Univerzitet u Tuzli
Fakultet elektrotehnike
Predmet: Inteligentni sistemi
Akademska godina: 2021./2022.
Student: Indir Karabegović
Datum: 03.04.2022. god.

Izvještaj zadaće broj 1

Proizvodnja bakarnih provodnika:

Dataset predstavlja podatke vezane za proizvodnju bakarne žice/provodnika u novembru 2020. godine. Cilj zadatka rada sa ovim setom jeste pronaći uzrok povećanja broja grešaka u proizvodnji bakranih provodnika. Dataset uključuje datume i smjene u kojima se radi, tipove grešaka/defekta itd.

Zadatak br. 1 (klasternig)

Uraditi K-means klastering u više klasa:

A) za proizvoljno odabrane 2 kolone (i plotati rezultate)

B) za sve kolone

Odrediti optimalan broj klastera i obrazložiti zašto je odabran taj broj.

Rješenje:

Prije nego što pristupimo K-means klasteringu, prvo ćemo analizirati dataset s kojim radimo. Informacije o datasetu su prikazane na narednoj slici:

```
indir@indir-ThinkPad:~/Fakultet/IntSys/Zadaca1$ python K-means.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Machine                 149 non-null    int64
 1   Shift                   149 non-null    object
 2   Operator                149 non-null    int64
 3   Date                    149 non-null    object
 4   Cable Failures          149 non-null    int64
 5   Cable Failure Downtime  149 non-null    int64
 6   Other Failures          149 non-null    int64
 7   Other Failure Downtime  149 non-null    int64
dtypes: int64(6), object(2)
memory usage: 9.4+ KB
None
(149, 8)
   Machine Shift  Operator       Date  Cable Failures  Cable Failure Downtime  Other Failures  Other Failure Downtime
0        1     A         1  11/6/2020               1                      35               1                      30
1        2     A         2  11/6/2020               1                      10               3                     150
2        2     B         3  11/6/2020               2                      40               2                     110
3        2     A         2  11/7/2020               5                     120               1                      80
4        2     B         3  11/7/2020               2                      40               1                      35
   Operator  Cable Failure Downtime
0         1                      35
1         2                      10
2         3                      40
3         2                     120
4         3                      40
```

Sa ove slike možemo vidjeti da imamo 8 kolona i 149 redova, pri čemu redovi idu od rednog broja 0 do rednog broja 148. Također, ispisali smo tip podataka za svaku kolonu, te smo nakon toga ispisali i prvih 5 redova dataset-a (da bismo vidjeli kakvu formu ima naš dataset). Specifično je da kolona "Date" i kolona "Shift" imaju tip podatka object. Za date kolone potrebno je izvršiti konverziju tipova podataka. Radi jednostavnosti, konverziju tipa podatka ćemo izvršiti samo za "Shift", dok ćemo kolonu "Date" izbrisati. Traženu analizu možemo uraditi i bez "Date" kolone.

Cilj zadatka pod a) je bio da proizvoljno izaberemo dvije kolone te da za njih vršimo plotanje rezultata. U ovom slučaju smo izabrali kolone "Operator" i "Cable Failure Downtime". Broj klastera smo prvo postavili na 2, te smo ga postepeno povećavali (test urađen na 2,4,8 i 12 klastera). Navedeni rezultati su prikazani na nrednim slikama:

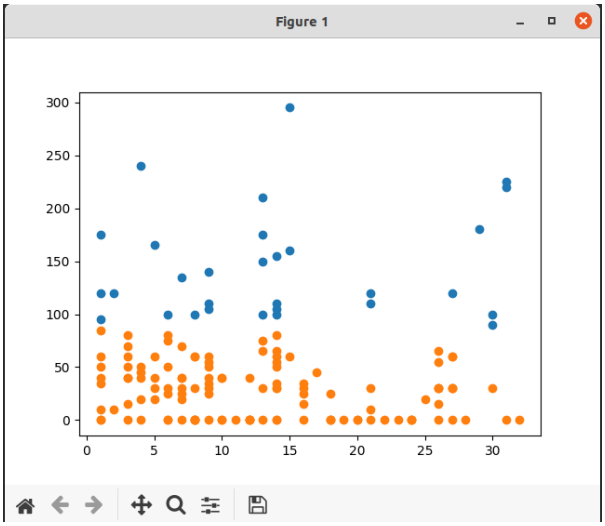


Figure 1: 2 kolone i 2 klastera

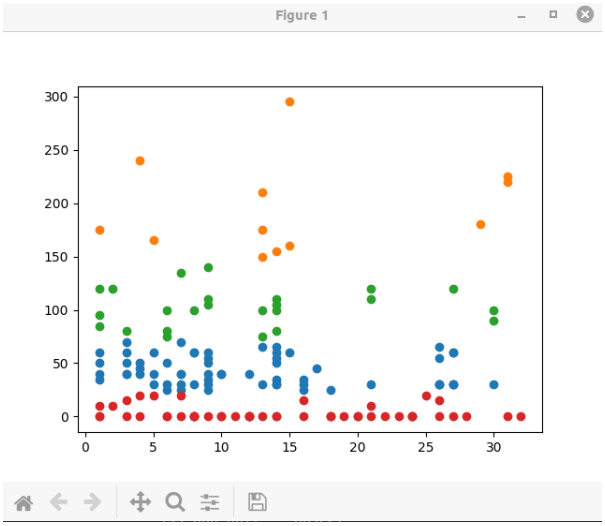


Figure 2: 2 kolone i 2 klastera

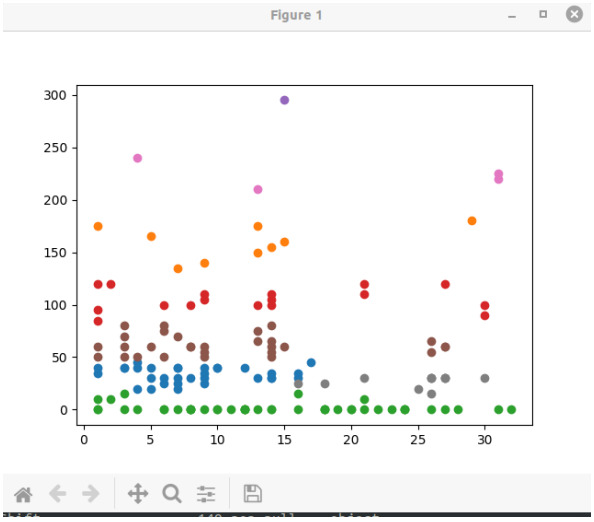


Figure 3: 2 kolone i 8 klastera

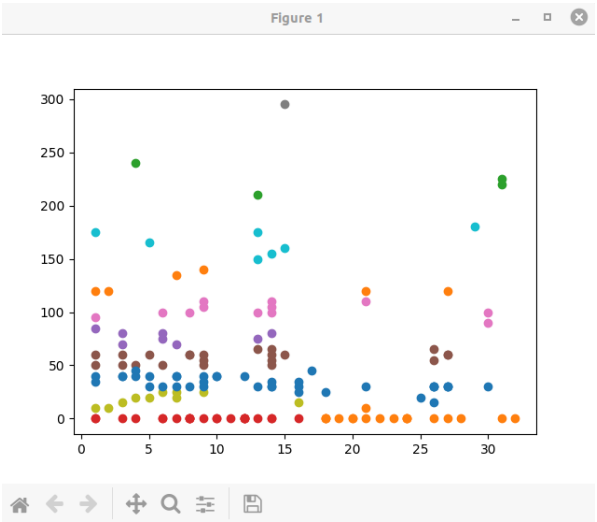


Figure 4: 2 kolone i 12 klastera

Cilj zadatka pod b) jeste da izaberemo sve kolone te da za njih vršimo plotanje rezultata. Kada izaberemo svih 7 kolona (kolonu "Date" smo izbirsali) imamo prevelik broj mogućih kombinacija. Shodno tome, za ispis rezultata smo u ovom slučaju izabrali kolone 0 i 6, te kolone 4 i 6. Za svaku kombinaciju smo testirali 2 i 4 klastera.

Rezultat za kolone 0 i 6:

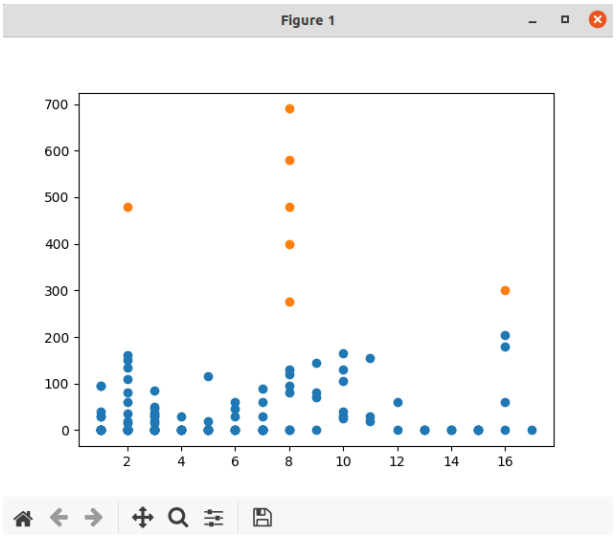


Figure 5: Sve kolone i 2 klastera (0,6)

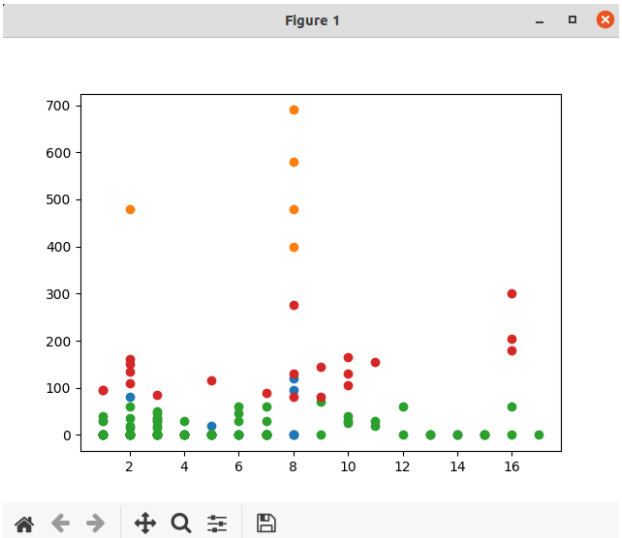


Figure 6: Sve kolone i 4 klastera (0,6)

Rezultati za kolone 4 i 6:

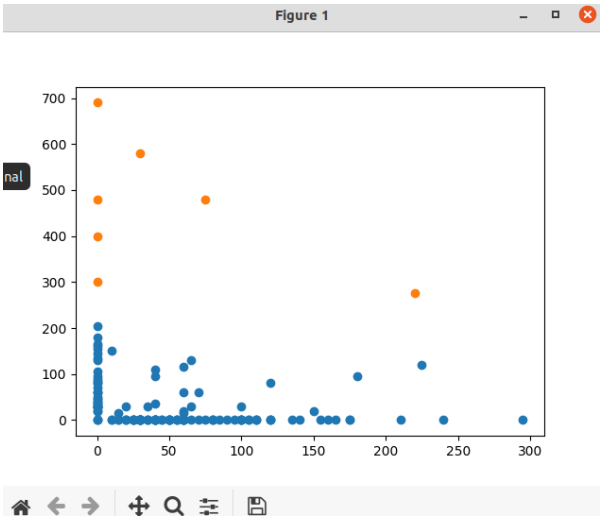


Figure 7: Sve kolone i 2 klastera (4,6)

