LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Department of Computer Science & Engineering

20AD53 Machine Learning Lab

Name of the Student: ______________________________

Registered Number: ______________________________

Branch & Section: __________& ___../Sec

Academic Year: 2022 – 23

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

CERTIFICATE

This is to certify that this is a bonafide record of the practical work done by Mr./Ms. ……………………………………………………………………………, bearing Regd. Num.: 20761A05……. of B.Tech ….. Semester, ………. Branch, …… Section in the <u>20AD53 Machine Learning Lab</u> during the

Academic Year: <u>2022– 23</u>

No. of Experiments/Modules held: <u>12</u>

No. of Experiments Done: <u>12</u>

Date: ….. / ____/ 2022 **Signature of the Faculty**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

S.No	DATE	EXPERIMENT	PAGE No.	REMARKS

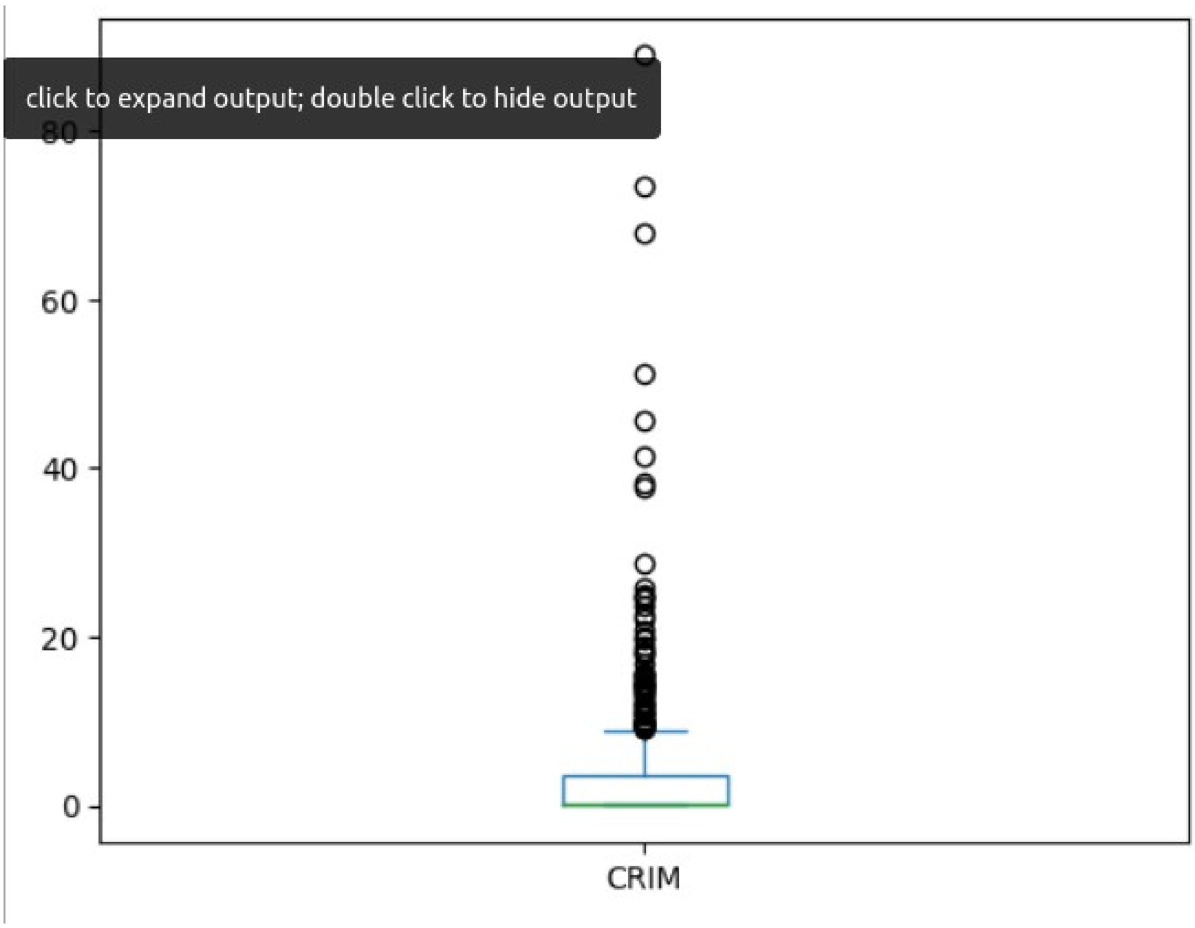
1.Basic statistical functions for data exploration

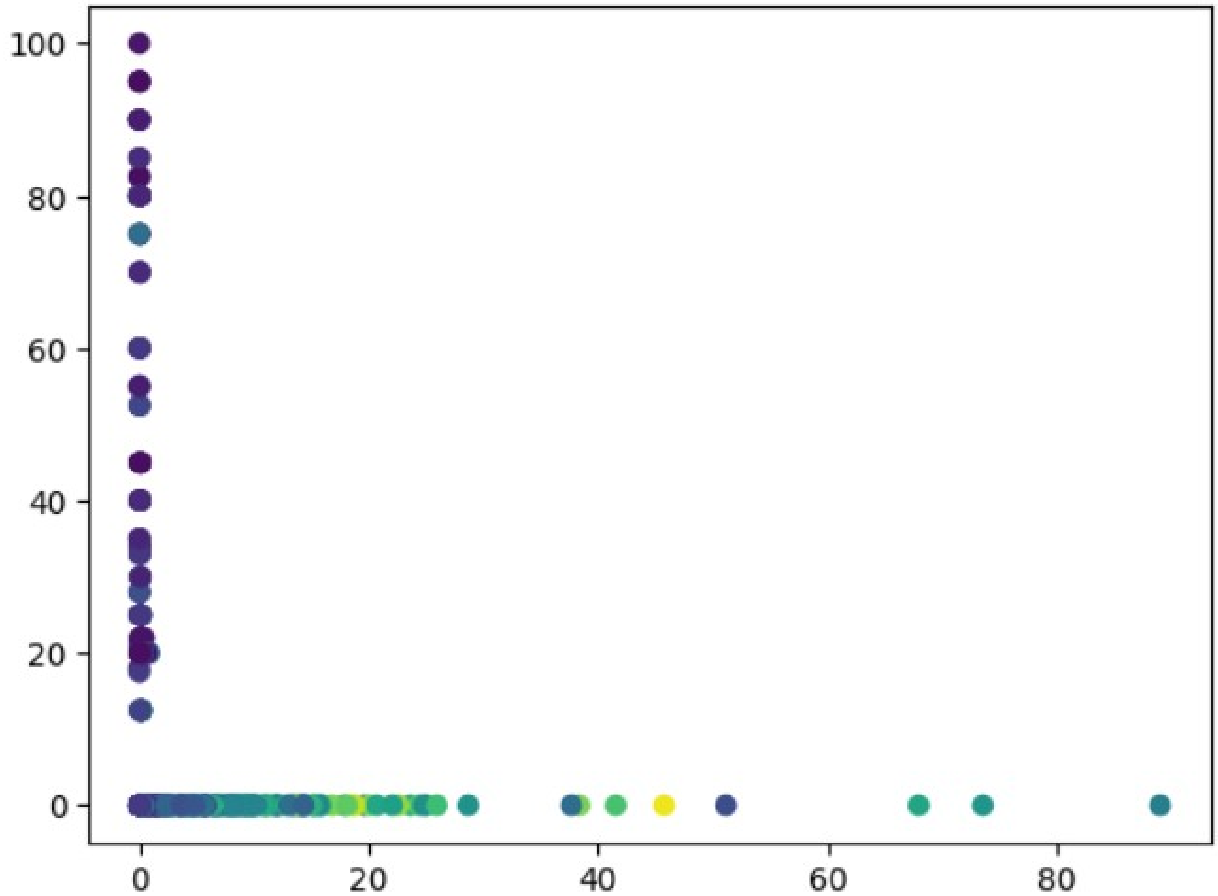
```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.datasets import load_boston
i=load_boston()
df=pd.DataFrame(i.data,columns=i.feature_names)
print(df.describe())
# median & mode
median=[]
m=[]
for i in df.columns:
    median.append(np.median(df[i]))
    m.append(stats.mode(df[i]))
print(median,'\n',m)
```

