**Name-Choudhury Abhisek Panda**

**Roll no- 18CSE059 , Sec-B**

**PYTHON PROGRAMS :-**

**REGRESSION-**

**import numpy as np**

**import matplotlib.pyplot as plt**

**def estimate\_coef(x, y):**

**# number of observations/points**

**n = np.size(x)**

**# mean of x and y vector**

**m\_x, m\_y = np.mean(x), np.mean(y)**

**# calculating cross-deviation and deviation about x**

**SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x**

**SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x**

**# calculating regression coefficients**

**b\_1 = SS\_xy / SS\_xx**

**b\_0 = m\_y - b\_1\*m\_x**

**return(b\_0, b\_1)**

**def plot\_regression\_line(x, y, b):**

**# plotting the actual points as scatter plot**

**plt.scatter(x, y, color = "m",**

**marker = "o", s = 30)**

**# predicted response vector**

**y\_pred = b[0] + b[1]\*x**

**# plotting the regression line**

**plt.plot(x, y\_pred, color = "g")**

**# putting labels**

**plt.xlabel('x')**

**plt.ylabel('y')**

**# function to show plot**

**plt.show()**

**def main():**

**# observations**

**x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])**

**y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])**

**# estimating coefficients**

**b = estimate\_coef(x, y)**

**print("Estimated coefficients:\nb\_0 = {} , \nb\_1 = {}".format(b[0], b[1]))**

**# plotting regression line**

**plot\_regression\_line(x, y, b)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**----------------------------------------------------------------------------**

