Group B : Machine Learning

Assignment no:1

Title of the Assignment: Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:

1. Pre-process the dataset.
2. Implement linear regression and random forest regression models.
3. Evaluate the models and compare their respective scores like R2, RMSE, etc.

Dataset Description: The project is about on world's largest taxi company Uber inc. In this project, we're looking to predict the fare for their future transactional cases. Uber delivers service to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. Eventually, it becomes really important to estimate the fare prices accurately.

Link for Dataset:https://www.kaggle.com/datasets/yasserh/uber-fares-dataset **Objective of the**

Assignment:
Students should be able to preprocess dataset and identify outliers, to check correlation and implement linear regression and random forest regression models. Evaluate them with respective scores like R2, RMSE etc.

Prerequisite:

1. Basic knowledge of Python
2. Concept of preprocessing data
3. Basic knowledge of Data Science and Big Data Analytics.

Contents of the Theory:

1. Data Preprocessing
2. Linear regression
3. Random forest regression models
4. Box Plot
5. Outliers
6. Haversine
7. Mathplotlib
8. Mean Squared Error

Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for amachine learning model. It is the rst and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean itand put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

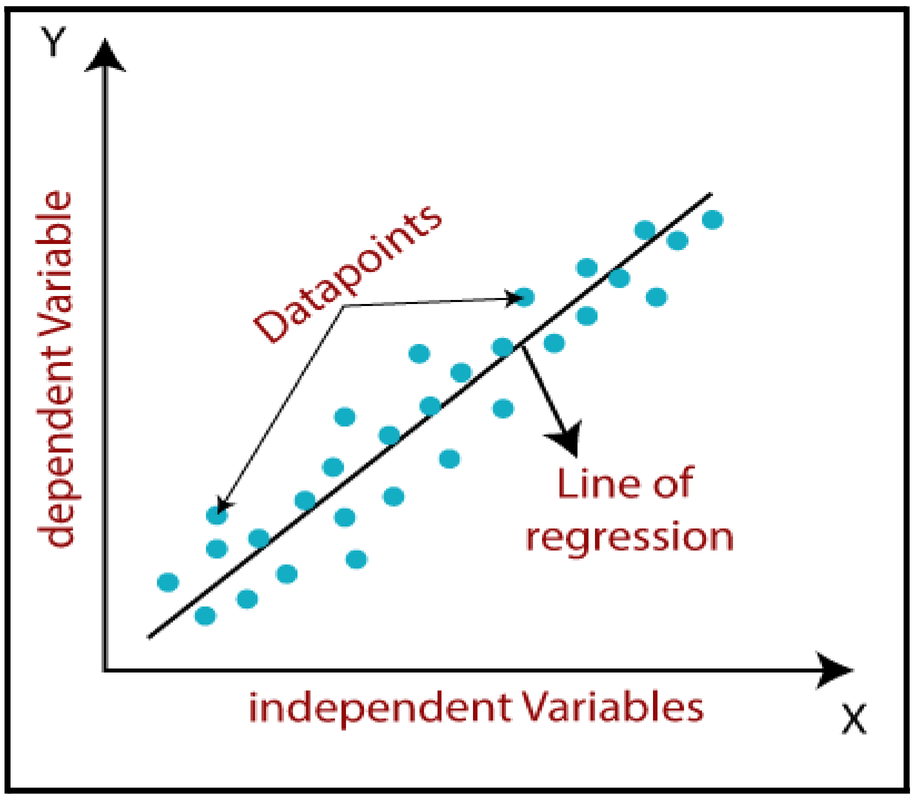
A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning thedata and making it suitable for a machine learning model which also increases the accuracy . □

Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as**sales, salary, age, product price,**etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it □nds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Random Forest Regression Models:

Random Forest is a popular machine learning algorithm that belongs to the supervisedlearning technique. It can be used for both Classi cation and Regression problems in ML. Itis based on the concept of**ensemble learning,**which is a process ofcombining multiple classi ers to solve a complex problem and to improve the performance of the model.

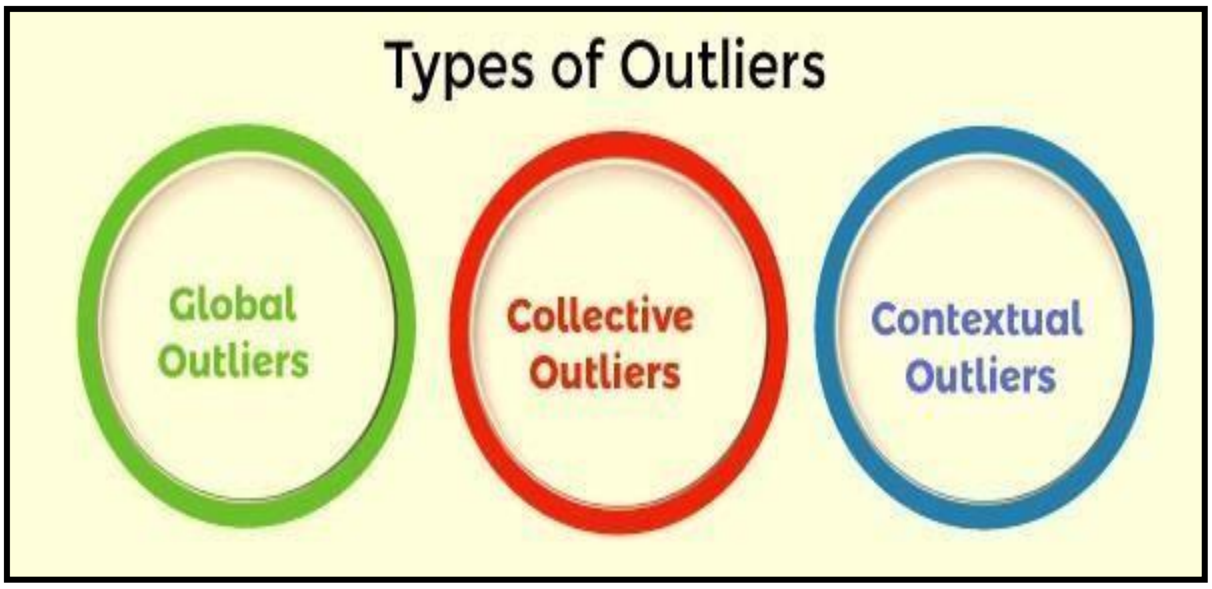
As the name suggests,**"Random Forest is a classi er that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictiveaccuracy of that dataset."**Instead of relying on one decision tree, the random forest takes the

prediction from each tree and based on the majority votes of predictions, and it predicts the nal output.

The greater number of trees in the forest leads to higher accuracy and prevents theproblem of over tting.

Outlier:

Themajor thing about the outliers is what you do with them. If you are going toanalyze any task to analyzedata sets, you will always have some assumptions based onhow this data is generated. If you nd some data points that are likely to contain some form of error, then these are de nitely outliers, and depending on the context, you want toovercome those errors. The data mining process involves the analysis and prediction of datathat the data holds. In 1969, Grubbs introduced the rst de nition of outliers.

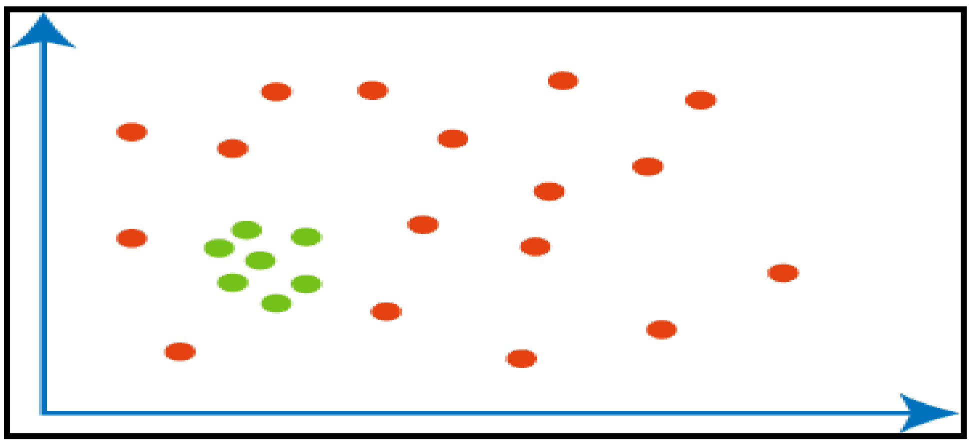


Global Outliers

Global outliers are also called point outliers. Global outliers are taken as the simplest form of outliers. When data points deviate from all the rest of the data points in a given data set, it is known as the global outlier. In most cases, all the outlier detection procedures are targeted to determine the global outliers. The green data point is the global outlier.

