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 **20AD53 - Machine Learning Lab** during the

Academic Year: 2022 - 2023

No. of Experiments/Modules held: 12

No. of Experiments Done: 12

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**	4-5	
2.		**Data Visualization: Box plot, scatter plot, histogram**	5-6	
3.		**Data Pre-processing: Handling missing values, outliers, normalization, Scaling**	6-7	
4.		**Principal Component Analysis (PCA)**	8	
5.		**Singular Value Decomposition (SVD)**	9	
6.		**Linear Discriminant Analysis (LDA)**	10	
7.		**Regression Analysis**	11-13	
8.		**Regularized Regression**	14-15	
9.		**K-Nearest Neighbour (KNN) Classifier**	16-17	
10.		**Support Vector Machines (SVMs)**	18	
11.		**Random Forest model**	19	
12.		**AdaBoost Classifier & XGBoost**	20-22	

1. Basic statistical functions for data exploration

Program:

```
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)
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)
```

OUTPUT:

```
    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np

        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
        CRIM    ZN  INDUS  CHAS    NOX  ...  RAD    TAX  PTRATIO       B  LSTAT
0    0.00632  18.0   2.31   0.0  0.538  ...  1.0  296.0     15.3  396.90   4.98
1    0.02731   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  396.90   9.14
2    0.02729   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  392.83   4.03
3    0.03237   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  394.63   2.94
4    0.06905   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  396.90   5.33
```

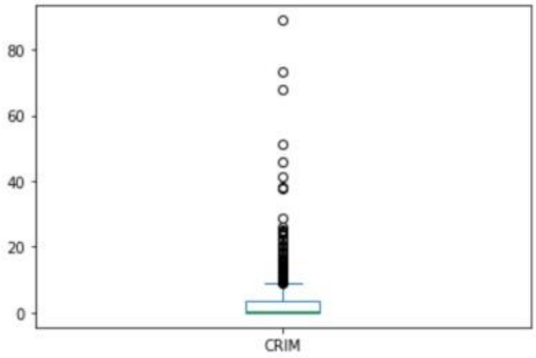
```
    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.
  warnings.warn(msg, category=FutureWarning)
        CRIM    ZN  INDUS  CHAS    NOX  ...  RAD    TAX  PTRATIO       B  LSTAT
0    0.00632  18.0   2.31   0.0  0.538  ...  1.0  296.0     15.3  396.90   4.98
1    0.02731   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  396.90   9.14
2    0.02729   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  392.83   4.03
3    0.03237   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  394.63   2.94
4    0.06905   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  396.90   5.33
..       ...   ...    ...   ...    ...  ...  ...    ...      ...     ...    ...
501  0.06263   0.0  11.93   0.0  0.573  ...  1.0  273.0     21.0  391.99   9.67
502  0.04527   0.0  11.93   0.0  0.573  ...  1.0  273.0     21.0  396.90   9.08
503  0.06076   0.0  11.93   0.0  0.573  ...  1.0  273.0     21.0  396.90   5.64
504  0.10959   0.0  11.93   0.0  0.573  ...  1.0  273.0     21.0  393.45   6.48
505  0.04741   0.0  11.93   0.0  0.573  ...  1.0  273.0     21.0  396.90   7.88

[506 rows x 13 columns]
             CRIM          ZN       INDUS  ...     PTRATIO           B       LSTAT
count  506.000000  506.000000  506.000000  ...  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779  ...   18.455534  356.674032   12.653063
std      8.601545   23.322453    6.860353  ...    2.164946   91.294864    7.141062
min      0.006320    0.000000    0.460000  ...   12.600000    0.320000    1.730000
25%      0.082045    0.000000    5.190000  ...   17.400000  375.377500    6.950000
50%      0.256510    0.000000    9.690000  ...   19.050000  391.440000   11.360000
75%      3.677083   12.500000   18.100000  ...   20.200000  396.225000   16.955000
max     88.976200  100.000000   27.740000  ...   22.000000  396.900000   37.970000

[8 rows x 13 columns]
[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.]), count=array([43])), ModeR
esult(mode=array([3.4952]), count=array([5])), ModeResult(mode=array([24.]), count=array([132])), ModeResult(mode=array([666.]), count=array([132])), ModeResult(mode=array([20.2]), co
unt=array([140])), ModeResult(mode=array([396.9]), count=array([121])), ModeResult(mode=array([6.36]), count=array([3]))]
```

