

**SINHAGAD INSTITUTE OF BUSINESS ADMINISTRATION AND RESEARCH**

**(SIBAR)-PUNEDEPARTMENTOFMCA**

MASTERINCOMPUTERAPPLICATIONFROM SAVITRIBAI PHULE UNIVERSITY PUNE

KRAI:ML,DLJOURNAL ON

Subject:IT34KnowledgeRepresentationandArtificial Intelligence.

IN

MASTERINCOMPUTERAPPLICATION

SUBMITTEDBY

**ShubhamSanjayPatil(105)**

SAVITRIBAIPHULEPUNEUNIVERSITY

SINHGADINSTITUTEOFBUSINESSADMINISTRATIONANDRESEARCH KONDHWA, PUNE-411048

2022-2023

***UNDERTHESUPERVISIONOF***

***Dr.NetraPatil***

**Sinhgad Technical Education Society’sSinhgadInstituteofBusinessAdministrationandResearch**

**Kondhwa(Bk.),Pune411048**

**Subject:IT34-KnowledgeRepresentationandArtificialIntelligence:ML,DLPractical**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **AssignmentQuestion** | **Page No.** | **Date** | **Sign** |
| 1 | WriteaPythonprogramtoFindthecorrelationmatrix. |  |  |  |
| 2 | Plotthecorrelationplotondatasetandvisualizegivinganoverviewofrelationships among data on iris data. |  |  |  |
| 3 | WriteaPythonprogramtopredictmpg(milespergallon)foracarbasedonvariablewt by applying simple linear regression on 'mtcars' dataset (Use Training data 80% and Testing Data 20%).  Recordtheperformance ofmodelintermsofMAE,MSE,RMSEandR-squaredvalue.  ChangeTrainingdatato70%andTestingData30%,compare&interpretthe performance of your model. |  |  |  |
| 4 | WriteaPythonprogramtopredictmpg(milespergallon)foracarbasedonvariableswt, cyl & disp by applying multi-linear regression on 'mtcars' dataset (Use Training data 80% and Testing Data 20%).  Recordtheperformance ofmodelintermsofMAE,MSE,RMSEandR-squaredvalue.  Removevariabledispfromthefeaturesetandcheckthe performanceagain.Compare& interpret the performance of |  |  |  |
| 5 | Write a Python program to predict mpg (miles per gallon) for a car based on variables wt, cyl & disp by applying multi-linear regression on 'mtcars' dataset (UseTraining data 80% and Testing Data 20%).  Recordtheperformance ofmodelintermsofMAE,MSE,RMSEandR-squaredvalue.  Replacedispbydratvariableinthefeaturesetandchecktheperformanceagain.Interpret the performance of your model. |  |  |  |
| 6 | Writea Python program to predictfruit (Apple or Orange)based onits size & weightby applying logistic regression on 'apples\_and\_oranges' dataset (Use Training data 80% And Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracy Scoremetric, Classification  Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthemodel performance. |  |  |  |
| 7 | WriteaPythonprogramtopredictfruit(AppleorOrange)basedonitssize&weight by applying K-Nearest Neighbour (KNN) model on 'apples\_and\_oranges' dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracy Scoremetric, Classification  Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthe model performance. |  |  |  |
| 8 | WriteaPythonprogramtopredictfruit(AppleorOrange)basedonitssize&weight byapplyingSupportVectorMachine(SVM)modelon'apples\_and\_oranges'dataset  (UseTrainingdata80%andTestingData20%). |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Evaluate the performance of themodel usingAccuracy Score metric, Classification Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthemodel performance. |  |  |  |
| 9 | WriteaPythonprogramtopredictspecies(Setosa,Versicolor,orViriginica)fora  new iris flower based on length & width of its petals and sepals by applying logistic regression model on 'iris' dataset (Use Training data 80% and Testing Data 20%).  Evaluate the performance of themodel usingAccuracy Score metric, Classification Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthemodel performance. |  |  |  |
| 10 | Write a Python program to predict species (Setosa, Versicolor, or Viriginica) for a new irisflowerbasedonlength&widthofitspetalsandsepalsbyapplyingK-Nearest Neighbour(KNN)modelon'iris'dataset(UseTrainingdata80%andTestingData20%). Evaluate the performance of the model usingAccuracy Score metric, Classification Report & Confusion Matrix,AUC ROC score for the model and interpret the model  performance. |  |  |  |
| 11 | Write a Python program to predict species (Setosa, Versicolor, or Viriginica) for a new iris flower based on length & width of its petals and sepals by applying Support Vector Machine (SVM) model on 'iris' dataset (Use Training data 80% and Testing Data 20%). EvaluatetheperformanceofthemodelusingAccuracyScoremetric,Classification  Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthe modelperformance. |  |  |  |
| 12 | Write a Python program to predict species (Setosa, Versicolor, or Viriginica) for a new iris flower based on length & width of its petals and sepals by applying Naive Bays Classification model on 'iris' dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracy Scoremetric, Classification  Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthemodel performance. |  |  |  |
| 13 | Write a Python program to predict species (Setosa, Versicolor, or Viriginica) for a new iris flower based on length & width of its petals and sepals by applying Decision Tree model on 'iris' dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracyScoremetric,Classification  Report&ConfusionMatrix,AUCROCscoreforthemodelandinterpretthe modelperformance. |  |  |  |
| 14 | Write aPythonprogram toimplementtheK-meansAlgorithmonunsuperviseddata of a mall, that contains the basic  information(ID,age,gender,income,spendingscore)aboutthecustomers.Findthe clusters based on the income and spending. |  |  |  |
| 15 | Write a Python program to implement the Agglomerative Hierarchical Clustering Algorithm on unsupervised data of a mall, that contains the basic information (ID, age, gender, income, spending score) about the customers. Find the clusters based on the income and spending. |  |  |  |

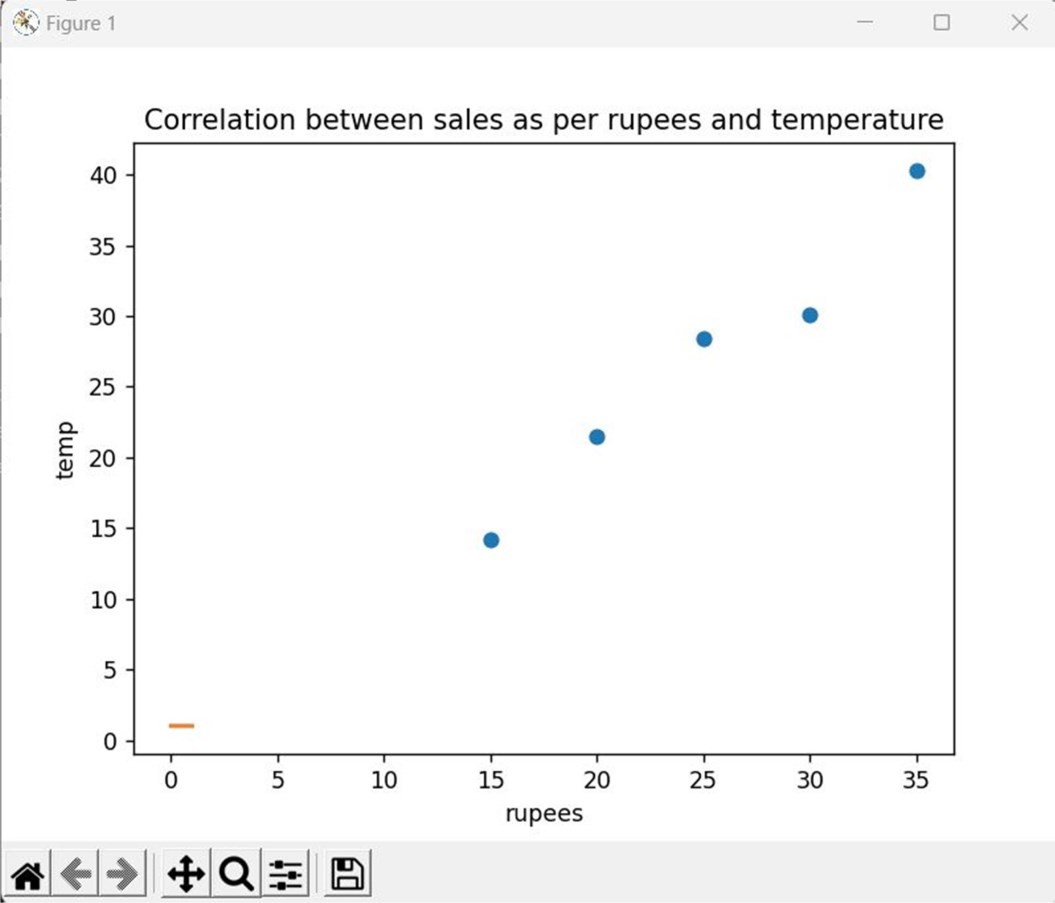
|  |  |
| --- | --- |
| 1 | WriteaPythonprogramtoFindthecorrelationmatrix. |

importnumpyasnpimportmatplotlib.pyplotaspltx=[15,20,25,30,35],y=[14.2,21.5,28.4,30.1,40.3]matrix= np.corrcoef(x, y) print(matrix) plt.scatter(x,y)

plt.title('Correlationbetweensalesasperrupeesandtemperature') plt.xlabel('rupees') plt.ylabel('temp') plt.plot(matrix)

plt.show()

Output:



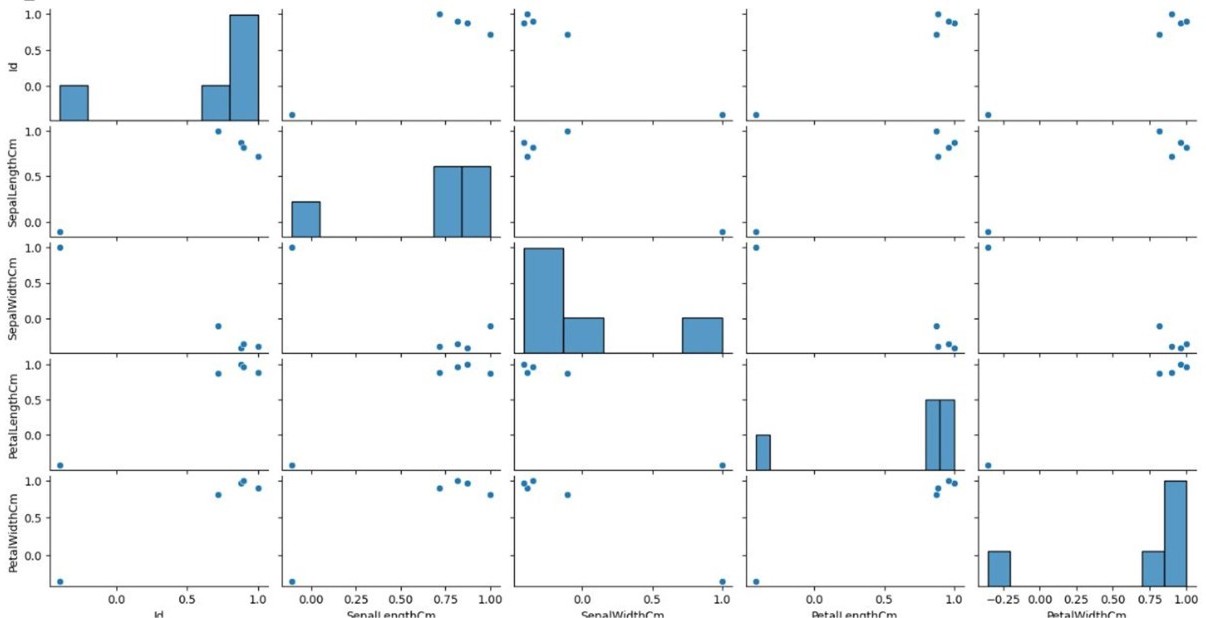
|  |  |
| --- | --- |
| 2 | Plotthecorrelationplotondatasetandvisualizegivinganoverviewofrelationshipsamongdataonirisdata. |