Collective Outliers

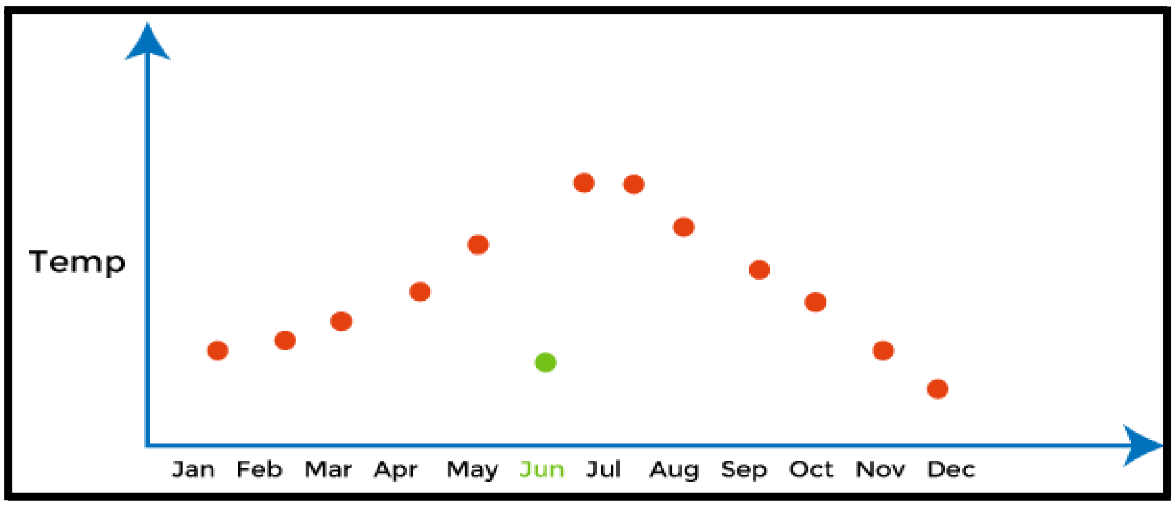
In a given set of data, when a group of data points deviates from the rest of the data set is called collective outliers. Here, the particular set of data objects may not be outliers, but when you consider the data objects as a whole, they may behave as outliers. To identify thetypes of different outliers, you need to go through background information about the relationship between the behavior of outliers shown by different data objects. For example, in an Intrusion Detection System, the DOS package from one system to another is taken as normal behavior. Therefore, if this happens with the various computer simultaneously, it is considered abnormal behavior, and as a whole, they are called collective outliers. The greendata points as a whole represent the collective outlier



.

Contextual Outliers

As the name suggests, "Contextual" means this outlier introduced within a context. For example, in the speech recognition technique, the single background noise. Contextual outliers are also known as Conditional outliers. These types of outliers happen if a data object deviates from the other data points because of any speci c condition in a given data set. As we know, there are two types of attributes of objects of data: contextual attributes and behavioral attributes. Contextual outlier analysis enables the users to examine outliers in different contexts and conditions, which can be useful in various applications. For example, A temperature reading of 45 degrees Celsius may behave as an outlier in a rainy season. Still, it will behave like a normal data point in the context of a summer season. In thegiven diagram, a green dot representing the low-temperature value in June is a contextual outlier since the same value in December is not an outlier.



Haversine:

The Haversine formula calculates the shortest distance between two points on a sphere using their latitudes and longitudes measured along the surface. It is important for use in navigation.

Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Mean Squared Error;

The**Mean Squared Error (MSE)**or**Mean Squared Deviation (MSD)**of an estimator measures the average of error squares i.e. the average squared difference between theestimated values and true value. It is a risk function, corresponding to the expected value ofthe squared error loss. It is always non – negative and values close to zero are better. TheMSE is the second moment of the error (about the origin) and thus incorporates both thevariance of the estimator and its bias.

Conclusion:

In this way we have explored Concept correlation and implement linear regression andrandom forest regression models.

Assignment Questions:

1. What is data preprocessing?
2. De ne Outliers?
3. What is Linear Regression?
4. What is Random Forest Algorithm?
5. Explain: pandas, numpy?

Assignment no:2

Title of the Assignment: Classify the email using the binary classification method. EmailSpam detection has two states:
a) Normal State – Not Spam,
b) Abnormal State – Spam.
Use K-Nearest Neighbors and Support Vector Machine for classification.
Analyze theirperformance.

Dataset Description: The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction : 1for spam, 0 for not spam. The remaining 3000 columns are the 3000 most common words inall theemails, after excluding the non-alphabetical characters/words. For each row, thecount of each word(column) in that email(row) is stored in the respective cells. Thus,information regarding all 5172 emails are stored in a compact dataframe rather than asseparate text files.

Link:https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv

Objective of the Assignment:

Students should be able to classify email using the binary Classification and implement email spam detection technique by using K-Nearest Neighbors and Support Vector Machine algorithm.

Prerequisite:

1. Basic knowledge of Python
2. Concept of K-Nearest Neighbors and Support Vector Machine for classification.

Contents of the Theory:

1. Data Preprocessing
2. Binary Classification
3. K-Nearest Neighbours
4. Support Vector Machine
5. Train, Test and Split Procedure

Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for amachine learning model. It is the□rst and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean itand put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning thedata and making it suitable for a machine learning model which also increases the accuracy and e ciency of a machine learning model.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scalin

Assignment no:3

Title of the Assignment: Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months

Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc.

Link for Dataset:https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling **Perform**

the following steps:

1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and test sets.
3. Normalize the train and test data.
4. Initialize and build the model. Identify the points of improvement and implement the same.
5. Print the accuracy score and confusion matrix (5 points).

Objective of the Assignment:
Students should be able to distinguish the feature and target set and divide the data set into training and test sets and normalize them and students should build the model on the basis of that.

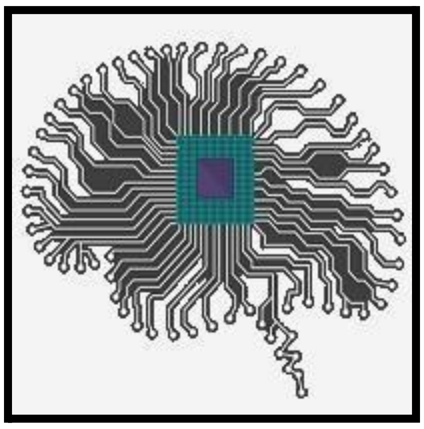
Prerequisite:

1. Basic knowledge of Python
2. Concept of Confusion Matrix

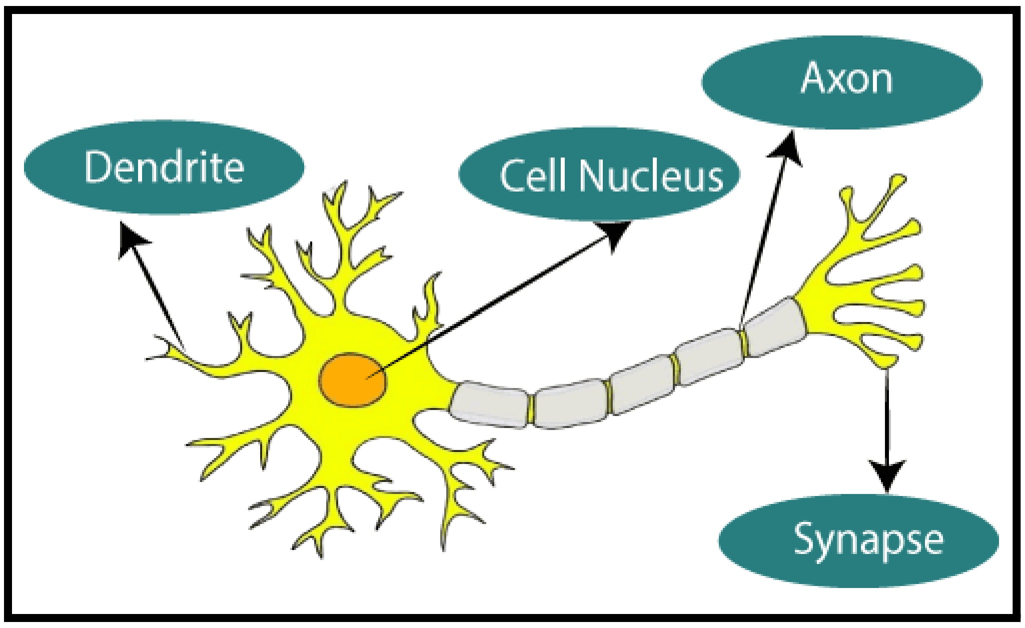
Contents of the Theory:

1. Artificial Neural Network
2. Keras
3. tensorflow
4. Normalization
5. Confusion Matrix

Artificial Neural Network:

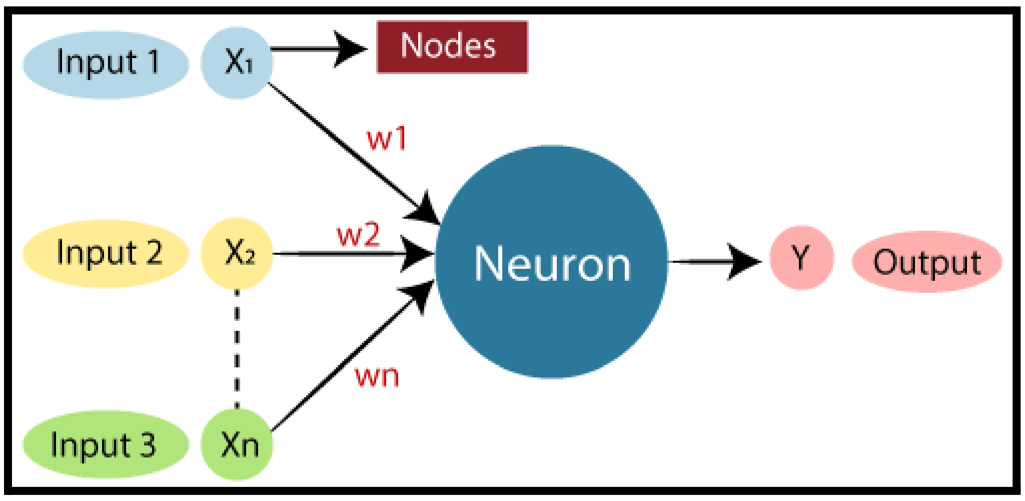


The term "Arti cial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, arti cial neural networks also have neurons that are interconnected to one another in various layers of thenetworks. These neurons are known as nodes.