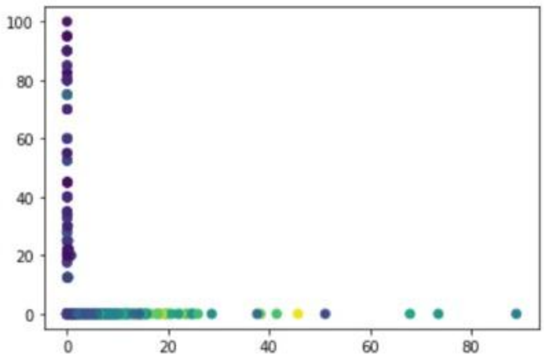
2. Data Visualization: Box plot, scatter plot, histogram

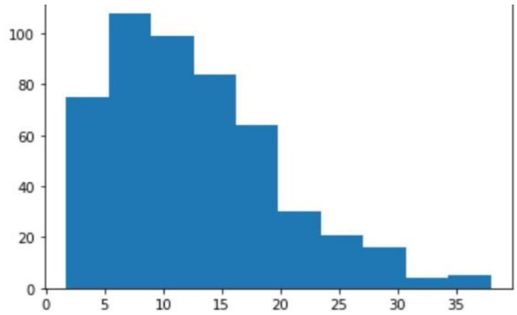
Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
i=load_boston()
df=pd.DataFrame(i.data,columns=i.feature_names)
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()
```

OUTPUT:







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

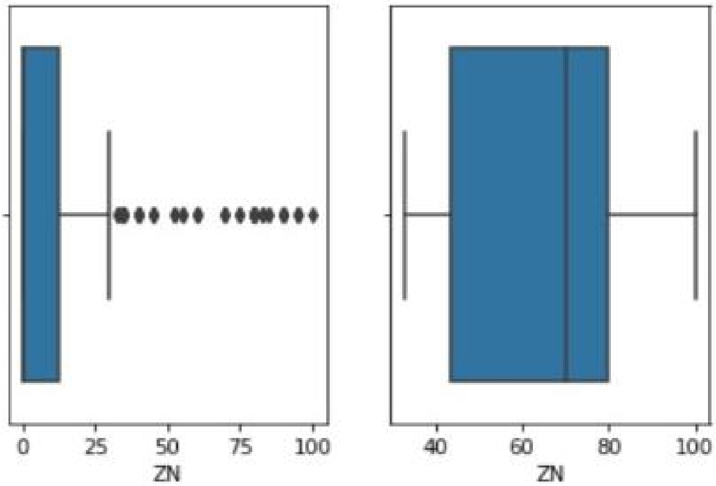
Program:

```
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
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.isna().sum(),'\n')
#checking outliers
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
```

OUTPUT:

```
Out[2]: 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]])
```

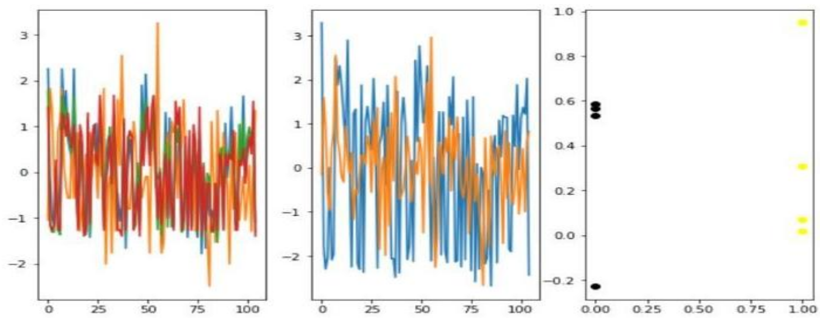


4. Principal Component Analysis (PCA)
Program:

