National University of Computer and Emerging Sciences



Lab Manual 06

CL461-Artificial Intelligence Lab

|  |  |
| --- | --- |
| Course Instructor | Dr. Mubasher Baig |
| Lab Instructor (s) | Saad Ali  Mahmood Hussain |
| Section | A |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Table of Contents

[**Objectives**](#_heading=h.30j0zll) **4**

[**Task Distribution**](#_heading=h.1fob9te) **4**

[3. Machine Learning Concepts:](#_heading=h.9eh5rzv2tpeu) **4**

[3.1 Classification:](#_heading=h.9eh5rzv2tpeu) 4

[3.2 Regression:](#_heading=h.9eh5rzv2tpeu) 4

[3.3 Underfitting](#_heading=h.9ilo38kad5w1) 5

[3.4 Overfitting:](#_heading=h.6t2cemqqr2k) 5

[3.5 Supervised Machine Learning:](#_heading=h.n5fylv3zg0lp) 5

[**4- K-Nearest Neighbors (KNN)**](#_heading=h.qpssovb5ayw8) **6**

[4.1 The KNN Algorithm](#_heading=h.4alqa81j1918) 6

[**5. Decision Tree:**](#_heading=h.2tef37ijb5qn) **6**

[5.1 Intuition Behind Splits:](#_heading=h.enxzdqyfkaog) 8

[**6. Model Evaluation:**](#_heading=h.xci1dok5pfvm) **9**

[6.1 Regression:](#_heading=h.pni01arina2z) 9

[6.2 Classification:](#_heading=h.f4cnio4d5sj6) 9

[6.2.1 Confusion Matrix:](#_heading=h.gye2qnux1u12) 10

[6.2.2 Classification Report](#_heading=h.cexut1q2hznm) 11

[**7. Implementation of Regression:**](#_heading=h.jod8x789zabl) **11**

[7.1 Data Acquisition:](#_heading=h.ad4dqcuc1nto) 11

[7.2 KNeighbor Regression:](#_heading=h.ionpjzch4x0o) 12

[7.2.1 Model Evaluation:](#_heading=h.u7cgrmyt3ra1) 13

[7.3 Decision Tree Regressor:](#_heading=h.g7vd0bytrhri) 13

[7.3.1 Model Evaluation:](#_heading=h.cswm4gk0wof5) 13

[**8. Implementation of the Classification:**](#_heading=h.1tz8zfsm2c6b) **14**

[8.1 Data Acquisition:](#_heading=h.eq1lmkfjxgb2) 14

[8.2 KNN Classifier](#_heading=h.d4ujk7zca53x) 14

[8.2.1 Model Evaluation.](#_heading=h.66oth6vlf9eh) 15

[8.3 Decision Tree Classifier:](#_heading=h.a6yypvm9kvkp) 15

[8.3.1 Model Evaluation:](#_heading=h.4ly4nraxyswm) 15

[**9. Misc**](#_heading=h.urao3dn3x2iw) **16**

[9.1 Correctly Identifying Neighbors for KNN:](#_heading=h.yyr68eh47jj7) 16

[9.2 Visualizing the Decision Trees.](#_heading=h.1xitkf7t4p4h) 17

[**10 Exercise: (25)**](#_heading=h.tfovdlb0b3vt) **18**

[10.1 Problem 1: (Predict the Survival) (15)](#_heading=h.4n68hpjslh3w) 18

[10.2 Problem 2: (Forecast USD-PKR currency exchange) (10)](#_heading=h.o0frl1s7bk9c) 18

[**11. Submission Instructions:**](#_heading=h.s1twrzd98wq) **19**

# Objectives

After performing this lab, students shall be able to understand classification and regression using state of the art machine learning model K-Nearest Neighbors and Decision Tree using SK-Learn.

# Task Distribution

|  |  |
| --- | --- |
| **Total Time** | **170 Minutes** |
| Concepts | 30 Minutes |
| Regression | 20 Minutes |
| Classification | 20 Minutes |
| Exercise | 90 Minutes |
| Online Submission | 10 Minutes |
|  |  |

# 3. Machine Learning Concepts:

Supervised Machine Learning is used whenever we want to predict a certain outcome from a given input, and we have examples of input/output pairs.