The given gure illustrates the typical diagram of Biological Neural Network.

The typical Arti cial Neural Network looks something like the given gure.



Dendrites from Biological Neural Network represent inputs in Arti cial Neural Networks, cellnucleusrepresents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and arti cial neural network:

Biological Neural Network	**Artificial Neural Network**
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

An**Arti cial Neural Network**in the eld of**Arti cial intelligence**where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option tounderstand things and makedecisions in a human-like manner. The arti cial neural networkis designed by programming computers to behave simply like interconnected brain cells.

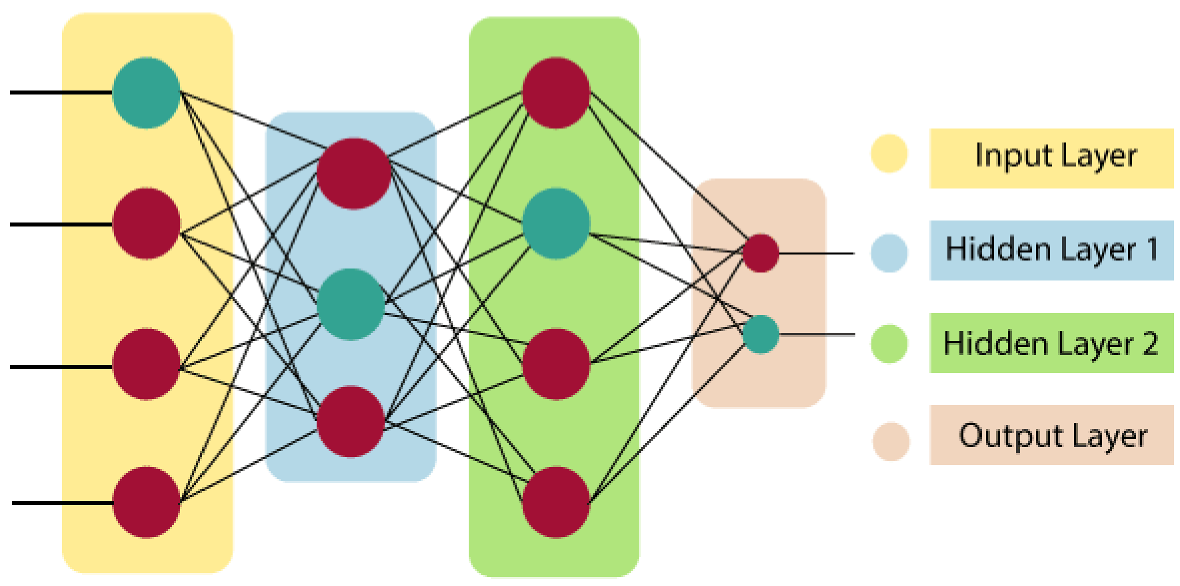
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up ofincredibly amazing parallel processors.

We can understand the arti cial neural network with an example, consider an example of adigital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," thenwe get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

To understand the concept of the architecture of an arti ficial neural network, we have to understand what a neural network consists of. In order to de ne a neural network that consists of a large number of arti cial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an arti cial neural network.

Arti cial Neural Network primarily consists of three layers:

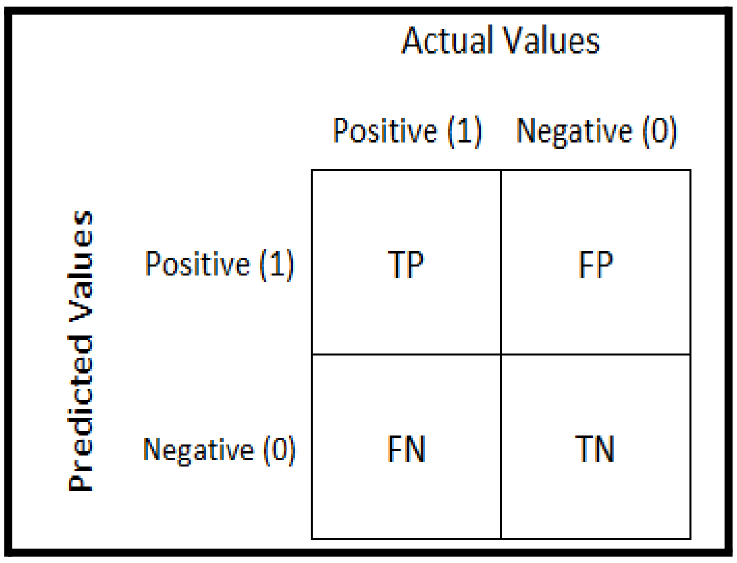


The hidden layer presents in-between input and output layers. It performs all the calculationsto nd hidden features and patterns.

Confusion Matrix:

The confusion matrix is a matrix used to determine the performance of the classi cation models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itselfcan be easily understood, but the related terminologies may beconfusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an**error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classi ers, the matrix is of 2*2 table, for 3 classes, it is3*3 table, and so on.
- The matrix is divided into two dimensions, that are**predicted values**and**actual values**along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actualvalues are the true values for the given observations.
- It looks like the below table:



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

The above table has the following cases:

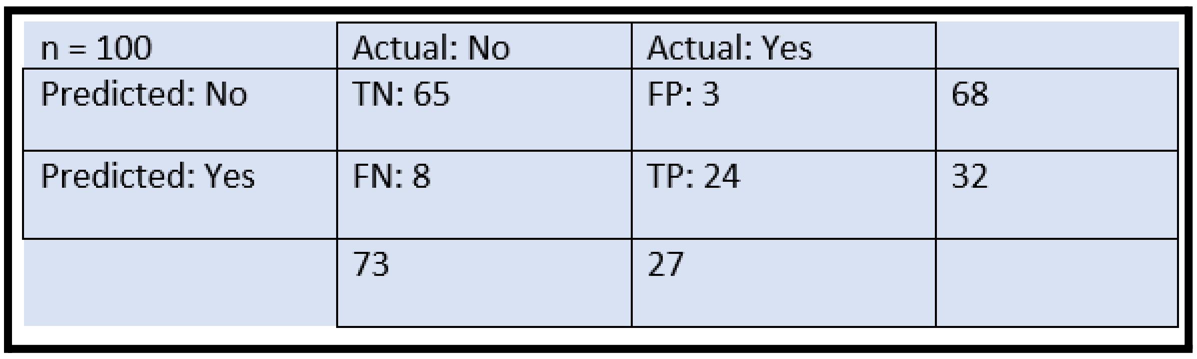
- **True Negative:**Model has given prediction No, and the real or actual value was alsoNo.
- **True Positive:**The model has predicted yes, and the actual value was also true.
- **False Negative:**The model has predicted no, but the actual value was Yes, it is alsocalled as**Type-II error**.
- **False Positive:**The model has predicted Yes, but the actual value was No. It is alsocalled a**Type-I error.**

Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classification models, when they makepredictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as itis either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters forthe model, such as accuracy, precision, etc.

Example: We can understand the confusion matrix using an example.

Suppose we are trying to create a model that can predict the result for the disease that iseither a person has that disease or not. So, the confusion matrix for this is given as:



n = 100	Actual: No	Actual: Yes	
Predicted: No	TN: 65	FP: 3	68
Predicted: Yes	FN: 8	TP: 24	32
	73	27	

From the above example, we can conclude that:

- The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, Yes defines that patient has the disease, and No defines that patient doesnot has that disease.
- The classifier has made a total of**100 predictions**. Out of 100 predictions,**89 are true predictions**, and**11 are incorrect predictions**.
- The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas theactual "Yes" was 27, and actual "No" was 73 times.

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using thismatrix. These calculations are given below:

- **Classification Accuracy:** It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts thecorrect output. It can be calculated as the ratio of the number of correct predictionsmade by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\textbf{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

- **Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as thenumber of incorrect

$$\textbf{Error rate}=\frac{FP+FN}{TP+FP+FN+TN}$$

predictions to all number of the predictions made by the classifier. The formula is given below:

- **Precision:** It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many ofthem were actually true. It can be calculate using the below formula

$$\text{Precision}=\frac{TP}{TP+FP}$$

- **Recall:** It is defined as the out of total positive classes, how our model predictedcorrectly.The recall must be as high as possible.

$$\text{Recall}=\frac{TP}{TP+FN}$$

- **F-measure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the belowformula:

$$\textbf{F-measure}=\frac{2*Recall*Precision}{Recall+Precision}$$

Other important terms used in Confusion Matrix:

- **Null Error rate:**It de nes how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "*the bestclassi er has a higher error rate than the null error rate.*"

- **ROC Curve:**The ROC is a graph displaying a classi er's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and thefalse Positive rate (on the x-axis).

Conclusion:

In this way we build a a neural network-based classi er that can determine whether they willleave or not in the next 6 months

Assignment Questions:

1) What is Normalization?
2) What is Standardization?
3) Explain Confusion Matrix ?
4) De ne the following: Classi cation Accuracy, Misclassi cation Rate, Precision.
5) One Example of Confusion Matrix?