Program:

importpandasaspdimportseabornassnsimportmatplotlib.pyplotaspltdf=pd.read\_csv("Iris.csv") rel = df.corr() print(rel)

sns.pairplot(rel)plt.show()

Output:



|  |  |
| --- | --- |
| 3 | WriteaPythonprogramtopredictmpg(milespergallon)foracarbasedonvariablewt by applying simple linear regression on 'mtcars' dataset (Use Training data 80% and Testing Data 20%).  Recordtheperformance ofmodelintermsofMAE,MSE,RMSEandR-squaredvalue.  ChangeTrainingdatato70%andTestingData30%,compare&interprettheperformanceofyour model. |

Program:

importpandasaspd

fromseabornimportload\_datasetimport statsmodels.formula.api as sm import statsmodels.stats.multicomp as multi

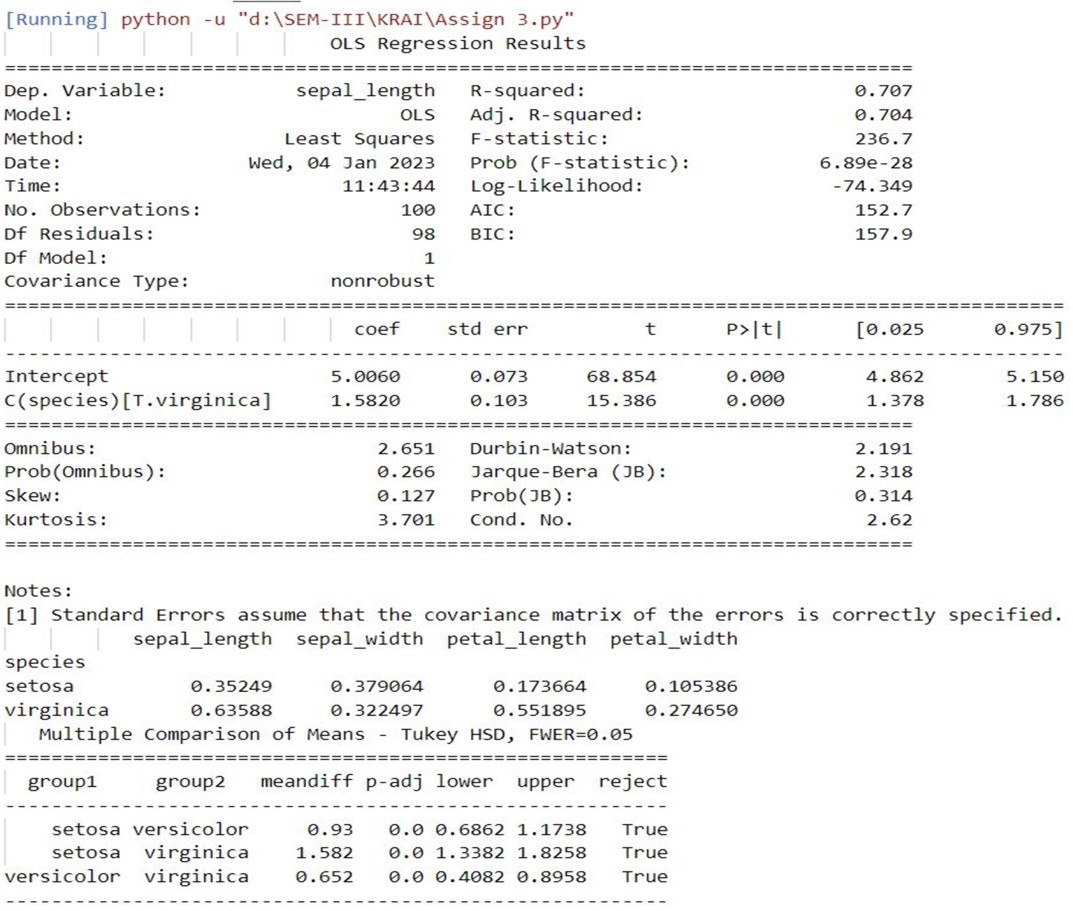
iris=load\_dataset("iris")

my\_subset=iris[iris["species"].isin(['setosa','virginica'])]

subset\_model=sm.ols(formula='sepal\_length~C(species)',data=my\_subset) print(subset\_model.fit().summary()) my\_subset.groupby("species").mean() print(my\_subset.groupby("species").std())

multi\_comp=multi.MultiComparison(iris['sepal\_length'],iris['species'])print(multi\_comp.tukeyhsd().summary())

Output:



|  |  |
| --- | --- |
| 4 | WriteaPythonprogramtopredictmpg(milespergallon)foracarbasedonvariableswt, cyl & disp by applying multi-linear regression on 'mtcars' dataset (Use Training data80% and Testing Data 20%).  RecordtheperformanceofmodelintermsofMAE,MSE,RMSEandR-squaredvalue. Remove variable disp from the feature set and check the performance again. |

Program:

importnumpyasnpimportmatplotlib.pyplotaspltimportpandasaspd dataset = pd.read\_csv('Salary\_Data.csv') X =

dataset.iloc[:,:-1].valuesy=dataset.iloc[:,1].values fromsklearn.model\_selectionimporttrain\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0) from sklearn.linear\_modelimportLinearRegressionregressor=LinearRegression()regressor.fit(X\_train, y\_train) y\_pred = regressor.predict(X\_test) plt.scatter(X\_train, y\_train, color = 'red') plt.plot(X\_train, regressor.predict(X\_train), color = 'blue') plt.scatter(X\_test, y\_test, color =

'red')

plt.plot(X\_train,regressor.predict(X\_train),color='blue')plt.title('Salary vs Experience (Test set)') plt.xlabel('Years of Experience') plt.ylabel('Salary') plt.show()

|  |  |
| --- | --- |
| 5 | Write a Python program to predict mpg (miles per gallon) for a car based on variables wt, cyl & disp by applying multi-linear regression on 'mtcars' dataset (UseTraining data 80% and Testing Data 20%).  Recordtheperformance ofmodelintermsofMAE,MSE,RMSEandR-squaredvalue.  Replacedispbydratvariableinthefeaturesetandchecktheperformanceagain.Interpret the performance of your model. |

Program:

importnumpyasnpimportmatplotlib.pyplotaspltimportpandasas pd from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LogisticRegression from sklearn.metrics import confusion\_matrix from sklearn.metrics import accuracy\_score from matplotlib.colors import ListedColormap dataset = pd.read\_csv('DMVWrittenTests.csv') X = dataset.iloc[:, [0, 1]].values y = dataset.iloc[:, 2].values

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.25,random\_state=0)sc= StandardScaler()

X\_train=sc.fit\_transform(X\_train)X\_test=sc.transform(X\_test) classifier = LogisticRegression() classifier.fit(X\_train, y\_train) y\_pred=classifier.predict(X\_test)cm=confusion\_matrix(y\_test, y\_pred) print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

df = pd.DataFrame({'Real Values':y\_test, 'PredictedValues':y\_pred}) X\_set, y\_set = X\_test, y\_test

X1,X2=np.meshgrid(np.arange(start=X\_set[:,0].min()-1,stop=X\_set[:,0].max()+1,step= 0.01),np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1,X2,classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),alpha=0.75,cmap

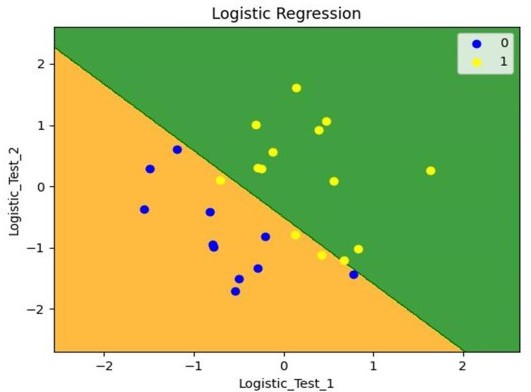
=ListedColormap(('orange', 'green')))

plt.xlim(X1.min(), X1.max()) plt.ylim(X2.min(), X2.max())fori,jinenumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set==j,0],X\_set[y\_set==j,1],c=ListedColormap(('blue','yellow'))(i),label=j) plt.title('Logistic Regression') plt.xlabel('Logistic\_Test\_1') plt.ylabel('Logistic\_Test\_2')

plt.legend()plt.show()

Output:



|  |  |
| --- | --- |
| 6 | WriteaPythonprogramtopredictfruit(AppleorOrange)basedonitssize&weightby applying logistic regression on 'apples\_and\_oranges' dataset (Use Training data 80% And Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracyScoremetric,ClassificationReport& Confusion Matrix,AUC ROC score for the model and interpret the model performance. |

Program:

from numpyimport uniquefrom numpyimport where fromsklearn.datasetsimportmake\_classificationfrom sklearn.clusterimportKMeansfrommatplotlibimport pyplot

#definedataset

X,\_=make\_classification(n\_samples=1000,n\_features=2,n\_informative=2,n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4)

#definethemodel

model=KMeans(n\_clusters=2) # fit the model model.fit(X)