```
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_state=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])
```

OUTPUT:

```
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)

Program:

```
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)
```

OUTPUT:

```
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 ]]
```

ś

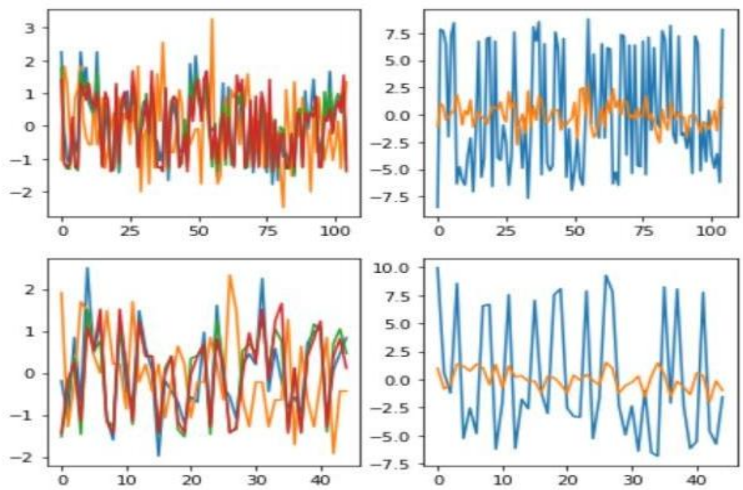
6. Linear Discriminant Analysis (LDA)

Program:

```
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)
```

OUTPUT:

```
Out[5]:  [<matplotlib.lines.Line2D at 0x7f8cb53cafa0>,
          <matplotlib.lines.Line2D at 0x7f8cb53d9040>]
```



7. Regression Analysis:

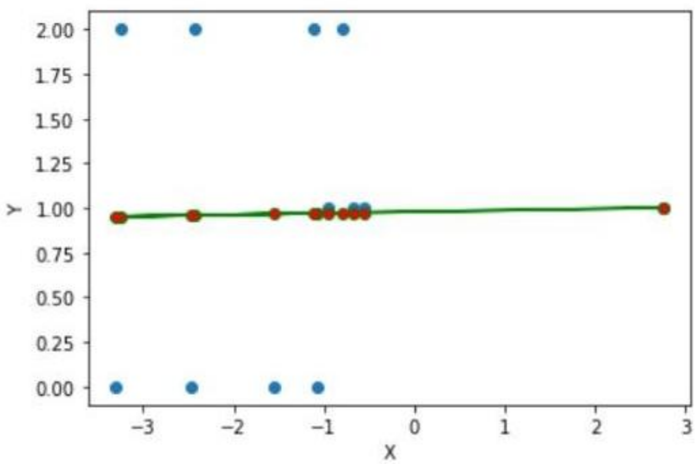
7 (A) .Linear Regression

Program:

```
from sklearn.model_selection import train_test_splitfrom
sklearn.preprocessing import StandardScaler from
sklearn.linear_model import LinearRegressionfrom
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_state=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")
```

OUTPUT:

```
plt.plot(x_test,y_pred,c='green',marker='o',markerfacecolor='red')
plt.xlabel("X")
plt.ylabel("Y")

slope: [0.00814197]
Intercept: 0.9789237089972992
Score: 0.00012075418594370557
Accuracy: -7.164353685640279e-05

Out[6]: Text(0, 0.5, 'Y')
```



7 (B) .Logistic Regression

Program:

```
import pandas as pdimport
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_matriximport
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.va
lues),yticklabels=pd.unique(y.values))
```

OUTPUT:

```
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

Out[7]: <AxesSubplot:>
```