Assignment no:4

Title of the Assignment: Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

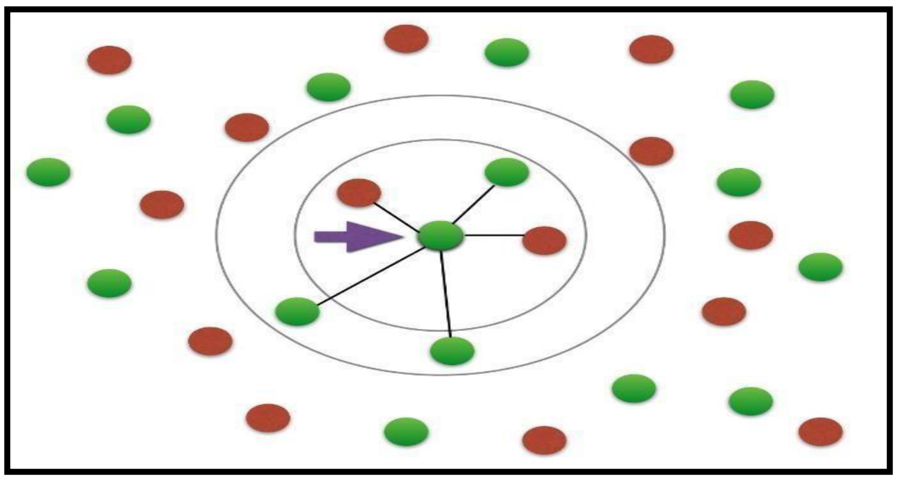
Dataset Description: We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not?
The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Link for Dataset: Diabetes predication system with KNN algorithm | Kaggle

Objective of the Assignment:
Students should be able to preprocess dataset and identify outliers, to check correlation and implement KNN algorithm and random forest classification models. Evaluate them with respective scores like confusion_matrix, accuracy_score, mean_squared_error, r2_score, roc_auc_score, roc_curve etc.

Prerequisite:

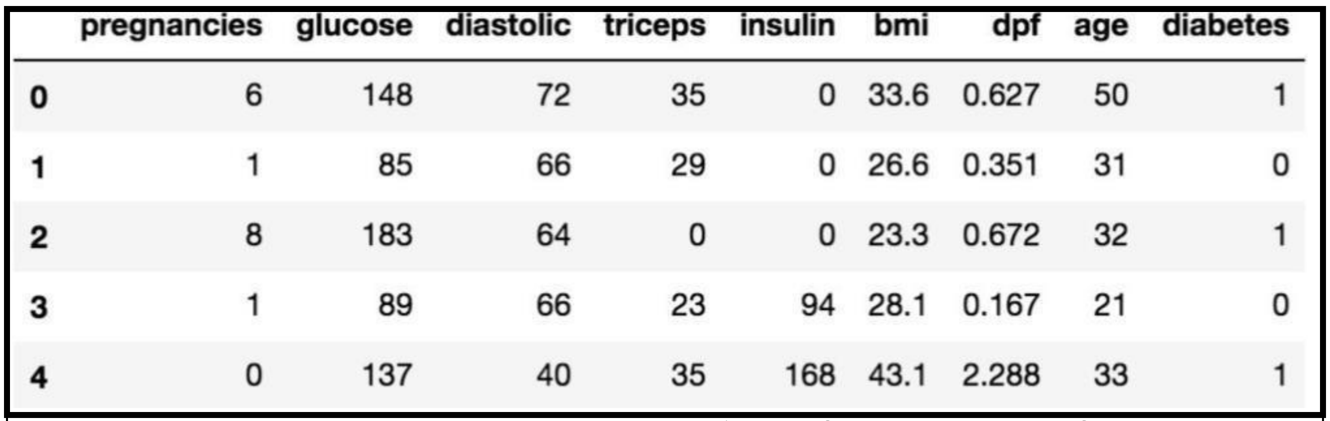
1. Basic knowledge of Python
2. Concept of Confusion Matrix
3. Concept of roc_auc curve.
4. Concept of Random Forest and KNN algorithms



k-Nearest-Neighbors (k-NN) is a supervised machine learning model. Supervised learning is when a model learns from data that is already labeled. A supervised learning model takes in a set of input objects and output values. The model then trains on that data to learn how to map the inputs to the desired output so it can learn to make predictions on unseen data.
k-NN models work by taking a data point and looking at the 'k' closest labeled data points. The data point is then assigned the label of the majority of the 'k' closest points.
For example, if k = 5, and 3 of points are 'green' and 2 are 'red', then the data point in question would be labeled 'green', since 'green' is the majority (as shown in the above graph).
Scikit-learn is a machine learning library for Python. In this tutorial, we will build a k-NN model using Scikit-learn to predict whether or not a patient has diabetes.
Reading in the training data

For our k-NN model, the first step is to read in the data we will use as input. For this example, we are using the diabetes dataset. To start, we will use Pandas to read in the data. I will not go into detail on Pandas, but it is a library you should become familiar with if you're looking to dive further into data science and machine learning.

```
import pandas as pd#read in the data using pandas
df = pd.read_csv('data/diabetes_data.csv')#check data has been read in properly
df.head()
```



	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next, let's see how much data we have. We will call the 'shape' function on our dataframe to see how many rows and columns there are in our data. The rows indicate the number of patients and the columns indicate the number of features (age, weight, etc.) in the dataset for each patient. 1

```
#check number of rows and columns in dataset
df.shape

Op → (768,9)
```

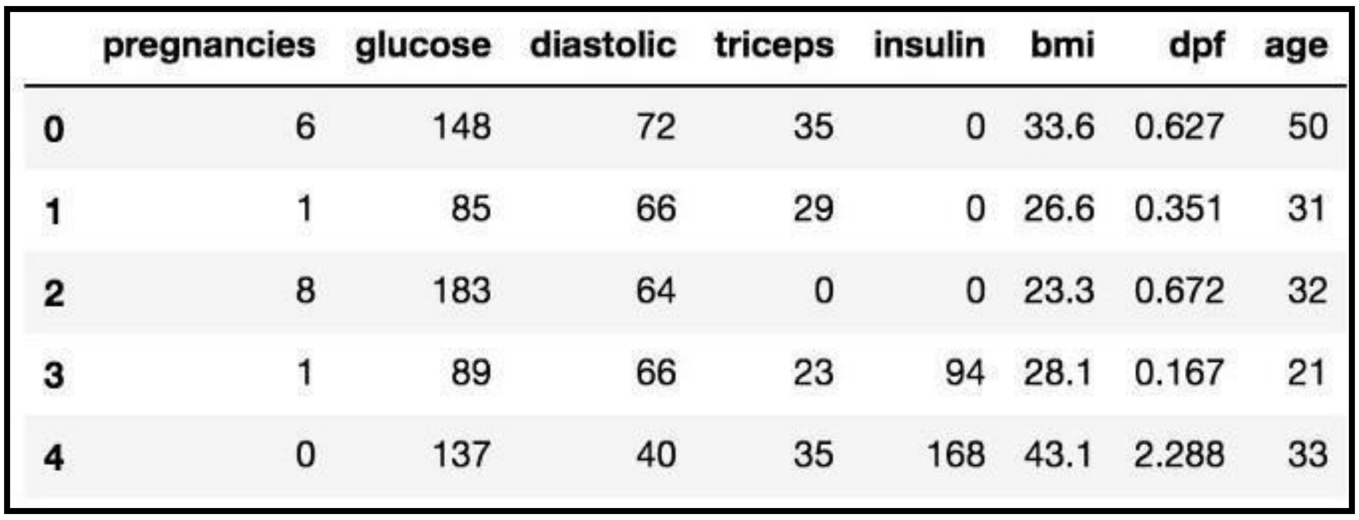
We can see that we have 768 rows of data (potential diabetes patients) and 9 columns (8 input features and 1 target output).

Split up the dataset into inputs and targets

Now let's split up our dataset into inputs (X) and our target (y). Our input will be every column except 'diabetes' because 'diabetes' is what we will be attempting to predict. Therefore, 'diabetes' will be ourtarget.

We will use pandas 'drop' function to drop the column 'diabetes' from our dataframe and store it in the variable 'X'. This will be our input.

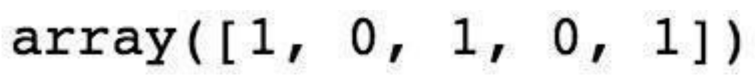
```
#create a dataframe with all training data except the target column
X = df.drop(columns=['diabetes'])#check that the target variable has been removed
X.head()
```



	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

We will insert the 'diabetes' column of our dataset into our target variable (y).

```
#separate target values
y = df['diabetes'].values#view target values
y[0:5]
```



```
array([1, 0, 1, 0, 1])
```

Split the dataset into train and test data

Now we will split the dataset into into training data and testing data. The training data is the data that the model will learn from. The testing data is the data we will use to see how well the model performs on unseen data.

Scikit-learn has a function we can use called 'train_test_split' that makes it easy for us to split our dataset into training and testing data.

```
from sklearn.model_selection import train_test_split#split dataset into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
```

'train_test_split' takes in 5 parameters. The first two parameters are the input and target data we split up earlier. Next, we will set 'test_size' to 0.2. This means that 20% of all the data will be used for testing, which leaves 80% of the data as training data for the model to learn from. Setting 'random_state' to 1 ensures that we get the same split each time so we can reproduce our results.