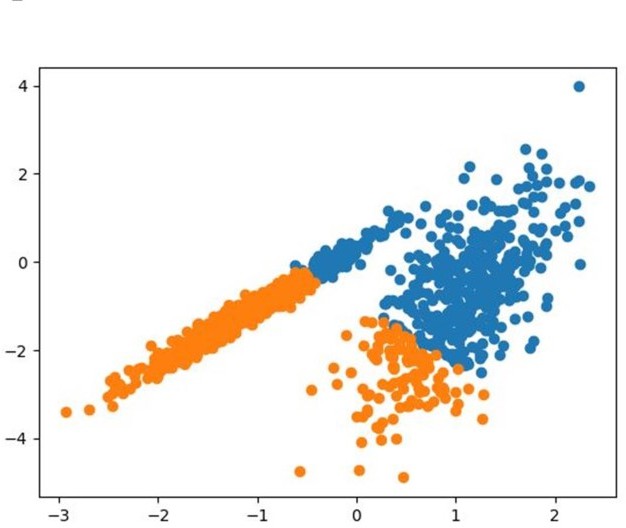
#assignaclustertoeachexampleyhat= model.predict(X) # retrieve unique clusters clusters = unique(yhat)

#createscatterplotforsamplesfromeachclusterforclusterin clusters:

#getrowindexesforsampleswiththisclusterrow\_ix

=where(yhat==cluster)#createscatterofthese samples

pyplot.scatter(X[row\_ix,0],X[row\_ix,1]) # show the plot pyplot.show()

Output:

|  |  |
| --- | --- |
| 7 | WriteaPythonprogramtopredictfruit(AppleorOrange)basedonitssize&weight by applying K-Nearest Neighbour (KNN) model on 'apples\_and\_oranges' dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracyScoremetric,Classification Report & Confusion Matrix,AUC ROC score for the model and interpret the  model performance. |

Program:

from numpyimport uniquefrom numpyimport where fromsklearn.datasetsimportmake\_classificationfrom sklearn.cluster import AgglomerativeClustering from matplotlib import pyplot

#definedataset

X,\_=make\_classification(n\_samples=1000,n\_features=2,n\_informative=2,n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4)

#definethemodel

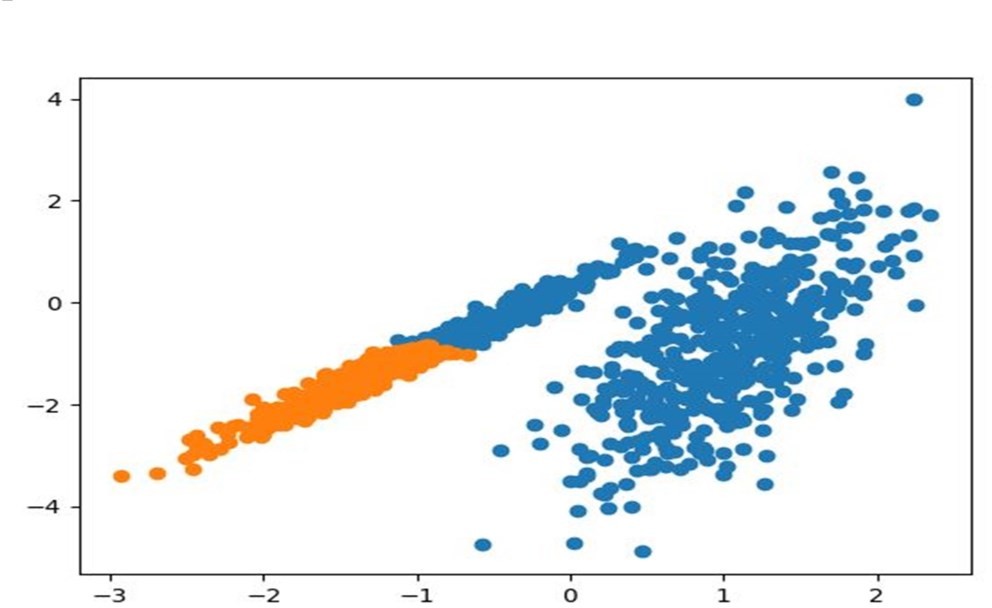
model=AgglomerativeClustering(n\_clusters=2) # fit model and predict clusters

yhat = model.fit\_predict(X) # retrieveuniqueclustersclusters= unique(yhat)

#createscatterplotforsamplesfromeachclusterfor cluster in clusters:

# get row indexes for samples with this cluster row\_ix=where(yhat==cluster)#createscatterof these samples

pyplot.scatter(X[row\_ix,0],X[row\_ix,1]) # show the plot

pyplot.show() Output:

|  |  |
| --- | --- |
| 8 | WriteaPythonprogramtopredictfruit(AppleorOrange)basedonitssize&weight byapplyingSupportVectorMachine(SVM)modelon'apples\_and\_oranges'dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracyScoremetric,ClassificationReport& Confusion Matrix,AUC ROC score for the model and interpret the model performance. |

Program:

importpandasaspdimportnumpyasnpimportmatplotlib.pyplotas plt from sklearn import datasets import scipy.cluster.hierarchy as sc import matplotlib.pyplot as plt from sklearn.cluster import AgglomerativeClustering

# Import iris data iris = datasets.load\_iris() iris\_data = pd.DataFrame(iris.data) iris\_data.columns = iris.feature\_names iris\_data['flower\_type'] = iris.target iris\_data.head() iris\_X = iris\_data.iloc[:, [0, 1, 2,3]].valuesiris\_Y=iris\_data.iloc[:,4].values"""plt.figure(figsize=(10,7))plt.scatter(iris\_X[iris\_Y==0,0], iris\_X[iris\_Y== 0, 1],s=100, c='blue',label='Type 1')plt.scatter(iris\_X[iris\_Y== 1, 0],iris\_X[iris\_Y== 1,

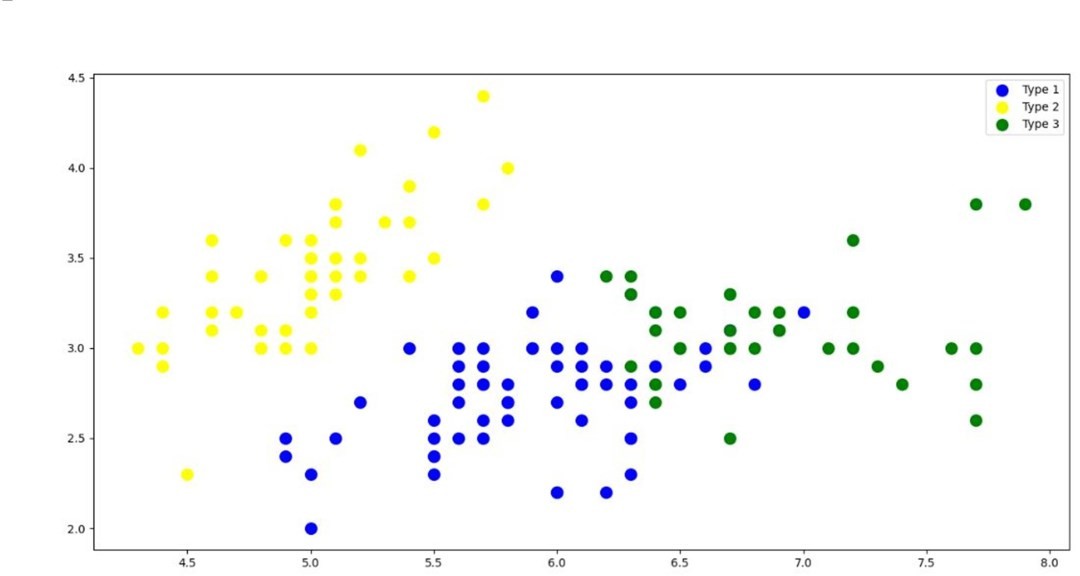
1],s=100,c='yellow',label='Type2')plt.scatter(iris\_X[iris\_Y==2,0],iris\_X[iris\_Y==2,1],s=100, c='green', label='Type 3') plt.legend() plt.show()""" # Plot dendrogram plt.figure(figsize=(20, 7)) plt.title("Dendrograms") # Create dendrogram

sc.dendrogram(sc.linkage(iris\_X,method='ward'))

plt.title('Dendrogram') plt.xlabel('Sample index') plt.ylabel('Euclidean distance') cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward') cluster.fit(iris\_X) labels = cluster.labels\_print(labels)plt.figure(figsize=(10,7))plt.scatter(iris\_X[labels==0,0],iris\_X[labels==0,1],s=

100,c='blue',label='Type1')plt.scatter(iris\_X[labels==1,0],iris\_X[labels==1,1],s=100,c='yellow',label

='Type2')plt.scatter(iris\_X[labels==2,0],iris\_X[labels== 2,1],s=100,c='green',label='Type3')plt.legend() plt.show()

Output:

|  |  |
| --- | --- |
| 9 | WriteaPythonprogramtopredictspecies(Setosa,Versicolor,orViriginica)fora  new iris flower based on length & width of its petals and sepals by applying logistic regression model on 'iris' dataset (Use Training data 80% and Testing Data 20%).  EvaluatetheperformanceofthemodelusingAccuracyScoremetric,ClassificationReport& Confusion Matrix,AUC ROC score for the model and interpret the model performance. |

Program:

importpandasaspdimportmatplotlib.pyplotas pltfrom efficient\_apriori import apriori