## 3.1 Classification:

In classification, the goal is to predict a class label, which is a choice from a predefined list of possibilities. Binary classification is distinguishing between exactly two classes, In Multiclass Classification, classification is between more than two classes. Classifying emails into Spam and not-Spam is an example of the Binary Classification while predicting the Plant Disease Class by looking at symptoms is an example of Multiclass classification.

## 3.2 Regression:

For regression tasks, the goal is to predict a continuous number, or a floating-point number in programming terms (or real number in mathematical terms). Predicting a Person’s annual income from their education, their age and where they live is an example of the Regression task. Another example of a regression task is predicting the yield of a corn farm given attributes such as previous yields, weather, and number of employees working on the farm.

## 3.3 Underfitting

A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. *(It’s just like trying to fit undersized pants!)* Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

## 3.4 Overfitting:

A statistical model is said to be overfitted, when we train it with a lot of data *(just like fitting ourselves in oversized pants!)*. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

## 3.5 Supervised Machine Learning:

In the Supervised machine learning, labels are included in the training set. There are a few models available in Scikit-Learn worth working with. Here is a list of models used for supervised machine learning.

* K-Nearest Neighbors
* Linear Models
* Naive Bayes Classifiers
* Decision Trees
* Ensembles of Decision Trees
* Kernelized Support Vector Machines

# 4- K-Nearest Neighbors (KNN)

KNN is the simplest of the supervised machine learning models, it only stores the training dataset and predicts the new data point, algorithms find the closest data points in the training datasets, called ‘nearest neighbors’.

## 4.1 The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data, Calculate the distance between the query example and the current example from the data. Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

# 5. Decision Tree:

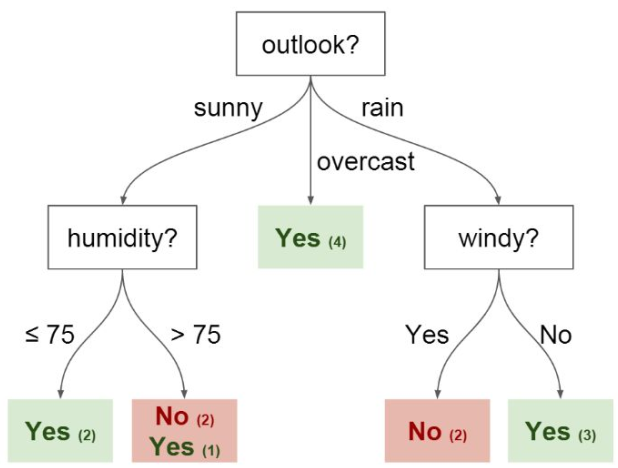
Let’s start off with a thought experiment to give some motivation behind using a decision tree method.

Imagine that I play Tennis every Saturday and I always invite a friend to come with me.

Sometimes my friend shows up, sometimes not. For him it depends on a variety of factors, such as: weather, temperature, humidity, wind etc..

I started keeping track of these features and whether or not he showed up to play with me.

