ML PRACS

1. LINEAR REGRESSION

*#import libraries*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

*#Read dataset*

dataset **=** pd**.**read\_csv("linear\_regression\_dataset.csv")

print(dataset**.**shape)

dataset**.**head()

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

y **=** dataset**.**iloc[:,1]**.**values

*#import linear regression and create object of it*

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

*#create object of LinearRegression*

regressor **=** LinearRegression()

*#train the data*

regressor**.**fit(x,y)

*#print value of coefficient*

print("Coefficient : ",regressor**.**coef\_)

*#print value of intercept*

print("intercept : ",regressor**.**intercept\_)

*#calculate accuracy*

accuracy **=** regressor**.**score(x,y)**\***100

print("Accuracy : ",accuracy)

y\_pred **=** regressor**.**predict([[8]])

print(y\_pred)

*#take input from user and predict the value*

hours **=** int(input("Enter the no of hours : "))

predicted\_value **=** regressor**.**predict([[hours]])

print(predicted\_value)

*#plotting points*

plt**.**plot(x,y,'o', label**=**"data point")

*#plot the line*

plt**.**plot(x,regressor**.**predict(x), color**=**'#ff0000', label**=**'regression line')

*# x-axis label*

plt**.**xlabel('Driving Hours')

*#y-axis label*

plt**.**ylabel('Risk Score')

plt**.**legend()

plt**.**show()

1. **DECISION TREE**

**import** numpy **as** np

**import** pandas **as** pd

*#reading dataset*

dataset **=** pd**.**read\_csv("decision\_Tree\_dataset.csv")

*#perform label encoding*

**from** sklearn.preprocessing **import** LabelEncoder

labelencoder\_x **=** LabelEncoder()

dataset **=** dataset**.**apply(LabelEncoder()**.**fit\_transform)

print(dataset)

x **=** dataset**.**iloc[ : , : **-**1]

y **=** dataset['Buys']

x **=** np**.**asarray(x)

*# applying decision tree classifier*

**from** sklearn.tree **import** DecisionTreeClassifier **as** DTC

c **=** DTC()

c **=** c**.**fit(x, y)

*#predicting*

c**.**predict([[2, 0, 1, 0],[1,0,0,0]])

1. **K MEANS**

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

*#read dataset*

dataset **=** pd**.**read\_csv("Kmeans\_dataset.csv")

print(dataset)

f1 **=** dataset['X']**.**values

f2 **=** dataset['Y']**.**values

x **=** np**.**array(list(zip(f1, f2)))

print(x)

*# initial centroid points*

centers **=** np**.**array([[0.1,0.6],[0.3,0.2]])

print(centers)

*# Apply K-Means Clustering*

**from** sklearn.cluster **import** KMeans

model **=** KMeans(n\_clusters**=**2, init**=**centers, n\_init**=**1)

*# n\_clusters = number of clusters*

*# init = initial centroids*

*# n\_init = number of init parameter*

*#train the algorithm*

model**.**fit(x)

*# print labels*

print(model**.**labels\_)

*# population around cluster 2*

print(np**.**count\_nonzero(model**.**labels\_ **==** 1))

*# new centroids*

print(model**.**cluster\_centers\_)

*# Cluster with initial centroids*

C\_x **=** np**.**array([0.1, 0.3])

C\_y **=** np**.**array([0.6, 0.2])

plt**.**scatter(f1, f2, c**=**'#050505', s**=**7)

plt**.**scatter(C\_x, C\_y, marker**=**'\*', s**=**200, c**=**'g')

*# cluster with new centroids*

N\_x **=** np**.**array([model**.**cluster\_centers\_[0][0], model**.**cluster\_centers\_[1][0]])

N\_y **=** np**.**array([model**.**cluster\_centers\_[0][1],model**.**cluster\_centers\_[1][1]])

plt**.**scatter(f1, f2, c**=**'#050505', s**=**7)

plt**.**scatter(N\_x, N\_y, marker**=**'\*', s**=**200, c**=**'g')

1. **KNN**

*# import packages*

**import** numpy **as** np

**import** pandas **as** pd

*# create dataset*

x **=** [[2, 4], [4, 4], [4, 6], [4, 2], [6, 2], [6, 4], [8, 2]]

y **=** ["Orange", "Blue", "Orange", "Orange", "Blue", "Orange", "Blue"]

In [2]:

**from** sklearn.neighbors **import** KNeighborsClassifier

classifier **=** KNeighborsClassifier(n\_neighbors**=**3) *#n\_neighbors indicates 'K'*

*# train the algorithm*

classifier**.**fit(x,y)

*# predict class for points (6,6)*

x\_test **=** np**.**array([6,6])

y\_pred **=** classifier**.**predict([x\_test])

print(y\_pred)

OUTPUT:

['Orange']