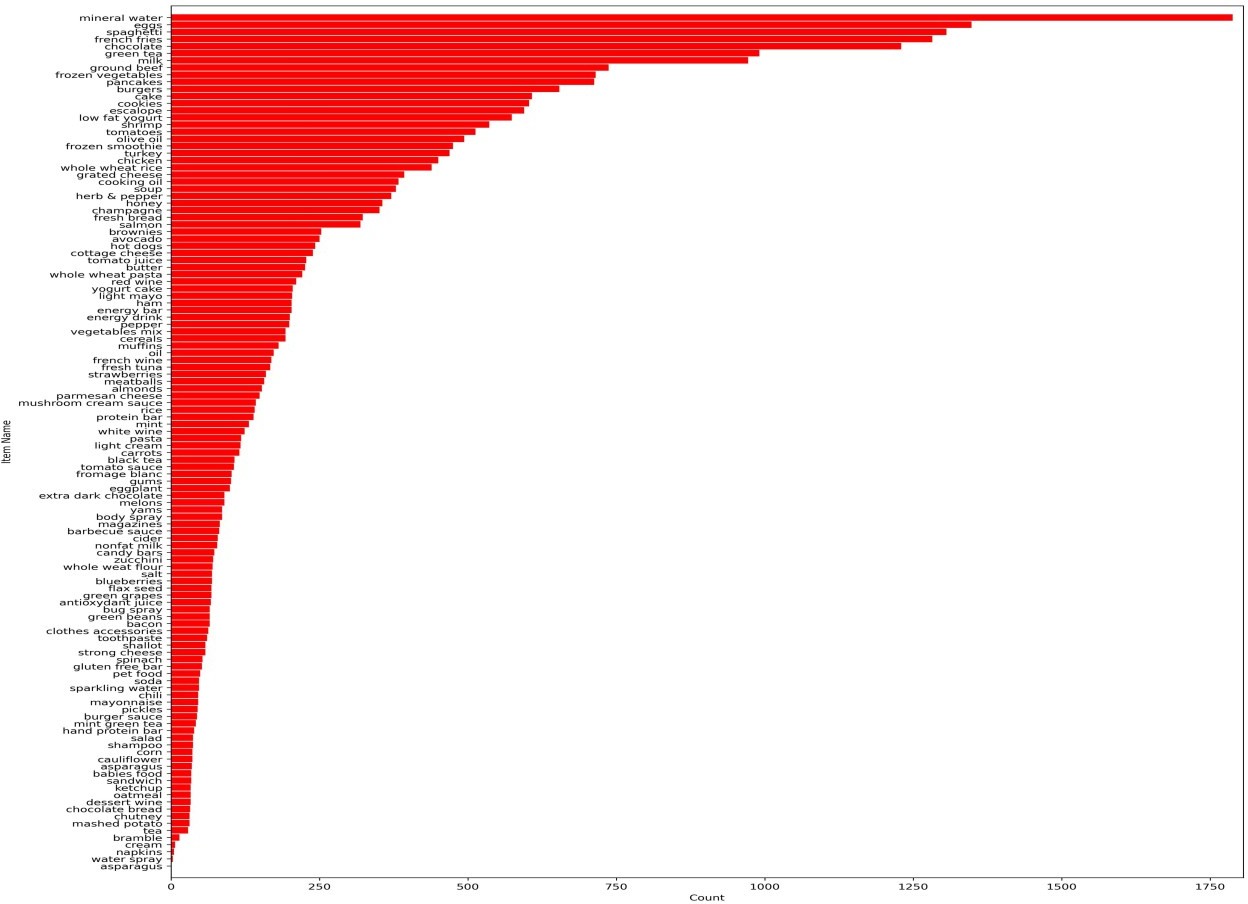
df=pd.read\_csv('Market\_Basket\_Optimisation.csv',encoding='utf-8',header=None) txns=df.values.reshape(-1).tolist() df\_list=pd.DataFrame(txns) df\_list['Count']=1 txns=df.values.reshape(-1).tolist()

df\_list=df\_list.groupby(by=[0],as\_index=False).count().sort\_values(by=['Count'],ascending=True)#count df\_list['Percentage'] = (df\_list['Count'] /df\_list['Count'].sum()) # percentage df\_list=df\_list.rename(columns={0 : 'Item'}) df\_list

plt.figure(figsize=(16,20),dpi=40)plt.ylabel('Item Name') plt.xlabel('Count')

plt.barh(df\_list['Item'],width=df\_list['Count'],color='r',height=0.9) plt.margins(0.01) plt.show()

Output:



Program:

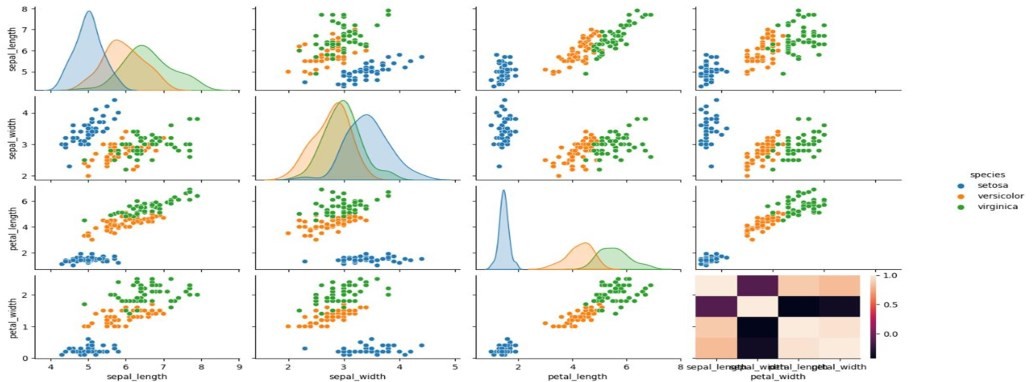
import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.preprocessing import LabelEncoder#for train test splitting from sklearn.model\_selection import train\_test\_split#for decision tree object from sklearn.tree import DecisionTreeClassifier#for checking testing results from sklearn.metrics import classification\_report, confusion\_matrix#for visualizing tree fromsklearn.treeimportplot\_treedf=sns.load\_dataset('iris')df.head()df.info()df.shapedf.isnull().any() sns.pairplot(data=df, hue = 'species') sns.heatmap(df.corr()) target = df['species'] df1 = df.copy() df1 = df1.drop('species', axis =1) X = df1 print(target) le = LabelEncoder() target = le.fit\_transform(target) print(target) y = target

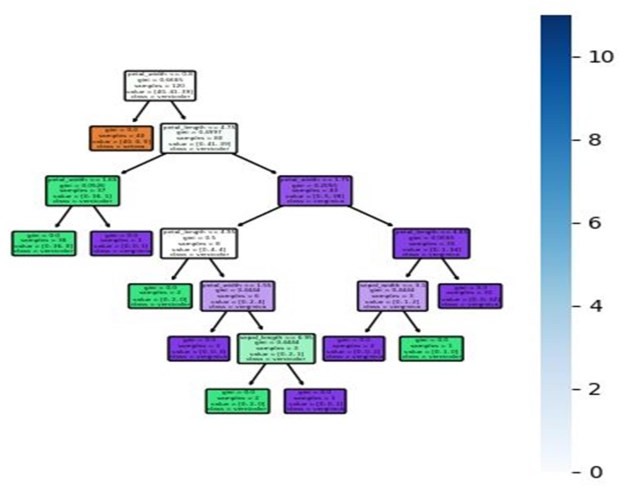
X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42)print("Training split input- ", X\_train.shape) print("Testing split input- ", X\_test.shape) dtree=DecisionTreeClassifier() dtree.fit(X\_train,y\_train) print('Decision Tree Classifier Created') y\_pred = dtree.predict(X\_test) print("Classification report - \n", classification\_report(y\_test,y\_pred))

cm=confusion\_matrix(y\_test,y\_pred)

plt.figure(figsize=(5,5))sns.heatmap(data=cm,linewidths=.5,annot=True,square=True,cmap= 'Blues') plt.ylabel('Actual label') plt.xlabel('Predictedlabel') all\_sample\_title = 'Accuracy Score:

{0}'.format(dtree.score(X\_test,y\_test))plt.title(all\_sample\_title,size=15) dec\_tree = plot\_tree(decision\_tree=dtree, feature\_names = df1.columns,

class\_names=["setosa","vercicolor","verginica"],filled=True,precision=4,rounded=True)plt.show() Output:



Program:

fromsklearn.datasetsimportload\_iris iris = load\_iris()

#storethefeaturematrix(X)andresponsevector(y)X=iris.datay

=iris.target

#splittingXandyintotrainingandtestingsetsfrom 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=3) # training the model on training set from

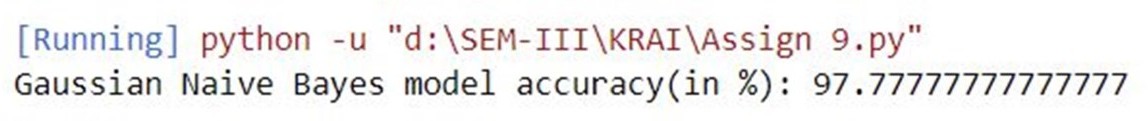
sklearn.naive\_bayesimportGaussianNBgnb= GaussianNB() gnb.fit(X\_train, y\_train)

#makingpredictionsonthetestingsety\_pred= gnb.predict(X\_test)

#comparingactualresponsevalues(y\_test)withpredictedresponsevalues(y\_pred)fromsklearnimport metrics

print("GaussianNaiveBayesmodelaccuracy(in%):",metrics.accuracy\_score(y\_test,y\_pred)\*100)

Output:



Program:

importpandasaspd

fromsklearn.model\_selectionimporttrain\_test\_split from sklearn.svm import SVC dataset\_url = "Fish.csv" fish = pd.read\_csv(dataset\_url)

fish

fish= pd.read\_csv(dataset\_url)

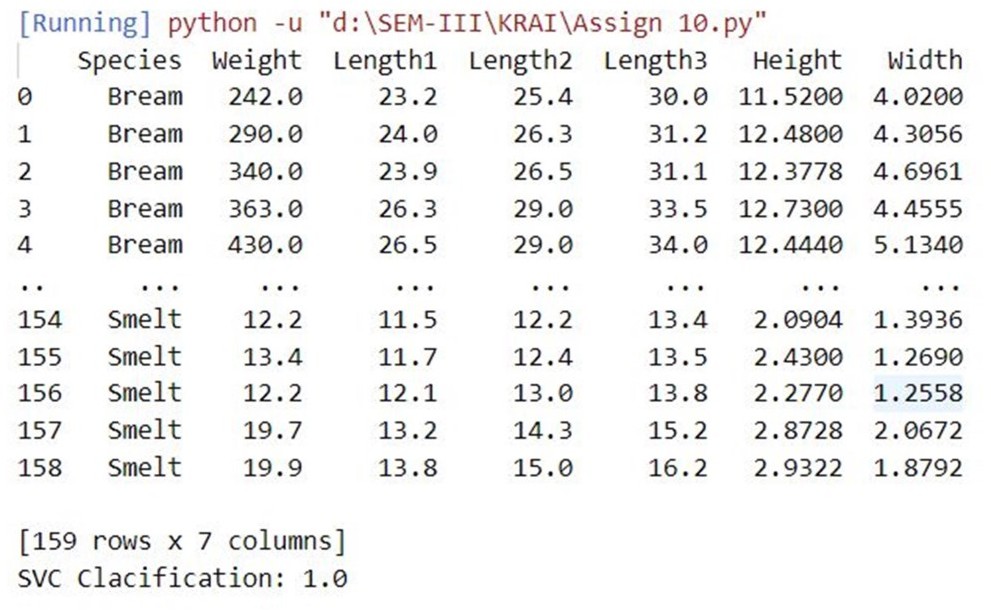
X=fish.drop(['Species'],axis='columns')y=fish.Species

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2) model = SVC(kernel = 'linear', C = 1) model.fit(X\_train,

y\_train)svm\_pred=model.predict(X\_test)accuracy= model.score(X\_test, y\_test) print("SVC Clacification:

{0}".format(accuracy))

OutPut:

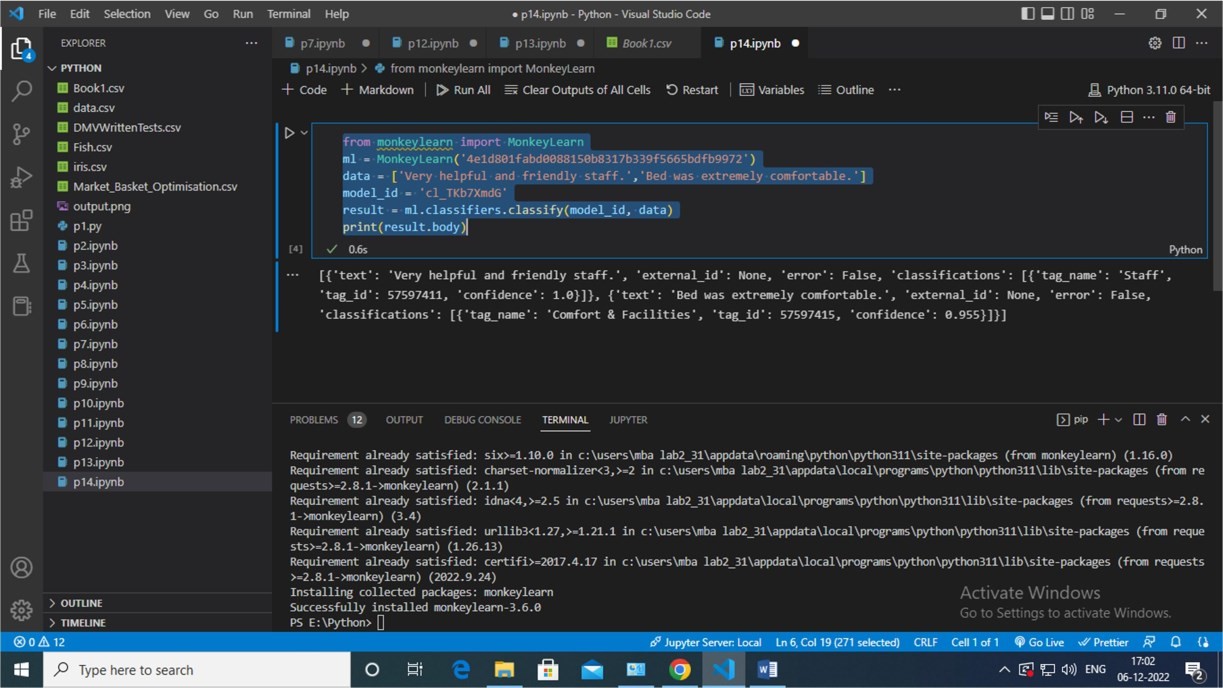


Program:

frommonkeylearnimportMonkeyLearn

ml = MonkeyLearn('4e1d801fabd0088150b8317b339f5665bdfb9972') data=['Veryhelpfulandfriendlystaff.','Bedwasextremelycomfortable.'] model\_id = 'cl\_TKb7XmdG'

result=ml.classifiers.classify(model\_id,data)print(result.body) Output:



Program:

importmatplotlib.pyplotaspltfromkneedimportKneeLocator from sklearn.datasets import make\_blobs from sklearn.cluster import KMeans from sklearn.metrics import silhouette\_score from sklearn.preprocessing import StandardScaler features, true\_labels = make\_blobs(

n\_samples=200,centers=3, cluster\_std=2.75, random\_state=42

)

features[:5]scaler=StandardScaler()

scaled\_features=scaler.fit\_transform(features)scaled\_features[:5]

kmeans\_kwargs={ "init": "random", "n\_init": 10,

"max\_iter":300,

"random\_state":42,

}sse =[]forkinrange(1,11):

kmeans=KMeans(n\_clusters=k,\*\*kmeans\_kwargs) kmeans.fit(scaled\_features)

sse.append(kmeans.inertia\_) plt.style.use("fivethirtyeight")plt.plot(range(1, 11), sse) plt.xticks(range(1, 11)) plt.xlabel("Number of Clusters") plt.ylabel("SSE") plt.show() silhouette\_coefficients = []

forkinrange(2,11):

kmeans=KMeans(n\_clusters=k,\*\*kmeans\_kwargs) kmeans.fit(scaled\_features)

score = silhouette\_score(scaled\_features, kmeans.labels\_) silhouette\_coefficients.append(score) plt.style.use("fivethirtyeight") plt.plot(range(2,11),silhouette\_coefficients)plt.xticks(range(2,11)) plt.xlabel("NumberofClusters")plt.ylabel("SilhouetteCoefficient") plt.show()

fromsklearn.clusterimportDBSCANfromsklearn.datasets import make\_moons from sklearn.metrics import adjusted\_rand\_score features, true\_labels = make\_moons( n\_samples=250, noise=0.05, random\_state=42

)

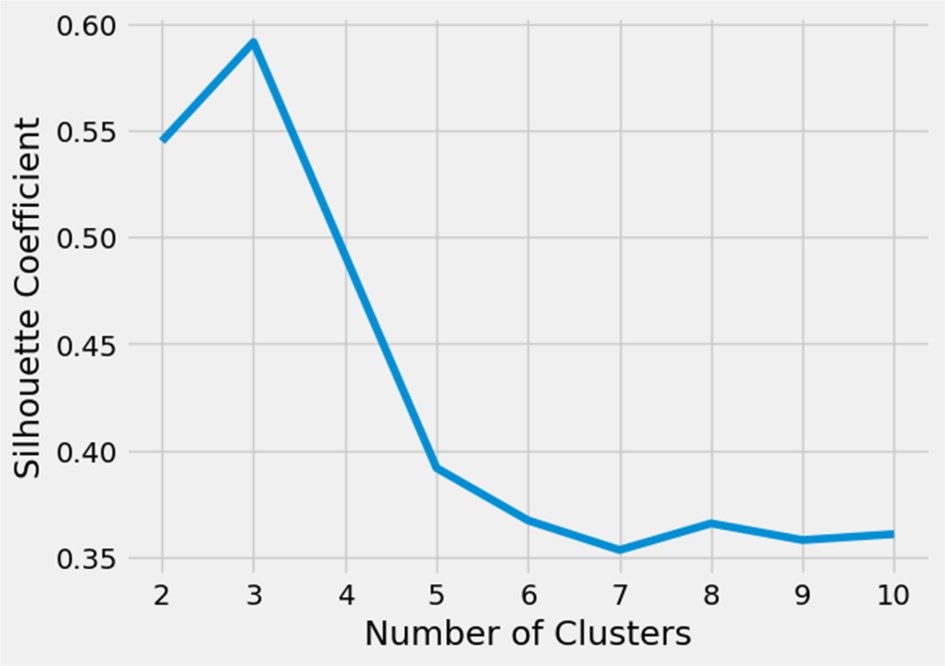
scaled\_features=scaler.fit\_transform(features)kmeans= KMeans(n\_clusters=2) dbscan = DBSCAN(eps=0.3) kmeans.fit(scaled\_features) dbscan.fit(scaled\_features) kmeans\_silhouette = silhouette\_score(scaled\_features, kmeans.labels\_

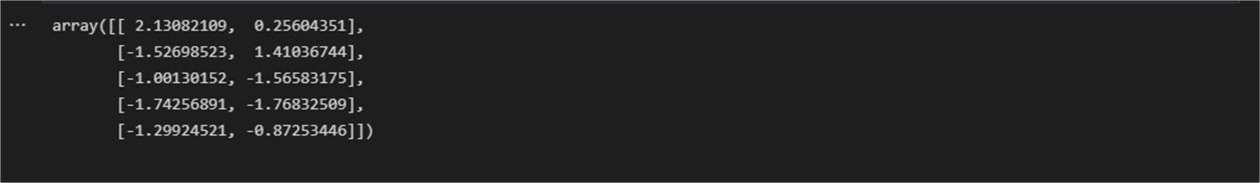
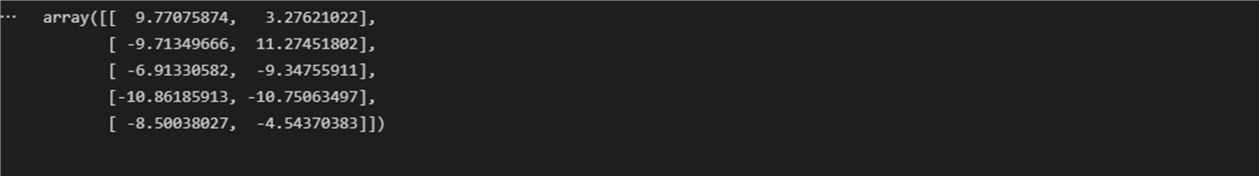
).round(2)

dbscan\_silhouette=silhouette\_score(scaled\_features,dbscan.labels\_

).round(2)kmeans\_silhouette

Output:





Program:

import numpy as np classNeuralNetwork():def init(self,learning\_rate): self.weights=np.array([np.random.randn(),np.random.randn()])self.bias= np.random.randn()

self.learning\_rate=learning\_rate

def\_sigmoid(self,x):

return1/(1+np.exp(-x))

def\_sigmoid\_deriv(self,x):

returnself.\_sigmoid(x)\*(1-self.\_sigmoid(x))

defpredict(self,input\_vector):

layer\_1 = np.dot(input\_vector, self.weights) + self.bias layer\_2= self.\_sigmoid(layer\_1) prediction = layer\_2 return prediction

def\_compute\_gradients(self,input\_vector,target):

layer\_1 = np.dot(input\_vector, self.weights) + self.bias layer\_2= self.\_sigmoid(layer\_1) prediction = layer\_2

derror\_dprediction = 2 \* (prediction - target) dprediction\_dlayer1= self.\_sigmoid\_deriv(layer\_1) dlayer1\_dbias = 1

dlayer1\_dweights = (0 \* self.weights) + (1 \*input\_vector) derror\_dbias=( derror\_dprediction \* dprediction\_dlayer1 \*dlayer1\_dbias

)

derror\_dweights=(

derror\_dprediction\*dprediction\_dlayer1\*dlayer1\_dweights

)

returnderror\_dbias,derror\_dweights

def\_update\_parameters(self,derror\_dbias,derror\_dweights):

self.bias = self.bias - (derror\_dbias \*self.learning\_rate) self.weights = self.weights - ( derror\_dweights \* self.learning\_rate) deftrain(self,input\_vectors, targets, iterations):

cumulative\_errors = [] forcurrent\_iterationin range(iterations):

random\_data\_index=np.random.randint(len(input\_vectors)) input\_vector = input\_vectors[random\_data\_index] target = targets[random\_data\_index]

derror\_dbias, derror\_dweights =self.\_compute\_gradients( input\_vector, target ) self.\_update\_parameters(derror\_dbias,derror\_dweights) ifcurrent\_iteration%100==0:

cumulative\_error = 0 fordata\_instance\_indexin range(len(input\_vectors)): data\_point