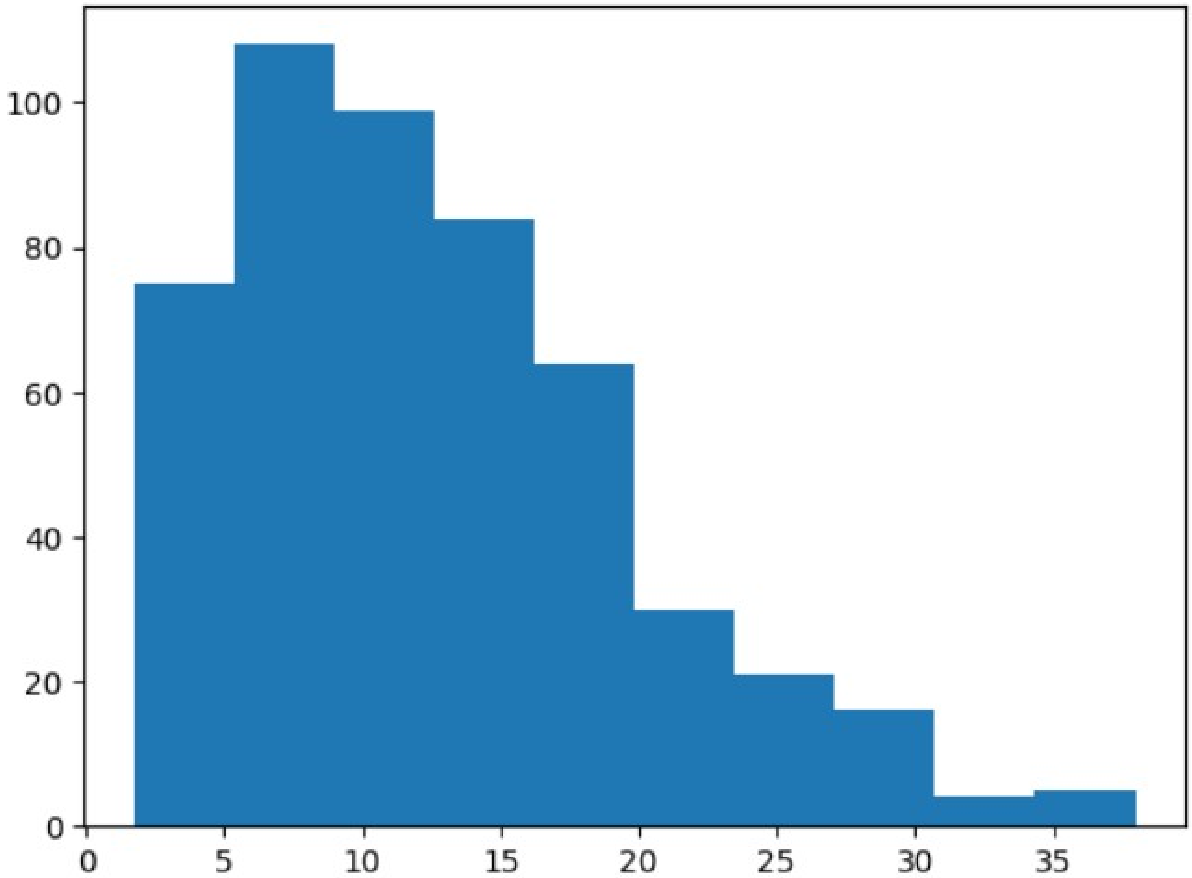
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT
count  506.000000
mean    12.653063
std      7.141062
min      1.730000
25%      6.950000
50%     11.360000
75%     16.955000
max     37.970000
[0.25651, 0.0, 9.69, 0.0, 0.538, 6.2085, 77.5, 3.2074499999999997, 5.0, 330.0, 19.05, 391.44, 11.36]
 [ModeResult(mode=array([0.01501]), count=array([2])), ModeResult(mode=array([0.]), count=array([372])), ModeResult
(mode=array([18.1]), count=array([132])), ModeResult(mode=array([0.]), count=array([471])), ModeResult(mode=array
([0.538]), count=array([23])), ModeResult(mode=array([5.713]), count=array([3])), ModeResult(mode=array([100.]), cou
nt=array([43])), ModeResult(mode=array([3.4952]), count=array([5])), ModeResult(mode=array([24.]), count=array([13
2])), ModeResult(mode=array([666.]), count=array([132])), ModeResult(mode=array([20.2]), count=array([140])), ModeRe
sult(mode=array([396.9]), count=array([121])), ModeResult(mode=array([6.36]), count=array([3]))]
```

2. Data Visualization: Box plot, scatter plot, histogram

```
import matplotlib.pyplot as plt
import seaborn as sns
df['CRIM'].plot(kind='box')
plt.show()
plt.scatter(df['CRIM'],df['ZN'],cmap='viridis',c=df['LSTAT'])
plt.show()
df['LSTAT'].hist().grid(False)
plt.show()
```




100
80
60
40
20
0
0
20
40
60
80

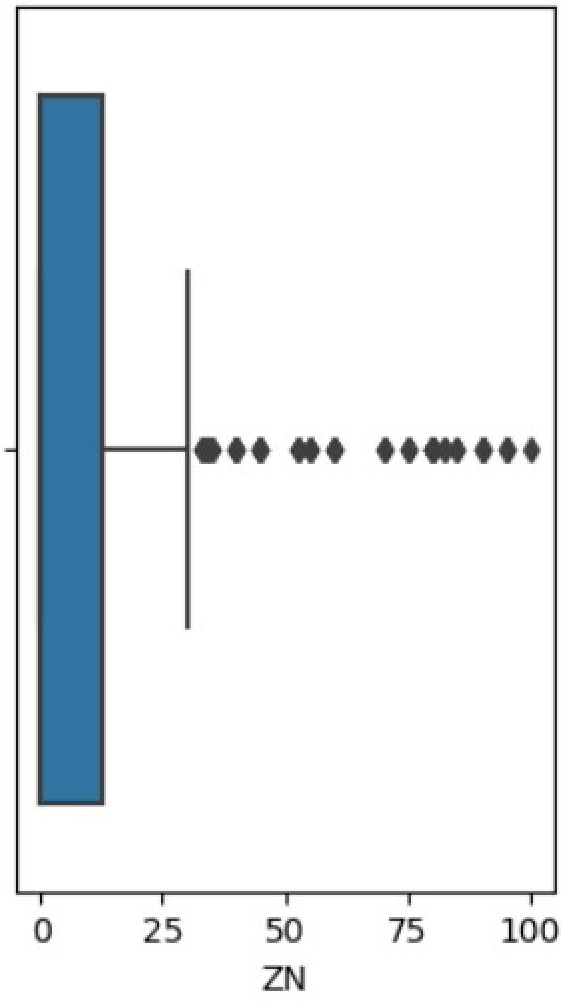

100
80
60
40
20
0
0
5
10
15
20
25
30
35

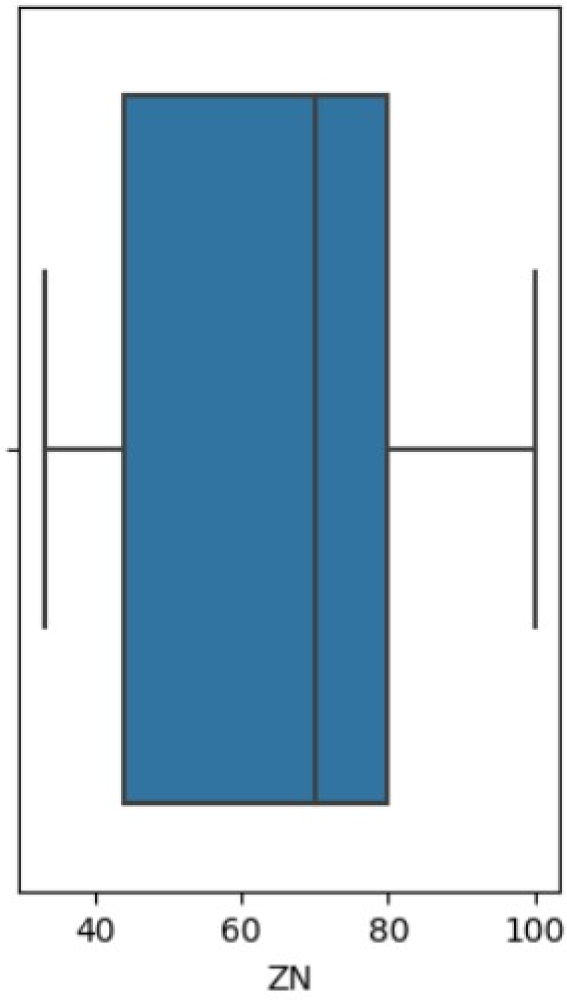
3.Data Pre-processing: Handling missing values, outliers, normalization, Scaling