Setting 'stratify' to y makes our training split represent the proportion of each value in the y variable. For example, in our dataset, if 25% of patients have diabetes and 75% don't have diabetes, setting 'stratify' to y will ensure that the random split has 25% of patients with diabetes and 75% of patients without diabetes.

Building and training the model

Next, we have to build the model. Here is the code:

```
from sklearn.neighbors import KNeighborsClassifier# Create KNN classifier
knn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier to the data
knn.fit(X_train,y_train)
```

First, we will create a new k-NN classifier and set 'n_neighbors' to 3. To recap, this means that if at least 2 out of the 3 nearest points to an new data point are patients without diabetes, then the new data point will be labeled as 'no diabetes', and vice versa. In other words, a new data point is labeled with by majority from the 3 nearest points.

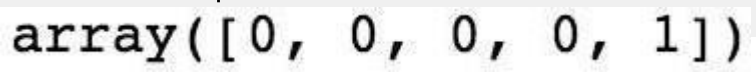
We have set 'n_neighbors' to 3 as a starting point. We will go into more detail below on how to better select a value for 'n_neighbors' so that the model can improve its performance.

Next, we need to train the model. In order to train our new model, we will use the ‘fit’ function and pass in our training data as parameters to fit our model to the training data.

Testing the model

Once the model is trained, we can use the ‘predict’ function on our model to make predictions on ourtest data. As seen when inspecting ‘y’ earlier, 0 indicates that the patient does not have diabetes and 1 indicates that the patient does have diabetes. To save space, we will only show print the first 5 predictions of our test set.

```
#show first 5 model predictions on the test data
kn
```

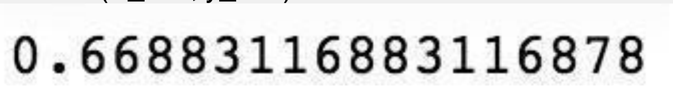


```
array([0, 0, 0, 0, 1])
```

We can see that the model predicted ‘no diabetes’ for the first 4 patients in the test set and ‘has diabetes’ for the 5th patient.

Now let’s see how our accurate our model is on the full test set. To do this, we will use the ‘score’ function and pass in our test input and target data to see how well our model predictions match up to the actual results.

```
#check accuracy of our model on the test data
knn.score(X_test, y_test)
```



```
0.6688311688311688
```

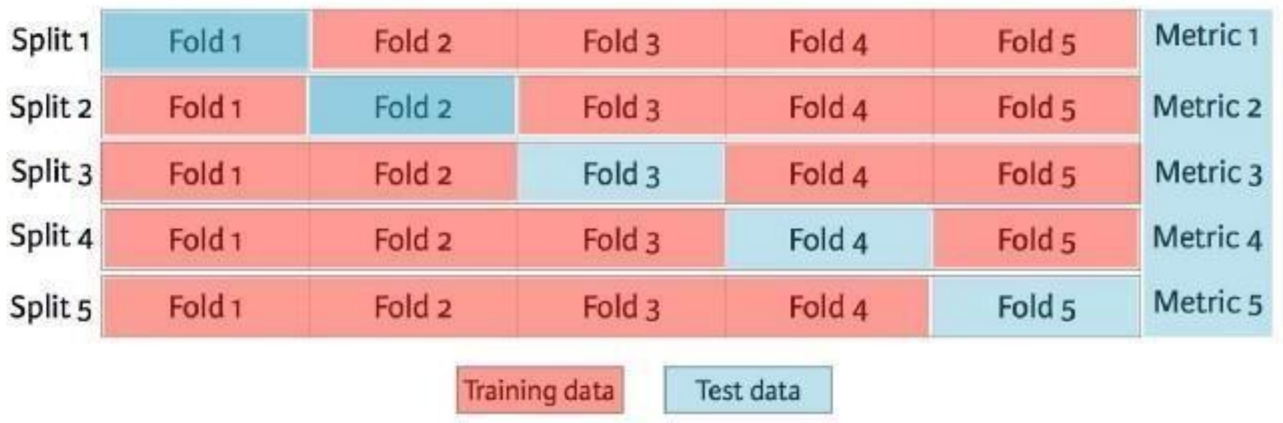
Our model has an accuracy of approximately 66.88%. It’s a good start, but we will see how we can increase model performance below.

Congrats! You have now built an amazing k-NN model!

k-Fold Cross-Validation

Cross-validation is when the dataset is randomly split up into ‘k’ groups. One of the groups is used asthe test set and the rest are used as the training set. The model is trained on the training set and scored on the test set. Then the process is repeated until each unique group as been used as the test set.

For example, for 5-fold cross validation, the dataset would be split into 5 groups, and the model would be trained and tested 5 separate times so each group would get a chance to be the test set. This can be seen in the graph below.



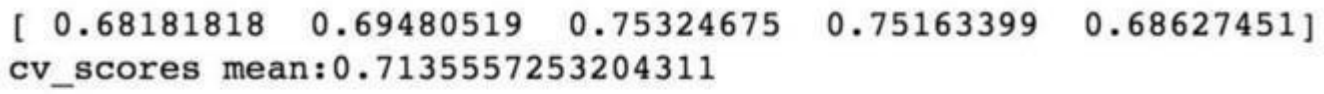
4-fold cross validation (image credit)

The train-test-split method we used in earlier is called ‘holdout’. Cross-validation is better than using the holdout method because the holdout method score is dependent on how the data is split into train and test sets. Cross-validation gives the model an opportunity to test on multiple splits so we can get a better idea on how the model will perform on unseen data.

In order to train and test our model using cross-validation, we will use the ‘cross_val_score’ function with a cross-validation value of 5. ‘cross_val_score’ takes in our k-NN model and our data as parameters. Then it splits our data into 5 groups and fits and scores our data 5 seperate times, recording the accuracy score in an array each time. We will save the accuracy scores in the ‘cv_scores’ variable.

To find the average of the 5 scores, we will use numpy’s mean function, passing in ‘cv_score’. Numpy is a useful math library in Python.

```
from sklearn.model_selection import cross_val_score
import numpy as np#create a new KNN model
knn_cv = KNeighborsClassifier(n_neighbors=3)#train model with cv of 5
cv_scores = cross_val_score(knn_cv, X, y, cv=5)#print each cv score (accuracy) and average them
print(cv_scores)
print(‘cv_scores mean:{}’.format(np.mean(cv_scores)))
```



```
[ 0.68181818  0.69480519  0.75324675  0.75163399  0.68627451]
cv_scores mean:0.7135557253204311
```

Using cross-validation, our mean score is about 71.36%. This is a more accurate representation of how our model will perform on unseen data than our earlier testing using the holdout method.

Hypertuning model parameters using GridSearchCV

When built our initial k-NN model, we set the parameter ‘n_neighbors’ to 3 as a starting point with no real logic behind that choice.

Hypertuning parameters is when you go through a process to find the optimal parameters for your model to improve accuracy. In our case, we will use GridSearchCV to find the optimal value for ‘n_neighbors’.

GridSearchCV works by training our model multiple times on a range of parameters that we specify. That way, we can test our model with each parameter and figure out the optimal values to get the best accuracy results.

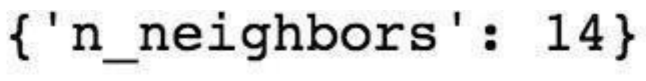
For our model, we will specify a range of values for ‘n_neighbors’ in order to see which value works best for our model. To do this, we will create a dictionary, setting ‘n_neighbors’ as the key and using numpy to create an array of values from 1 to 24.

Our new model using grid search will take in a new k-NN classifier, our param_grid and a cross- validation value of 5 in order to find the optimal value for ‘n_neighbors’.

```
from sklearn.model_selection import GridSearchCV#create new a knn model
knn2 = KNeighborsClassifier()#create a dictionary of all values we want to test for n_neighbors
param_grid = {‘n_neighbors’: np.arange(1, 25)}#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)#fit model to data
knn_gscv.fit(X, y)
```

After training, we can check which of our values for ‘n_neighbors’ that we tested performed the best. To do this, we will call ‘best_params_’ on our model.

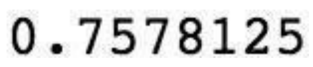
```
#check top performing n_neighbors value
knn_gscv.best_params_
```



```
{'n_neighbors': 14}
```

We can see that 14 is the optimal value for ‘n_neighbors’. We can use the ‘best_score_’ function to check the accuracy of our model when ‘n_neighbors’ is 14. ‘best_score_’ outputs the mean accuracy of the scores obtained through cross-validation.

```
#check mean score for the top performing value of n_neighbors
knn_gscv.best_score_
```



```
0.7578125
```

Conclusion:

In this way we build a a neural network-based classifier that can determine whether they willleave or not in the next 6 months