=input\_vectors[data\_instance\_index] target = targets[data\_instance\_index] prediction=self.predict(data\_point) error = np.square(prediction - target) cumulative\_error = cumulative\_error + error

cumulative\_errors.append(cumulative\_error) returncumulative\_errors

learning\_rate=0.1

neural\_network=NeuralNetwork(learning\_rate)

importmatplotlib.pyplotasplt input\_vectors = np.array([

[3, 1.5],

[2, 1],

[4, 1.5],

[3, 4],

[3.5,0.5],

[2, 0.5],

[5.5,1],

[1, 1],

]

)

targets=np.array([0,1,0,1,0,1,1,0])

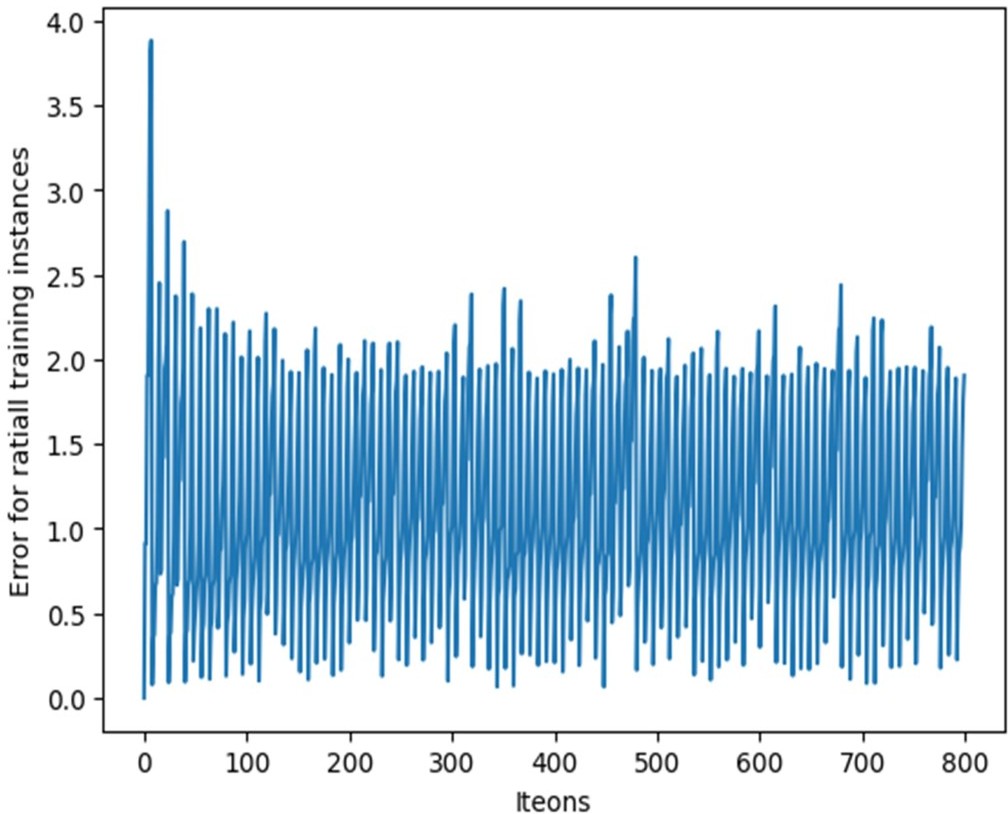
learning\_rate=0.1

neural\_network=NeuralNetwork(learning\_rate)

training\_error=neural\_network.train(input\_vectors,targets,10000) plt.plot(training\_error) plt.xlabel("Iteons")

plt.ylabel("Errorforratialltraininginstances")plt.savefig("cumulative\_error.png")

Output:



Program:

importmath

importmatplotlib.pyplotaspltfrommatplotlib.patchesimportWedgedef draw\_neural\_net(ax, left, right, bottom, top, layer\_sizes):

n\_layers=len(layer\_sizes)

v\_spacing = (top - bottom)/float(max(layer\_sizes)) h\_spacing= (right - left)/float(len(layer\_sizes) - 1) for n, layer\_size in enumerate(layer\_sizes):

layer\_top = v\_spacing\*(layer\_size - 1)/2. + (top +bottom)/2. formin range(layer\_size):

circle=plt.Circle((n\*h\_spacing+left,layer\_top-m\*v\_spacing),v\_spacing/4.,color='w',ec='k',zorder=4) ax.add\_artist(circle)

for n, (layer\_size\_a, layer\_size\_b) in enumerate(zip(layer\_sizes[:-1], layer\_sizes[1:])): layer\_top\_a= v\_spacing\*(layer\_size\_a - 1)/2. + (top +bottom)/2. layer\_top\_b = v\_spacing\*(layer\_size\_b - 1)/2. + (top +bottom)/2.

for m in range(layer\_size\_a): for o in range(layer\_size\_b): line = plt.Line2D([n\*h\_spacing+left,(n+1)\*h\_spacing+ left],[layer\_top\_a - m\*v\_spacing,layer\_top\_b - o\*v\_spacing], c='k')

ax.add\_artist(line)

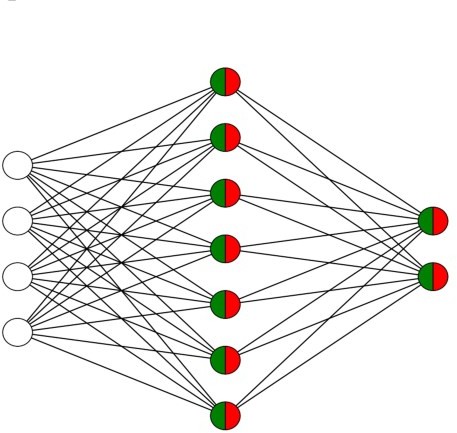
for n, (layer\_size) in enumerate(layer\_sizes): layer\_top=v\_spacing\* (layer\_size - 1) / 2. + (top +bottom) / 2. for m in range(layer\_size):

center = (n \* h\_spacing + left, layer\_top - m \*v\_spacing) radius= v\_spacing / 4. if n > 0:

wedge\_left =Wedge(center, r=radius, theta1=90,theta2=270, color='w',fc='g', ec='k', zorder=4) wedge\_right=Wedge(center,r=radius,theta1=270,theta2=90,color='w',fc='r',ec='k',zorder=4) ax.add\_artist(wedge\_left)

ax.add\_artist(wedge\_right) else:

circle = plt.Circle(center, radius, color='w',ec='k', zorder=4) ax.add\_artist(circle)fig=plt.figure(figsize=(12,12))ax=fig.gca()ax.axis('off') draw\_neural\_net(ax, .1, .9, .1, .9, [4, 7, 2]) fig.savefig('nn.png')

Output:

Program:

importosimportsysimport cv2

importnumpyasnp

input\_f='letter.data'img\_resize\_factor

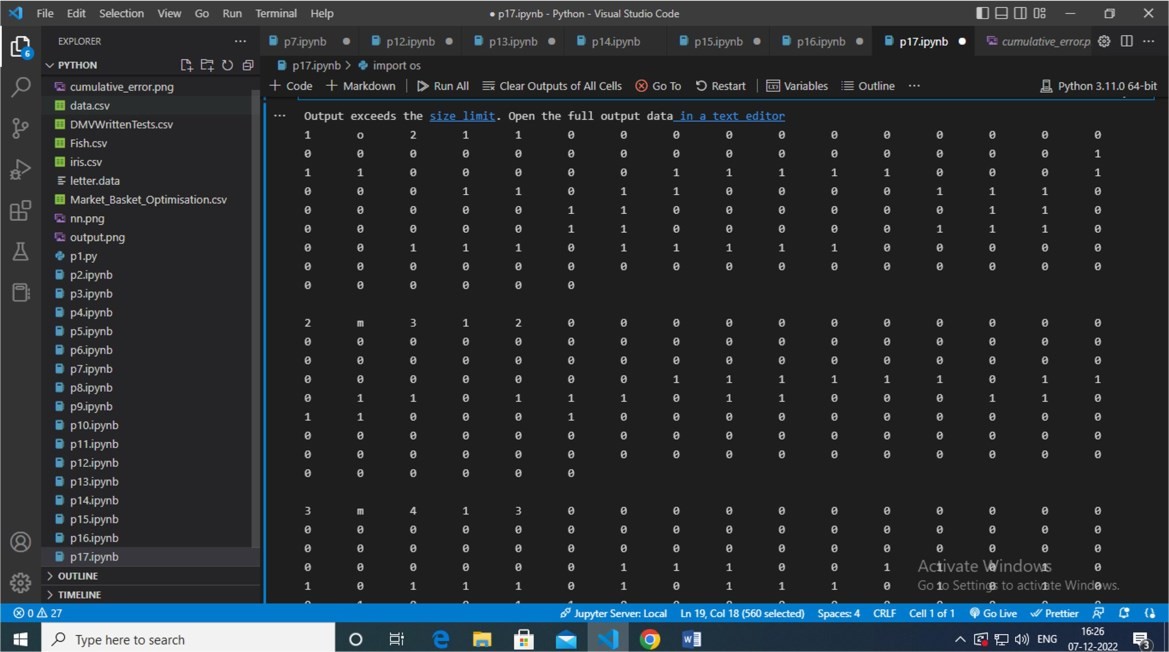
= 12 start, end = 6, -1 height, width = 16, 8 with open(input\_f, 'r') as f: for line in f.readlines():

data = np.array([255\*float(x) for x in line.split('\t')[start:end]]) img= np.reshape(data, (height, width))

img\_scaled = cv2.resize(img, None, fx=img\_resize\_factor, fy=img\_resize\_factor) print(line) cv2.imshow('Img', img\_scaled)

c = cv2.waitKey() ifc== 27: break

Output:



Program:

importcv2importnumpyasnp import os

fromrandomimportshuffle from tqdm import tqdm

#Download the images of dogs and cats from the following URL(https://[www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data)](http://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data))andextractit