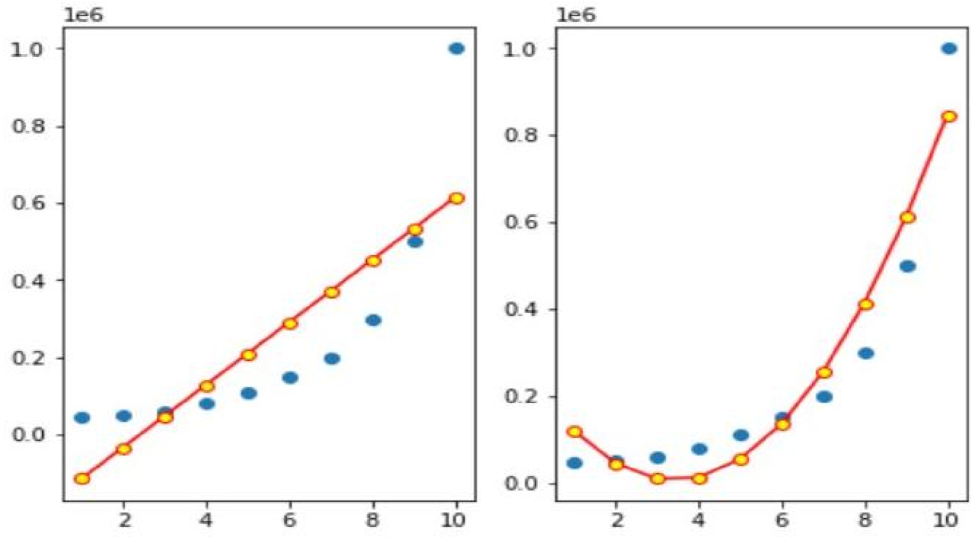
7 C). Polynomial Regression

Program:

```
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')
```

OUTPUT:

```
[<matplotlib.lines.Line2D at 0x7f8cafff1730>]
```



8. Regularized Regression

Program:

```
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_state=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)
```

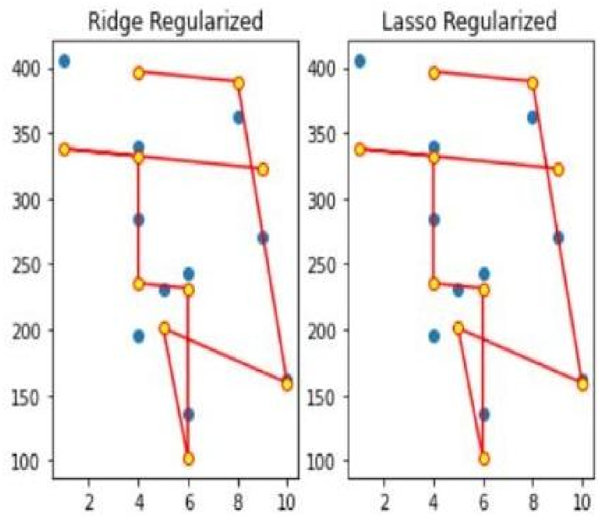
```
plt.title('Ridge Regularized')
plt.scatter(x_test['month'][:10],y_test[:10])
plt.plot(x_test['month'][:10],y_pred[:10],c='red',marker='o',markerfacecolor='yellow')
plt.subplot(1,2,2)
plt.title('Lasso Regularized')
plt.scatter(x_test['month'][:10],y_test[:10])
plt.plot(x_test['month'][:10],y_pred1[:10],c='red',marker='o',markerfacecolor='yellow')
```

OUTPUT:

```
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: 0.857261218618013
accuracy: 0.8240219138855976
```

```
Out[10]: [<matplotlib.lines.Line2D at 0x7f8cafe9dd00>]
```



9. K-Nearest Neighbour (KNN) Classifier

Program:

```
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_state=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)))
```