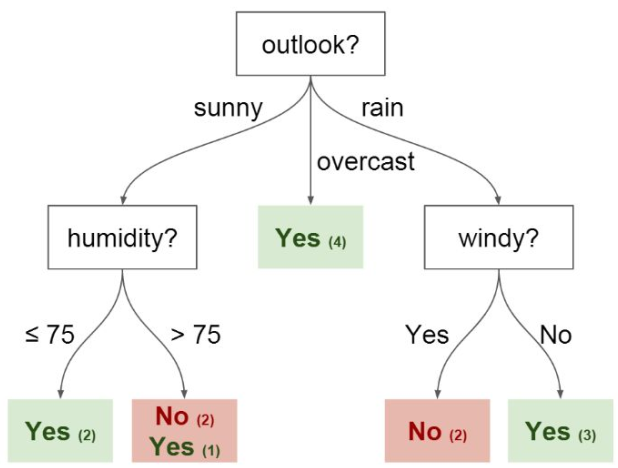
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Temperature | Outlook | Humidity | Windy | Played |
| Mild | Sunny | 80 | No | Yes |
| Hot | Sunny | 75 | Yes | No |
| Hot | Overcast | 77 | No | Yes |
| Cool | Rain | 70 | No | Yes |
| Cool | Overcast | 72 | Yes | Yes |
| Mild | Sunny | 77 | No | No |
| Cool | Sunny | 70 | No | Yes |
| Mild | Rain | 69 | No | Yes |
| Cool | Sunny | 65 | Yes | Yes |
| Mild | Overcast | 77 | Yes | Yes |
| Hot | Overcast | 74 | No | Yes |
| Mild | Rain | 77 | Yes | No |
| Cool | Rain | 73 | Yes | No |



I want to use this data to predict whether or not he will

show up to play.

An intuitive way to do this is through a Decision Tree.

In this tree we have:

● Nodes

○ Split for the value of a

certain attribute

● Edges

○ Outcome of a split to

next node.

● Root

○ The node that performs

the first split

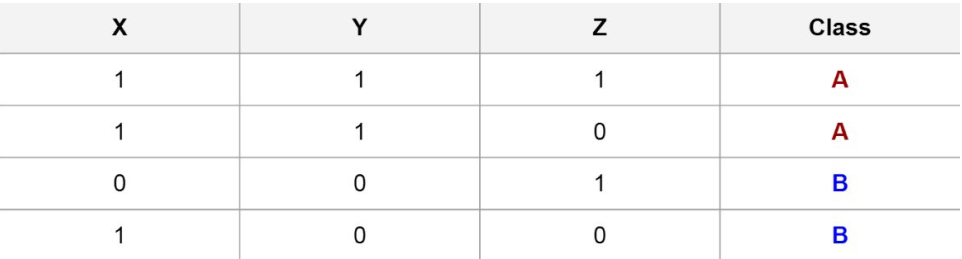
● Leaves

○ Terminal nodes that

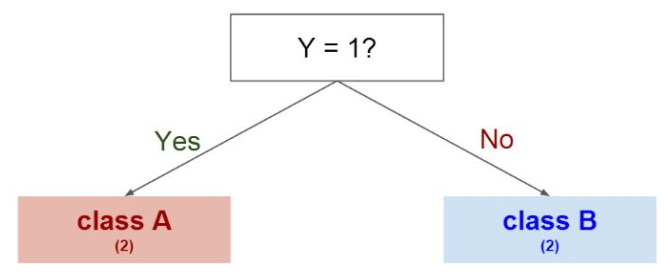
predict the outcome

## 5.1 Intuition Behind Splits:

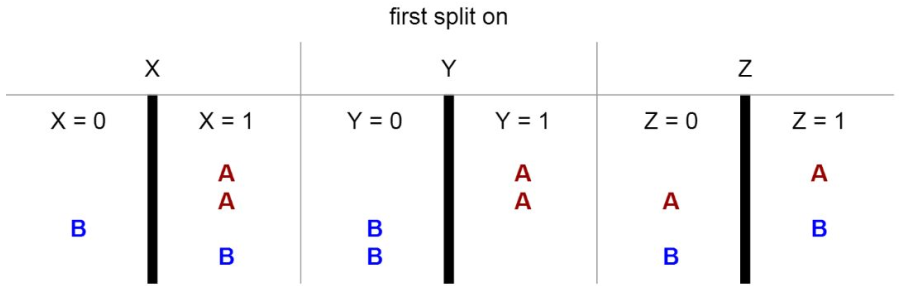
Imaginary Data with 3 features (X,Y, and Z) with two possible classes.



Splitting on Y gives us a clear separation between classes



We could have also tried splitting on other features first:



In the next half of the manuals, we will be implementing both KNN and Decision Trees using Scikit-Learn Python.

# 6. Model Evaluation:

There are different methods to evaluate both regression and classification based machine learning models.