```
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.datasets import load_boston
i=load_boston()
df=pd.DataFrame(i.data,columns=i.feature_names)
#checking missing values
print(df.isna().sum(),'\n')
#checking outliers
#Capping
q1=df['ZN'].quantile(0.25)
q3=df['ZN'].quantile(0.75)
IQR=q3-q1
Upper=q3+1.5*IQR
Lower=q1-1.5*IQR
df[df['ZN'] > Upper]
df[df['ZN'] < Lower]
new_df = df[df['ZN'] > Upper ]
plt.subplot(1,2,1)
sns.boxplot(x=df['ZN'])
plt.subplot(1,2,2)
sns.boxplot(x=new_df['ZN'])
#normalize the data
s=StandardScaler()
df=s.fit_transform(df)
df
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
dtype: int64
```

```
array([[-0.41978194,  0.28482986, -1.2879095 , ..., -1.45900038,
         0.44105193, -1.0755623 ],
       [-0.41733926, -0.48772236, -0.59338101, ..., -0.30309415,
         0.44105193, -0.49243937],
       [-0.41734159, -0.48772236, -0.59338101, ..., -0.30309415,
         0.39642699, -1.2087274 ],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.98304761],
       [-0.40776407, -0.48772236,  0.11573841, ...,  1.17646583,
         0.4032249 , -0.86530163],
       [-0.41500016, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.66905833]])
```

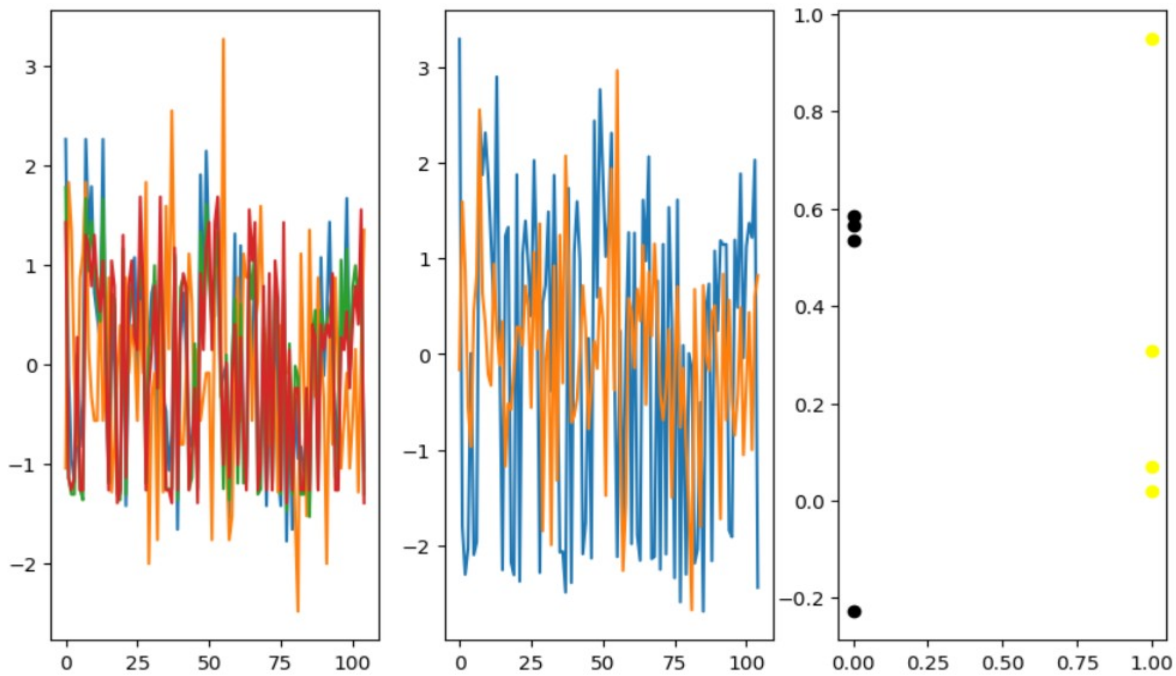




4. Principal Component Analysis (PCA)

```
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
iris=load_iris()
x,y=iris.data,iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=1)
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
x_1=x_train
pca=PCA(n_components=2)
x_train=pca.fit_transform(x_train)
comps=pca.components_
print("components:\n",pca.components_)
print("Co variance\n",pca.get_covariance())
plt.figure(figsize=(10,6))
plt.subplot(1,3,1)
plt.plot(x_1)
plt.subplot(1,3,2)
plt.plot(x_train)
plt.subplot(1,3,3)
c=['black','yellow','red','green']
for i in range(0,len(comps)):
    for j in range(0,len(comps[i])):
        plt.scatter(i,comps[i][j],c=c[i])
```

```
components:
 [[ 0.53412301 -0.22791604  0.58442075  0.5667621 ]
 [ 0.30791713  0.94856275  0.01976354  0.07088833]]
Co variance
 [[ 0.97606998 -0.09317132  0.88720786  0.87407504]
 [-0.09317132  1.00793386 -0.36025464 -0.30724325]
 [ 0.88720786 -0.36025464  1.05395785  0.93708159]
 [ 0.87407504 -0.30724325  0.93708159  1.00049985]]
```



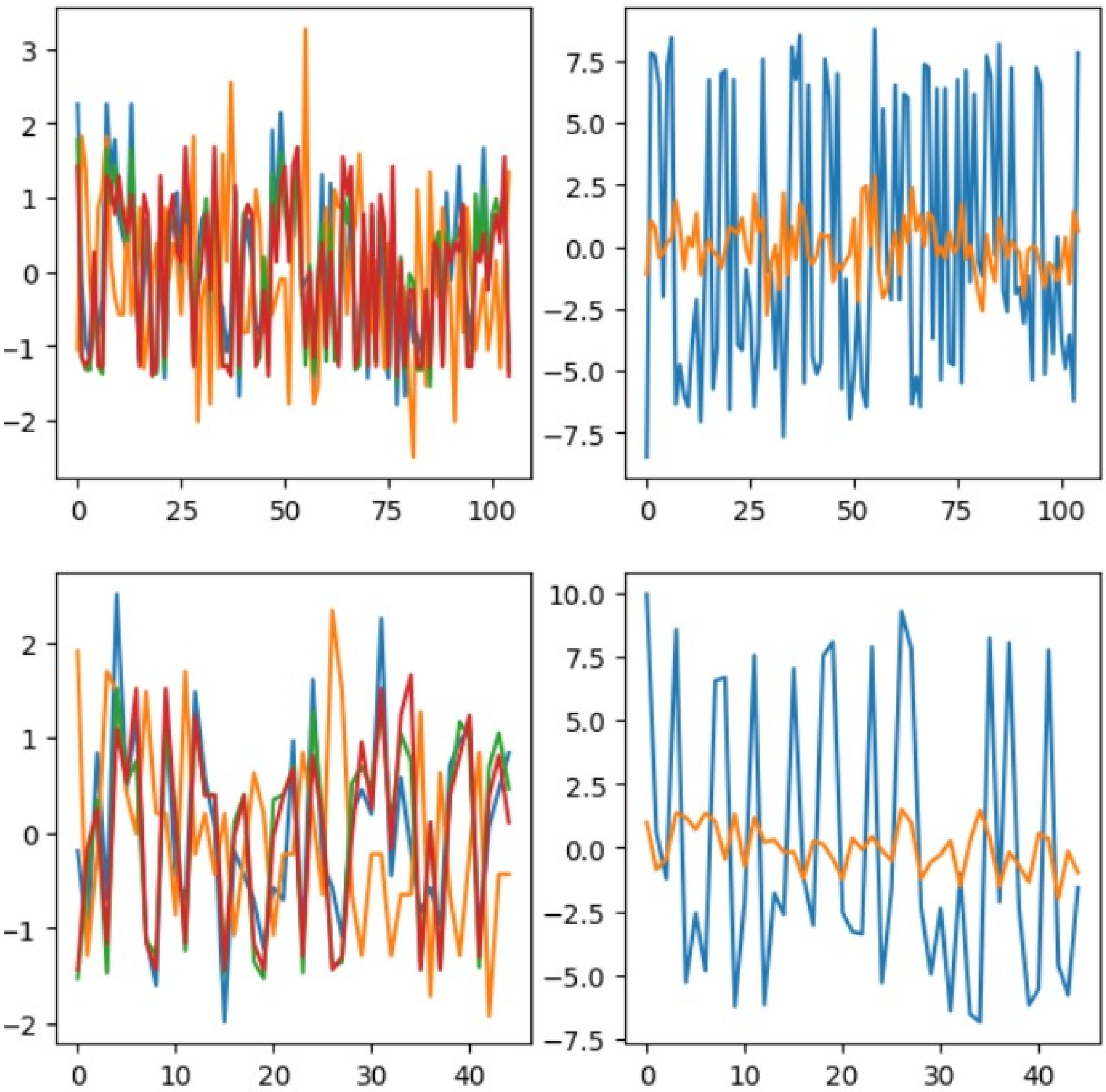
5. Singular Value Decomposition (SVD)