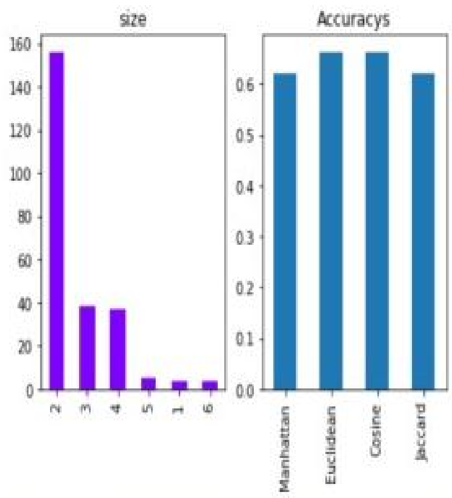
```
D4.append(accuracy_score(y_test,knn3.predict(x_test)))
D=np.array((D1,D2,D3,D4)).T
dff=pd.DataFrame(D,columns=['Manhattan','Euclidean','Cosine','Jaccard'],index=[i
for i in range(1,11)])
print(dff)
plt.subplot(1,2,1)
plt.title('size')
df['size'].value_counts().plot(kind='bar',cmap='rainbow')
plt.subplot(1,2,2)
plt.title('Accuracys')
np.max(dff).plot(kind='bar')
```

OUTPUT:

	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

```
/home/shesiram/20761A0576/lib/python3.8/site-packages/numpy/core/fromnumeric.py:84: FutureWarning: In a future version, DataFrame.max(axis=None) will return a scalar max over the entire DataFrame. To retain the old behavior, use 'frame.max(axis=0)' or just 'frame.max()'
  return reduction(axis=axis, out=out, **passkwargs)
```

Out[12]: <AxesSubplot:title={'center':'Accuracys'}>

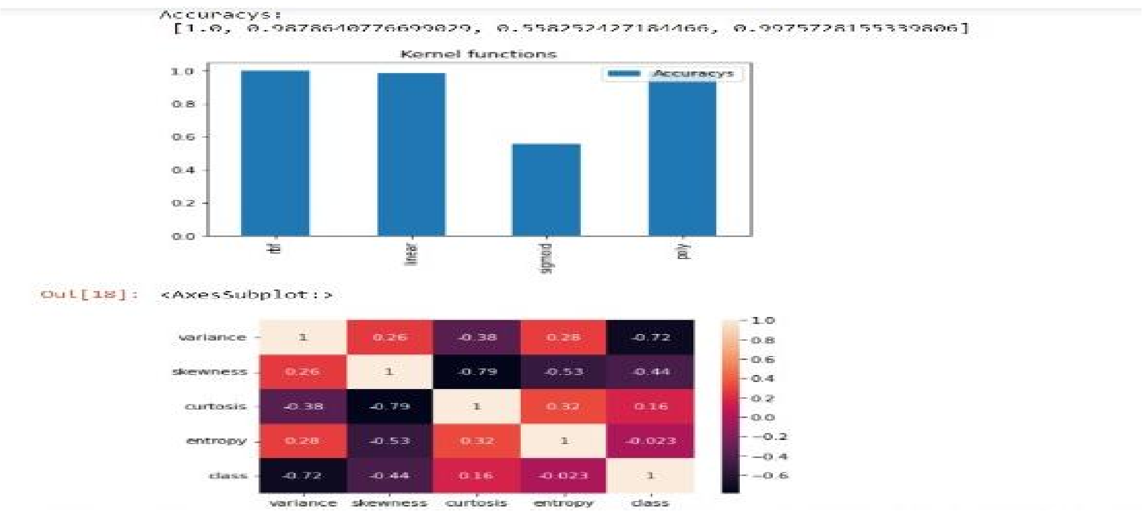


10. Support Vector Machines (SVMs)

Program:

```
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', 'skewness', 'curtosis', 'entropy']]
y=df['class']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=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)
```

OUTPUT:



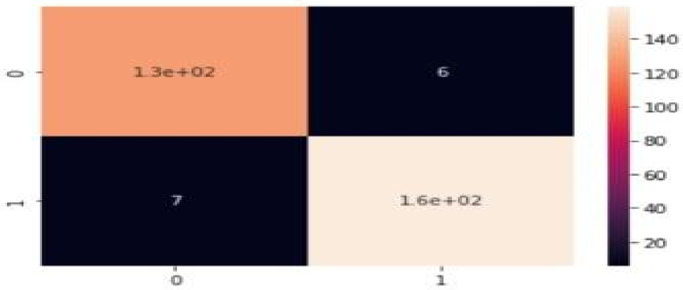
11. Random Forest model

Program:

```
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=Fals
e)
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)
```

OUTPUT:

```
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]
Out[19]: <AxesSubplot:>
```



12.(a)AdaBoost Classifier

Program:

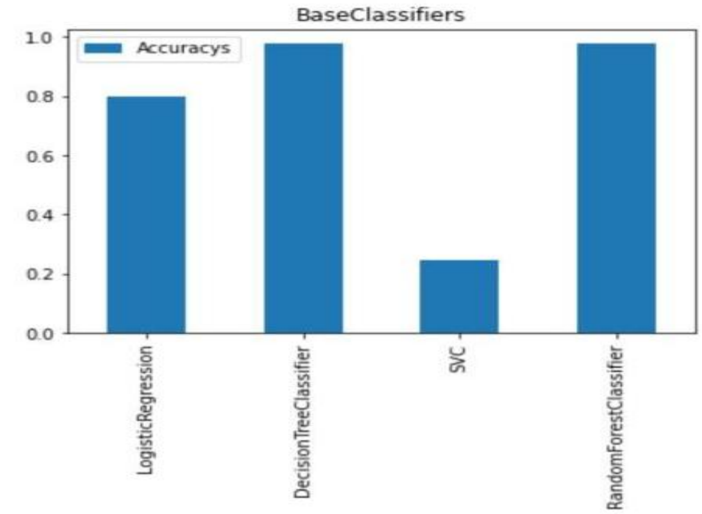
```
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
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','RandomForestClassifi
er']
instance=[lrc,dtc,svm,rfc]
s=[]
for i in instance:
    ada=AdaBoostClassifier(n_estimators=100,base_estimator=i,learning_rate=1.0,rando
    m_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')
```

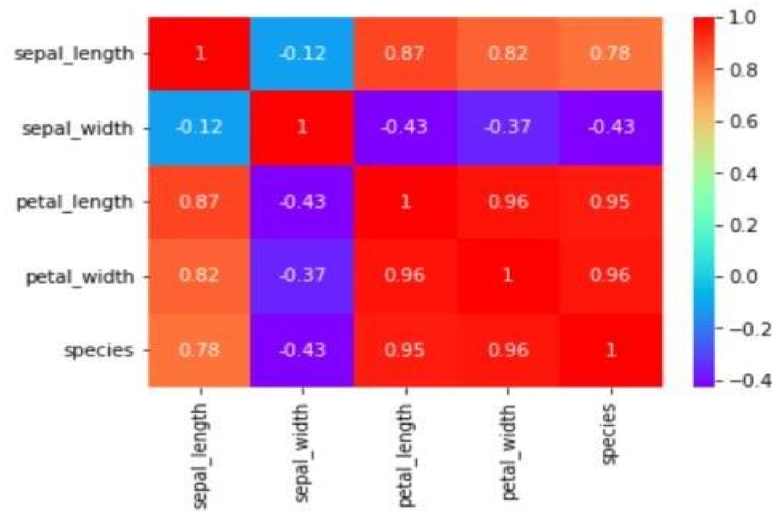
OUTPUT:

```
sns.heatmap(df.corr(),annot=True,cmap='rainbow')
```

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




```
Out[22]: <AxesSubplot:>
```



(b). XGBoost

Program:

```
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
from sklearn.preprocessing import LabelEncoder
df=sns.load_dataset('iris')
le=LabelEncoder()
df['species']=le.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_state=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)
```

OUTPUT:

```
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]
```