## 6.1 Regression:

Here we are starting from regression models first.

There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

1. Mean Squared Error (MSE).
2. Root Mean Squared Error (RMSE).
3. Mean Absolute Error (MAE)

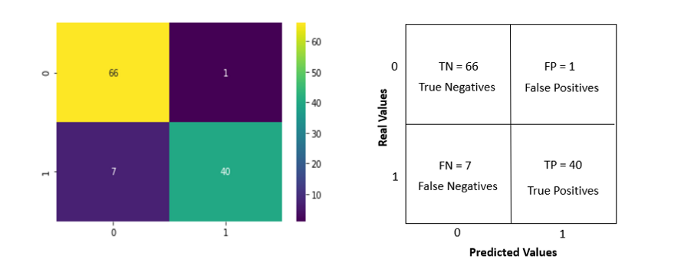
## 6.2 Classification:

Evaluation of a model is always critical and important as well. Two techniques used to evaluate a classification model are the confusion matrix and the classification report.

To Understand well, a brief explanation of both confusion matrix and classification report is required.

### 6.2.1 Confusion Matrix:

The confusion matrix is an N x N table (where N is the number of classes) that contains the number of correct and incorrect predictions of the classification model.



The generalized terms contained in confusion matrix are explained below:

The values returned by the confusion matrix are divided into the following categories: **True Positive (TP):**The model predicted positive, and the real value is positive. **True Negative (TN):**The model predicted negative, and the real value is negative.

**False Positive (FP):**The model predicted positive, but the real value is negative (Type I error).

**False Negative (FN):**The model predicted negative, but the real value is positive (Type II error).

**Confusion Matrix:**

To apply confusion matrix, first of all import the required libraries,

from sklearn.metrics import confusion\_matrix

and then put the predicted labels (from our model) and original testing labels of our dataset into confusion matrix:

confusion\_matrix(y\_test, clf.predict(X\_test))

### 6.2.2 Classification Report

Classification report is another performance evaluation matrix of our model. Before going into greater depth of Implementation, the different parts of classification report are explained:

**Precision:**

The precision returns the proportion of true positives among all the values predicted as positive.

Precision **=** TP/(TP + FP) = 40/(40 + 1) = 0.98 = 98%

**Recall:**

The recall returns the proportion of positive values correctly predicted.

Recall= TP/(TP + FN) = 40/(40 + 7) = 0.85 = 85%

**F1 Score:**

The f1-score is the harmonic mean of precision and recall. It is often used to compare classifiers.

F1-score **=** (2 x Precision x Recall)/(Precision + Recall) = 0.91 = 91%

The harmonic mean gives more weight to the lower value, so a high F1-score means that both precision and recall are high.

# 7. Implementation of Regression:

We can start implementing all these concepts in Scikit-Learn by getting a hands-on Regression model first.

We are starting from KNN first.

## 7.1 Data Acquisition:

First of all, let's acquire the data first using the pandas framework. We have to import required libraries first.

import pandas as pd

import numpy as np

# to split the data into train-test parts

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

# KNN Regressor

from sklearn.neighbors import KNeighborsRegressor

#Model Evaluation

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

#Decision Tree Regressor

from sklearn.tree import DecisionTreeRegressor

At the second step, we have to import the data using Pandas.

df=pd.read\_csv('USA\_Housing.csv')

The dataset is used to predict the house prices given area income, number of rooms, number of bedrooms and population of vicinity.

To move forward, one has to clean the data first by getting rid of all the unwanted columns. In this example, the *address* is the column to be dropped because the machine learning model disallows text data inclusion.

df=df.drop(['Address'],axis=1)

Next step in each machine learning section is to split the data into input-output columns, input columns are also called features while output columns are called Labels.

x=df.drop(['Price'],axis=1)

y=df['Price']

Now it's time to split the data into training-testing modules. We are dividing the data into 70-30 ratio, such as 70% goes to training and 30% goes to testing part.