```
from numpy.linalg import svd
A = np.array([[3,4,3],[1,2,3],[4,2,1]])
S,U,VT=svd(A)
print("Singular Matrix:\n",S,'\n',"U:\n",U,'\n','Vector Matrix:\n',VT)
```

```
Singular Matrix:
 [[-0.73553325 -0.18392937 -0.65204358]
 [-0.42657919 -0.62196982  0.65664582]
 [-0.52632788  0.76113306  0.37901904]]
 U:
 [7.87764972 2.54031671 0.69958986]
 Vector Matrix:
 [[-0.60151068 -0.61540527 -0.5093734 ]
 [ 0.73643349 -0.18005275 -0.65210944]
 [ 0.30959751 -0.76737042  0.5615087 ]]
```

6. Linear Discriminant Analysis (LDA)

```
from sklearn.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data=load_iris()
x,y=data.data,data.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.fit_transform(x_test)
lda=LDA(n_components=2)
X_train=lda.fit_transform(x_train,y_train)
X_test=lda.transform(x_test)
plt.figure(figsize=(7,7))
plt.subplot(2,2,1)
plt.plot(x_train)
plt.subplot(2,2,2)
plt.plot(X_train)
plt.subplot(2,2,3)
plt.plot(x_test)
plt.subplot(2,2,4)
plt.plot(X_test)
```



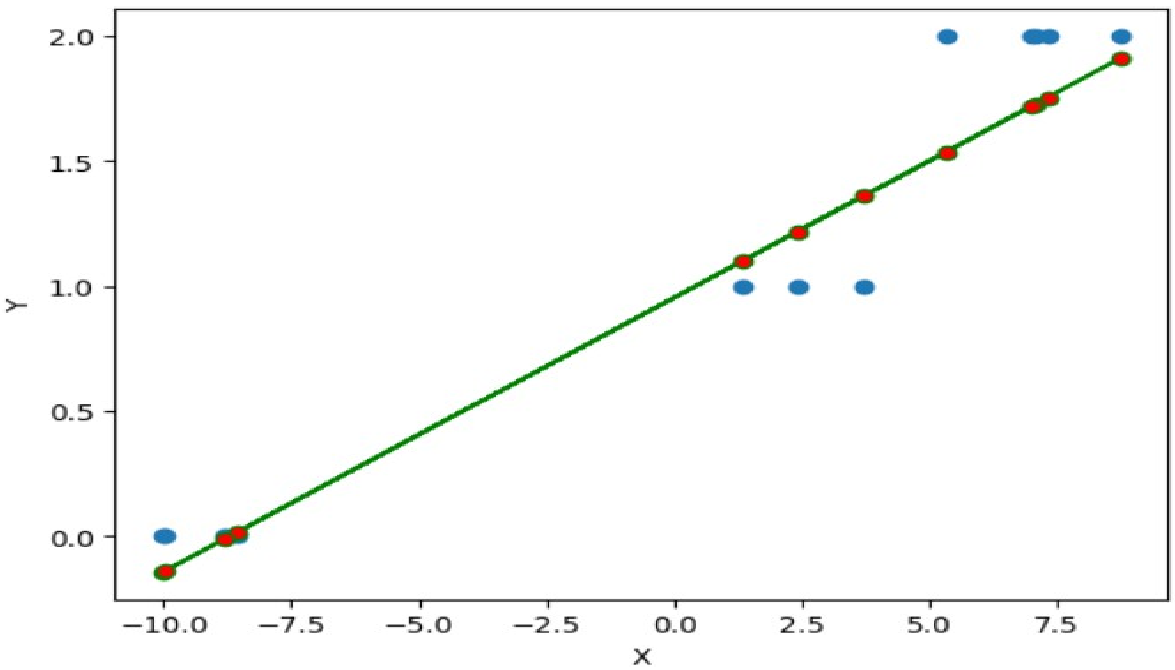
3
2
1
0
−1
−2
0
25
50
75
100
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
25
50
75
100
2
1
0
−1
−2
0
10
20
30
40
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
10
20
30
40

7. Regression Analysis: Linear regression, Logistic regression, Polynomial regression

i)Linear regression

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import seaborn as sns
from sklearn.datasets import make_blobs
x,y=make_blobs(n_samples=40,n_features=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=1)
lr=LinearRegression()
lr.fit(x_train,y_train)
print("slope:",lr.coef_)
print("Intercept:",lr.intercept_)
print("Score:",lr.score(x_train,y_train))
y_pred=lr.predict(x_test)
print('Accuracy:',r2_score(y_test,y_pred))
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred,c='green',marker='o',markerfacecolor='red')
plt.xlabel("X")
plt.ylabel("Y")
```

```
slope: [0.10950792]
Intercept: 0.9540122319042185
Score: 0.891501220920705
Accuracy: 0.9255364544536799

Text(0, 0.5, 'Y')
```

