsize=18)
336     plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
337
338     plt.subplot(1,2,1)
339     plt.title("Train Data", fontsize=12)
340     sns.heatmap(confusionmatrix_decisiontree_train, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
341
342     plt.subplot(1,2,2)
343     plt.title("Test Data", fontsize=12)
344     sns.heatmap(confusionmatrix_decisiontree_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
345
346     #Random Forest Plot
347     plt.figure(figsize=(6,6))
348
349     plt.suptitle("Random Forest Confusion Matrixes", fontsize=18)
350     plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
351
352     plt.subplot(1,2,1)
353     plt.title("Train Data", fontsize=12)
354     sns.heatmap(confusionmatrix_randomforest_train, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
355
356     plt.subplot(1,2,2)
357     plt.title("Test Data", fontsize=12)
358     sns.heatmap(confusionmatrix_randomforest_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
359
360     #Comparison Train Data Plot
361     plt.figure(figsize=(24,12))
362
363     plt.suptitle("Confusion Matrixes", fontsize=24)
364     plt.subplots_adjust(wspace = 0.4, hspace= 0.4)
365
366     plt.subplot(3,2,1)
367     plt.title("K Nearest Neighbors Confusion Matrix")
368     sns.heatmap(confusionmatrix_knn_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
369
370     plt.subplot(3,2,2)
371     plt.title("Support Vector Machine Confusion Matrix")
372     sns.heatmap(confusionmatrix_svm_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
373
374     plt.subplot(3,2,3)
375     plt.title("Naive Bayes Confusion Matrix")
376     sns.heatmap(confusionmatrix_naivebayes_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
377
378     plt.subplot(3,2,4)
379     plt.title("Decision Tree Confusion Matrix")
380     sns.heatmap(confusionmatrix_decisiontree_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
381
382     plt.subplot(3,2,5)
383     plt.title("Random Forest Confusion Matrix")
384     sns.heatmap(confusionmatrix_randomforest_test, annot=True, cmap="gray", fmt="d", cbar=False, annot_kws={"size": 24})
385

```