RINEX

MAJOR PROJECTS

Submitted by,

THEJASWINI J N

M.Sc. [Mathematics]

(completed)

JSS COLLEGE OF ARTS,COMMERCE

AND SCIENCE, OOTY ROAD,

MYSORE.

CONTENT

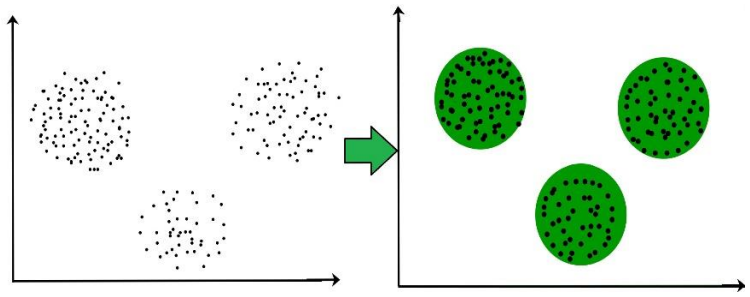
1. K-Means Clustering

Clustering:

The task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex–

The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.



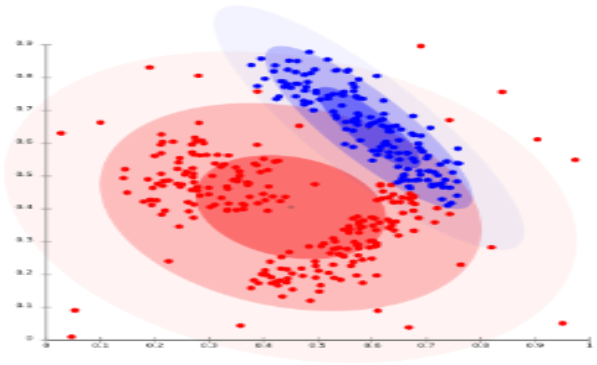
Clustering Algorithms:

The clustering Algorithms are of many types. The following overview will only list the most prominent examples of clustering algorithms,

Distribution based methods:

It is a clustering model in which we will fit the data on the probability that how it may belong to the same distribution. The grouping done may be *normal or gaussian*. Gaussian distribution is more prominent where we have a fixed number of distributions and all the upcoming data is fitted into it such that the distribution of data may get maximized.

This result in grouping which is shown in the figure:-



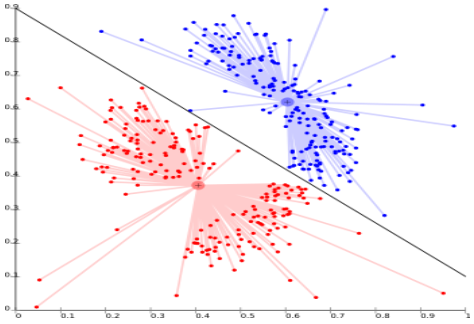
This model works well on synthetic data and diversely sized clusters. But this model may have problems if the constraints are not used to limit the model's complexity. Furthermore, Distribution-based clustering produces clusters that assume concisely defined mathematical models underlying the data, a rather strong assumption for some data distributions.

For Ex- The expectation-maximization *algorithm* which uses multivariate normal distributions is one of the popular examples of this algorithm.

Centroid based methods:

This is basically one of the iterative clustering algorithms in which the clusters are formed by the closeness of data points to the centroid of clusters. Here, the cluster center. i.e., centroid is formed such that the distance of data points is minimum with the center. This problem is basically one of the NP-Hard problems and thus solutions are commonly approximated over a number of trials.

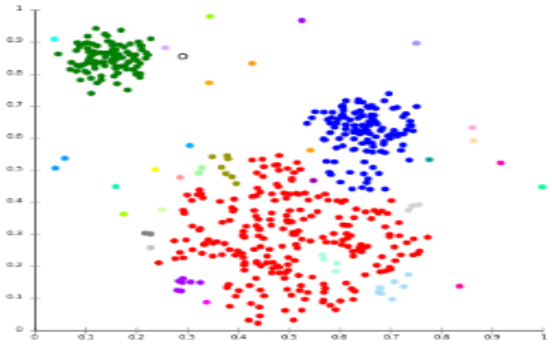
For Ex- K – means algorithm is one of the popular examples of this algorithm.



The biggest problem with this algorithm is that we need to specify K in advance. It also has problems in clustering density-based distributions.

Connectivity based methods:

The core idea of the connectivity-based model is similar to Centroid based model which is basically defining clusters on the basis of the closeness of data points. Here we work on a notion that the data points which are closer have similar behaviour as compared to data points that are farther. It is not a single partitioning of the data set, instead, it provides an extensive hierarchy of clusters that merge with each other at certain distances. Here the choice of distance function is subjective. These models are very easy to interpret but it lacks scalability.

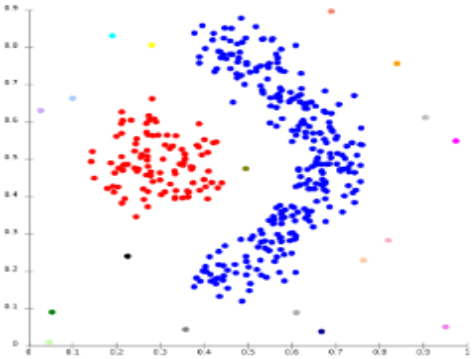


For Ex- hierarchical algorithm and its variants.

Density Models:

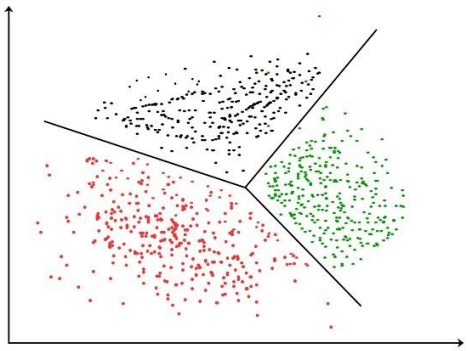
In this clustering model, there will be searching of data space for areas of the varied density of data points in the data space. It isolates various density regions based on different densities present in the data space.

For Ex-DBSCAN and OPTICS.



Here I’m using, K-means clustering algorithm

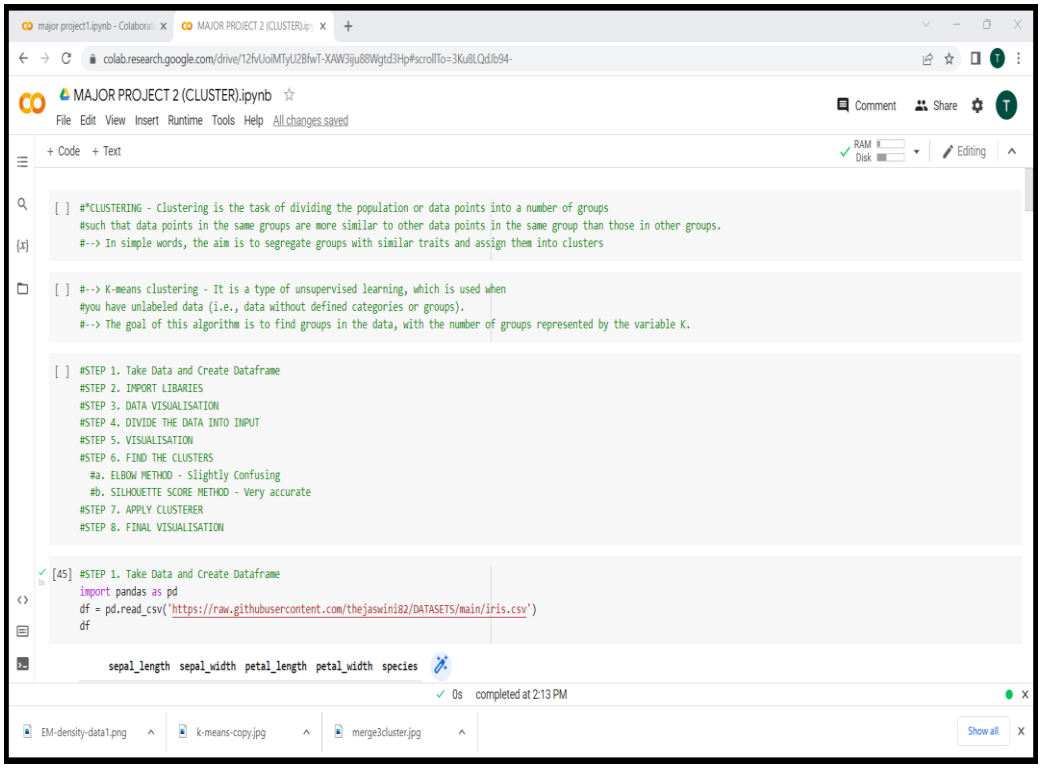
It is the simplest unsupervised learning algorithm that solves clustering problem. K-means algorithm partitions n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster.