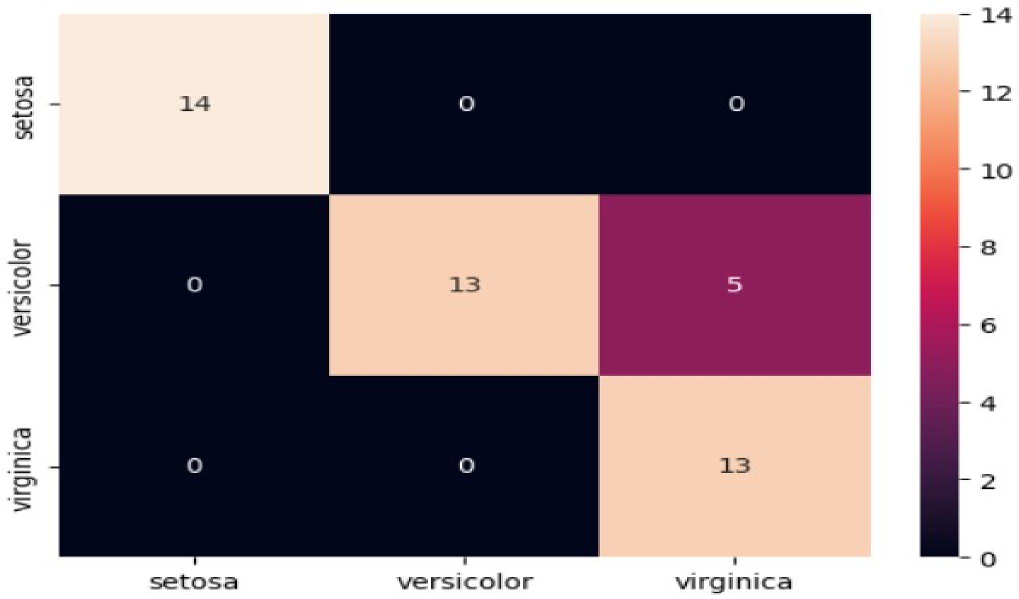


ii)Logistic regression

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix
import seaborn as sns
df=sns.load_dataset('iris')
x=df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y=df['species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
lrc=LogisticRegression(solver='liblinear',random_state=0)
lrc.fit(x_train,y_train)
y_pred=lrc.predict(x_test)
print("classes:",lrc.classes_)
print("cofficient:\n",lrc.coef_)
print("Intercept:",lrc.intercept_)
print("Score:",lrc.score(x_train,y_train))
print("Predicted Probabilites:",lrc.predict_proba(x_test[:1]))
print("accuracy:",accuracy_score(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,xticklabels=pd.unique(y.values),yticklabels=pd.unique(y.values))
```

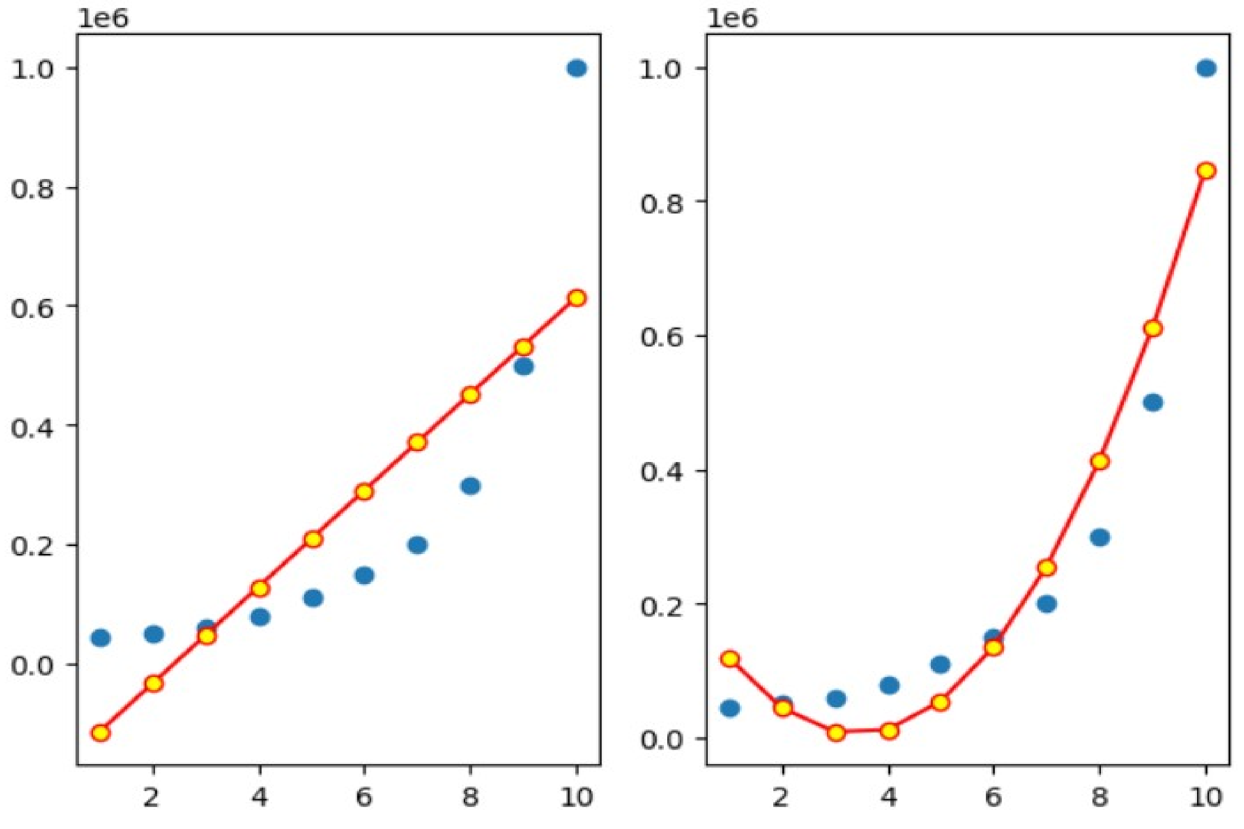
```
classes: ['setosa' 'versicolor' 'virginica']
cofficient:
 [[ 0.40624466  1.34920549 -2.12162959 -0.94996585]
 [ 0.628089   -1.72885637  0.30292395 -0.99777747]
 [-1.6655058  -0.95801685  2.19808191  2.08216345]]
Intercept: [ 0.26653163  0.67420401 -0.86971453]
Score: 0.9523809523809523
Predicted Probabilites: [[9.25059393e-01 7.49291898e-02 1.14167043e-05]]
accuracy: 0.8888888888888888
```

```
: <AxesSubplot: >
```



iii)Polynomial regression

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
df=pd.read_csv('data.csv')
x=df[['Level']]
y=df['Salary']
poly=PolynomialFeatures(degree=2)
x_poly=poly.fit_transform(x)
lin=LinearRegression().fit(x,y)
y_pred=lin.predict(x)
lin2=LinearRegression().fit(x_poly,y)
y_pred1=lin2.predict(x_poly)
plt.figure(figsize=(7,5))
plt.subplot(1,2,1)
plt.scatter(x,y)
plt.plot(x,y_pred,marker='o',markerfacecolor='yellow',c='red')
plt.subplot(1,2,2)
plt.scatter(x,y)
plt.plot(x,y_pred1,marker='o',markerfacecolor='yellow',c='red')
```

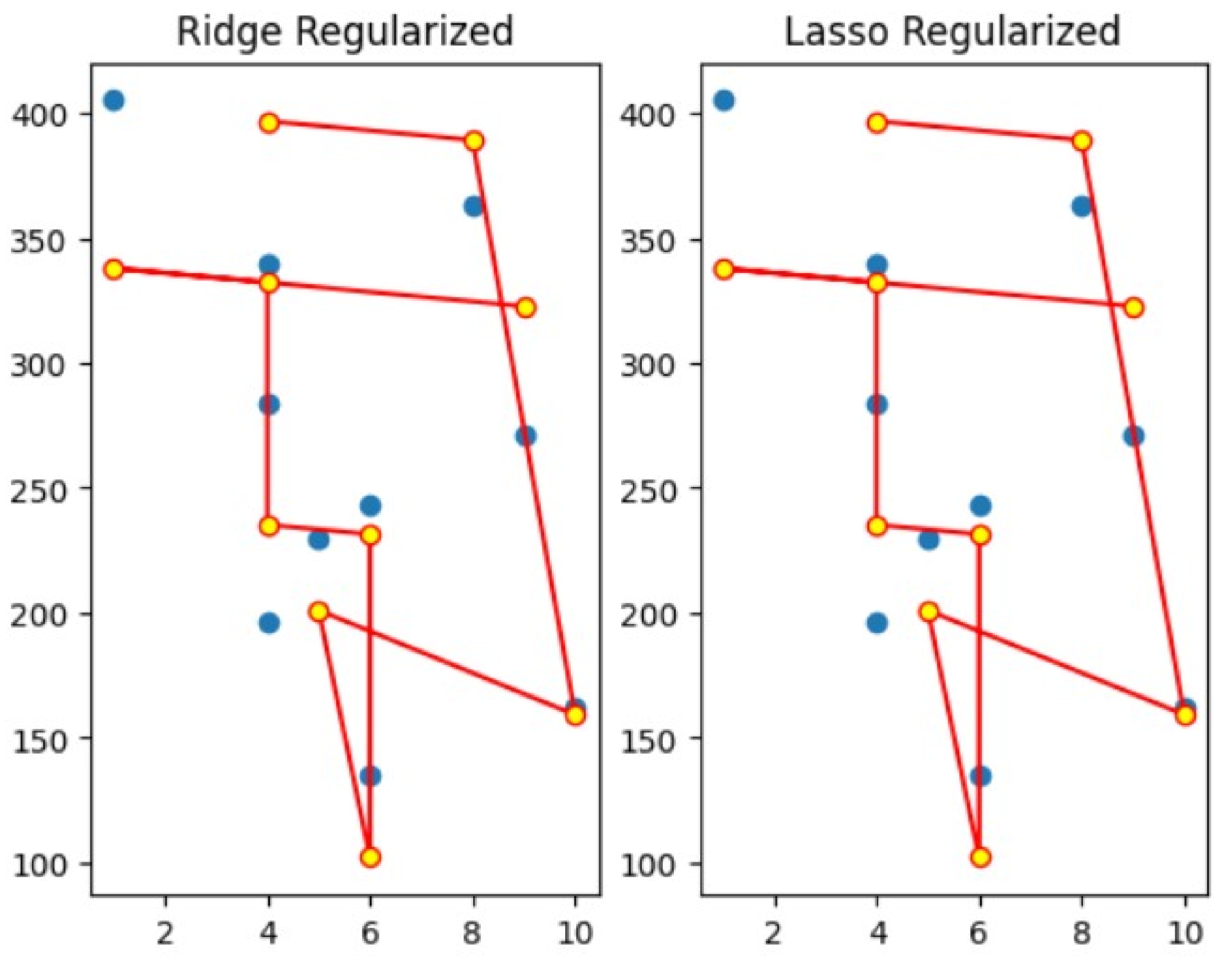


8. Regularized Regression

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler,LabelEncoder,scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge,Lasso
from sklearn.metrics import r2_score
import seaborn as sns
df=sns.load_dataset('flights')
le=LabelEncoder()
df['month']=le.fit_transform(df['month'])
x=df[['year','month']]
y=df['passengers']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=1)
ridge=Ridge(alpha=0.3)
ridge.fit(x_train,y_train)
y_pred=ridge.predict(x_test)
lasso=Lasso(alpha=0.1)
lasso.fit(x_train,y_train)
y_pred1=lasso.predict(x_test)
print('Ridge Regularized Regression:')
print('slope:',ridge.coef_)
print('Intercept:',ridge.intercept_)
print('score:',ridge.score(x_train,y_train))
print('accuracy:',r2_score(y_test,y_pred),'\n')
print('Lasso Regularized Regression:')
print('slope:',lasso.coef_)
print('Intercept:',lasso.intercept_)
print('score:',lasso.score(x_train,y_train))
print('accuracy:',r2_score(y_test,y_pred1))
plt.subplot(1,2,1)
plt.title('Ridge Regularized')
plt.scatter(x_test['month'][:10],y_test[:10])
plt.plot(x_test['month']
[:10],y_pred[:10],c='red',marker='o',markerfacecolor='yellow')
plt.subplot(1,2,2)
plt.title('Lasso Regularized')
plt.scatter(x_test['month'][:10],y_test[:10])
```

plt.plot(x_test['month']
[:10],y_pred1[:10],c='red',marker='o',markerfacecolor='yellow')