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.3)

## 7.2 KNeighbor Regression:

To incorporate the machine learning model, first of all we need to import the library.

knn=KNeighborsRegressor(n\_neighbors=4)

Now fit the model to training parts of the data.

knn.fit(X\_train,Y\_train)

So far, Our model is trained. Before moving towards the model evaluation part, we can generate predictions from the model.

To generate predictions for single values, here *60000, 4, 7, 3, 30000* are the average area income, house age, number of rooms , number of bedrooms and area population respectively.

knn.predict([[60000,4,7,3,30000]])

### 7.2.1 Model Evaluation:

Now we need to evaluate the model first, prior to going for deployment. For this context, first make model predictions on testing data.

Y\_pred=knn.predict(X\_test)

Now find *absolute\_mean\_error.*

mean\_absolute\_error(Y\_test,Y\_pred)

Then *absolute\_square\_error.*

mean\_squared\_error(Y\_test,Y\_pred)

For *root\_mean\_square\_error*

np.sqrt(mean\_squared\_error(Y\_test,Y\_pred))

## 7.3 Decision Tree Regressor:

To apply, Decision Tree Regression on the same data, we first need to import the library first.

from sklearn.tree import DecisionTreeRegressor

Then the same steps need to be followed as one did for KNN Regression.

Creating the model. In the first half of the problem set, we need to create the model.

dtr=DecisionTreeRegressor()

Now training the model.

dtr.fit(X\_train,Y\_train)

Making predictions from the data.

Y\_pred=dtr.predict(X\_test)

### 7.3.1 Model Evaluation:

To evaluate the model, we need to find the errors of the model. These errors are:

# Mean Absolute Error

mean\_absolute\_error(Y\_test,Y\_pred)

# Mean Squared Error

mean\_squared\_error(Y\_test,Y\_pred)

# Root Mean Squared Error

np.sqrt(mean\_squared\_error(Y\_test,Y\_pred))

# 8. Implementation of the Classification:

As discussed, KNN and Decision Tree both are used for regression and classification as well. In this portion of the manuals we will be covering classification aspects of KNN and Decision Trees.

## 8.1 Data Acquisition:

First step is to import the data.

df=pd.read\_csv('Iris.csv')

Lets divide the data into labels and features.

x=df.drop(['Species'],axis=1)

y=df['Species']

Now split the data into train-test parts.

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.3)

## 8.2 KNN Classifier

Import the model first.

from sklearn.neighbors import KNeighborsClassifier

Creating the model object with 12 neighbors, neighbors has been chosen randomly. In the later section we will go through correctly identifying the number of neighbors for KNN.

knn=KNeighborsClassifier(n\_neighbors=12)

Train the model.

knn.fit(X\_train,Y\_train)

Making predictions from models.

Y\_Pred=knn.predict(X\_test)

### 8.2.1 Model Evaluation.

To evaluate a classification model, we will be using Confusion Matrix and Classification Reports

First of all, we need to import required libraries.

from sklearn.metrics import confusion\_matrix, classification\_report

Confusion Matrix

print(confusion\_matrix(Y\_Pred,Y\_test))

Classification Report:

print(classification\_report(Y\_Pred,Y\_test))

## 8.3 Decision Tree Classifier:

In this section of the manuals, we will be applying the Decision Tree algorithm to our same data.

The Code snippet is given.

# import the model

from sklearn.tree import DecisionTreeClassifier

#creating empty model

dtc=DecisionTreeClassifier()

# model training

dtc.fit(X\_train,Y\_train)

# making predictions from model

Y\_Pred=dtc.predict(X\_test)

### 8.3.1 Model Evaluation:

Now the evaluation part comes into play. Lets generate confusion matrix and classification reports together.

print(confusion\_matrix(Y\_Pred,Y\_test))

print(classification\_report(Y\_Pred,Y\_test))

# 9. Misc

## 9.1 Correctly Identifying Neighbors for KNN:

Number of neighbors in KNN is an important parameter. Choosing n\_neighbors is always critical while working with KNN.