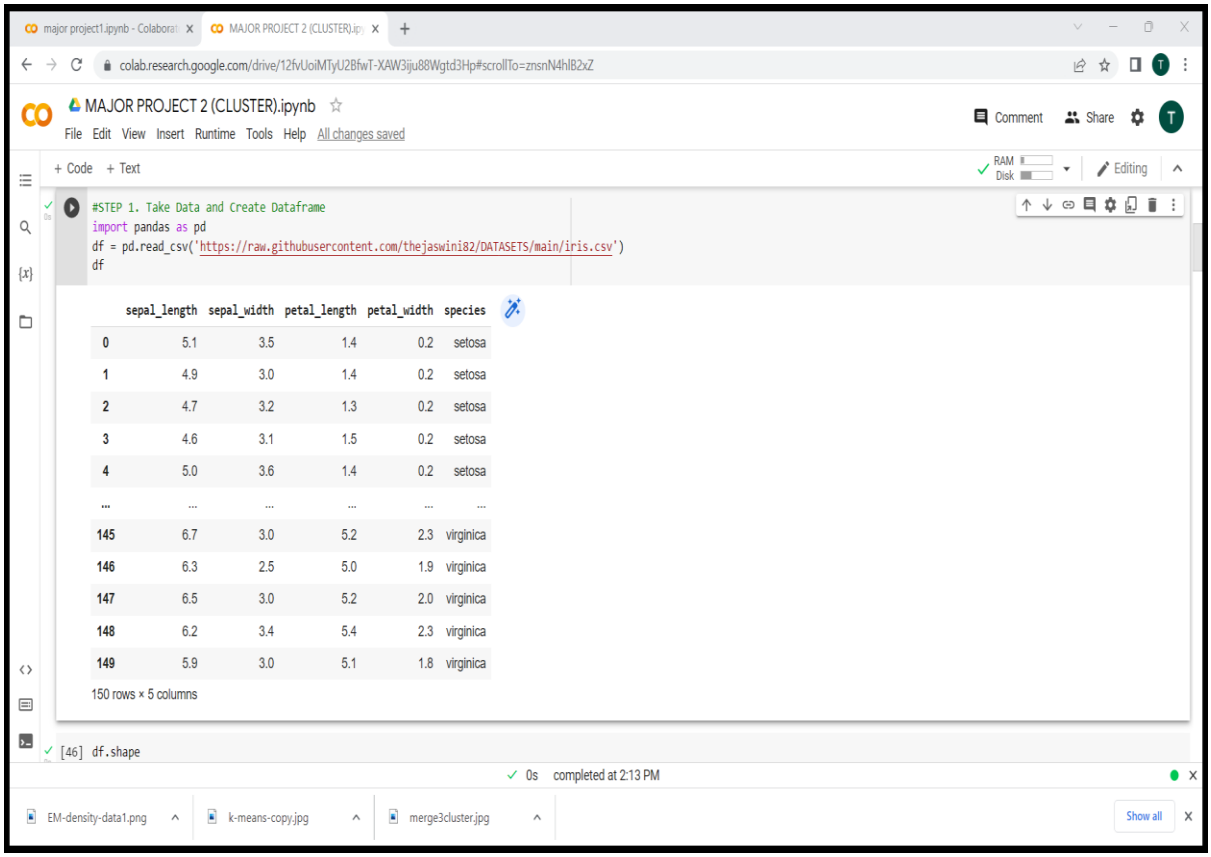


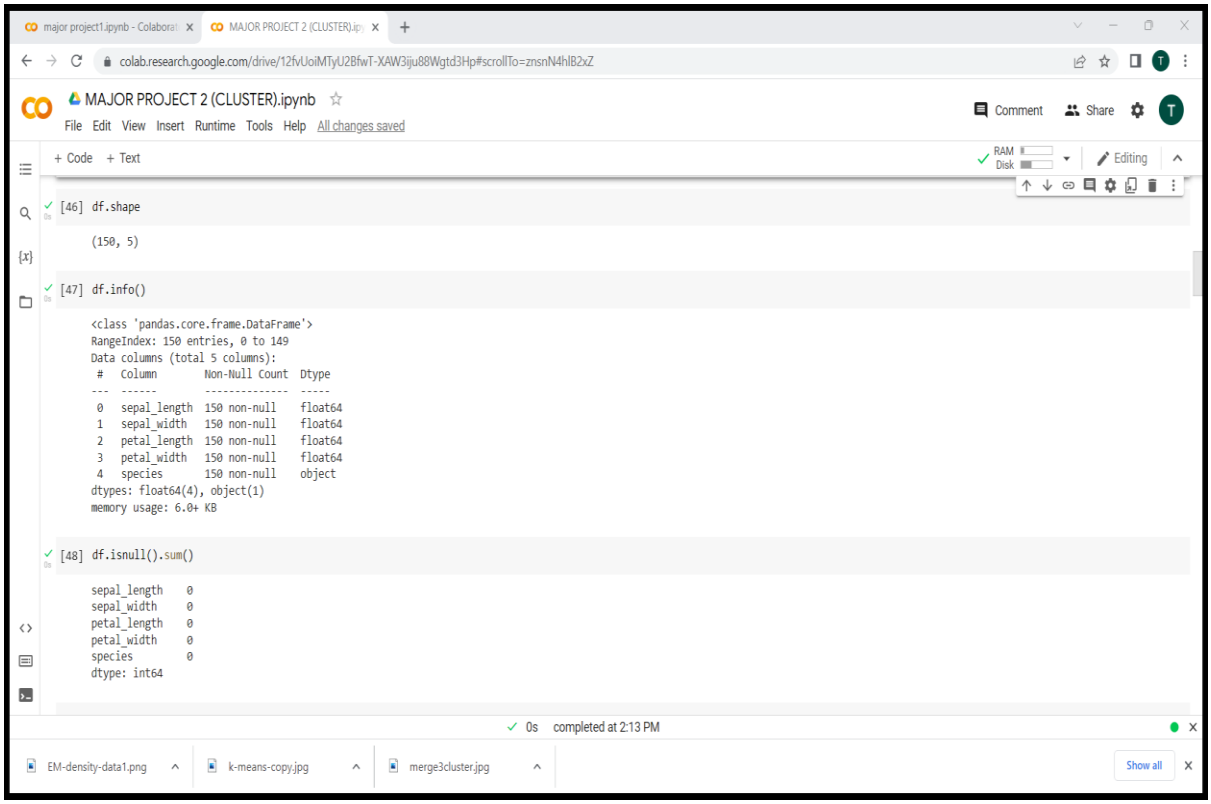
Applications of Clustering in different fields

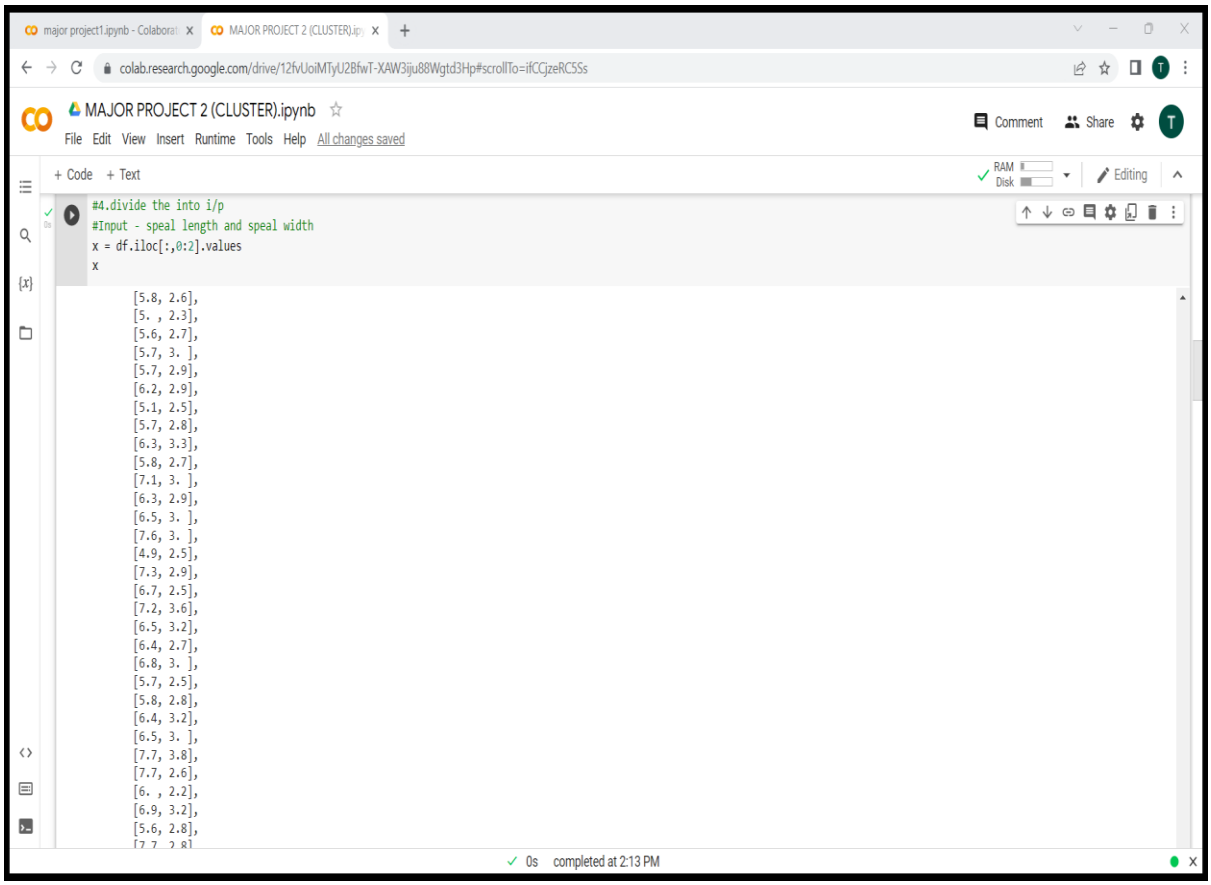
- **Marketing:** It can be used to characterize & discover customer segments for marketing purposes.
- **Biology:** It can be used for classification among different species of plants and animals.
- **Libraries:** It is used in clustering different books on the basis of topics and information.
- **Insurance:** It is used to acknowledge the customers, their policies and identifying the frauds.

SCREENSHOT:


major project1.ipynb - Colabora
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=3Ku8LQdJb94-
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[ ] #*CLUSTERING - Clustering is the task of dividing the population or data points into a number of groups
#such that data points in the same groups are more similar to other data points in the same group than those in other groups.
#--> In simple words, the aim is to segregate groups with similar traits and assign them into clusters
[ ] #--> K-means clustering - It is a type of unsupervised learning, which is used when
#you have unlabeled data (i.e., data without defined categories or groups).
#--> The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K.
[ ] #STEP 1. Take Data and Create Dataframe
#STEP 2. IMPORT LIBARIES
#STEP 3. DATA VISUALISATION
#STEP 4. DIVIDE THE DATA INTO INPUT
#STEP 5. VISUALISATION
#STEP 6. FIND THE CLUSTERS
#a. ELBOW METHOD - Slightly Confusing
#b. SILHOUETTE SCORE METHOD - Very accurate
#STEP 7. APPLY CLUSTERER
#STEP 8. FINAL VISUALISATION
[45] #STEP 1. Take Data and Create Dataframe
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/thejaswini82/DATASETS/main/iris.csv')
df
sepal_length sepal_width petal_length petal_width species
0s completed at 2:13 PM
EM-density-data1.png
k-means-copy.jpg
merge3cluster.jpg
Show all


major project1.ipynb - Colabora
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=znsnN4hlB2xZ
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
#STEP 1. Take Data and Create Dataframe
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/thejaswini82/DATASETS/main/iris.csv')
df
sepal_length sepal_width petal_length petal_width species
0 5.1 3.5 1.4 0.2 setosa
1 4.9 3.0 1.4 0.2 setosa
2 4.7 3.2 1.3 0.2 setosa
3 4.6 3.1 1.5 0.2 setosa
4 5.0 3.6 1.4 0.2 setosa
... ...
145 6.7 3.0 5.2 2.3 virginica
146 6.3 2.5 5.0 1.9 virginica
147 6.5 3.0 5.2 2.0 virginica
148 6.2 3.4 5.4 2.3 virginica
149 5.9 3.0 5.1 1.8 virginica
150 rows × 5 columns
[46] df.shape
0s completed at 2:13 PM
EM-density-data1.png
k-means-copy.jpg
merge3cluster.jpg
Show all


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=znsnN4hlB2xZ
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[46] df.shape
(150, 5)
[47] df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 sepal_length 150 non-null float64
1 sepal_width 150 non-null float64
2 petal_length 150 non-null float64
3 petal_width 150 non-null float64
4 species 150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
[48] df.isnull().sum()
sepal_length 0
sepal_width 0
petal_length 0
petal_width 0
species 0
dtype: int64
0s completed at 2:13 PM
EM-density-data1.png
k-means-copy.jpg
merge3cluster.jpg
Show all