```
Ridge Regularized Regression:
slope: [32.3137578  -1.90613899]
Intercept: -62865.90705172744
score: 0.8572612820122529
accuracy: 0.8239841277324573

Lasso Regularized Regression:
slope: [32.31311528 -1.89857863]
Intercept: -62864.692308705315
score: 0.8572612218618013
accuracy: 0.8240219138855976
```

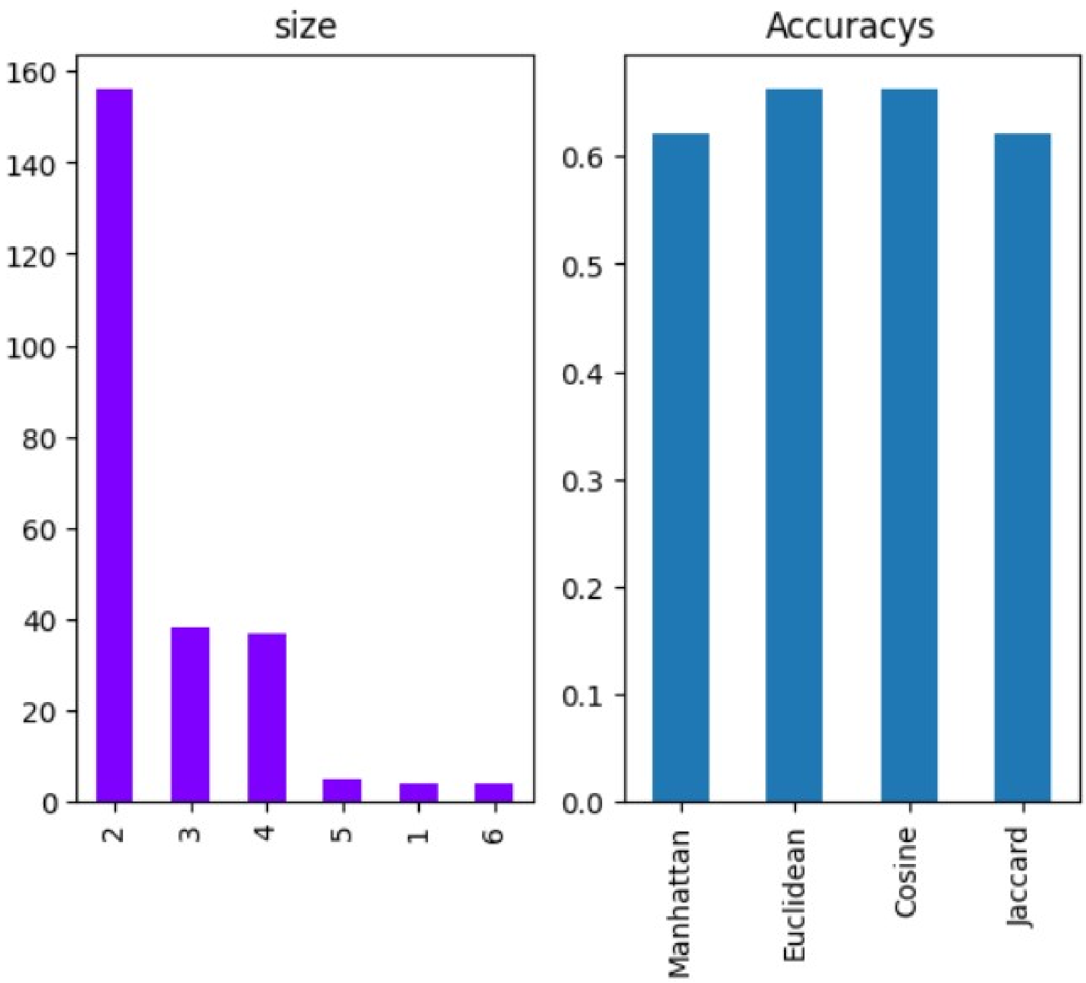


9. K-Nearest Neighbour (kNN) Classifier

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import seaborn as sns
df=sns.load_dataset('tips')
l=LabelEncoder()
df['sex']=l.fit_transform(df['sex'])
df['smoker']=l.fit_transform(df['smoker'])
df['day']=l.fit_transform(df['day'])
df['time']=l.fit_transform(df['time'])
x=df[['total_bill','tip','sex','smoker','day','time']]
y=df['size']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=1)
D1=[]
D2=[]
D3=[]
D4=[]
for i in range(1,11):
   knn=KNeighborsClassifier(n_neighbors=i)
   knn1=KNeighborsClassifier(p=1,n_neighbors=i)

knn2=KNeighborsClassifier(metric='cosine',algorithm='brute',n_neighbors
=i)
   knn3=KNeighborsClassifier(metric='jaccard',n_neighbors=i)
   knn.fit(x_train,y_train)
   knn1.fit(x_train,y_train)
   knn2.fit(x_train,y_train)
   knn3.fit(x_train,y_train)
   D1.append(accuracy_score(y_test,knn.predict(x_test)))
   D2.append(accuracy_score(y_test,knn1.predict(x_test)))
   D3.append(accuracy_score(y_test,knn2.predict(x_test)))
   D4.append(accuracy_score(y_test,knn3.predict(x_test)))
D=np.array((D1,D2,D3,D4)).T
dff=pd.DataFrame(D,columns=['Manhattan','Euclidean','Cosine','Jaccard'],i
ndex=[i for i in range(1,11)])
```

```
print(dff)
plt.subplot(1,2,1)
plt.title('size')
df['size'].value_counts().plot(kind='bar',cmap='rainbow')
plt.subplot(1,2,2)
plt.title('Accuracys')
np.max(dff).plot(kind='bar')
```

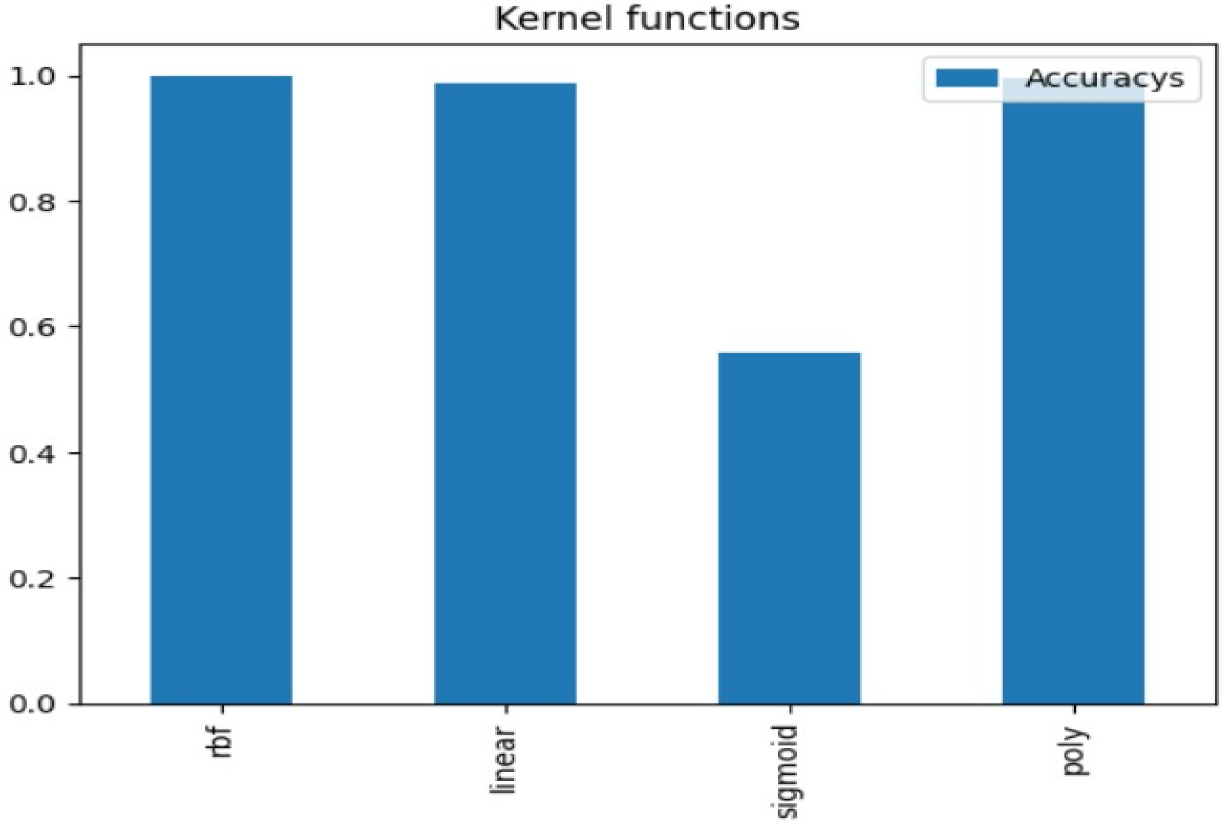
	Manhattan	Euclidean	Cosine	Jaccard
1	0.581081	0.594595	0.486486	0.527027
2	0.621622	0.608108	0.594595	0.527027
3	0.608108	0.581081	0.581081	0.621622
4	0.608108	0.594595	0.594595	0.621622
5	0.594595	0.608108	0.608108	0.621622
6	0.594595	0.635135	0.608108	0.621622
7	0.608108	0.635135	0.621622	0.621622
8	0.594595	0.621622	0.635135	0.621622
9	0.608108	0.635135	0.608108	0.621622
10	0.621622	0.662162	0.662162	0.621622