Assignment no:5

Title of the Assignment: Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

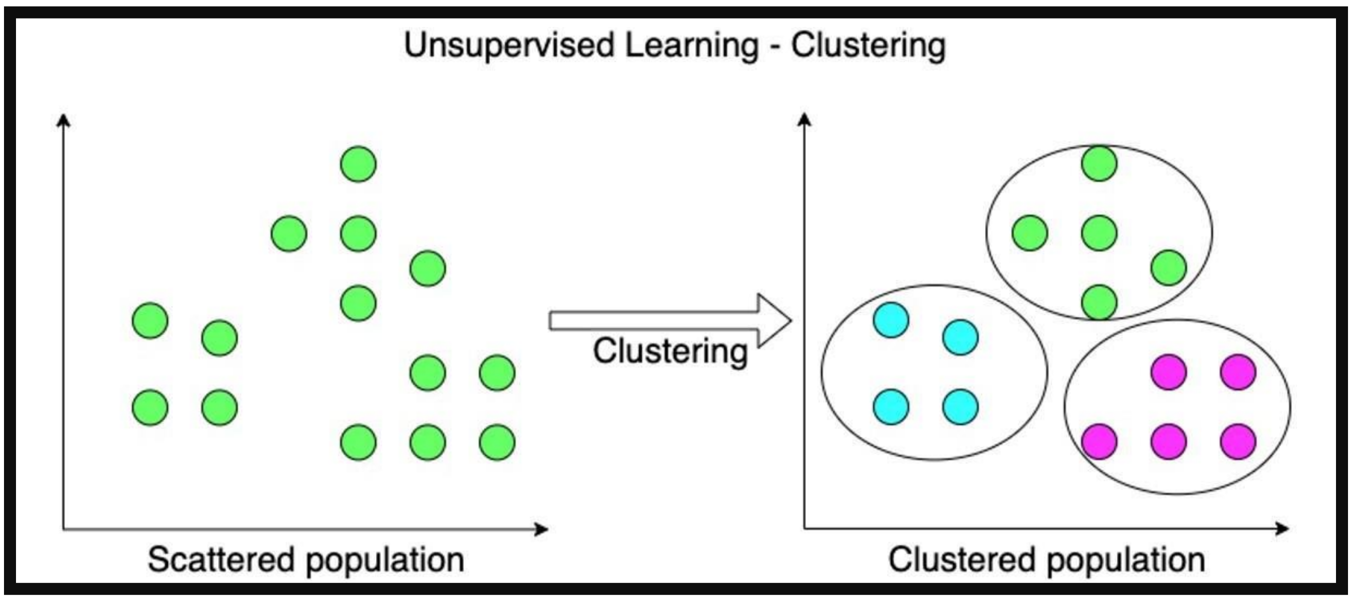
Dataset Description: The data includes the following features:
1. Customer ID
2. Customer Gender
3. Customer Age
4. Annual Income of the customer (in Thousand Dollars)
5. Spending score of the customer (based on customer behavior and spending nature)

Objective of the Assignment:
Students should able to understand how to use unsupervised learning to segment different-different clusters or groups and used to them to train your model to predict future things.

Prerequisite:

1. Knowledge of Python
2. Unsupervised learning
3. Clustering
4. Elbow method

Clustering algorithms try to find natural clusters in data, the various aspects of how the algorithms to cluster data can be tuned and modified. Clustering is based on the principle that items within the same cluster must be similar to each other. The data is grouped in such a way that related elements are close to each other.



Diverse and different types of data are subdivided into smaller groups.

Uses of Clustering
Marketing:

In the field of marketing, clustering can be used to identify various customer groups with existing customer data. Based on that, customers can be provided with discounts, offers, promo codes etc.

Real Estate:

Clustering can be used to understand and divide various property locations based on value and importance. Clustering algorithms can process through the data and identify various groups of property on the basis of probable price.

BookStore and Library management:

Libraries and Bookstores can use Clustering to better manage the book database. With proper book ordering, better operations can be implemented.

Document Analysis:

Often, we need to group together various research texts and documents according to similarity. And in such cases, we don’t have any labels. Manually labelling large amounts of data is also not possible. Using clustering, the algorithm can process the text and group it into different themes.

These are some of the interesting use cases of clustering.

K-Means Clustering

K-Means clustering is an unsupervised machine learning algorithm that divides the given data into the given number of clusters. Here, the “K” is the given number of predefined clusters, that need to be created.

It is a centroid based algorithm in which each cluster is associated with a centroid. The main idea is to reduce the distance between the data points and their respective cluster centroid.

The algorithm takes raw unlabelled data as an input and divides the dataset into clusters and the process is repeated until the best clusters are found.

K-Means is very easy and simple to implement. It is highly scalable, can be applied to both small and large datasets. There is, however, a problem with choosing the number of clusters or K. Also, with the increase in dimensions, stability decreases. But, overall K Means is a simple and robust algorithm that makes clustering very easy.

```
#Importing the necessary librariesimport
numpy as np
import pandas as pd
import matplotlib.pyplot as pltimport
seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

The necessary libraries are imported.

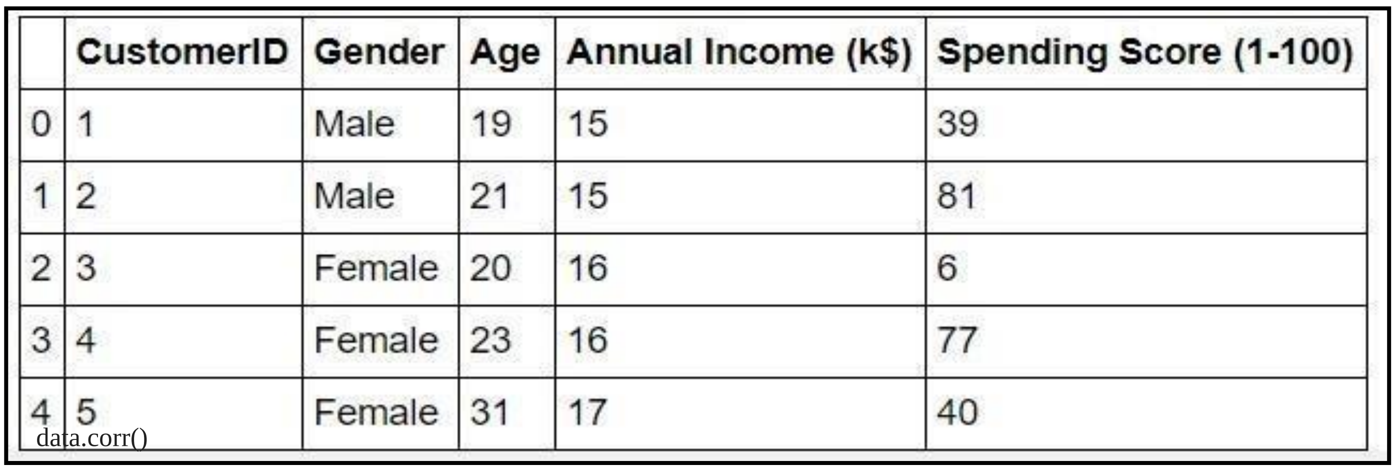
```
#Reading the excel file data=pd.read_excel("Mall_Customers.xlsx")
```

The data is read. I will share a link to the entire code and excel data at the end of the article.

The data has 200 entries, that is data from 200 customers.

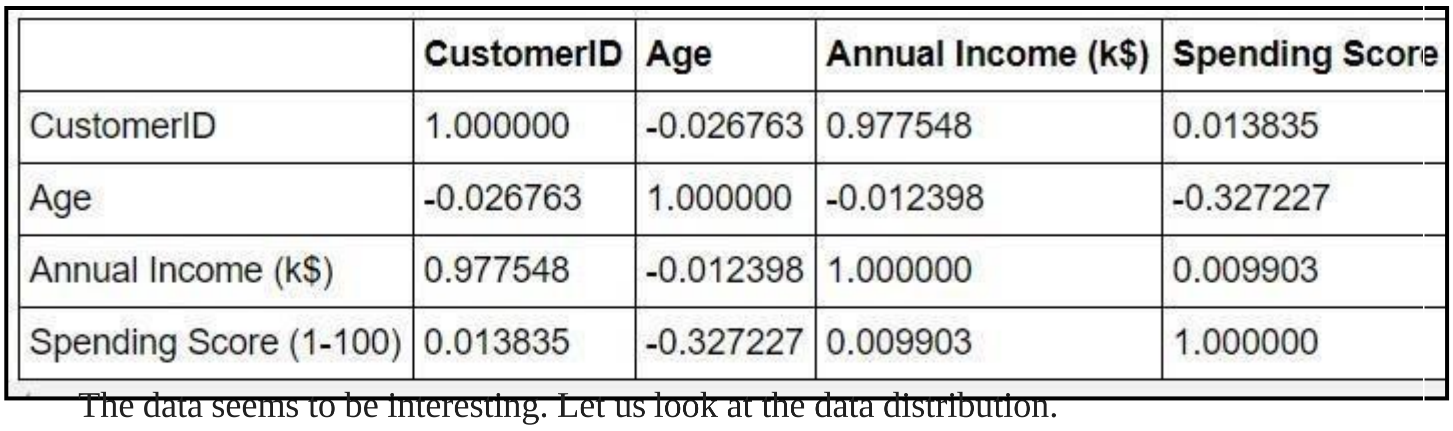
```
data.head()
```

So let us have a look at the data.



	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
data.corr()
```

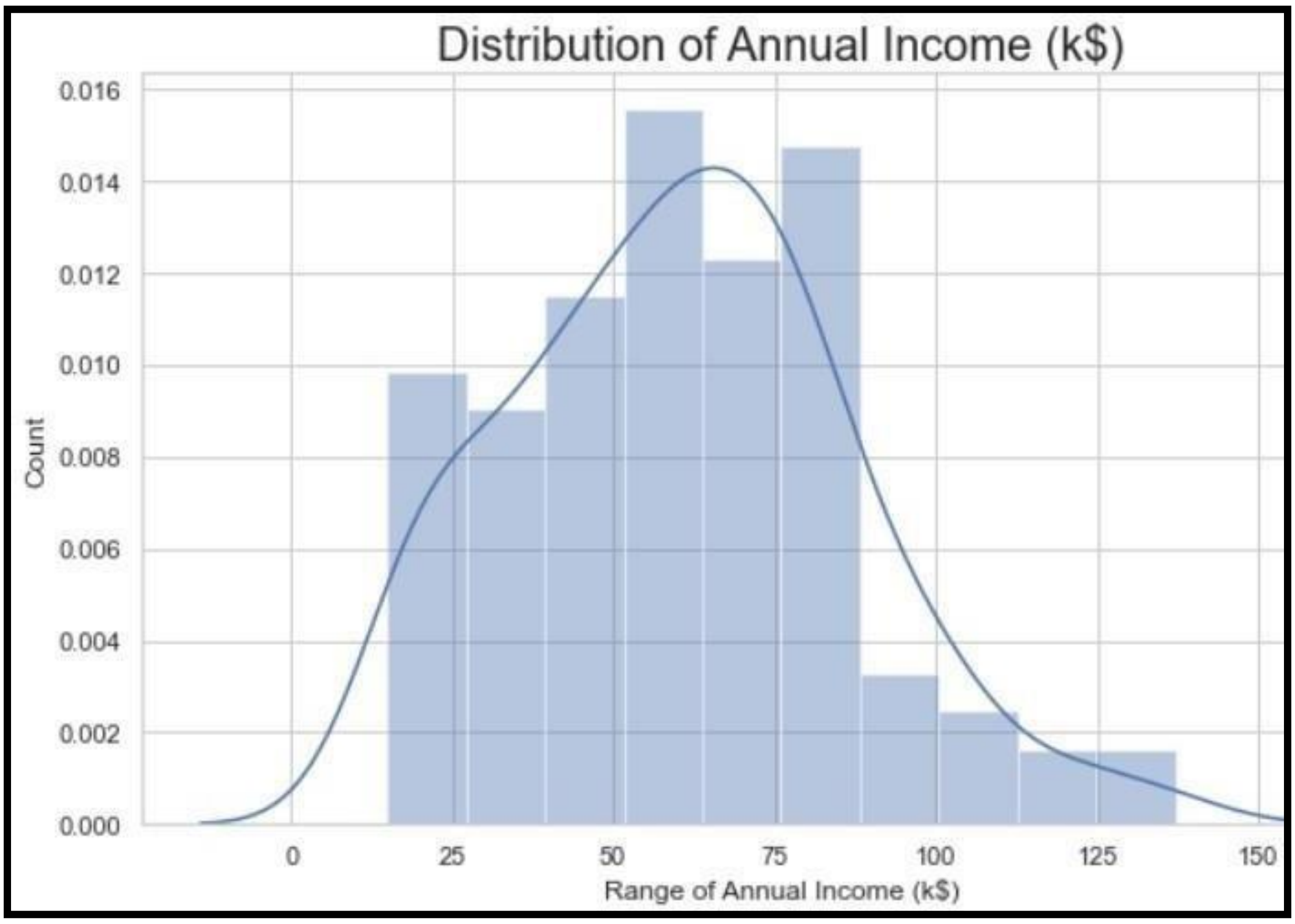


	CustomerID	Age	Annual Income (k$)	Spending Score
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

The data seems to be interesting. Let us look at the data distribution.

Annual Income Distribution:

```
#Distribution of Annnual Income
plt.figure(figsize=(10, 6)) sns.set(style = 'whitegrid')
sns.distplot(data['Annual Income (k$)'])
plt.title('Distribution of Annual Income (k$)', fontsize = 20)plt.xlabel('Range of
Annual Income (k$)')
plt.ylabel('Count')
```

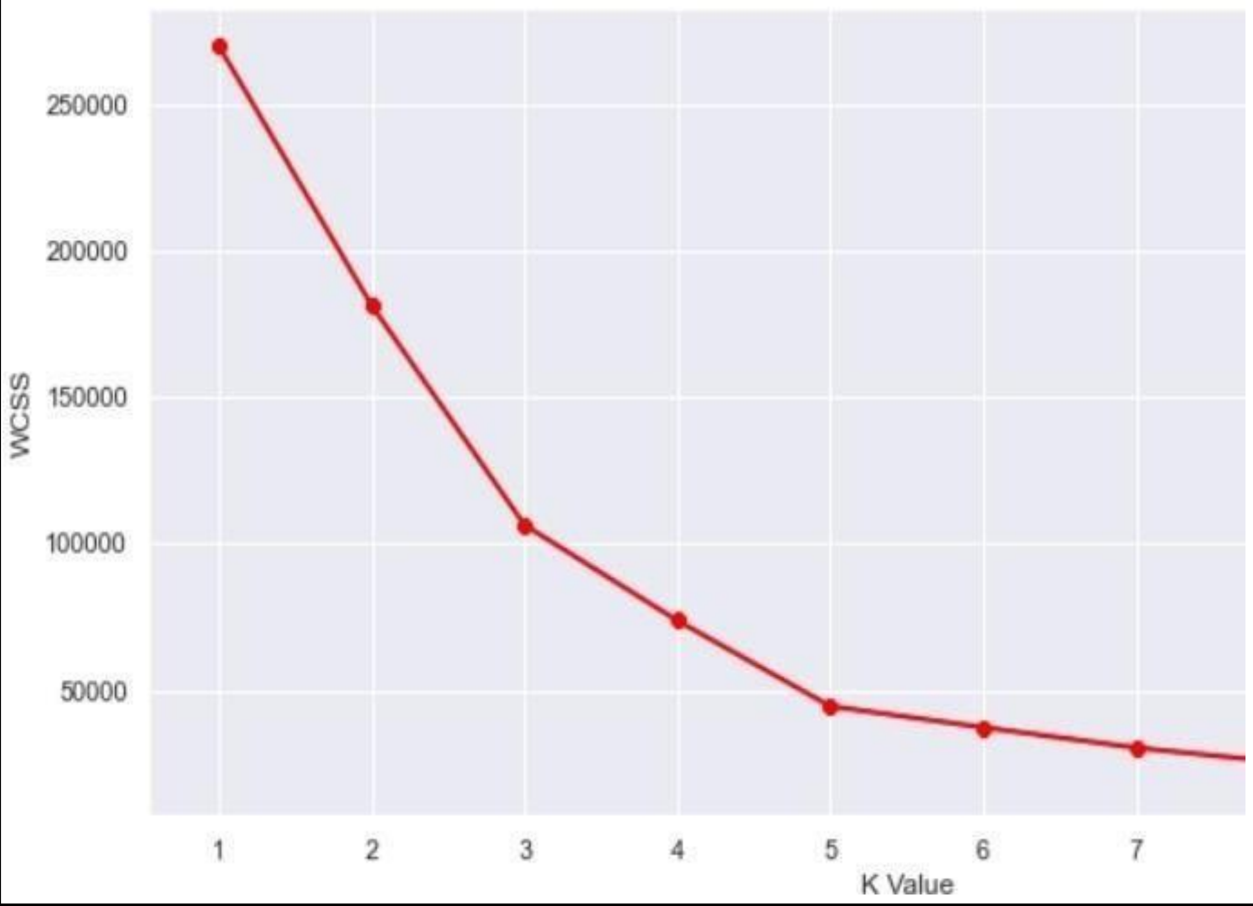


```
#Importing KMeans from sklearn from
sklearn.cluster import KMeans
```

Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k. Next, we choose the k for which WSS first starts to diminish. This value of K gives us the best number of clusters to make from the raw data.

```
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)km.fit(X)
    wcss.append(km.inertia_)
#The elbow curve plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS") plt.show()
```

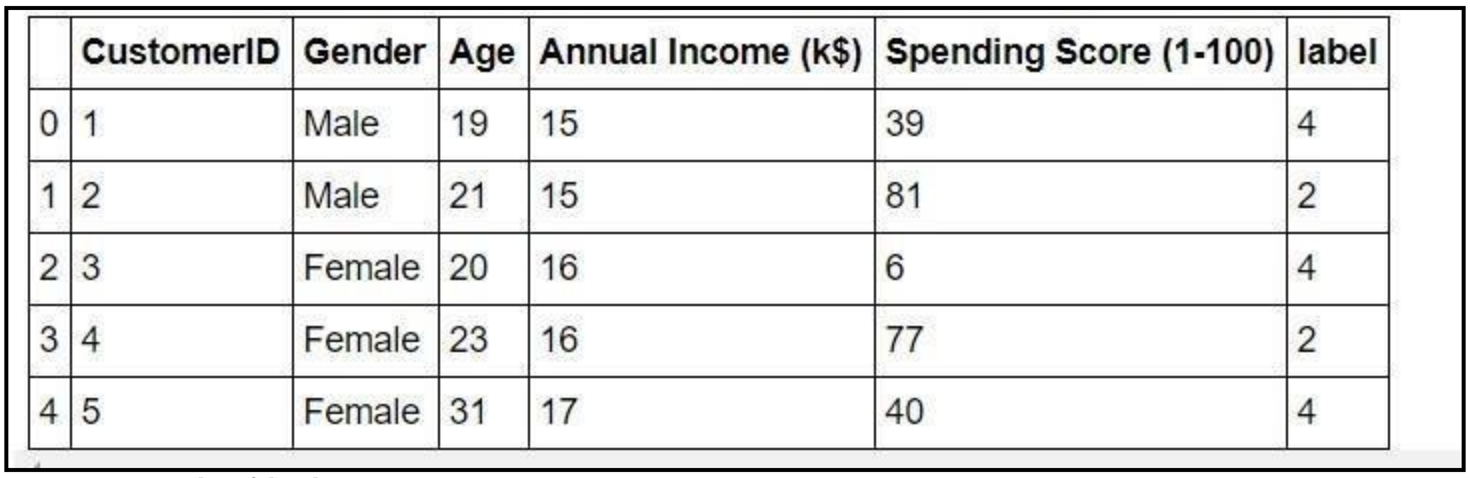
The plot:



This is known as the elbow graph, the x-axis being the number of clusters, the number of clusters is taken at the elbow joint point. This point is the point where making clusters is most relevant as here the value of WCSS suddenly stops decreasing. Here in the graph, after 5 the drop is minimal, so we take 5 to be the number of clusters.