TRAIN\_DIR='E:/Python/train' TEST\_DIR = 'E:/Python/test' IMG\_SIZE = 50

LR= 1e-3

MODEL\_NAME='dogvscats-{}-{}.model'.format(LR,'6conv-basic-video') def label\_img(img):

word\_label=img.split('.')[-3] ifword\_label

=='cat':return[1,0]

elifword\_label=='dog':return[0,1]

def create\_train\_data(): training\_data = [] forimgin tqdm(os.listdir(TRAIN\_DIR)): label = label\_img(img)

path=os.path.join(TRAIN\_DIR,img)

img=cv2.resize(cv2.imread(path,cv2.IMREAD\_GRAYSCALE),(IMG\_SIZE,IMG\_SIZE)) training\_data.append([np.array(img), np.array(label)]) shuffle(training\_data)

np.save('train\_data.npy', training\_data) return training\_data

def process\_test\_data(): testing\_data = [] forimgin

tqdm(os.listdir(TEST\_DIR)): path = os.path.join(TEST\_DIR,img) img\_num= img.split('.')[0]

img=cv2.resize(cv2.imread(path,cv2.IMREAD\_GRAYSCALE),(IMG\_SIZE,IMG\_SIZE)) testing\_data.append([np.array(img), img\_num])

np.save('test\_data.npy',testing\_data) returntesting\_data import tflearn

fromtflearn.layers.convimportconv\_2d,max\_pool\_2dfromtflearn.layers.core

importinput\_data,dropout,fully\_connectedfromtflearn.layers.estimatorimport regression

convnet=input\_data(shape=[None,IMG\_SIZE,IMG\_SIZE,1],name='input')

convnet=conv\_2d(convnet,32,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=conv\_2d(convnet,64,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=conv\_2d(convnet,32,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=conv\_2d(convnet,64,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=conv\_2d(convnet,32,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=conv\_2d(convnet,64,2,activation='relu') convnet = max\_pool\_2d(convnet, 2)

convnet=fully\_connected(convnet,1024,activation='relu')convnet= dropout(convnet, 0.8)

convnet=fully\_connected(convnet,2,activation='softmax')convnet= regression(convnet, optimizer='adam', learning\_rate=LR, loss='categorical\_crossentropy', name='targets')

model=tflearn.DNN(convnet,tensorboard\_dir='log') if os.path.exists('{}.meta'.format(MODEL\_NAME)):

model.load(MODEL\_NAME) print('modelloaded!')

train = train\_data[:-500] test = train\_data[-500:]X=np.array([i[0]fori in train]).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

Y=[i[1]fori intrain]

test\_x=np.array([i[0]foriintest]).reshape(-1,IMG\_SIZE,IMG\_SIZE,1)test\_y=[i[1]foriin test]

model.fit({'input':X},{'targets':Y},n\_epoch=5,validation\_set=({'input':test\_x},{'targets': test\_y}),

snapshot\_step=500,show\_metric=True,run\_id=MODEL\_NAME)

model.save(MODEL\_NAME)

#tensorboard--logdir=/home/ngacha/MachineLearning/log

importmatplotlib.pyplotasplt#test\_data= process\_test\_data()

test\_data=np.load('test\_data.npy') fig = plt.figure()

fornum,datainenumerate(test\_data[:12]): img\_num = data[1]

img\_data=data[0]

y = fig.add\_subplot(3,4,num+1) orig = img\_data data=img\_data.reshape(IMG\_SIZE,IMG\_SIZE,1)

model\_out=model.predict([data])[0]

if np.argmax(model\_out) == 1: str\_label='Dog' else:str\_label= 'Cat'

y.imshow(orig, cmap='gray') plt.title(str\_label) y.axes.get\_xaxis().set\_visible(False)

y.axes.get\_yaxis().set\_visible(False)plt.show()

with open('submission-file.csv', 'w') as f: f.write('id,label\n')withopen('submission-file.csv','w') as f:

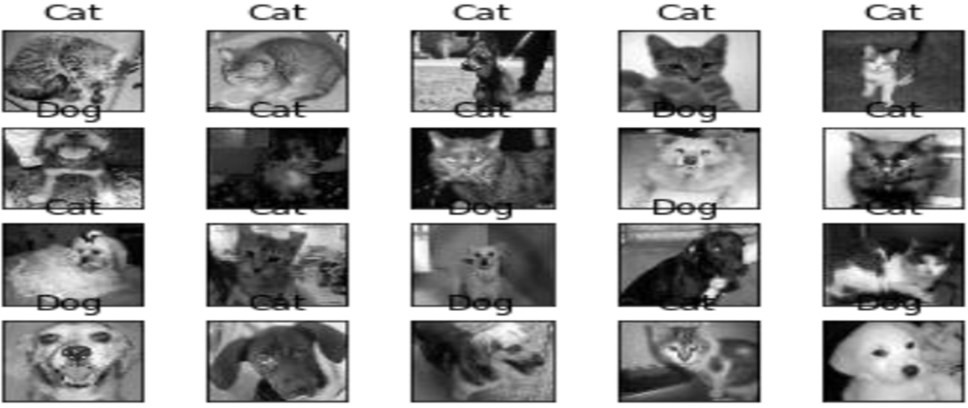
fordataintqdm(test\_data):

img\_num = data[1] img\_data=data[0] orig = img\_data

data = img\_data.reshape(IMG\_SIZE, IMG\_SIZE, 1) model\_out= model.predict([data])[0]

f.write('{},{}\n'.format(img\_num,model\_out[1]))

Output:



Program:

%load\_extautoreload%autoreload2

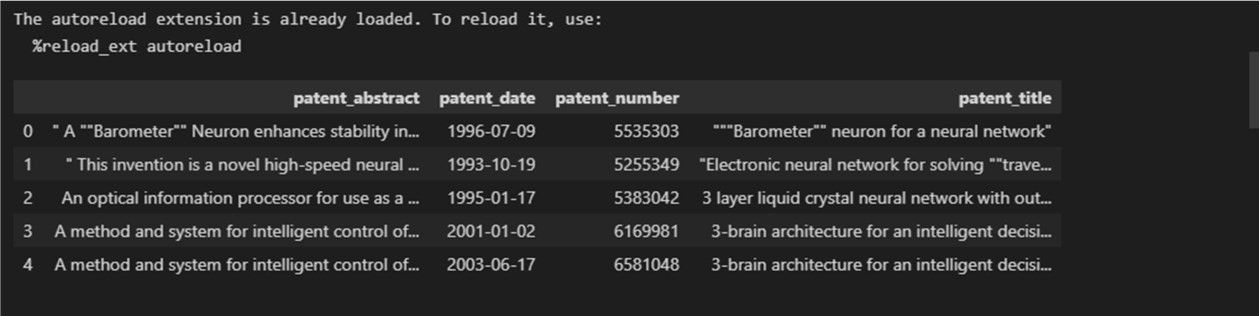
fromIPython.core.interactiveshellimportInteractiveShell from IPython.display import HTML InteractiveShell.ast\_node\_interactivity = 'all'

importwarnings

warnings.filterwarnings('ignore', category = RuntimeWarning) warnings.filterwarnings('ignore',category=UserWarning)importpandasaspd import numpy as np from utils import \*

data=pd.read\_csv('neural\_network\_patent\_query.csv')data.head()

Output:



Program:

fromkeras.modelsimportSequential,load\_model

from keras.layers import LSTM, Dense, Dropout, Embedding,Masking, Bidirectional from tensorflow.keras.optimizersimportAdamfromtensorflow.keras.utilsimportplot\_modelmodel= Sequential() model.add(Embedding(

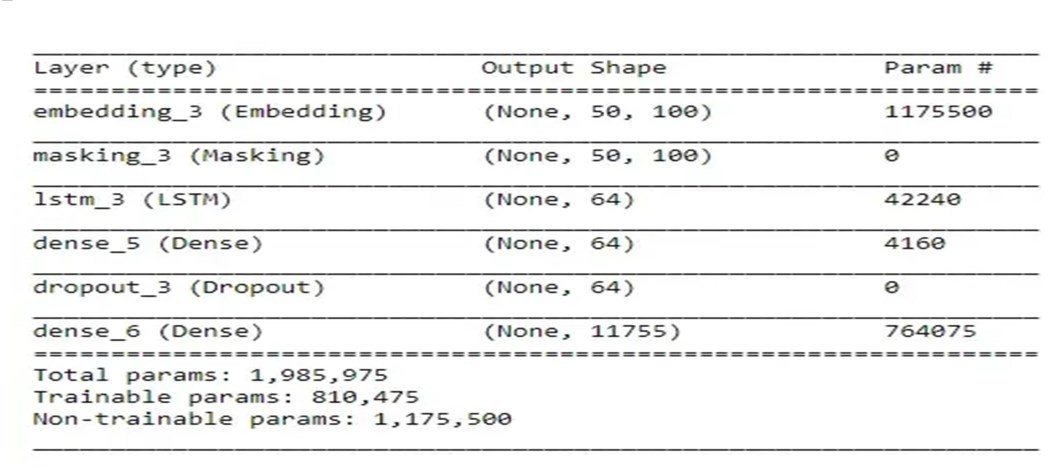
input\_dim=len(word\_idx) + 1, output\_dim=100,weights=None, trainable=True))

model.add(

LSTM(64,return\_sequences=False,dropout=0.1,recurrent\_dropout=0.1)

)

model.add(Dense(64, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(len(word\_idx) + 1, activation='softmax')) model.compile( optimizer='adam', loss='categorical\_crossentropy',metrics=['accuracy']) model.summary()

Output:

Program:

importnumpyasnpimportmatplotlib.pyplotaspltimportkerasfromkeras.layersimport Input, Dense, Reshape, Flatten, Dropout from keras.layers import BatchNormalization, Activation,ZeroPadding2D from keras.layers import LeakyReLU from keras.layers.convolutional import UpSampling2D, Conv2D from keras.models import Sequential, Model

fromtensorflow.keras.optimizersimportAdam,SGD

(X, y), (\_, \_) = keras.datasets.cifar10.load\_data() X = X[y.flatten()==8]image\_shape=(32,32,3)latent\_dimensions

=100defbuild\_generator():model=Sequential()

model.add(Dense(128\*8\*8,activation="relu",input\_dim=latent\_dimensions))

model.add(Reshape((8,8,128)))model.add(UpSampling2D())

model.add(Conv2D(128,kernel\_size=3,padding="same"))model.add(BatchNormalization(momentum=0.78)) model.add(Activation("relu"))model.add(UpSampling2D())

model.add(Conv2D(64,kernel\_size=3,padding="same"))model.add(BatchNormalization(momentum=0.78)) model.add(Activation("relu"))

model.add(Conv2D(3,kernel\_size=3,padding="same")) model.add(Activation("tanh"))noise = Input(shape=(latent\_dimensions,))image=model(noise)

return Model(noise, image) def build\_discriminator():model=Sequential()

model.add(Conv2D(32,kernel\_size=3,strides=2,input\_shape=image\_shape,padding="same "))

model.add(LeakyReLU(alpha=0.2))model.add(Dropout(0.25)) model.add(Conv2D(64,kernel\_size=3,strides=2,padding="same"))

model.add(ZeroPadding2D(padding=((0,1),(0,1))))model.add(BatchNormalization(momentum=0.82)) model.add(LeakyReLU(alpha=0.25))model.add(Dropout(0.25))

model.add(Conv2D(128,kernel\_size=3,strides=2,padding="same")) model.add(BatchNormalization(momentum=0.82)) model.add(LeakyReLU(alpha=0.2))model.add(Dropout(0.25)) model.add(Conv2D(256, kernel\_size=3, strides=1,padding="same")) model.add(BatchNormalization(momentum=0.8)) model.add(LeakyReLU(alpha=0.25))model.add(Dropout(0.25)) model.add(Flatten())

model.add(Dense(1,activation='sigmoid'))image= Input(shape=image\_shape)

validity=model(image)

returnModel(image,validity)def display\_images():r, c = 4,4

noise=np.random.normal(0,1,(r\*c,latent\_dimensions))generated\_images= generator.predict(noise)generated\_images = 0.5 \* generated\_images + 0.5

fig,axs=plt.subplots(r,c)

count = 0for i in range(r): forjinrange(c): axs[i,j].imshow(generated\_images[count,:,:,]) axs[i,j].axis('off')

count+=1

plt.show() plt.close()

discriminator = build\_discriminator() discriminator.compile(loss='binary\_crossentropy',optimizer=Adam(0.0002,0.5),metrics=['acc uracy']) discriminator.trainable = False generator = build\_generator() z = Input(shape=(latent\_dimensions,)) image = generator(z) valid = discriminator(image) combined\_network = Model(z, valid) combined\_network.compile(loss='binary\_crossentropy',optimizer=Adam(0.0002,0.5)) num\_epochs=15000 batch\_size=32 display\_interval=2500

losses=[]X=(X/127.5) -1.