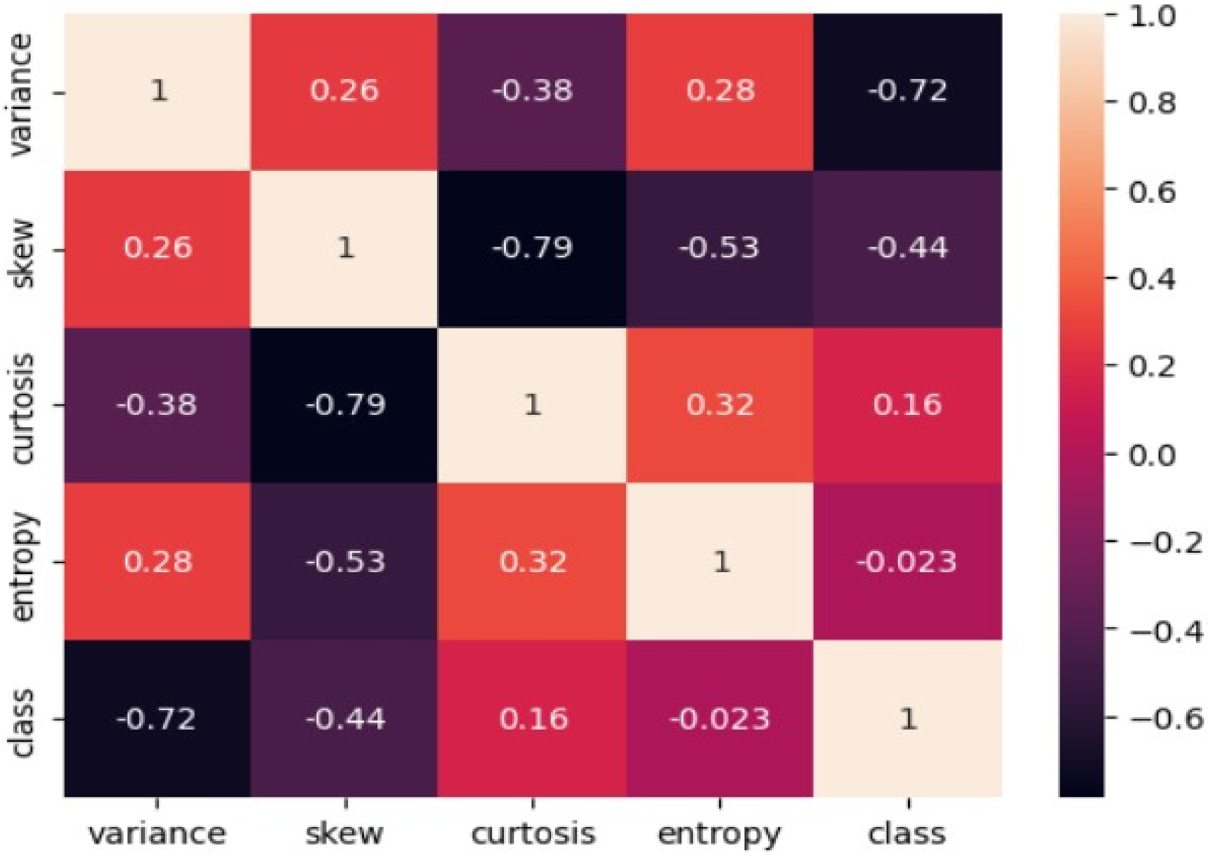


10. Support Vector Machines (SVMs)

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
df=pd.read_csv('banknote-authentication.csv')
x=df[['variance', 'skew', 'curtosis', 'entropy']]
y=df['class']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=0)
s=[]
kernel=['rbf','linear','sigmoid','poly']
for i in kernel:
    svm=SVC(kernel=i,gamma='auto',C=1)
    svm.fit(x_train,y_train)
    s.append(accuracy_score(y_test,svm.predict(x_test)))
print("Accuracys:\n",s)
df1=pd.DataFrame(np.array(s).reshape(4,1),columns=['Accuracys'])
df1.plot(kind='bar')
plt.title('Kernel functions')
plt.xticks(range(4),kernel)
plt.show()
sns.heatmap(df.corr(),annot=True)
```

```
Accuracys:
 [1.0, 0.9878640776699029, 0.558252427184466, 0.9975728155339806]
```



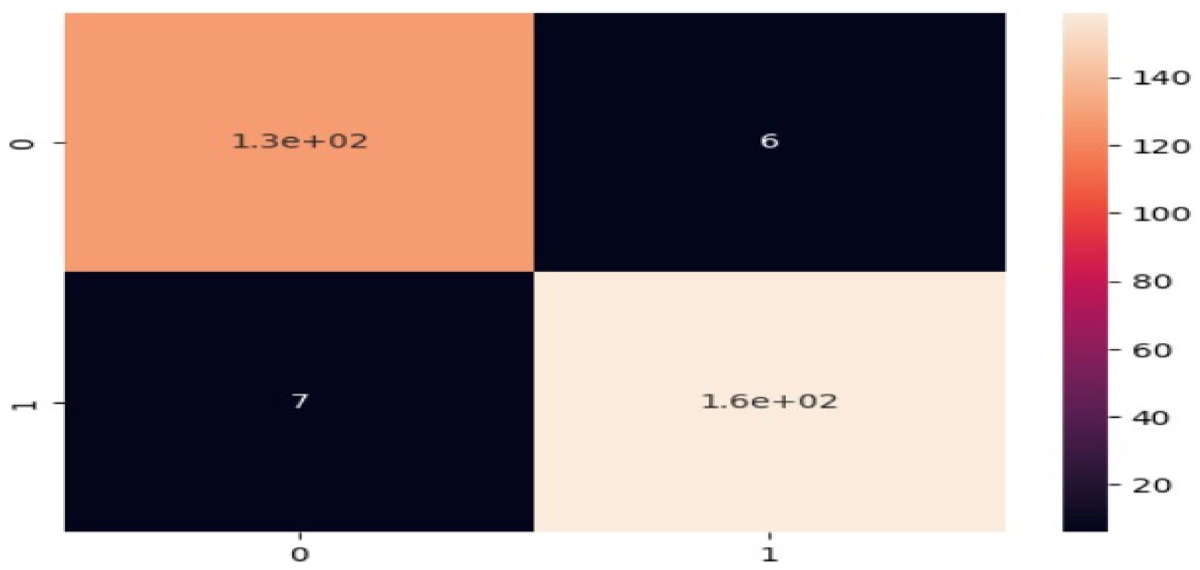


11. Random Forest model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
import seaborn as sns
from sklearn.metrics import accuracy_score,confusion_matrix
x,y=make_classification(n_samples=1000,n_features=4,random_state=0,shuffle=False)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
rfc=RandomForestClassifier(max_depth=2,n_estimators=1000,random_state=0)
rfc.fit(x_train,y_train)
y_pred=rfc.predict(x_test)
print('Accuarcy:',accuracy_score(y_test,y_pred))
print('Score:',rfc.score(x_train,y_train))
print('Predicted probabilites:\n',rfc.predict_proba(x))
print('Base Estimator:',rfc.base_estimator_)
print('Classes:',rfc.classes_)
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True)
```

```
Accuarcy: 0.9566666666666667
Score: 0.9557142857142857
Predicted probabilites:
 [[0.97220336 0.02779664]
 [0.94492029 0.05507971]
 [0.97215891 0.02784109]
 ...
 [0.03255269 0.96744731]
 [0.03198959 0.96801041]
 [0.0626095  0.9373905 ]]
Base Estimator: DecisionTreeClassifier()
Classes: [0 1]
```

```
: <AxesSubplot: >
```