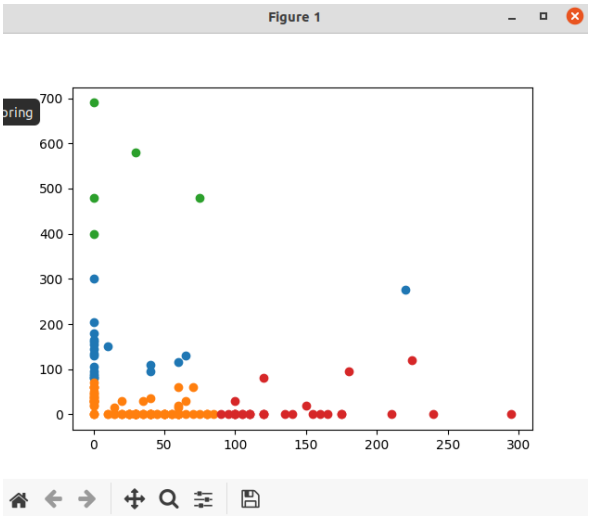


Figure 8: Sve kolone i 4 klastera (4,6)

Kao zaključak ovog zadatka, potrebno je odrediti optimalan broj klastera. Kada određujemo broj klastera koje želimo imati nad našim dataset-om, potrebno je prvo analizirati dataset. Ako pogledamo prvi slučaj (dvije proizvoljne kolone), možemo vidjeti da smo ga testirali na različitom broju klastera. Sa dva klastera imamo jasnu liniju razgraničenja. To je slučaj i sa 4 klastera, ali već su neki elementi blizu jedni drugima (preklapanje dvije boje). Povećanjem broja klastera navedena preklapanja se povećavaju i imamo sve manje jasan prikaz. Sa 12 klastera slika izgleda jako konfuzno. U drugom

slučaju vidimo da već sa povećanjem broja klastera na 4 dobijamo izuzetno čudan prikaz i čudna preklapanja. U prvom slučaju broj klastera koji se može uzeti, a da je pri tome pikazan jasan, je 4. U drugom slučaju, broj klastera koji je veći od 2 nam daje konfuzan prikaz. Na kraju, bitno je napomenuti da broj klastera zavisi od seta do seta podataka s kojim se radi.

Kod za navedeni zadatak je prikazan u nastavku:

```
from sklearn.cluster import KMeans

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# ZA confusion matricu
from sklearn.metrics import confusion_matrix

# UCITAVANJE DATASET-A
data = pd.read_csv("Cable-Production-Line-Dataset.csv")

# ISPISIVANJE INFORMACIJA O DATASET-U
print(data.info())

# ISPISIVANJE DIMENZIJA DATASET-A
print(data.shape)

# ISPISIVANJE PRVIH 5 REDOVA DATASET-A
print(data.head())

# PLOTANJE VRIJEDNOSTI U IZ ODREĐENIH KOLONA
# data['Machine'].value_counts().plot(kind='bar')
# plt.show()
# data['Operator'].value_counts().plot(kind='bar')
# plt.show()
# data['Shift'].value_counts().plot(kind='bar')
# plt.show()

# BRISEMO DATUM JER JE TIPA OBJECT. S OVIM SMO IZBJEGLI KONVERZIJU TIPA PODATAKA
# Date kolona nam nije toliko bitna za realizaciju ovog zadatka
del data['Date']


# UKOLIKO U KOLONI IMAMO TEKSTUALNE VRIJEDNOSTI TREBA IH PRETVORITI U NUMERIČKE
data['Shift'] = [0 if each == "A" else 1 for each in data['Shift']]
```

```
# ----------------------------------------------------------------------------------------
# K-means za prozivoljno odabrane dvije kolone
#
#x = data[['Operator', 'Cable Failure Downtime']]
# print(x.head())

# PREBACIVANJE VRIJEDNOSTI U MATRICU ZA PREDIKCIJU (X) (PANDAS TABELA u
NUMPY MATRICU)
#x = x.values

# Potrebno uzeti razlicit broj klastera, te na osnovu toga vrsiti analizu
# Testirati sa: 2,4,8,12
#model = KMeans(n_clusters=12)
# fit the model
# model.fit(x)
# assign a cluster to each example
#yhat = model.predict(x)
# retrieve unique clusters
#clusters = np.unique(yhat)
# create scatter plot for samples from each cluster
# for cluster in clusters:
# get row indexes for samples with this cluster
# row_ix = np.where(yhat == cluster)
# create scatter of these samples #mijenjati brojeve 2 i 3 da se vide različiti atributi na 2D plotu
# plt.scatter(x[row_ix, 0], x[row_ix, 1])
# show the plot
# plt.show()
# ----------------------------------------------------------------------------------------

#

x = data[['Machine', 'Shift', 'Operator', 'Cable Failures',
'Cable Failure Downtime', 'Other Failures', 'Other Failure Downtime']]
print(x.head())

# PREBACIVANJE VRIJEDNOSTI U MATRICU ZA PREDIKCIJU (X) (PANDAS TABELA u
NUMPY MATRICU)
x = x.values

# Potrebno uzeti razlicit broj klastera, te na osnovu toga vrsiti analizu
# Testirati sa: 2,4
model = KMeans(n_clusters=4)
# fit the model
model.fit(x)
# assign a cluster to each example
```

```
yhat = model.predict(x)
# retrieve unique clusters
clusters = np.unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
# get row indexes for samples with this cluster
row_ix = np.where(yhat == cluster)
# create scatter of these samples
# mijenjati brojeve kolona da se vide različiti atributi na 2D plotu
plt.scatter(x[row_ix, 4], x[row_ix, 6])
# show the plot
plt.show()
# ----------------------------------------------------------------------------------------------------
```