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=ifCCjzeRC5Ss
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
#4.divide the into i/p
#Input - speal length and speal width
x = df.iloc[:,0:2].values
x
[5.8, 2.6],
[5. , 2.3],
[5.6, 2.7],
[5.7, 3.],
[5.7, 2.9],
[6.2, 2.9],
[5.1, 2.5],
[5.7, 2.8],
[6.3, 3.3],
[5.8, 2.7],
[7.1, 3.],
[6.3, 2.9],
[6.5, 3.],
[7.6, 3.],
[4.9, 2.5],
[7.3, 2.9],
[6.7, 2.5],
[7.2, 3.6],
[6.5, 3.2],
[6.4, 2.7],
[6.8, 3.],
[5.7, 2.5],
[5.8, 2.8],
[6.4, 3.2],
[6.5, 3.],
[7.7, 3.8],
[7.7, 2.6],
[6. , 2.2],
[6.9, 3.2],
[5.6, 2.8],
0s completed at 2:13 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=IqYa_jj9OWgk

MAJOR PROJECT 2 (CLUSTER).ipynb

File Edit View Insert Runtime Tools Help All changes saved

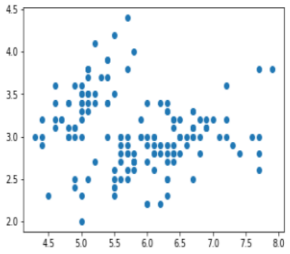
Comment Share

+ Code + Text

RAM Disk Editing

```
[53] #STEP 5. VISUALISATION
     import matplotlib.pyplot as plt
     plt.scatter(df['sepal_length'],df['sepal_width'])
```

```
<matplotlib.collections.PathCollection at 0x7fab98ecdfd0>
```




```
[54] #-->Here we have got only one cluster before applying any clustering technique
     #-->Here our main task is to find out the number of clusters(k)
```

```
[56] import numpy as np
     np.sqrt(150)
```

```
12.24744871391589
```

```
#-->150 is the total no of points
```

0s completed at 2:13 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=2gPSLq9KawBJ

MAJOR PROJECT 2 (CLUSTER).ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

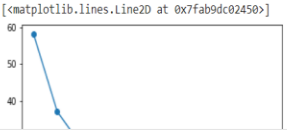
RAM Disk Editing

```
[57] #-->150 is the total no of points
     #-->No of cluster - k
     #-->k value should not exceed the square root of the total no of points
     #-->Hence k value should be in the range of 2 to 13
```

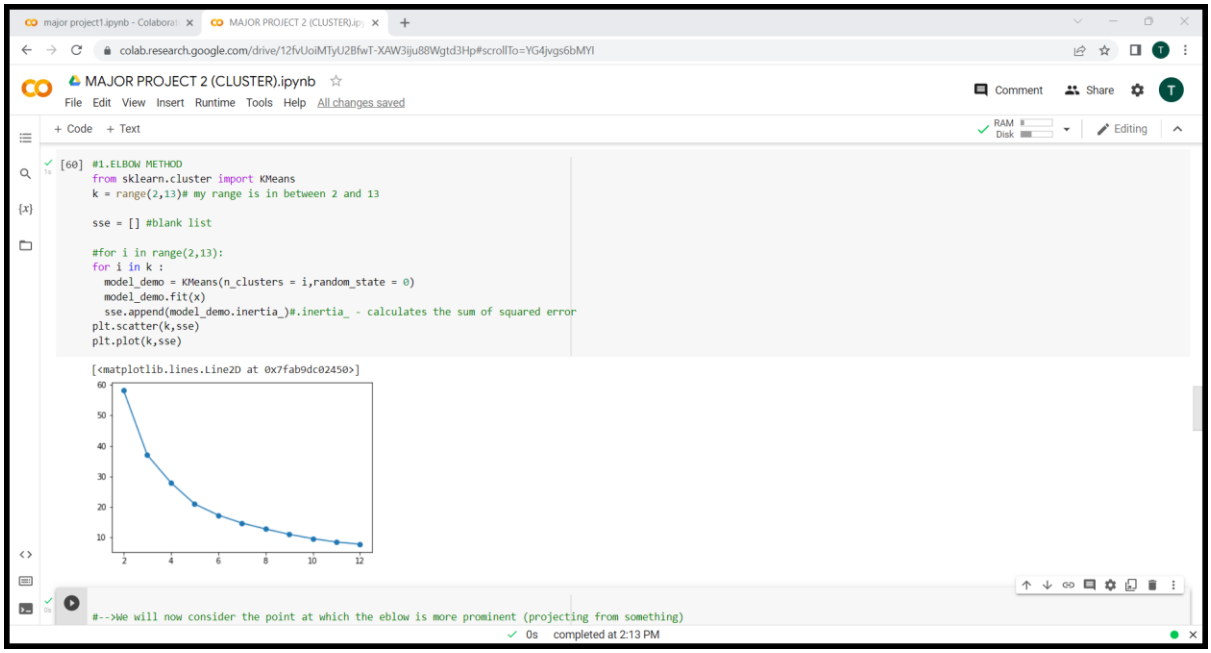
```
#STEP.6
 #1.ELBOW METHOD - Slightly Confusing
 #2.SILHOUETTE SCORE METHOD - Very accurate
```

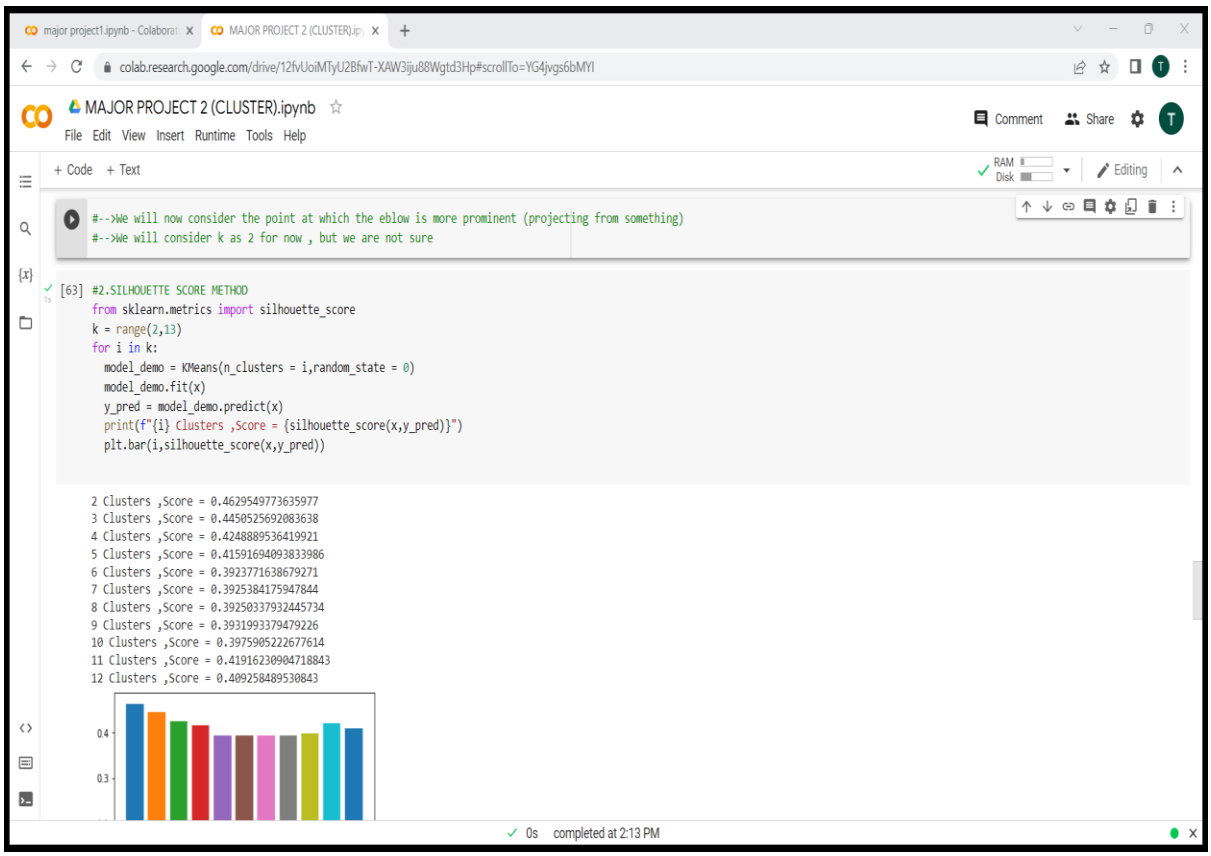
```
[60] #1.ELBOW METHOD
     from sklearn.cluster import KMeans
     k = range(2,13)# my range is in between 2 and 13

     sse = [] #blank list

     #for i in range(2,13):
     for i in k :
       model_demo = KMeans(n_clusters = i,random_state = 0)
       model_demo.fit(x)
       sse.append(model_demo.inertia_)#.inertia_ - calculates the sum of squared error
     plt.scatter(k,sse)
     plt.plot(k,sse)
```

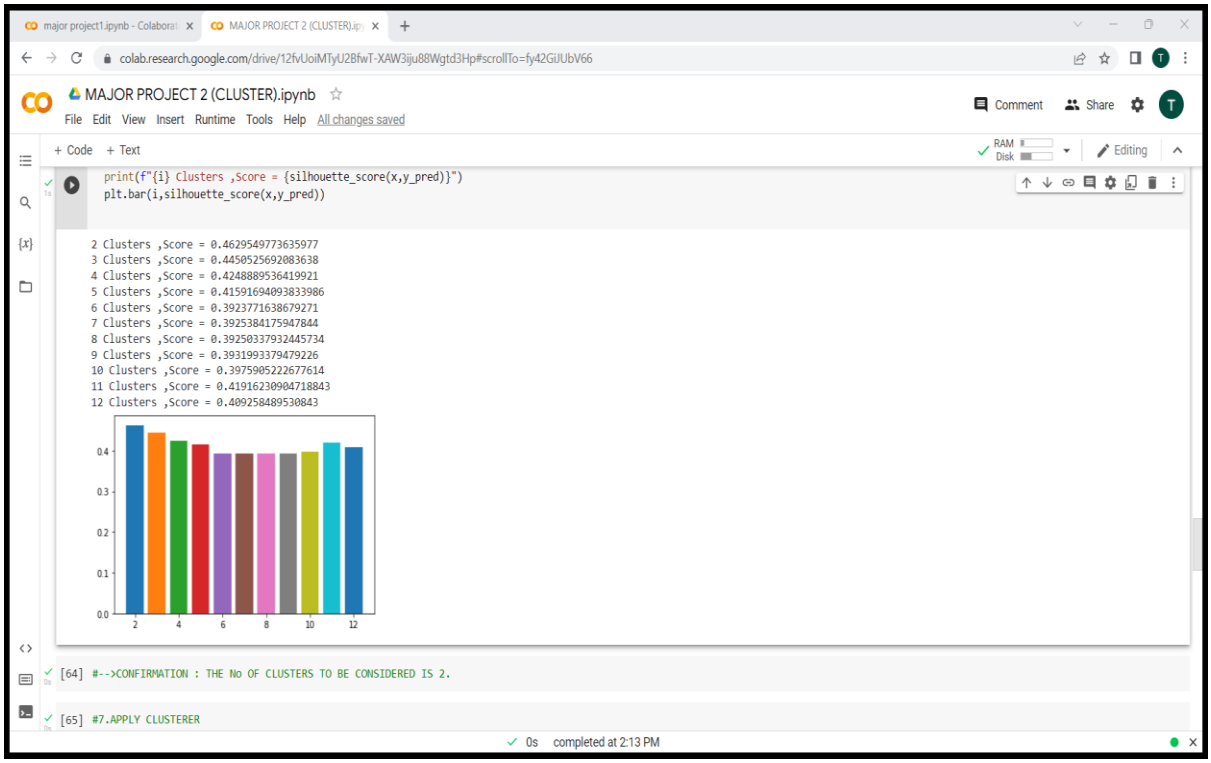
```
[<matplotlib.lines.Line2D at 0x7fab9dc02450>]
```

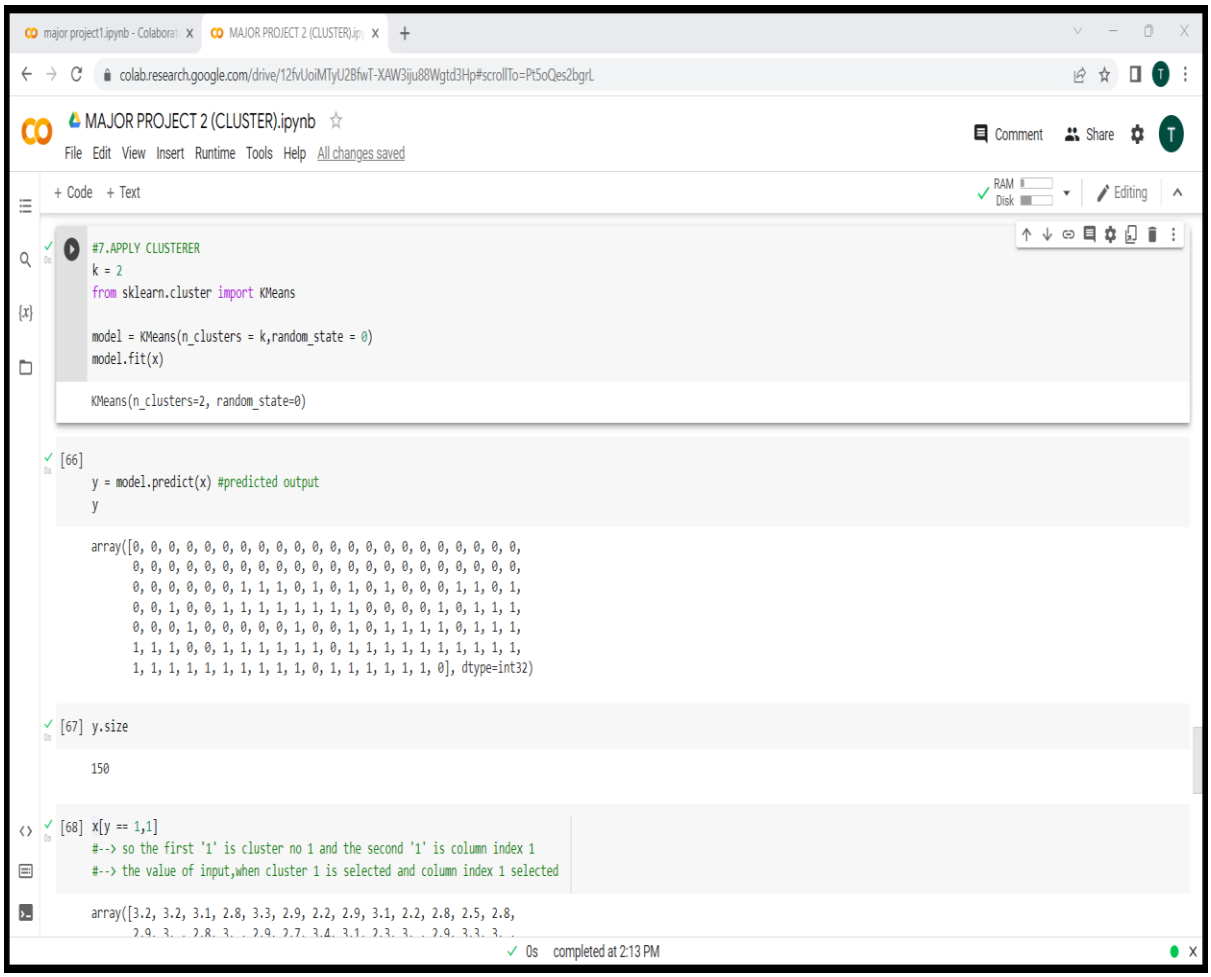


0s completed at 2:13 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=YG4jvgs6bMYI
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[60] #1.ELBOW METHOD
from sklearn.cluster import KMeans
k = range(2,13)# my range is in between 2 and 13
sse = [] #blank list
#for i in range(2,13):
for i in k :
model_demo = KMeans(n_clusters = i,random_state = 0)
model_demo.fit(x)
sse.append(model_demo.inertia_)#.inertia_ - calculates the sum of squared error
plt.scatter(k,sse)
plt.plot(k,sse)
[<matplotlib.lines.Line2D at 0x7fab9dc02450>]
60
50
40
30
20
10
2
4
6
8
10
12
#-->We will now consider the point at which the eblow is more prominent (projecting from something)
0s completed at 2:13 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=YG4jvgs6bMYI
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Editing
#-->We will now consider the point at which the eblow is more prominent (projecting from something)
#-->We will consider k as 2 for now , but we are not sure
[63] #2.SILHOUETTE SCORE METHOD
from sklearn.metrics import silhouette_score
k = range(2,13)
for i in k:
model_demo = KMeans(n_clusters = i,random_state = 0)
model_demo.fit(x)
y_pred = model_demo.predict(x)
print(f"{i} Clusters ,Score = {silhouette_score(x,y_pred)}")
plt.bar(i,silhouette_score(x,y_pred))
2 Clusters ,Score = 0.4629549773635977
3 Clusters ,Score = 0.4450525692083638
4 Clusters ,Score = 0.4248889536419921
5 Clusters ,Score = 0.41591694093833986
6 Clusters ,Score = 0.3923771638679271
7 Clusters ,Score = 0.3925384175947844
8 Clusters ,Score = 0.39250337932445734
9 Clusters ,Score = 0.3931993379479226
10 Clusters ,Score = 0.3975905222677614
11 Clusters ,Score = 0.41916230904718843
12 Clusters ,Score = 0.409258489530843
0.4
0.3
0s completed at 2:13 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=fy42GiJUbV66
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
print(f"{i} Clusters ,Score = {silhouette_score(x,y_pred)}")
plt.bar(i,silhouette_score(x,y_pred))
2 Clusters ,Score = 0.4629549773635977
3 Clusters ,Score = 0.4450525692083638
4 Clusters ,Score = 0.4248889536419921
5 Clusters ,Score = 0.41591694093833986
6 Clusters ,Score = 0.3923771638679271
7 Clusters ,Score = 0.3925384175947844
8 Clusters ,Score = 0.39250337932445734
9 Clusters ,Score = 0.3931993379479226
10 Clusters ,Score = 0.3975905222677614
11 Clusters ,Score = 0.41916230904718843
12 Clusters ,Score = 0.409258489530843
0.4
0.3
0.2
0.1
0.0
2
4
6
8
10
12
[64] #-->CONFIRMATION : THE No OF CLUSTERS TO BE CONSIDERED IS 2.
[65] #7.APPLY CLUSTERER
0s completed at 2:13 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=Pt5oQes2bgrL
MAJOR PROJECT 2 (CLUSTER).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
#7.APPLY CLUSTERER
k = 2
from sklearn.cluster import KMeans
model = KMeans(n_clusters = k,random_state = 0)
model.fit(x)
KMeans(n_clusters=2, random_state=0)
[66]
y = model.predict(x) #predicted output
y
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0], dtype=int32)
[67] y.size
150
[68] x[y == 1,1]
#--> so the first '1' is cluster no 1 and the second '1' is column index 1
#--> the value of input,when cluster 1 is selected and column index 1 selected
array([3.2, 3.2, 3.1, 2.8, 3.3, 2.9, 2.2, 2.9, 3.1, 2.2, 2.8, 2.5, 2.8,
0s completed at 2:13 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=ttHC3mXkb090

MAJOR PROJECT 2 (CLUSTER).ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
x[y == 1,1]
#--> so the first '1' is cluster no 1 and the second '1' is column index 1
#--> the value of input,when cluster 1 is selected and column index 1 selected
```

```
array([3.2, 3.2, 3.1, 2.8, 3.3, 2.9, 2.2, 2.9, 3.1, 2.2, 2.8, 2.5, 2.8,
       2.9, 3. , 2.8, 3. , 2.9, 2.7, 3.4, 3.1, 2.3, 3. , 2.9, 3.3, 3. ,
       2.9, 3. , 3. , 2.9, 2.5, 3.6, 3.2, 2.7, 3. , 3.2, 3. , 3.8, 2.6,
       2.2, 3.2, 2.8, 2.7, 3.3, 3.2, 2.8, 3. , 2.8, 3. , 2.8, 3.8, 2.8,
       2.8, 2.6, 3. , 3.4, 3.1, 3. , 3.1, 3.1, 3.1, 3.2, 3.3, 3. , 2.5,
       3. , 3.4])
```

```
[69] np.unique(y,return_counts = True)
```

```
(array([0, 1], dtype=int32), array([83, 67]))
```

```
[70] #STEP 8. FINAL VISUALISATION
     plt.figure(figsize = (10,5))
     for i in range(k):
       plt.scatter(x[y == i,0],x[y == i,1],label = f'Cluster {i}')
     plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s = 300,c = 'yellow',
                 label = 'Centroids')
     plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fab98c625d0>
```

0s completed at 2:13 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/12fvUoiMTyU2BfwT-XAW3iju88Wgtd3Hp#scrollTo=R5zE73x5cDiq

MAJOR PROJECT 2 (CLUSTER).ipynb

File Edit View Insert Runtime Tools Help All changes saved

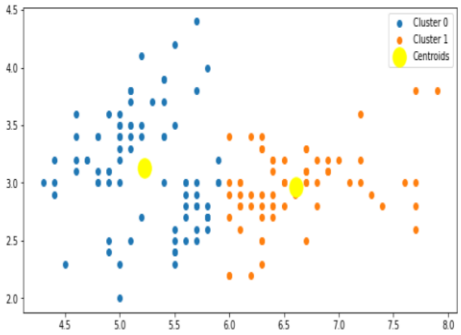
Comment Share

+ Code + Text

RAM Disk Editing

[70] #STEP 8. FINAL VISUALISATION
plt.figure(figsize = (10,5))
for i in range(k):
plt.scatter(x[y == i,0],x[y == i,1],label = f'Cluster {i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s = 300,c = 'yellow',
label = 'Centroids')
plt.legend()

<matplotlib.legend.Legend at 0x7fab98c625d0>




```
#CONCLUSION:
# The results of the K-means clustering algorithm are: The centroids of the K clusters, which can be used to label new data.
# K-means clustering is the unsupervised machine learning algorithm that is part of a much deep pool of data techniques
#and operations in the realm of Data Science. It is the fastest and most efficient algorithm to categorize data points
# into groups even when very little information is available about data.
```

0s completed at 2:13 PM

Conclusion:

K-means clustering is the unsupervised machine learning algorithm that is part of a much deep pool of data techniques and operations in the realm of Data Science. It is the fastest and most efficient algorithm to categorize data points into groups even when very little information is available about data.

GITHUBLINK:

https://raw.githubusercontent.com/thejaswini82/PROJECTS/main/K_MEANS_CLUSTERING.ipynb

2.Logistic regression

Regression model Are used to predict a numerical value.

Example: Predicting the sale price of a house.

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation.

Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.

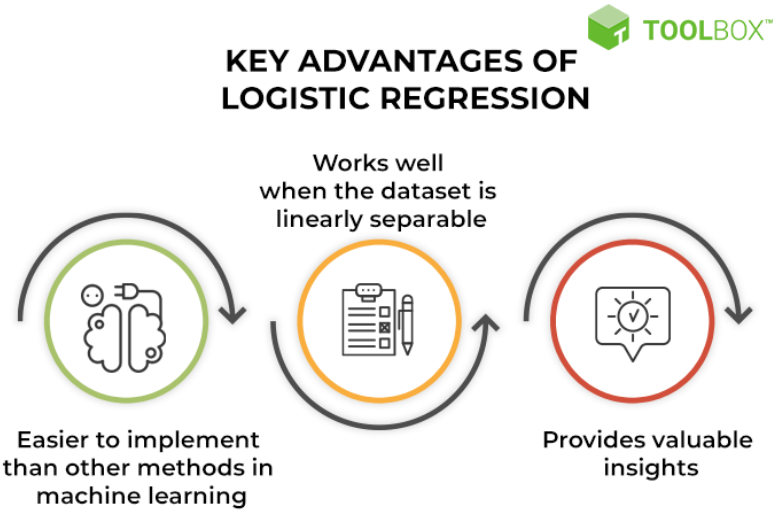
For example, 0 – represents a negative class; 1 – represents a positive class. Logistic regression is commonly used in binary classification problems where the outcome variable reveals either of the two categories (0 and 1).

Some examples of such classifications and instances where the binary response is expected or implied are:

1. **Determine the probability of heart attacks**
2. **Possibility of enrolling into a university**
3. **Identifying spam emails**

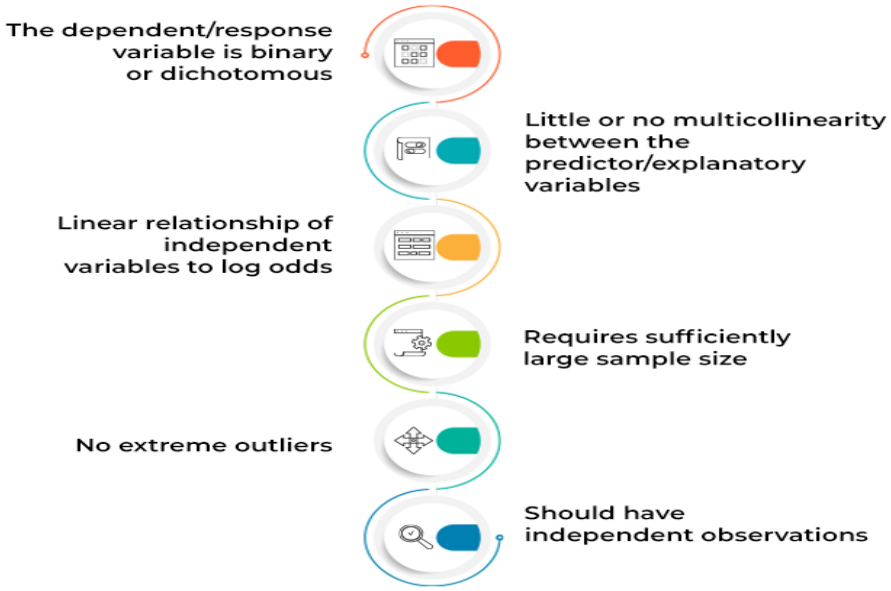
Key advantages of logistic regression

The logistic regression analysis has several advantages in the field of machine learning.



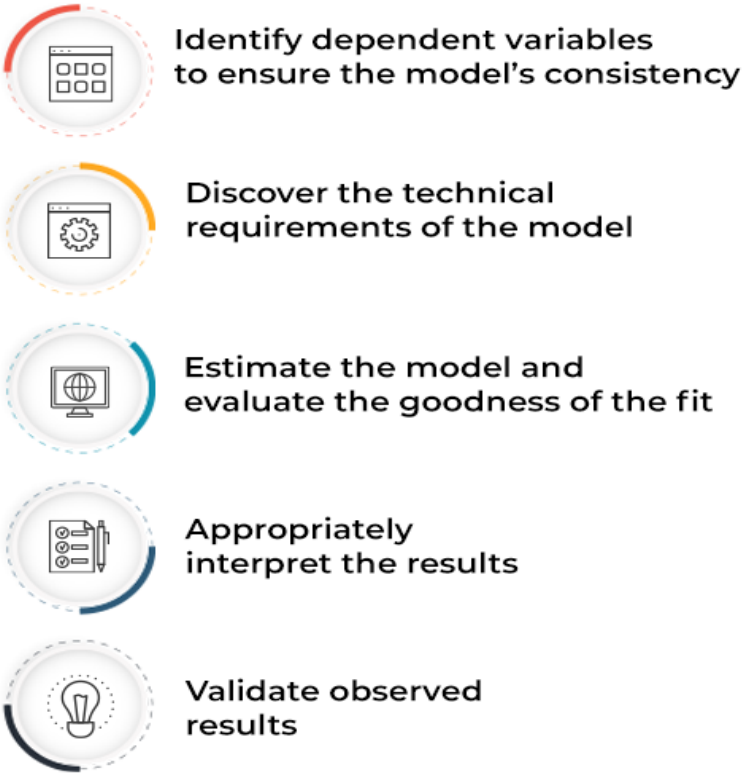
TOOLBOX™

KEY ASSUMPTIONS FOR IMPLEMENTING LOGISTIC REGRESSION



TOOLBOX™

LOGISTIC REGRESSION BEST PRACTICES

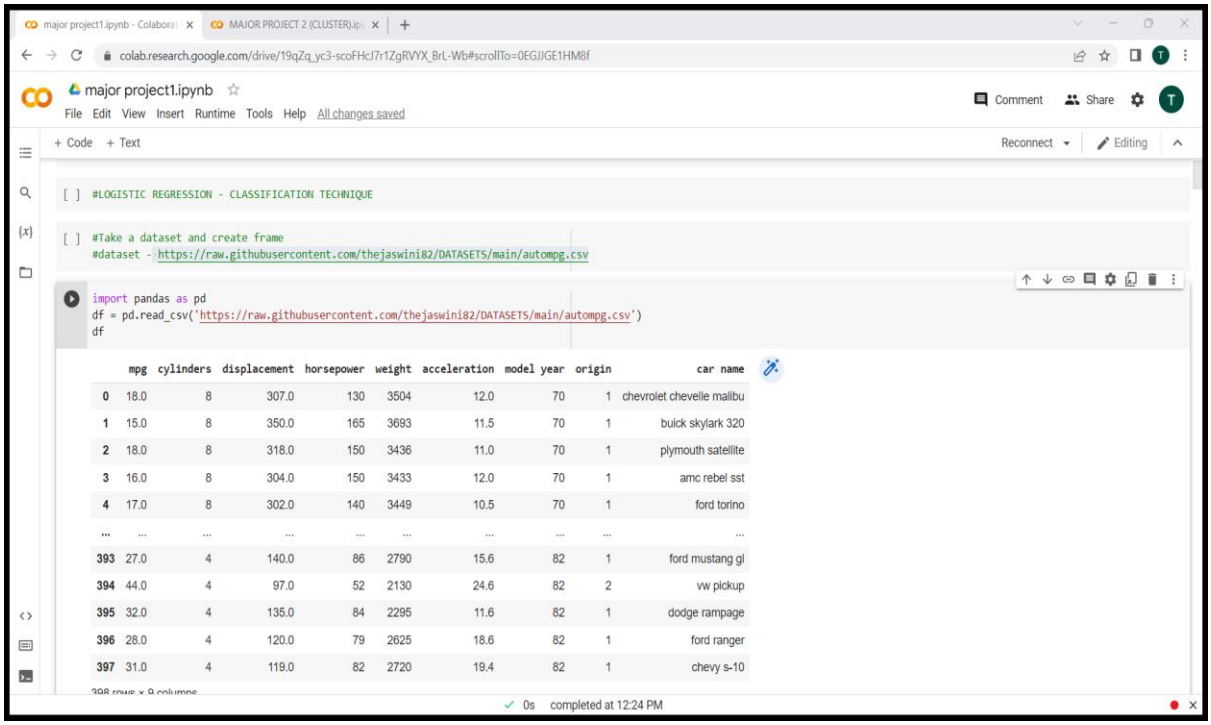


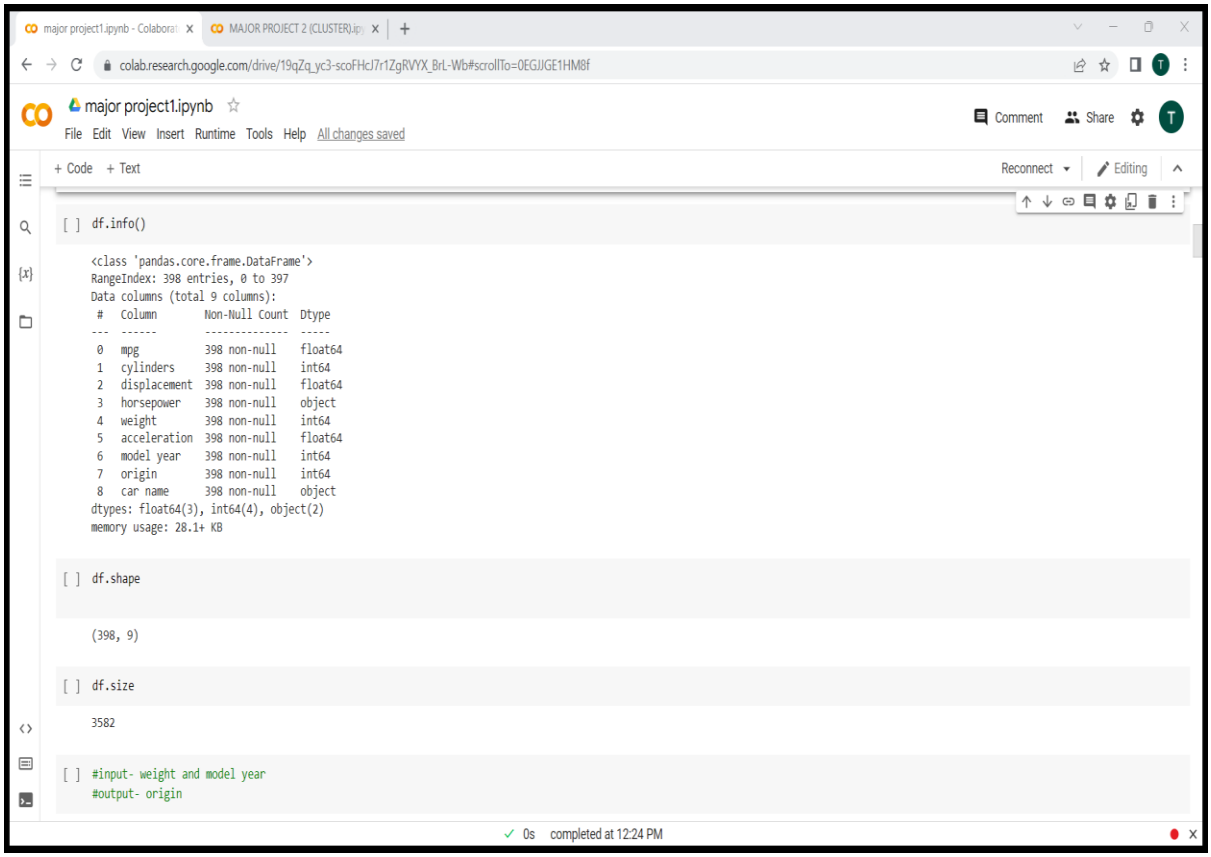
HERE IS ONE LOGISTIC REGRESSION

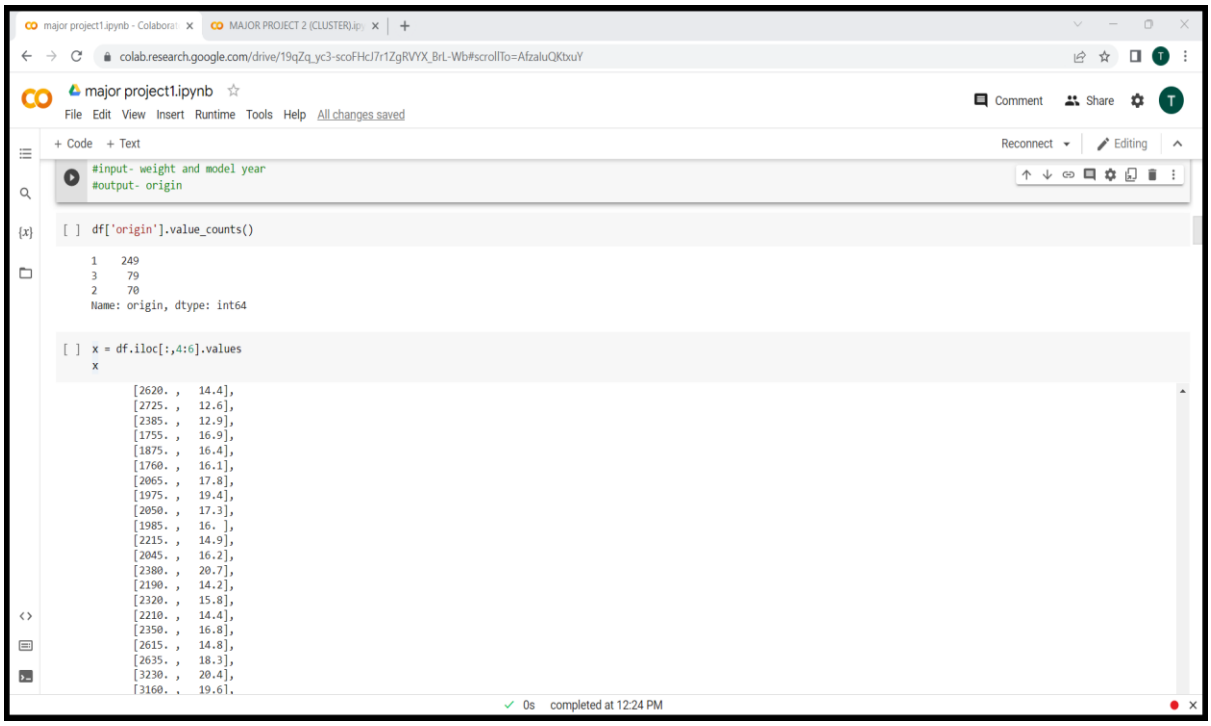
We have taken some Dataset

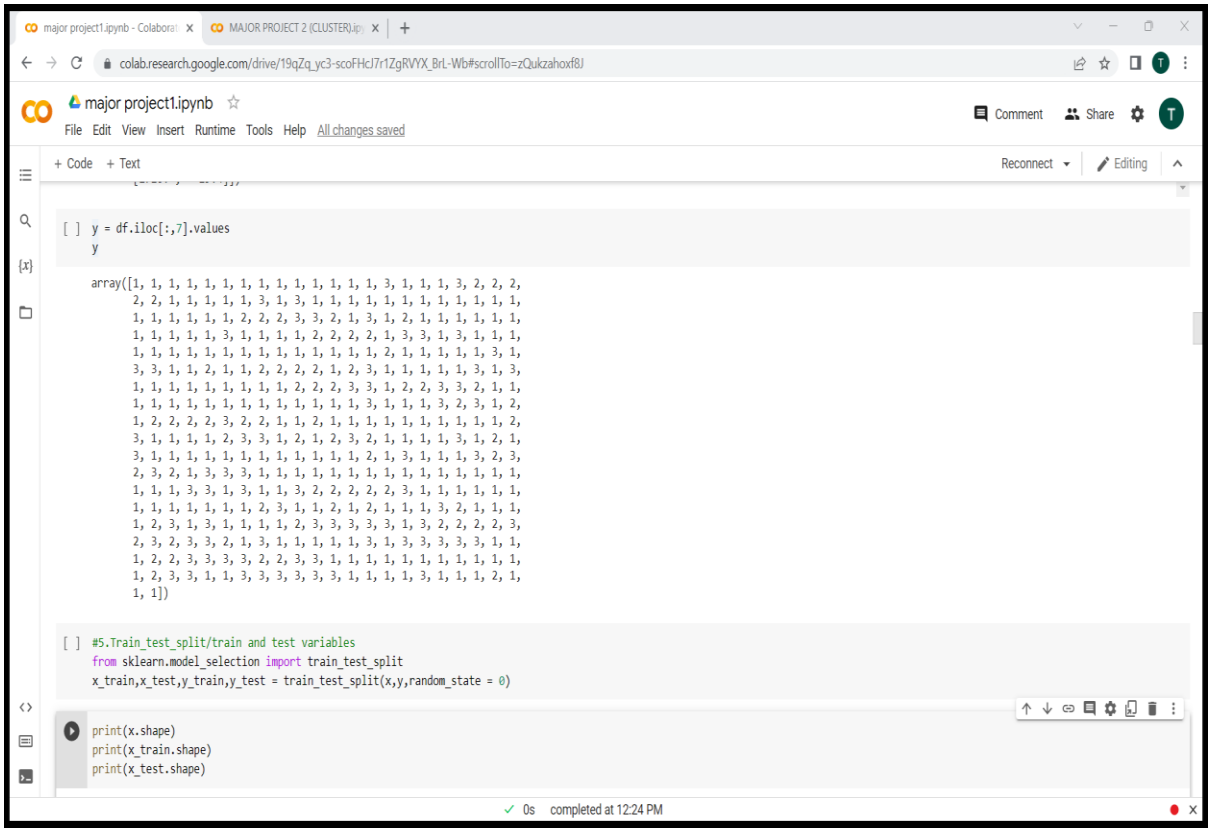
Datasource:
https://raw.githubusercontent.com/thejaswini82/DATASETS/main/autompg.csv

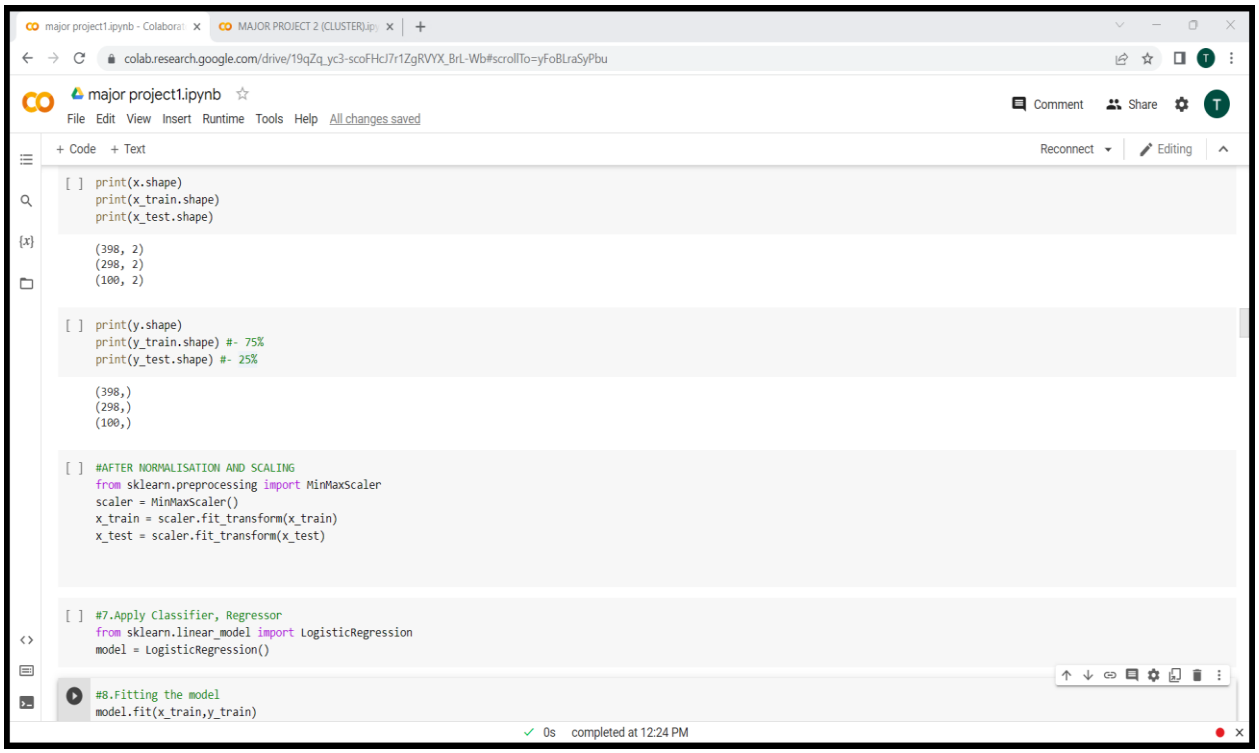
SCREENSHOT:




major project1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
[ ] df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
Column Non-Null Count Dtype
0 mpg 398 non-null float64
1 cylinders 398 non-null int64
2 displacement 398 non-null float64
3 horsepower 398 non-null object
4 weight 398 non-null int64
5 acceleration 398 non-null float64
6 model year 398 non-null int64
7 origin 398 non-null int64
8 car name 398 non-null object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
[ ] df.shape
(398, 9)
[ ] df.size
3582
[ ] #input- weight and model year
#output- origin
0s completed at 12:24 PM


major project1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
#input- weight and model year
#output- origin
[ ] df['origin'].value_counts()
1 249
3 79
2 70
Name: origin, dtype: int64
[ ] x = df.iloc[:,4:6].values
x
[2620. , 14.4],
[2725. , 12.6],
[2385. , 12.9],
[1755. , 16.9],
[1875. , 16.4],
[1760. , 16.1],
[2065. , 17.8],
[1975. , 19.4],
[2050. , 17.3],
[1985. , 16.],
[2215. , 14.9],
[2045. , 16.2],
[2380. , 20.7],
[2190. , 14.2],
[2320. , 15.8],
[2210. , 14.4],
[2350. , 16.8],
[2615. , 14.8],
[2635. , 18.3],
[3230. , 20.4],
[3160. , 19.6],
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=zQukzahoxf8J
major project1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
[ ] y = df.iloc[:,7].values
y
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 2, 2, 2,
2, 2, 1, 1, 1, 1, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 2, 1, 3, 1, 2, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 2, 2, 2, 2, 1, 3, 3, 1, 3, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 3, 1,
3, 3, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 3, 1, 1, 1, 1, 1, 3, 1, 3,
1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 1, 2, 2, 3, 3, 2, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 2, 3, 1, 2,
1, 2, 2, 2, 2, 3, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
3, 1, 1, 1, 1, 2, 3, 3, 1, 2, 1, 2, 3, 2, 1, 1, 1, 1, 3, 1, 2, 1,
3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 1, 1, 3, 2, 3,
2, 3, 2, 1, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 3, 3, 1, 3, 1, 1, 3, 2, 2, 2, 2, 3, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 1, 1, 2, 1, 2, 1, 1, 1, 3, 2, 1, 1, 1,
1, 2, 3, 1, 3, 1, 1, 1, 1, 2, 3, 3, 3, 3, 3, 1, 3, 2, 2, 2, 2, 3,
2, 3, 2, 3, 3, 2, 1, 3, 1, 1, 1, 1, 1, 3, 1, 3, 3, 3, 3, 3, 1, 1,
1, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 2, 3, 3, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 3, 1, 1, 1, 2, 1,
1, 1])
[ ] #5.Train_test_split/train and test variables
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
print(x.shape)
print(x_train.shape)
print(x_test.shape)
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=yFoBLraSyPbu
major project1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Reconnect
Editing
[ ] print(x.shape)
print(x_train.shape)
print(x_test.shape)
(398, 2)
(298, 2)
(100, 2)
[ ] print(y.shape)
print(y_train.shape) #- 75%
print(y_test.shape) #- 25%
(398,)
(298,)
(100,)
[ ] #AFTER NORMALISATION AND SCALING
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
[ ] #7.Apply Classifier, Regressor
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
#8.Fitting the model
model.fit(x_train,y_train)
0s completed at 12:24 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=zwe3j1agynuA

major project1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Reconnect Editing

```
[ ] #8.Fitting the model
    model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[ ] #9.Predict the output
    y_pred = model.predict(x_test)
    y_pred #PREDCITED VALUES
```

```
array([1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 3, 3, 1, 1, 1, 3, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 3, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 1, 3, 1, 1, 1,
       1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 1])
```

```
[ ] y_test
```

```
array([1, 1, 1, 2, 1, 3, 3, 1, 1, 1, 2, 3, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1,
       3, 2, 1, 3, 1, 1, 1, 1, 1, 3, 1, 3, 1, 3, 1, 1, 2, 1, 1, 1, 1, 3,
       3, 2, 1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 2, 1, 3, 1, 3, 3, 1, 1, 1, 2, 2, 2, 3, 3, 1, 1,
       2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 1, 1])
```

```
[ ] #Accuracy
    from sklearn.metrics import accuracy_score
    accuracy_score(y_pred,y_test)* 100
```

```
69.0
```

```
#Individual Prediction
a = scaler.transform([[4500,60]])
```

0s completed at 12:24 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=UOhVD2owy2Oa

+ Code + Text

Reconnect

```
[ ] #Accuracy
    from sklearn.metrics import accuracy_score
    accuracy_score(y_pred,y_test)* 100
```

```
69.0
```

```
[ ] #Individual Prediction
    a = scaler.transform([[4500,60]])
```

```
[ ]
    model.predict(a)
```

```
array([1])
```

```
[ ] a
```

```
array([[0.86463013, 3.1595092 ]])
```

```
CONSLUION:
  #--> we had taken weight and model year  has input
  #--> we had taken origin has output
  #--> The had given x as input and y as output
  #--> From train_test_split, we get x and y train.shape of 75%
  #--> and x and y test.shape of 25%
  #--> By fitting the model output is predicted
  #--> Before Normalisation or Scaling, accuracy will be low(67)
  #--> After Normalisation or Scaling, accuracy will be high (69)
  #-->  we also get Individual Prediction
  #--> These analysis help us to now about more data
  #--> Model depends on nature of data as well as size of the data
```

Double-click (or enter) to edit

0s completed at 12:24 PM

Conclusion:

The outcome in logistic regression analysis is often coded as 0 or 1, where 1 indicates that the outcome of interest is present, and 0 indicates that the outcome of interest is absent. Logistic regression is used for binary or multi-class classification, and the target variable always has to be categorical.

GITHUBLINK:

https://raw.githubusercontent.com/thejaswini82/PROJECTS/main/LOGISTIC_AND_LINEAR_REGRESSION.ipynb

3.Linear Regression

Linear Regression can be considered a Machine Learning algorithm that allows us to map numeric inputs to numeric outputs, fitting a line into the data points. In other words, Linear Regression is a way of modelling the relationship between one or more variables. From the Machine Learning perspective, this is done to ensure generalization — giving the model the ability to predict outputs for inputs it has never seen before.

Example of simple linear regression

The table below shows some data from the early days of the Italian clothing company Benetton. Each row in the table shows Benetton's sales for a year and the amount spent on advertising that year. In this case, our outcome of interest is sales—it is what we want to predict. If we use advertising as the predictor variable, linear regression estimates that **Sales = 168 + 23 Advertising**. That is, if advertising expenditure is increased by one million Euro, then sales will be expected to increase by 23 million Euros, and if there was no advertising we would expect sales of 168 million Euros.

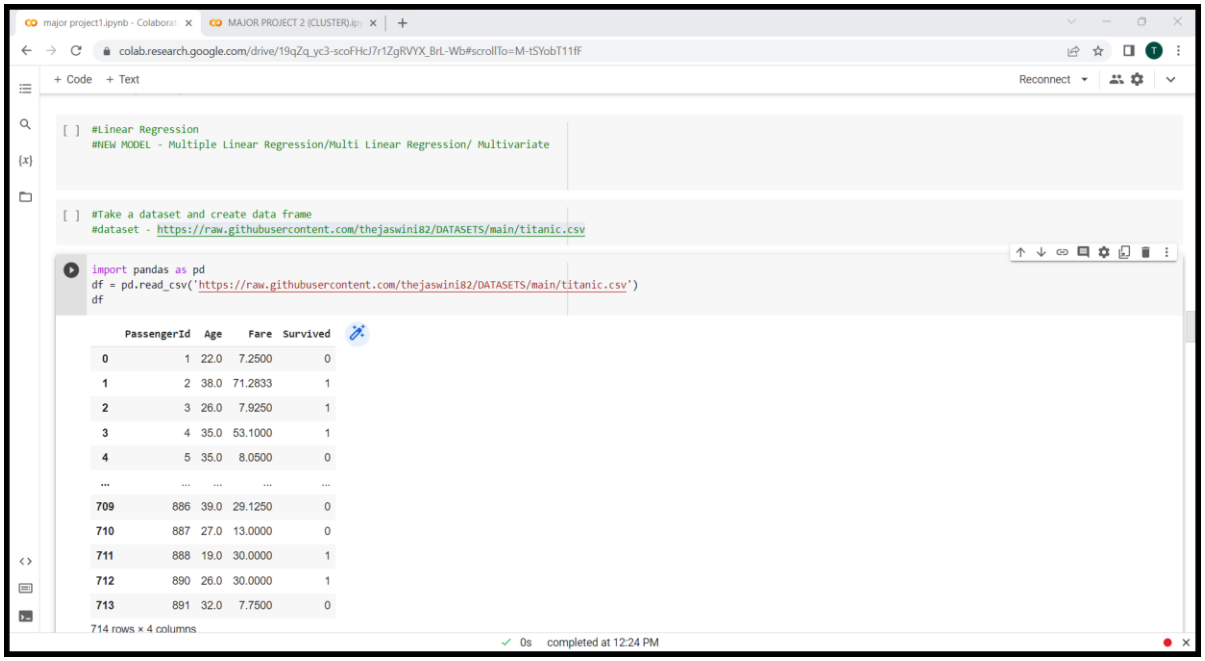
Year	Sales (Million Euro)	Advertising (Million Euro)
1	651	23
2	762	26
3	856	30
4	1,063	34
5	1,190	43
6	1,298	48
7	1,421	52
8	1,440	57
9	1,518	58

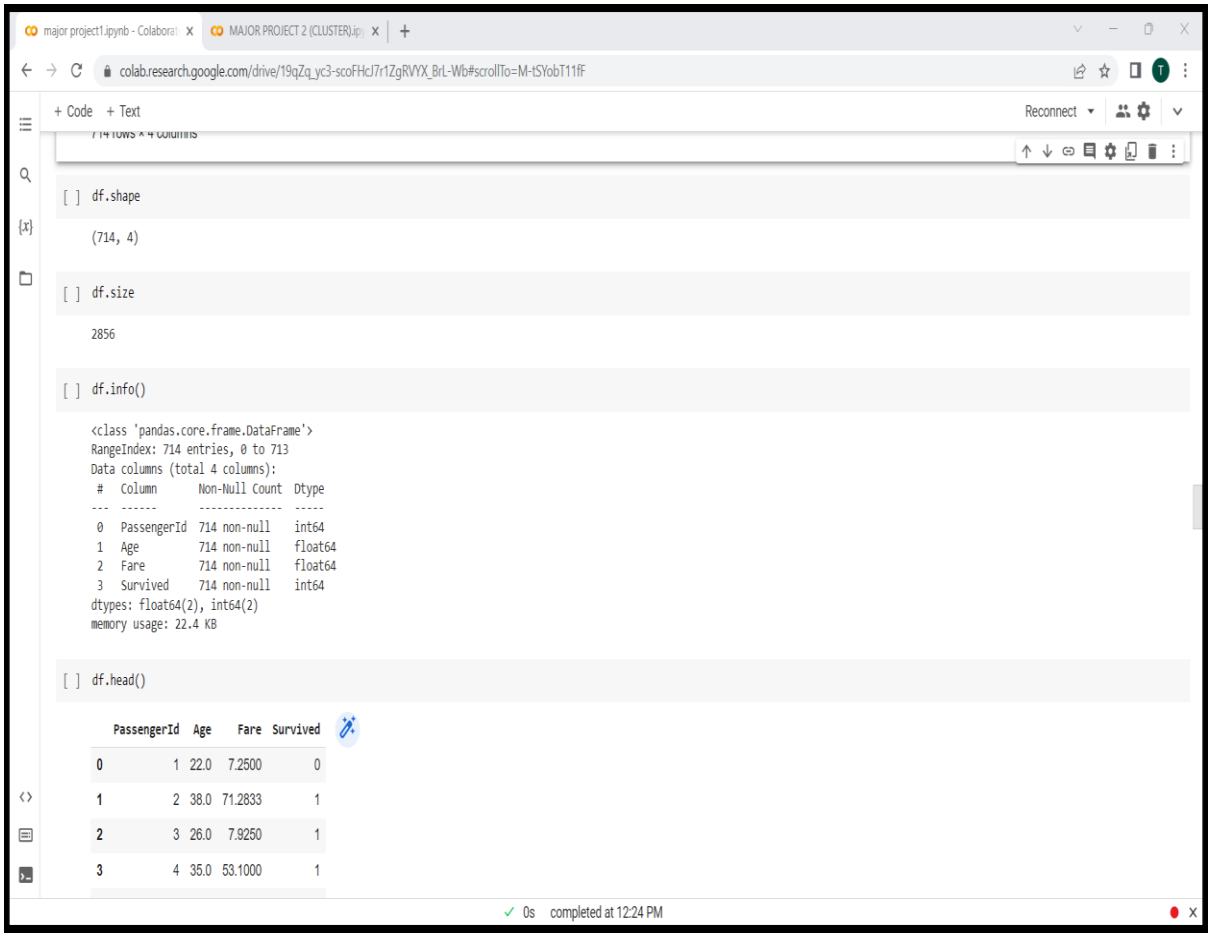
HERE IS ONE LINEAR REGRESSION

We have taken some Dataset:

https://raw.githubusercontent.com/thejaswini82/DATASETS/main/titanic.csv

SCREENSHOT:


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
#Linear Regression
#NEW MODEL - Multiple Linear Regression/Multi Linear Regression/ Multivariate
#Take a dataset and create data frame
#dataset - https://raw.githubusercontent.com/thejaswini82/DATASETS/main/titanic.csv
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/thejaswini82/DATASETS/main/titanic.csv')
df
PassengerId Age Fare Survived
0 1 22.0 7.2500 0
1 2 38.0 71.2833 1
2 3 26.0 7.9250 1
3 4 35.0 53.1000 1
4 5 35.0 8.0500 0
... ...
709 886 39.0 29.1250 0
710 887 27.0 13.0000 0
711 888 19.0 30.0000 1
712 890 26.0 30.0000 1
713 891 32.0 7.7500 0
714 rows × 4 columns
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
df.shape
(714, 4)
df.size
2856
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 714 entries, 0 to 713
Data columns (total 4 columns):
Column Non-Null Count Dtype
0 PassengerId 714 non-null int64
1 Age 714 non-null float64
2 Fare 714 non-null float64
3 Survived 714 non-null int64
dtypes: float64(2), int64(2)
memory usage: 22.4 KB
df.head()
PassengerId Age Fare Survived
0 1 22.0 7.2500 0
1 2 38.0 71.2833 1
2 3 26.0 7.9250 1
3 4 35.0 53.1000 1
0s completed at 12:24 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF

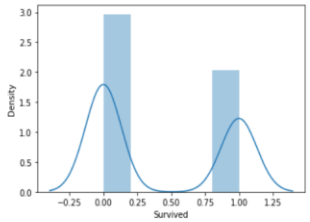
+ Code + Text | Reconnect