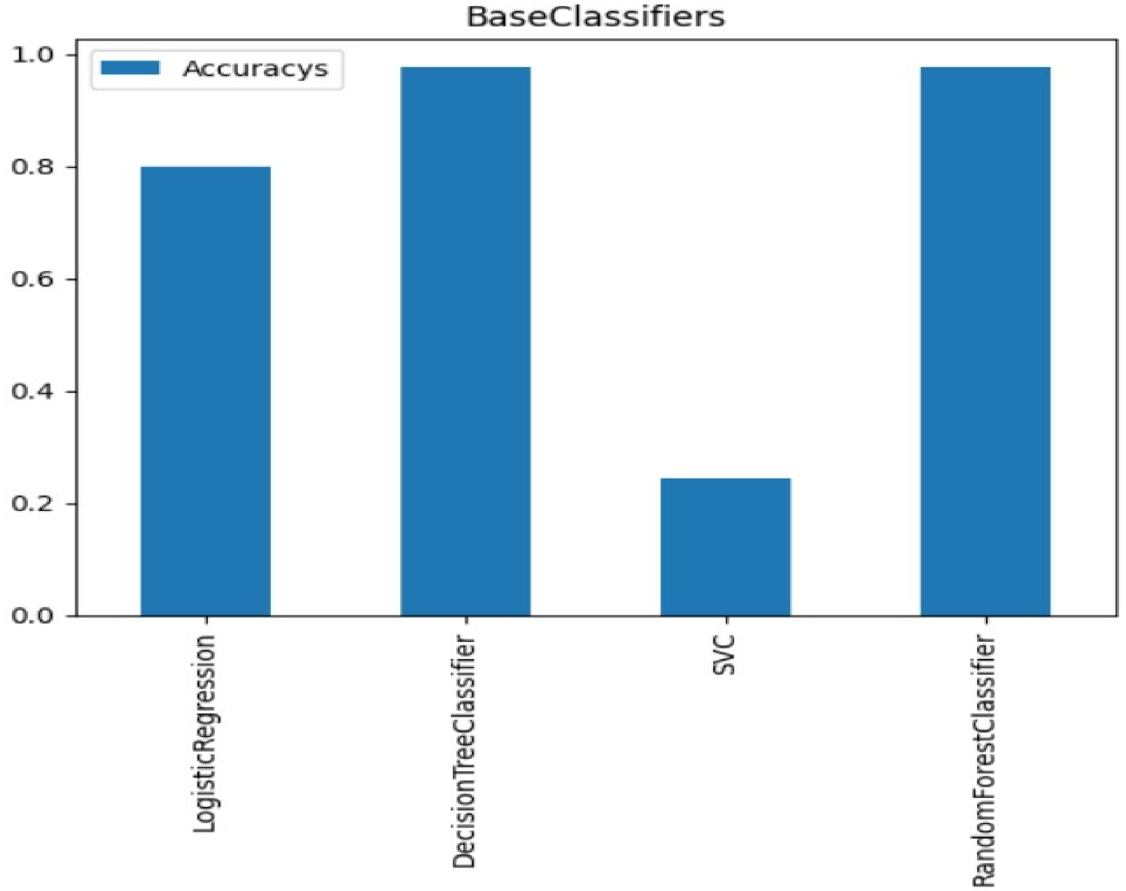


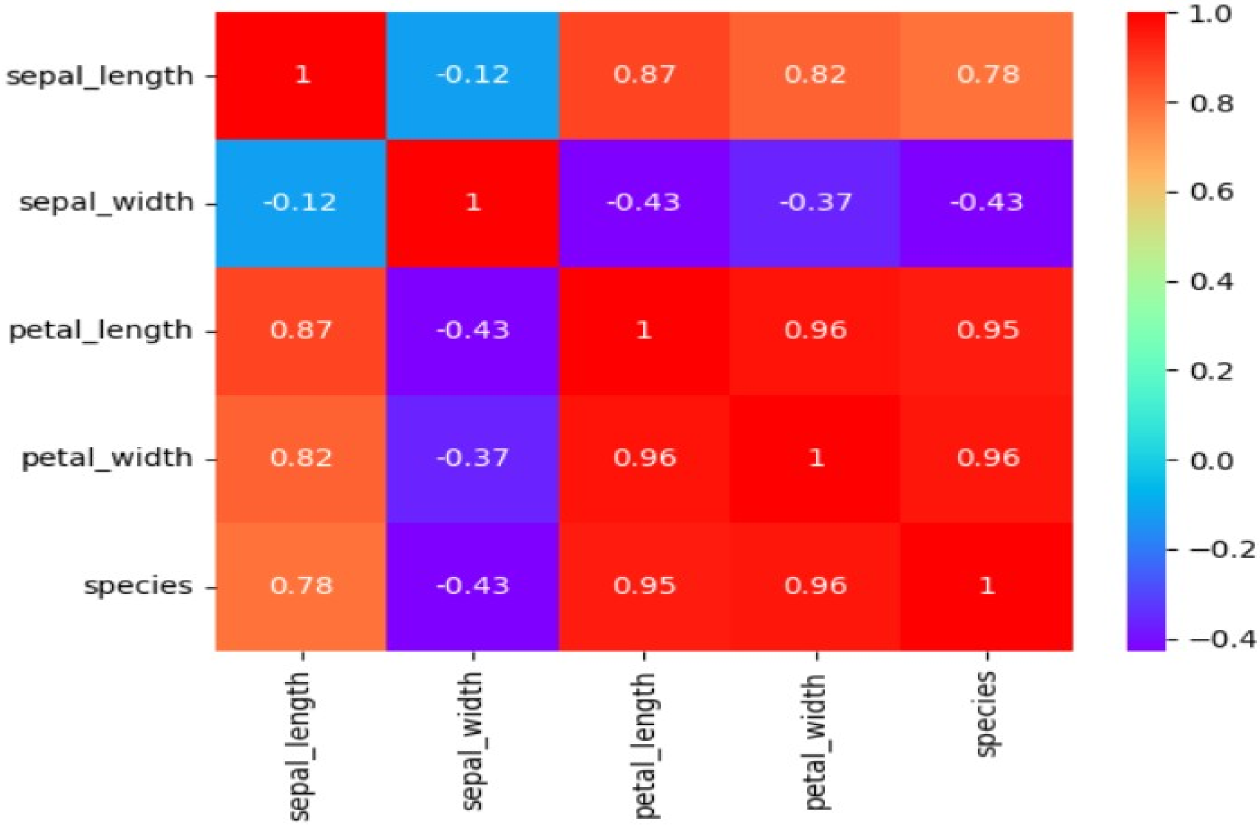
12. AdaBoost Classifier and XGBoost

i)AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import seaborn as sns
df=sns.load_dataset('iris')
df['species']=LabelEncoder().fit_transform(df['species'])
x=df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y=df['species']
svm=SVC()
lrc=LogisticRegression()
dtc=DecisionTreeClassifier()
rfc=RandomForestClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
classifier=['LogisticRegression','DecisionTreeClassifier','SVC','RandomForestClassifier']
instance=[lrc,dtc,svm,rfc]
s=[]
for i in instance:

ada=AdaBoostClassifier(n_estimators=100,base_estimator=i,learning_rate=1.0,random_state=0,algorithm='SAMME')
    ada.fit(x_train,y_train)
    s.append(accuracy_score(y_test,ada.predict(x_test)))
print("Accuracys:\n",s)
df1=pd.DataFrame(np.array(s).reshape(4,1),columns=['Accuracys'])
df1.plot(kind='bar')
plt.title('BaseClassifiers')
plt.xticks(range(4),classifier)
plt.show()
sns.heatmap(df.corr(),annot=True,cmap='rainbow')
```

```
Accuracys:
 [0.8, 0.9777777777777777, 0.24444444444444444, 0.9777777777777777]
```





ii)XGboost

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import seaborn as sns
from sklearn.metrics import accuracy_score,confusion_matrix
from xgboost  import XGBClassifier as XGB
df=sns.load_dataset('iris')
df['species']=LabelEncoder().fit_transform(df['species'])
x=df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y=df['species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=0)
xgb=XGB()
xgb.fit(x_train,y_train)
y_pred=xgb.predict(x_test)
sns.heatmap(df.corr(),annot=True,cmap='viridis')
print('Accuarcy:',accuracy_score(y_test,y_pred))
print('Predicted values:\n',y_pred)
```

```
Accuarcy: 0.9777777777777777
Predicted values:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2 1 1 2 0 2 0 0]
```