Zadatak br. 2 (klasifikacija)

1. Uraditi KNN, Bayes i SVM klasifikaciju po tome da li je bio “other failure” na mašini ili nije i uporediti rezultate (ROC krivulje)
2. Izdvojiti podatke samo kada je barem jedan “cable failure”
3. Uraditi KNN, Bayes i SVM klasifikaciju po broju “cable failure-a”
    a) Za SVM isprobati različite kernele (linear, polynomial, rbf)
    b) Za KNN isprobati različit broj susjeda i plotati kavalitet estimatora
    c) Usporediti rezultate (confusion matrica) za svaki klasifikator

Napisati komentar i obrazložiti rezultate

Rješenje:

1. Cilj ovog zadatka je bio da uradimo klasifikaciju po tome da li je bio “other failure” na mašini ili nije. Nakon pokretanja koda dobili smo sljedeća rješenja za KNN, SVM I Bayes klasifikaciju:

```
Score for Number of Neighbors= 3: 0.9666666666666667
SVM Classification Score is : 1.0
Naive Bayes Classification Score:1.0
```

Bitan dio koda koji smo morali dodati je prikazan u sljedećem frame-u:

```
data['Other Failures'] = [0 if each ==
0 else 1 for each in data['Other Failures']]
```

Nakon toga smo vršili navedene klasifikacije te dobili sljedeće rezultate:

KNN klasifikacija pri čemu je broj susjeda jednak 3 (n_neighbors = 3):

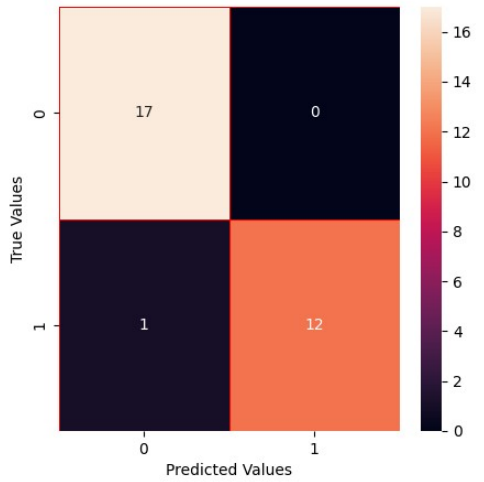


Figure 9: KNN confusion matrica

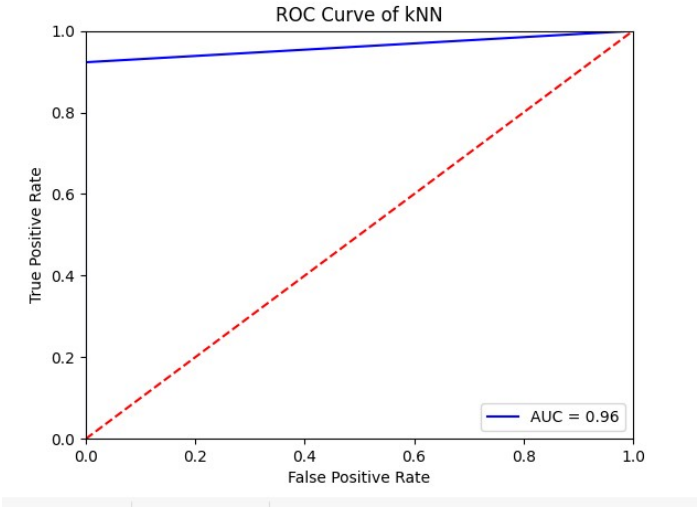


Figure 10: KNN ROC krivulja

SVM klasifikacija pri čemu je izabran linearni kernel i random state je jednak 42:

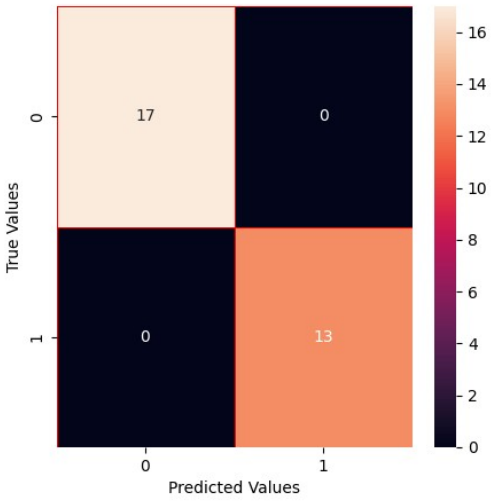


Figure 11: SVM confusion matrica

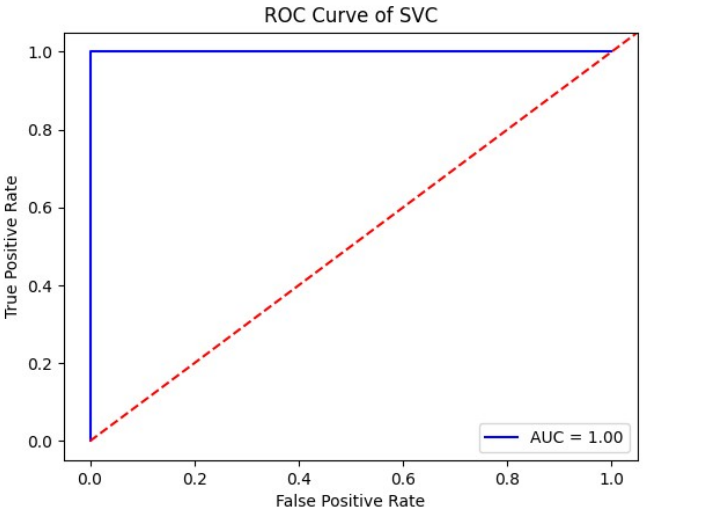


Figure 12: SVM ROC krivulja

Naive Bayes klasifikacija:

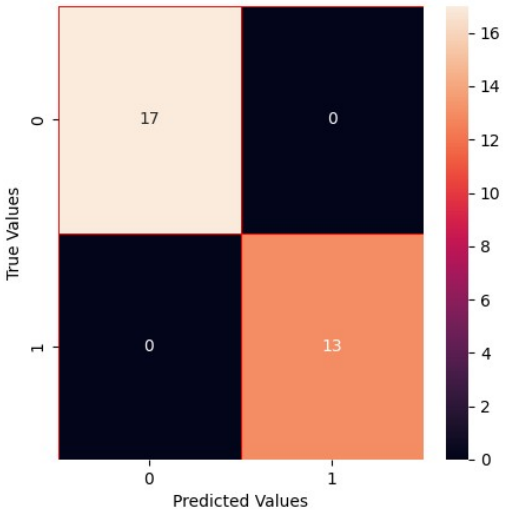


Figure 13: Naive Bayes confusion matrica

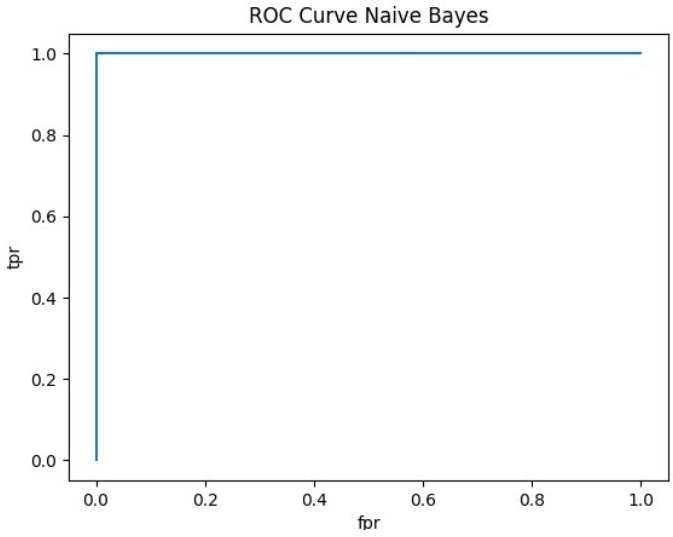


Figure 14: Naive Bayes ROC krivulja

Ukoliko uporedimo ROC krivulje i rezutate koje smo dobili za navedene klasifikatore, možemo primjetiti sljedeće:
Kod KNN klasifikatora (sa 3 susjeda) rezultat koji smo dobili je 0.9666667, što se može vidjeti sa ispisa u terminalu i sa ROC krivulje. Rezultat koji smo dobili za SVM i Bayes klasifikaciju je jednak 1.0000 što opet možemo vidjeti sa ispisa u terminalu i sa ROC krivulje. Kako znamo da ROC krivulja predstavlja kvalitetu klasifikatora kod binarne klasifikacije, možemo zaključiti da su u ovom slučaju SVM i Bayes dali idealne rezultate dok je kod KNN-a nešto lošiji rezultat. Što je veća površina ispod krivulje to je klasifikator bolji. Za SVM i Bayes vidimo da je površina maksimalna, dok je kod KNN-a nešto manja od maksimalne.

Kod za navedeni zadatak je prikazan u nastavku:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
print("Score for Number of Neighbors= 3: {}".format(knn.score(x_test, y_test)))
method_names.append("KNN")

method_scores.append(knn.score(x_test, y_test))
# Confusion Matrix
y_pred = knn.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()
```

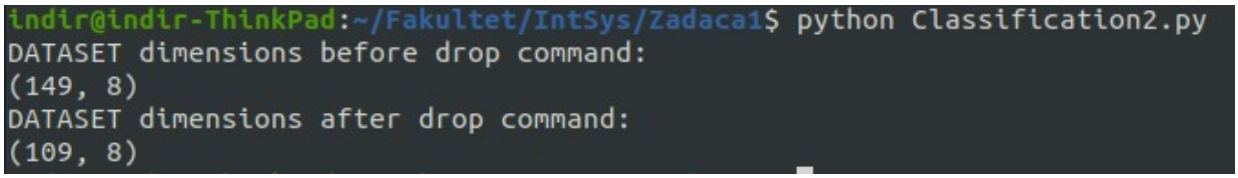
```
#ROC za KNN
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
knn_cv = KNeighborsClassifier(n_neighbors=3)
y_scores = knn.predict_proba(x_test)
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc_auc)
plt.legend(loc='lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of kNN')
plt.show()

#SVC klasifikacija
svm = SVC(random_state=42, kernel='linear', probability=True)
svm.fit(x_train, y_train)
print("SVM Classification Score is : {}".format(svm.score(x_test, y_test)))
method_names.append("SVM")
method_scores.append(svm.score(x_test, y_test))
# Confusion Matrix
y_pred = svm.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()