valid=np.ones((batch\_size,1))valid+=0.05\* np.random.random(valid.shape)

fake=np.zeros((batch\_size,1))fake+=0.05\* np.random.random(fake.shape) for epoch in range(num\_epochs):

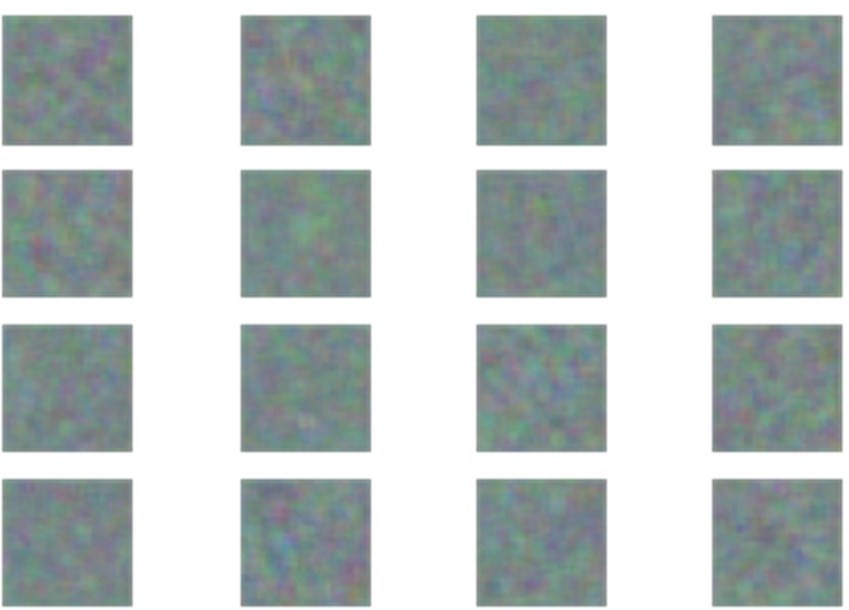
index=np.random.randint(0,X.shape[0],batch\_size)images=X[index]

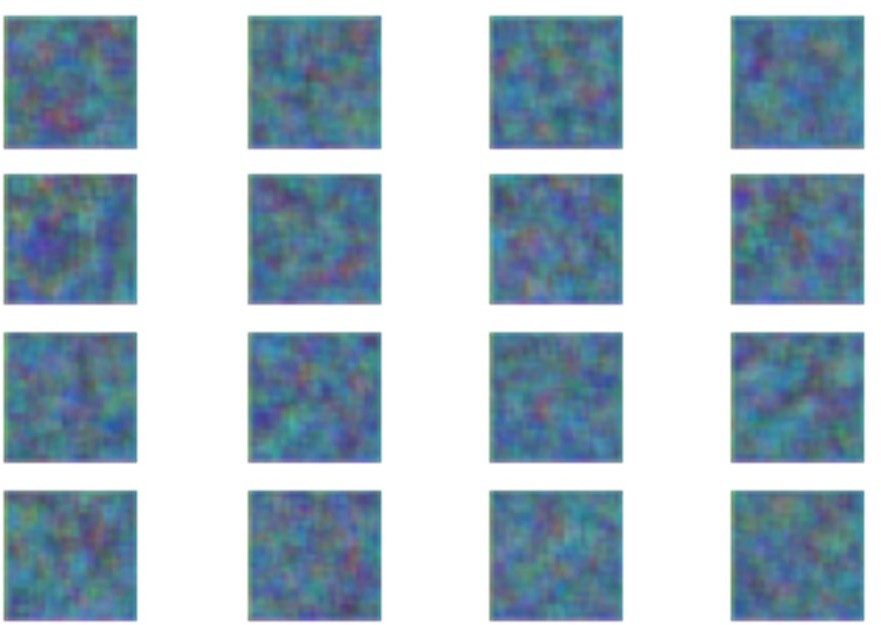
noise = np.random.normal(0, 1, (batch\_size,latent\_dimensions))generated\_images = generator.predict(noise)discm\_loss\_real =discriminator.train\_on\_batch(images, valid) discm\_loss\_fake=discriminator.train\_on\_batch(generated\_images,fake)discm\_loss=0.5

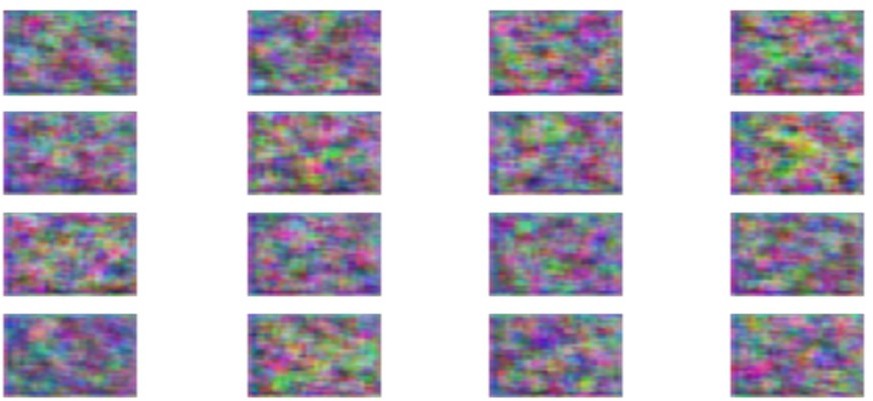
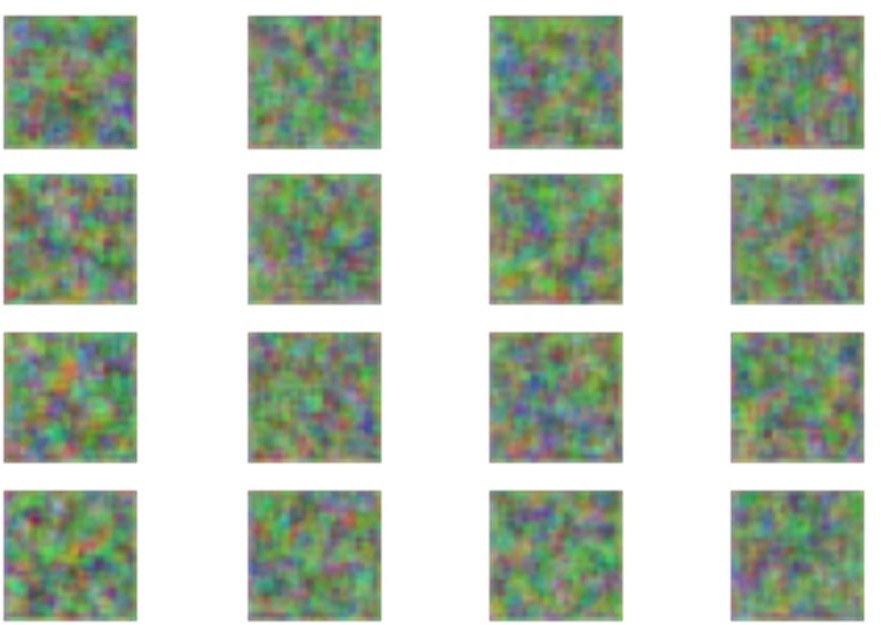
\*np.add(discm\_loss\_real,discm\_loss\_fake)genr\_loss

=combined\_network.train\_on\_batch(noise,valid)ifepoch%display\_interval==6: display\_images()

Output:







Program:

importm3u8importrequests

from bs4 import BeautifulSoup from tqdm import tqdm\_notebook as tqdm import subprocess import requestschunk\_size=256sess=requests.session()

ganna\_url="https://gaana.com/album/ek-love-ya-original-motionpicture-soundtrack"bob\_ganna\_url= "https://gaana.com/d0d959d9-fac9-4f38-ad6fa80fc09a468f" log\_sing = "?sign=1646556464- Zi7mUDtXYR-0-32a0f29f4bf3977b05ade364f1cbe442" m3File1

="https://vodhlsgaana.akamaized.net/hls/72/4992872/39791107/96/hdntl=exp=1646643931~a cl=%2f\*~data=hdntl~hmac=735cd0c11a600b98f7fd283dce25dfaab0f5c006e5fea07ef50cdcfd d525960b/index.m3u8"m3File=m3File1+log\_singr=sess.get(m3File,stream=True)m3u8\_master= m3u8.loads(r.text)

m3u8\_master

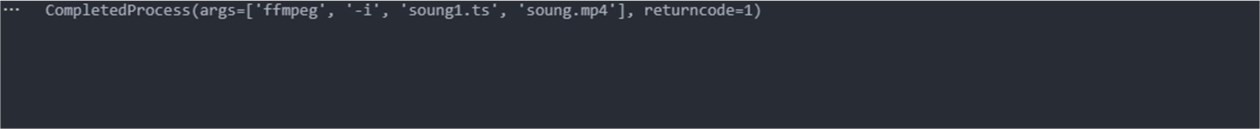
playlist\_uri=m3u8\_master.data['segments']

main\_url="https://streamcdn.gaana.com/songs/67/5206667/40781248/40781248\_96.mp4/" # playlist\_url = main\_url + playlist\_uri # r =

requests.get(playlist\_url)withopen('soung1.ts','wb') as f:

forsegmentinplaylist\_uri:

uri = segment['uri'] url = main\_url + uri r=requests.get(url) f.write(r.content)

subprocess.run(['ffmpeg', '-i','soung1.ts','soung.mp4'],shell=True) Output:

