1)simple linear Regression usin python

2)find sse,ssr,sst using linear Regression

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_iris**

**data=load\_iris()**

**x=data.data**

**y=data.target**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)**

**from sklearn.linear\_model import LinearRegression**

**sc=LinearRegression()**

**sc.fit(x\_train,y\_train)**

**ypred=sc.predict(x\_test)**

**print("score",sc.score(x\_test,y\_test))**

**sse=np.sum((y\_test-ypred)\*\*2)**

**ssr=np.sum((ypred-y.mean())\*\*2)**

**sst=sse+ssr**

**print("sse =",sse)**

**print("ssr= ",ssr)**

**print("sst= ",sst)**

3)implement Decision tree using iris dataset

4) implement precision,recall,f1-score of decision tree

22)implement confusion matrix,accuracy score using decision tree

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_iris**

**data=load\_iris()**

**x=data.data**

**y=data.target**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)**

**from sklearn.tree import DecisionTreeClassifier**

**sc=DecisionTreeClassifier(criterion='entropy')**

**sc.fit(x\_train,y\_train)**

**y\_pred=sc.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

7) knn using iris dataset

8) find accuracy score and confusion matrix of knn

9) knn using iris dataset

10) demonstrate precision,recall,f1-score of knn

11) correct and wrong prediction and accuracy of knn

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_iris**

**data=load\_iris()**

**x=data.data**

**y=data.target**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)**

**from sklearn.preprocessing import StandardScaler**

**sc\_x=StandardScaler()**

**xtrain=sc\_x.fit\_transform(x\_train)**

**xtest=sc\_x.transform(x\_test)**

**from sklearn.neighbors import KNeighborsClassifier**

**sc=KNeighborsClassifier(n\_neighbors=5)**

**sc.fit(xtrain,y\_train)**

**y\_pred=sc.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

12) print correct and wrong prediction and print accuracy of naïve bayes classifier model

13) implement naïve bayes classification using iris dataset

14) Print Precision,recall,F1-score of naïve bayes classification model

15) Demonstrate Accuracy and confusion matrix of naïve bayes classification

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_iris**

**data=load\_iris()**

**x=data.data**

**y=data.target**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)**

**from sklearn.preprocessing import StandardScaler**

**sc\_x=StandardScaler()**

**xtrain=sc\_x.fit\_transform(x\_train)**

**xtest=sc\_x.transform(x\_test)**

**from sklearn.naive\_bayes import GaussianNB**

**sc= GaussianNB()**

**sc.fit(xtrain,y\_train)**

**y\_pred=sc.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

20) and 21) implement decision tree using breast cancer dataset

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_breast\_cancer**

**data=load\_breast\_cancer()**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)**

**from sklearn.tree import DecisionTreeClassifier**

**sc=DecisionTreeClassifier(criterion='entropy')**

**sc.fit(x\_train,y\_train)**

**y\_pred=sc.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

24) Implement knn to classify breast cancer dataset

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_breast\_cancer**

**data=load\_breast\_cancer()**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)**

**from sklearn.preprocessing import StandardScaler**

**sc=StandardScaler()**

**xtrain=sc.fit\_transform(x\_train)**

**xtest=sc.transform(x\_test)**

**from sklearn.neighbors import KNeighborsClassifier**

**sc\_x=KNeighborsClassifier(n\_neighbors=5)**

**sc\_x.fit(x\_train,y\_train)**

**y\_pred=sc.predict(x\_test)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

25) implement naïve bayes classifier to classify breast cancer dataset

**import pandas as pd**

**import numpy as np**

**from sklearn.datasets import load\_breast\_cancer**

**data=load\_breast\_cancer()**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(data.data[:,0:9],data.target,test\_size=0.3,random\_state=42)**

**from sklearn.preprocessing import StandardScaler**

**sc=StandardScaler()**

**xtrain=sc.fit\_transform(x\_train)**

**xtest=sc.transform(x\_test)**

**from sklearn.naive\_bayes import GaussianNB**

**sc\_x=GaussianNB()**

**sc\_x.fit(x\_train,y\_train)**

**from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report**

**print(confusion\_matrix(y\_test,y\_pred))**

**print("Accuracy score",accuracy\_score(y\_test,y\_pred))**

**print(" classification report",classification\_report(y\_test,y\_pred))**

19)write a program to implement multivariable linear regression and find sse,ssr,sst

**import pandas as pd**

**import numpy as np**

**data=pd.read\_csv("abc.csv")**

**x=data.iloc[:,:-1].values**

**y=data.iloc[:,2].values**

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=0)**

**from sklearn.linear\_model import LinearRegression**

**sc=LinearRegression()**

**sc.fit(x\_train,y\_train)**

**ypred=sc.predict(x\_test)**

**print("score",sc.score(x\_test,y\_test))**

**sse=np.sum((y\_test-ypred)\*\*2)**

**ssr=np.sum((ypred-y.mean())\*\*2)**

**sst=sse+ssr**

**print("sse =",sse)**

**print("ssr= ",ssr)**

**print("sst= ",sst)**

18) Implement Fuzzy c mean clustering technique

**import pandas as pd**

**import numpy as np**

**import skfuzzy as f**

**np.random.seed(0)**

**data=np.random.rand(100,2)**

**nc=3**

**cnt,u,u0,d,jm,p,fpc=f.cluster.cmeans(data.T,nc,2,error=0.005,maxiter=10)**

**clm=np.argmax(u,axis=0)**

**print("cluster center : ",cnt)**

**print("cluster membership:",clm)**