# ROC za SVM
x_train, x_test, y_train, y_test = train_test_split(
x, y, random_state=42, test_size=0.2)
svc = SVC(random_state=42, probability=True, kernel='linear')
y_scores = svc.fit(x_train, y_train).decision_function(x_test)
fpr, tpr, thresholds = roc_curve(y_test, y_scores[:])
roc_auc = auc(fpr, tpr)
```

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label='AUC = %0.2f' % roc_auc)
plt.legend(loc='lower right')
plt.plot([0, 1.05], [0, 1.05], 'r--')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of SVC')
plt.show()

naive_bayes = GaussianNB()
naive_bayes.fit(x_test, y_test)
print("Naive Bayes Classification Score:{}".format(
naive_bayes.score(x_test, y_test)))
method_names.append("Naive Bayes")
method_scores.append(naive_bayes.score(x_test, y_test))
# Confusion Matrix
y_pred = naive_bayes.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()


#ROC za Bayes-a
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)clf2 = GaussianNB()

clf2.fit(x_test, y_test)
probas = clf2.predict_proba(x_test)
fpr, tpr, thresholds = roc_curve(y_test, probas[:, 1])
# Do not change this code! This plots the ROC curve.
# Just replace the fpr and tpr above with the values from your roc_curve
plt.plot(fpr, tpr, label='NB') # plot the ROC curve
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('ROC Curve Naive Bayes')
plt.show()
print(roc_auc_score(y_test, probas[:, 1]))
```

2. Cilj ovog zadatka jeste da izdvojimo podatke iz dataseta tako da nam u datasetu ostanu podaci kada imamo jedan ili više “cable failures”. To ćemo uraditi tako što ćemo izbrisati sve redove u kojima imamo vrijednost kolone “cable failures” jednaku nuli. Linija koda koja je ključna za ovaj zadatak jeste:

```
data.drop(data[data['Cable Failures'] == 0].index, inplace=True)
```

Nakon pokretanja koda dobili smo sljedeći ispis u terminalu:



Sa slike možemo vidjeti da su se dimenzije dataset-a promijenile, odnosno da smo uspješno izbrisali redove sa vrijednosti “Cable Failures” == 0.

Kod za navedeni zadatak je prikazan u narednom frame-u:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# UCITAVANJE DATASET-A
data = pd.read_csv("Cable-Production-Line-Dataset.csv")

# ISPISIVANJE INFORMACIJA O DATASET-U
# print(data.info())

# ISPISIVANJE DIMENZIJA DATASET-A
print("DATASET dimensions before drop command: ")
print(data.shape)

# BRISANJE REDOVA KOD KOJIH JE CABLE FAILURES == 0
data.drop(data[data['Cable Failures'] == 0].index, inplace=True)

# ISPISIVANJE DIMENZIJA I INFORMACIJA POSLIJE DROP KOMANDE
# print(data.info())
# ISPISIVANJE DIMENZIJA DATASET-A
print("DATASET dimensions after drop command: ")
print(data.shape)
```

3. Kako smo u prethodnom zadatku izbrisali redove u kojima je “Cable Failures” == 0, sada je potrebno odraditi klasifikaciju (KNN, SVM, Bayes) i pri tome posmatrati rezultate za različite slučajeve (mijenjamo broj susjeda kod KNN-a, vrstu kernela kod SVM itd).

Pokretanjem programa smo dobili sljedeće rezultate:

```
lndir@lndir-ThinkPad:~/Fakultet/IntSys/Zadaca1$ python Classification3.py
DATASET dimensions before drop command:
(149, 8)
DATASET dimensions after drop command:
(109, 7)
(109, 7) (109,)
Score for Number of Neighbors= 2 : 0.3181818181818182
Score for Number of Neighbors= 5 : 0.36363636363636365
Score for Number of Neighbors= 7 : 0.45454545454545453
SVM Classification Score for linear kernel is : 0.5
SVM Classification Score for poly kernel is : 0.2727272727272727
SVM Classification Score for rbf kernel is : 0.36363636363636365
Naive Bayes Classification (GaussianNB) Score : 0.5
Naive Bayes Classification (ComplementNB) Score : 0.36363636363636365
```

Na narednim slikama su u istom redoslijedu prikazane confusion matrice (KNN i SVM):

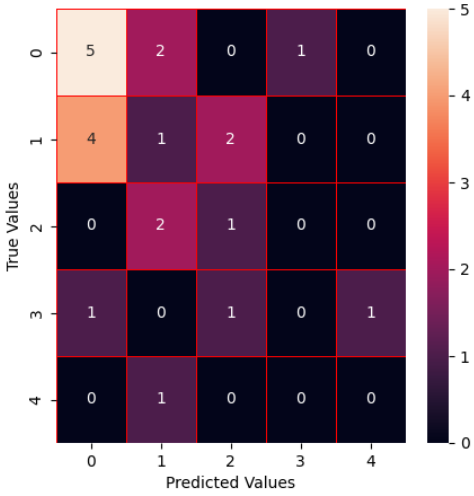


Figure 15: KNN (neighbors=2)

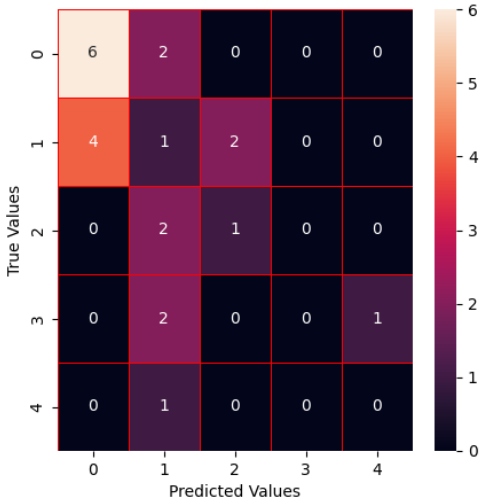


Figure 16: KNN (neighbors=5)

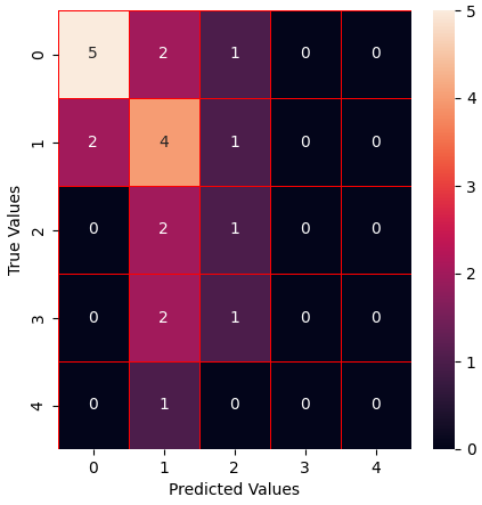


Figure 18: KNN (neighbors=7)

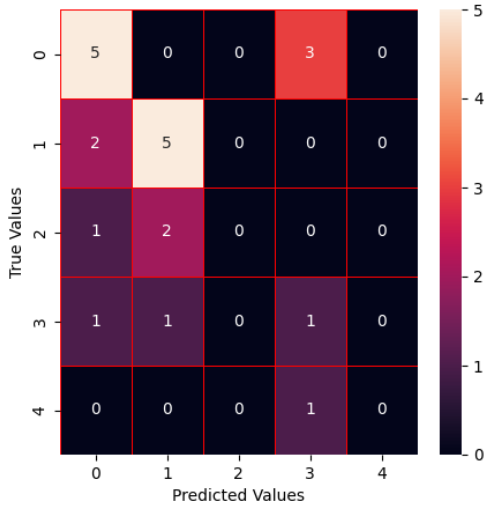


Figure 17: SVM - linear kernel

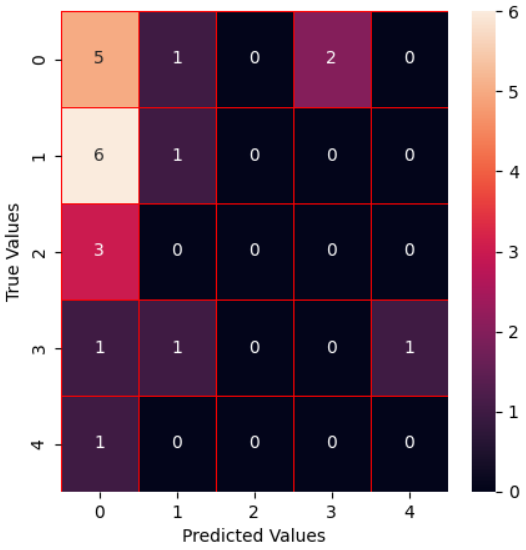


Figure 20: SVM - poly kernel

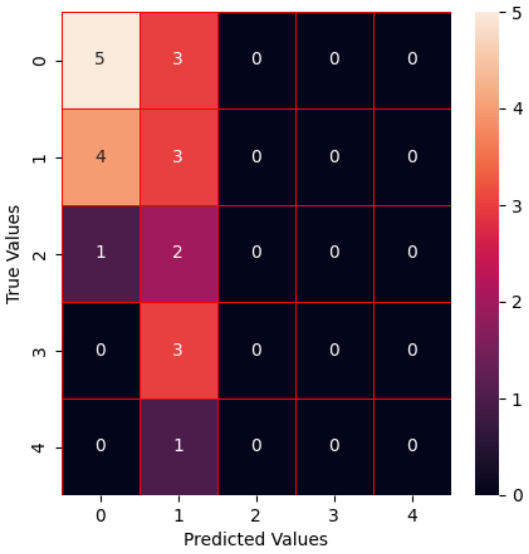


Figure 19: SVM - rbf kernel

Confusion matrice za Bayes-a:

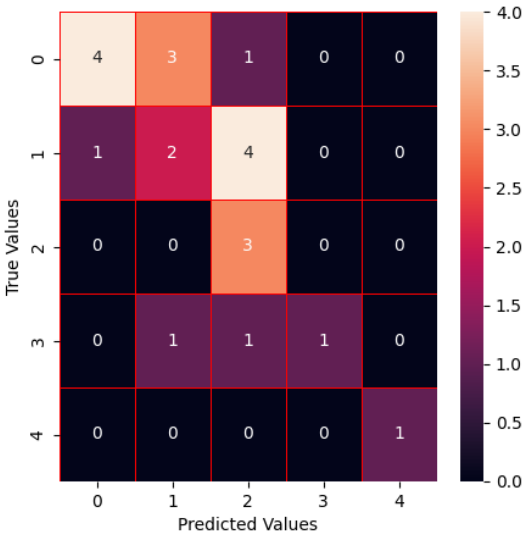


Figure 21: Bayes - GaussianNB

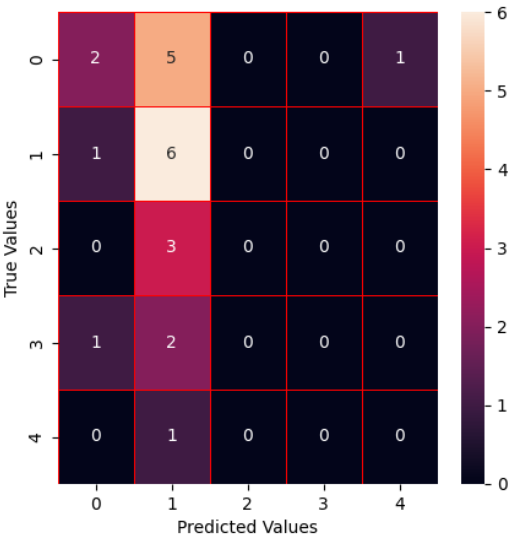


Figure 22: Bayes - ComplementNB

Kao što znamo, confusion matrica nam služi za ocjenu kvaliteta klasifikatora. Dijagonalni elementi su ispravno prediktovane tačke, a elementi van dijagonale su pogrešno klasificirane tačke. Iz ovoga možemo zaključiti da je bolje imati što veće vrijednosti na dijagonali. Ako posmatramo KNN sa 7 susjeda vidimo da imamo na dijagonali u sumi vrijednost 10 i rezultat je 0.45. Ako posmatramo elemente na dijagonali kod SVM klasifikacije sa linearnim kernelom, vidimo da je suma vrijednosti na dijagonali jednaka 11, te je i rezultat nešto veći (0.5). Za razliku od njih SVM klasifikacija sa poly kernelom ima u sumi 6 elemenata na dijagonali, pa je i dobijeni

rezultat znatno lošiji (0.27). U ovom slučaju smo najbolje rezultate dobili za SVM klasifikaciju sa linearnim kernelom i za Navie Bayes sa Gaussian raspodjelom.

Kod navednog zadatka je dat u nastavku:

```
from sklearn.cluster import KMeans
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.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.svm import SVC
# ZA confusion matricu
from sklearn.metrics import confusion_matrix
# UCITAVANJE DATASET-A
data = pd.read_csv("Cable-Production-Line-Dataset.csv")
# ISPISIVANJE DIMENZIJA DATASET-A
print("DATASET dimensions before drop command: ")
print(data.shape)
# BRISEMO DATUM JER JE TIPA OBJECT. S OVIM SMO IZBJEGLI KONVERZIJU TIPA
PODATAKA
del data['Date']
data.drop(data[data['Cable Failures'] == 0].index, inplace=True)
# ISPISIVANJE DIMENZIJA DATASET-A
print("DATASET dimensions after drop command: ")
print(data.shape)
X = data[['Machine', 'Shift', 'Operator', 'Cable Failures',
'Cable Failure Downtime', 'Other Failures', 'Other Failure Downtime']]
y = data['Cable Failures']
print(X.shape, y.shape)
# UKOLIKO U KOLONI IMAMO TEKSTUALNE VRIJEDNOSTI TREBA IH PRETVORITI U
NUMERIČKE
data['Shift'] = [0 if each == "A" else 1 for each in data['Shift']]
x = data[['Machine', 'Shift', 'Operator', 'Cable Failure Downtime',
'Other Failures', 'Other Failure Downtime']]

# PREBACIVANJE VRIJEDNOSTI U MATRICU ZA PREDIKCIJU (X) I VEKTOR CILJNIH
VRIJEDNOSTI (Y) (PANDAS TABELA u NUMPY MATRICU)
x = x.values
y = y.values.ravel()
```

```
x_train, x_test, y_train, y_test = train_test_split(
x, y, test_size=0.2, random_state=42)
# test_size=0.2 means %20 test datas, %80 train datas
method_names = []
method_scores = []
# These are for barplot in conclusion

# KNN n_neighbors=2
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(x_train, y_train)
print("Score for Number of Neighbors= 2 : {}".format(knn.score(x_test, y_test)))
method_names.append("KNN")
method_scores.append(knn.score(x_test, y_test))

# Confusion Matrix
y_pred = knn.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()

# KNN n_neighbors=5
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train, y_train)
print("Score for Number of Neighbors= 5 : {}".format(knn.score(x_test, y_test)))
method_names.append("KNN")

method_scores.append(knn.score(x_test, y_test))
# Confusion Matrix
y_pred = knn.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()
```

```
# KNN n_neighbors=7
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
print("Score for Number of Neighbors= 7 : {}".format(knn.score(x_test, y_test)))
method_names.append("KNN")

method_scores.append(knn.score(x_test, y_test))
# Confusion Matrix
y_pred = knn.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()
# SVC klasifikacija kernel='linear'
svm = SVC(random_state=42, kernel='linear', probability=True)
svm.fit(x_train, y_train)
print("SVM Classification Score for linear kernel is : {}".format(
svm.score(x_test, y_test)))
method_names.append("SVM")
method_scores.append(svm.score(x_test, y_test))
# Confusion Matrix
y_pred = svm.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()

# SVC klasifikacija kernel='poly'
svm = SVC(random_state=42, kernel='poly', probability=True)
svm.fit(x_train, y_train)
print("SVM Classification Score for poly kernel is : {}".format(
svm.score(x_test, y_test)))
method_names.append("SVM")
method_scores.append(svm.score(x_test, y_test))
```

```
# Confusion Matrix
y_pred = svm.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()

# SVC klasifikacija kernel='rbf'
svm = SVC(random_state=42, kernel='rbf', probability=True)
svm.fit(x_train, y_train)
print("SVM Classification Score for rbf kernel is : {}".format(
svm.score(x_test, y_test)))
method_names.append("SVM")
method_scores.append(svm.score(x_test, y_test))
# Confusion Matrix
y_pred = svm.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()

naive_bayes = GaussianNB()
naive_bayes.fit(x_test, y_test)
print("Naive Bayes Classification (GaussianNB) Score : {} ".format(
naive_bayes.score(x_test, y_test)))
method_names.append("Naive Bayes")
method_scores.append(naive_bayes.score(x_test, y_test))

# Confusion Matrix
y_pred = naive_bayes.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
```

```
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()

#naive_bayes = GaussianNB()
naive_bayes = ComplementNB()
#naive_bayes = MultinomialNB()
#naive_bayes = BernoulliNB()
naive_bayes.fit(x_test, y_test)
print("Naive Bayes Classification (ComplementNB) Score : {} ".format(
naive_bayes.score(x_test, y_test)))
method_names.append("Naive Bayes")

method_scores.append(naive_bayes.score(x_test, y_test))
# Confusion Matrix
y_pred = naive_bayes.predict(x_test)
conf_mat = confusion_matrix(y_test, y_pred)
# Visualization Confusion Matrix
f, ax = plt.subplots(figsize=(5, 5))
sns.heatmap(conf_mat, annot=True, linewidths=0.5,
linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Predicted Values")
plt.ylabel("True Values")
plt.show()
```