**KNN Algorithm-**

**# Importing libraries**

**import pandas as pd**

**import numpy as np**

**import math**

**import operator**

**#### Start of STEP 1**

**# Importing data**

**data = pd.read\_csv('C:/Users/tripa/Desktop/ML/iris.csv')**

**#### End of STEP 1**

**print(data.head(5))**

**# Defining a function which calculates euclidean distance between two data points**

**def euclideanDistance(data1, data2, length):**

**distance = 0**

**for x in range(length):**

**distance += np.square(data1[x] - data2[x])**

**return np.sqrt(distance)**

**# Defining our KNN model**

**def knn(trainingSet, testInstance, k):**

**distances = {}**

**sort = {}**

**length = testInstance.shape[1]**

**#### Start of STEP 3**

**# Calculating euclidean distance between each row of training data and test data**

**for x in range(len(trainingSet)):**

**#### Start of STEP 3.1**

**dist = euclideanDistance(testInstance, trainingSet.iloc[x], length)**

**distances[x] = dist[0]**

**#### End of STEP 3.1**

**#### Start of STEP 3.2**

**# Sorting them on the basis of distance**

**sorted\_d = sorted(distances.items(), key=operator.itemgetter(1))**

**#### End of STEP 3.2**

**neighbors = []**

**#### Start of STEP 3.3**

**# Extracting top k neighbors**

**for x in range(k):**

**neighbors.append(sorted\_d[x][0])**

**#### End of STEP 3.3**

**classVotes = {}**

**#### Start of STEP 3.4**

**# Calculating the most freq class in the neighbors**

**for x in range(len(neighbors)):**

**response = trainingSet.iloc[neighbors[x]][-1]**

**if response in classVotes:**

**classVotes[response] += 1**

**else:**

**classVotes[response] = 1**

**#### End of STEP 3.4**

**#### Start of STEP 3.5**

**sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)**

**return(sortedVotes[0][0], neighbors)**

**#### End of STEP 3.5**

**testSet = [[7.2, 3.6, 5.1, 2.5]]**

**test = pd.DataFrame(testSet)**

**#### Start of STEP 2**

**# Setting number of neighbors = 1**

**print('\n\nWith 1 Nearest Neighbour \n\n')**

**k = 1**

**#### End of STEP 2**

**# Running KNN model**

**result,neigh = knn(data, test, k)**

**# Predicted class**

**print('\nPredicted Class of the datapoint = ', result)**

**# Nearest neighbor**

**print('\nNearest Neighbour of the datapoints = ',neigh)**

**print('\n\nWith 3 Nearest Neighbours\n\n')**

**# Setting number of neighbors = 3**

**k = 3**

**# Running KNN model**

**result,neigh = knn(data, test, k)**

**# Predicted class**

**print('\nPredicted class of the datapoint = ',result)**

**# Nearest neighbor**

**print('\nNearest Neighbours of the datapoints = ',neigh)**

**print('\n\nWith 5 Nearest Neighbours\n\n')**

**# Setting number of neighbors = 3**

**k = 5**

**# Running KNN model**

**result,neigh = knn(data, test, k)**

**# Predicted class**

**print('\nPredicted class of the datapoint = ',result)**

**# Nearest neighbor**

**print('\nNearest Neighbours of the datapoints = ',neigh)**

**R-Introduction:**

***Definition:***

* **R is an interpreted programming language**used to analyze **statistical information, graphical representation, reporting,** and **data modeling.**
* R is the implementation of the**S programming** language, which is combined with **lexical scoping semantics.**
* Its most common use is to analyze and visualize data.R generally comes with the Command-line interface.

***Evolution of R:***

* R programming language was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand.
* The R Development Core Team currently develops R.

***Why R programming Language:***

* R programming is an open-source free language which is currently one of the most requested programming language in the Data Science job market.
* R is a a platform-independent language and it is used as a leading tool for machine learning, statistics, and data analysis.
* R programming language allows us to integrate with other languages (C, C++) and it has a vast community of users and it’s growing day by day.

***Advantages of R:***

* R programming is platform independent which runs on any operating systems.
* In R, everyone is welcome to provide new packages, bug fixes, and code enhancements.

***Disadvantages of R:***

* In the R programming language, the standard of some packages is less than perfect.
* Although, R commands give little pressure to memory management. SoR programming language may consume all available memory.

***Applications of R:***

* We use R for Data Science.
* R is used by many quantitative analysts as its programming tool.
* Tech giants like Google, Facebook, bing, Accenture, Wipro and many more using R nowadays.

**R installation:**

**R programming** is a very popular language and to work on that we have to install two things, i.e., R and RStudio. R and RStudio works together to create a project on R.

***Installation of R:***

1. First, we have to download the R setup from <https://cloud.r-project.org/bin/windows/base/>.
2. When we click on **Download R for windows**, our downloading will be started of R setup. Once the downloading is finished, we have to run the setup of R in the following way:

* Select the path where we want to download the R and proceed to Next.
* Select all components which we want to install, and then we will proceed to **Next.**
* In the next step, we haveto select either customized start-up or accept the default, and then we proceed to **Next**.
* When we proceed to next, our installation of R in our system will get started.
* In the last, we will click on finish to successfully install R in our system.

***Installation of RStudio:***

1. First, we have to visit the RStudio official site.

([https://rstudio.com/products/rstudio/download/](https://rstudioproject.com/products/rstudio/download/))

1. Select the RStudio desktop for open-source license and click on download.
2. Select the appropriate installer and download it.Once the downloading is finished, we have to run the setup of R in the following way:

* Click on Next on welcome page.
* Click on Install.
* Click on Finish.

1. Now, RStudio is ready to work.

**some basic commands and output:**

***Mathematical Functions:***

|  |  |  |  |
| --- | --- | --- | --- |
| **FUNCTION** |  | **INPUT** | **OUTPUT** |
| abs(x) | abs(-10) | 10 |
| log(x, base=y) | log(100, base=10) | 2 |
| exp(x) | exp(5) | 148.4132 |
| sqrt(x) | sqrt(25) | 5 |
| factorial(x) | factorial(3) | 6 |
| pi | pi | 3.141593 |

***Logical Functions:***

|  |  |  |  |
| --- | --- | --- | --- |
| **FUNCTION** |  | **INPUT** | **OUTPUT** |
| Greater than | 5>6 | FALSE |
| Less than | 4<5 | TRUE |
| Less than and Equal to | 12<=10 | FALSE |
| Greater than and Equal to | 19>=15 | TRUE |
| Equal to | 7==8 | FALSE |
| Not equal to | 13!=14 | TRUE |
| AND | 3 & 4 | TRUE |
| OR | 3 | 4 | TRUE |
| NOT | !3 | FALSE |

***Other Functions:***

|  |  |  |  |
| --- | --- | --- | --- |
| **FUNCTION** |  | **INPUT** | **OUTPUT** |
| Colon (:) | 1:6 | 1 2 3 4 5 6 |
| %in% | 5 %in% 6  5 %in% 5 | FALSE  TRUE |