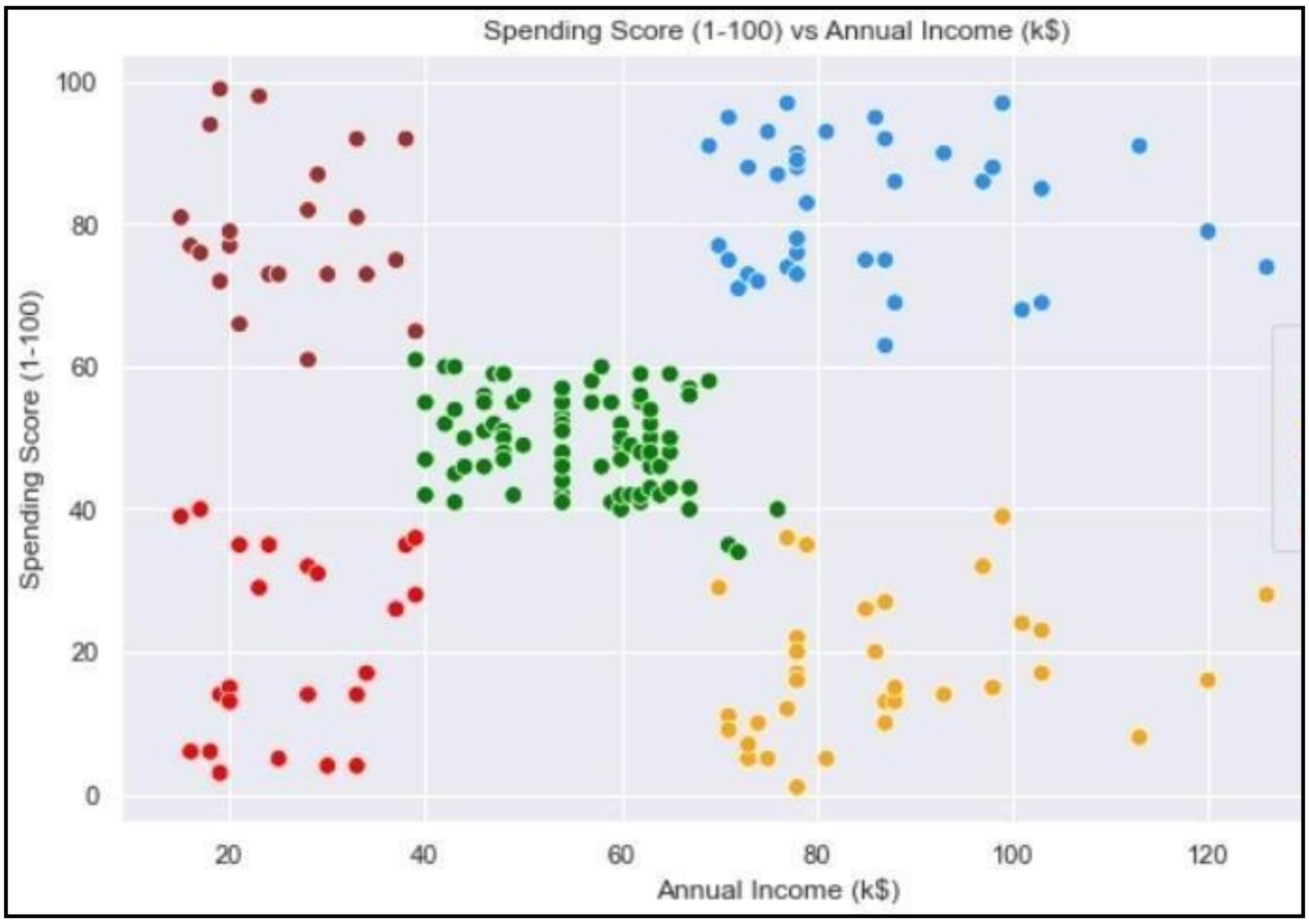
```
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df1["label"] = y
#The new dataframe with the clustering done
df1.head()
```

The labels added to the data.



	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	Male	19	15	39	4
1	2	Male	21	15	81	2
2	3	Female	20	16	6	4
3	4	Female	23	16	77	2
4	5	Female	31	17	40	4

```
#Scatterplot of the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
                        palette=['green','orange','brown','dodgerblue','red'],legend='full',data = df1   ,s = 60 )
plt.xlabel('Annual Income (k$)') plt.ylabel('Spending
Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')plt.show()
```



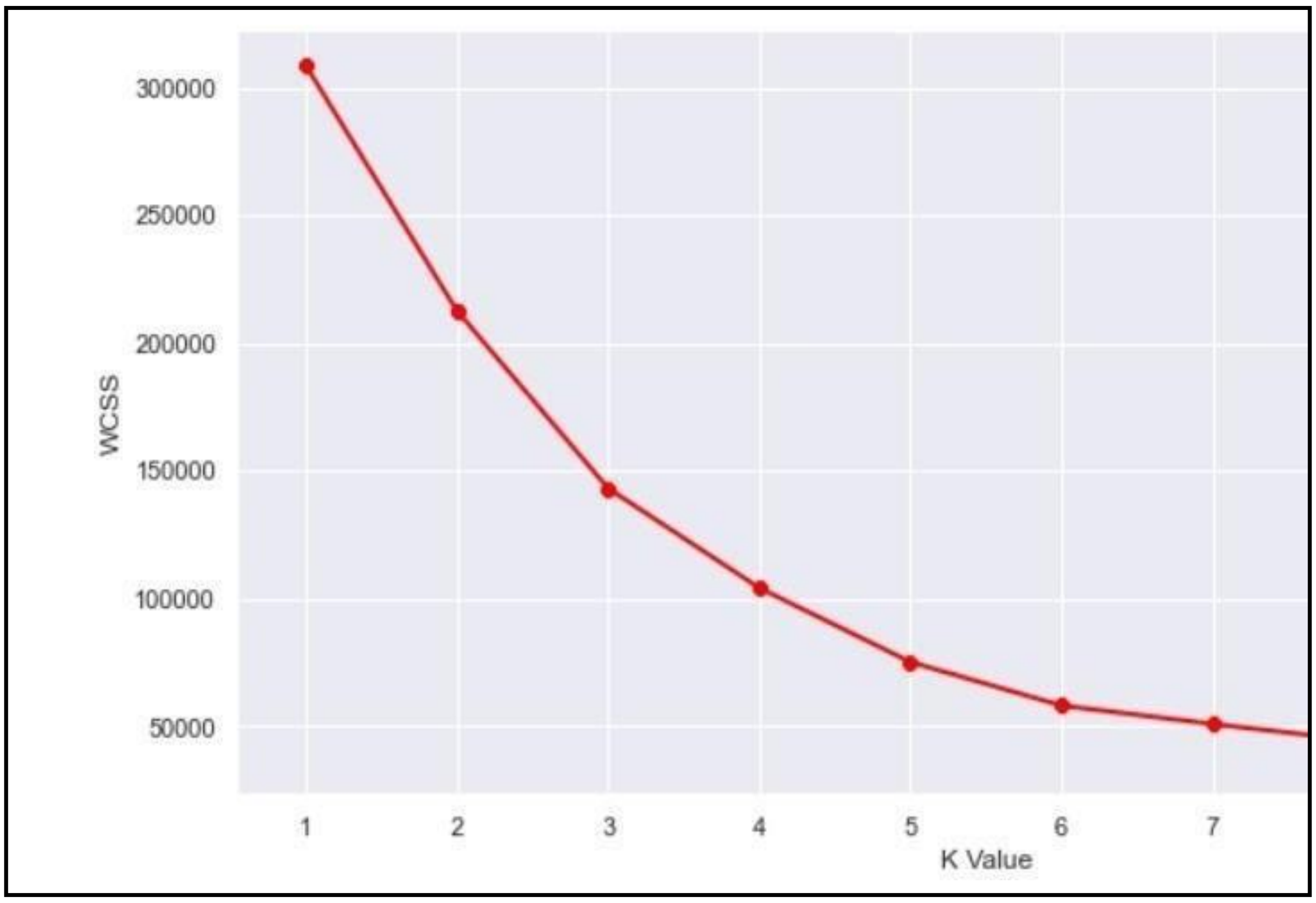
We can clearly see that 5 different clusters have been formed from the data. The red cluster is the customers with the least income and least spending score, similarly, the blue cluster is the customers with the most income and most spending score.

k-Means Clustering on the basis of 3D data

Now, we shall be working on 3 types of data. Apart from the spending score and annual income of customers, we shall also take in the age of the customers.

```
#Taking the features
X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]
#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values ofk.
wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS") plt.show()
```

The WCSS curve.


WCSS
K Value