Here is the Implementation details: first import the required libraries.

import matplotlib.pyplot as plt

By estimating the maximum number of neighbors for KNN (according to data) can be 40. More or less we want optimal neighbors count to 40. for this a code snippet is given below.

error\_matrix = []

for i in range(1,40):

knn = KNeighborsClassifier(n\_neighbors=i)

knn.fit(X\_train,Y\_train)

Y\_pred=knn.predict(X\_test)

error\_matrix.append(np.mean(Y\_pred != Y\_test))

we make a loop to run upto 40 times to calculate the error. Now we can visualize the graph as follows,

plt.figure(figsize=(10,7))

plt.plot(range(1,40),error\_matrix,color='blue',linestyle='dashed',marker='o',

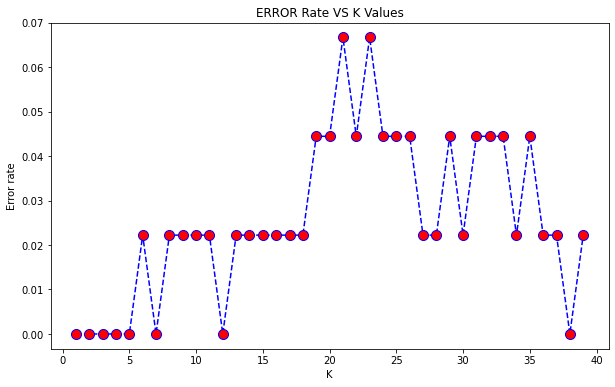
markerfacecolor='red',markersize=10)

plt.title('ERROR Rate VS K Values')

plt.xlabel('K')

plt.ylabel('Error rate')

The result is given below.



According to the graph, optimal values of neighbors are when the error rate is minimum.

## 9.2 Visualizing the Decision Trees.

Internal representation of a decision tree can easily be determined through built-in features of the Decision Tree model.

The coding snippet is given.

#necessary library to import

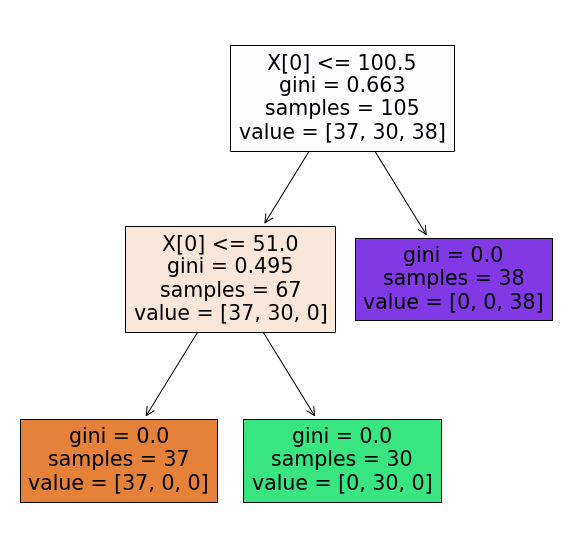
from sklearn import tree

#Visualization

fig = plt.figure(figsize=(10,10))

tree.plot\_tree(dtc,filled=True)

The result is:



# 10 Exercise: (25)

## 10.1 Problem 1: (Predict the Survival) (15)

You are required to predict the survival for the passengers traveling titanic. But this is not that simple. Before moving towards machine learning, one has to drop columns, fill in the missing values and convert text columns into categories.

To replace text values to categories, Please use *replace* function.

Use both of the models, also fine tune the KNN (identify the correct number of neighbors) to make it more accurate and find out which model performs better. (train.csv file is attached)

## 10.2 Problem 2: (Forecast USD-PKR currency exchange) (10)

Forecast USD-PKR currency exchange using both KNN and Random Tree models. Identify the model which performs better. (currency.csv file is attached)

# 11. Submission Instructions:

1. A data file is attached. For Practice Exercise, One has to use this file.
2. For Examples given in manual, no dataset is required because the dataset used is available in sklearn library.
3. To make the submission, Create a Jupyter Notebook File (lab5\_rollno.ipynb), create a .zip file along with the data file and submit on the Portal.