```
[ ] df.tail()
```

	PassengerId	Age	Fare	Survived
709	886	39.0	29.125	0
710	887	27.0	13.000	0
711	888	19.0	30.000	1
712	890	26.0	30.000	1
713	891	32.0	7.750	0

```
[ ] #VISUALISATION
    import seaborn as sns
    sns.distplot(df['Survived']) #distribution plot
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9bbd50f410>
```



0s completed at 12:24 PM

major project1.ipynb - Colaborat | MAJOR PROJECT 2 (CLUSTER).ipy

colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF

+ Code + Text | Reconnect

```
[ ] #We want to consider only the numeric data
    #So we will create a new dataframe with only numeric data
    df_numeric = df.select_dtypes(include = ['float64','int64'])
    df_numeric
```

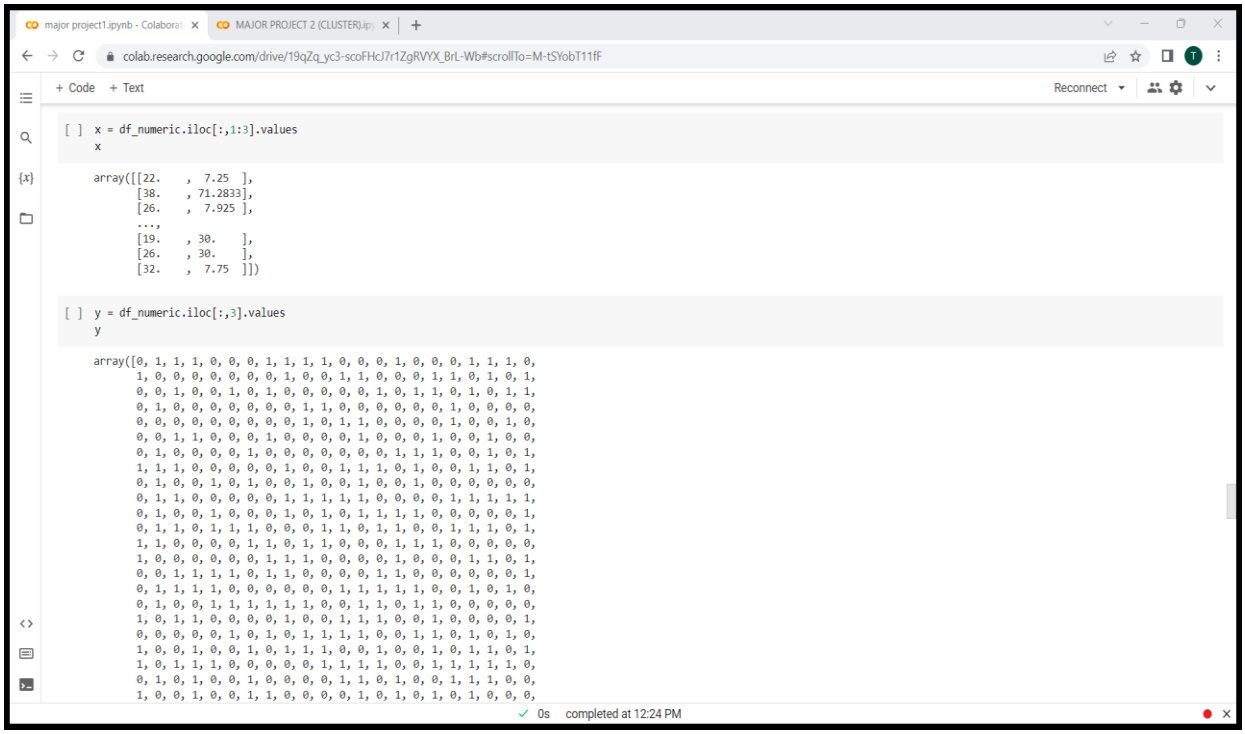
	PassengerId	Age	Fare	Survived
0	1	22.0	7.2500	0
1	2	38.0	71.2833	1
2	3	26.0	7.9250	1
3	4	35.0	53.1000	1
4	5	35.0	8.0500	0
...	...	...	...	...
709	886	39.0	29.1250	0
710	887	27.0	13.0000	0
711	888	19.0	30.0000	1
712	890	26.0	30.0000	1
713	891	32.0	7.7500	0

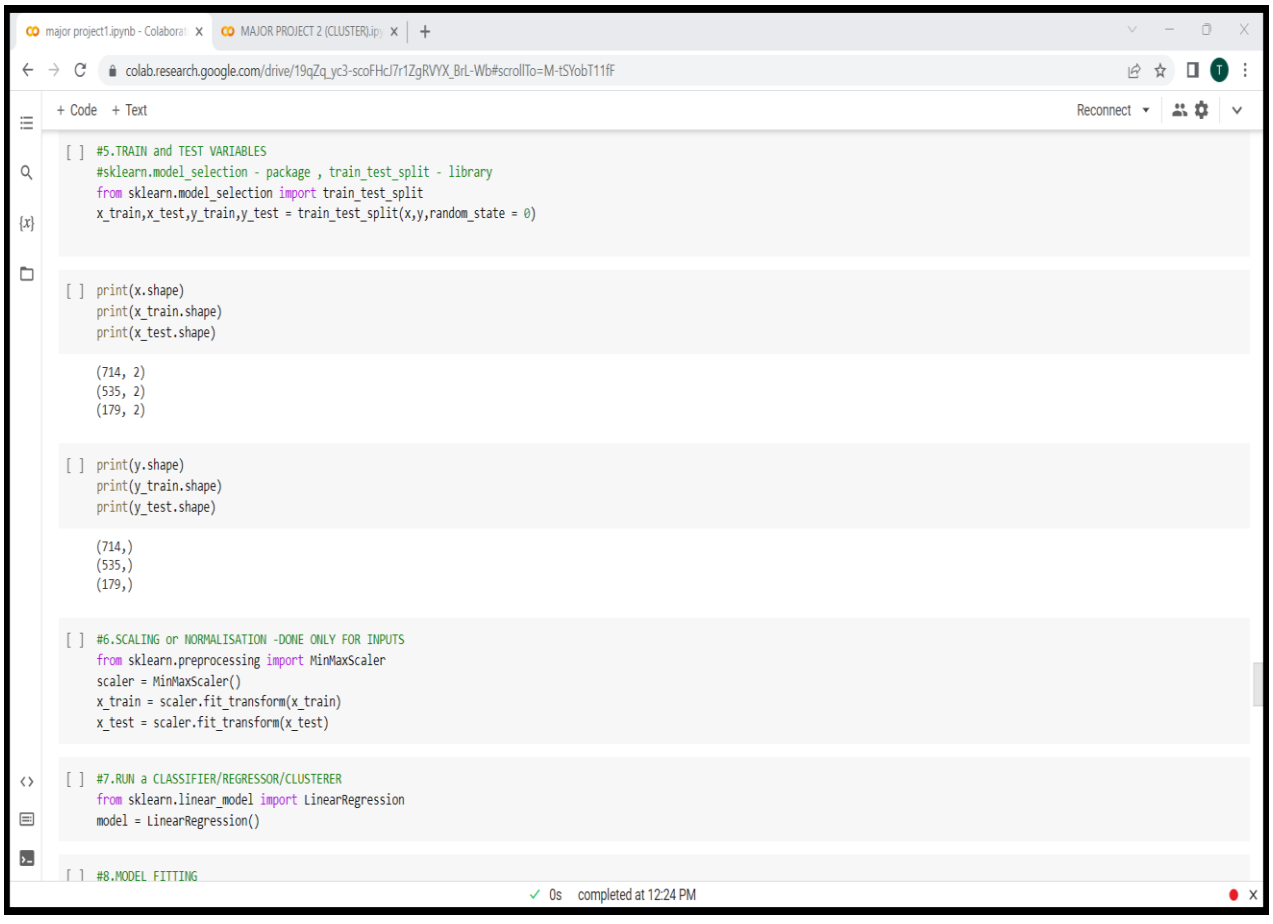
714 rows × 4 columns

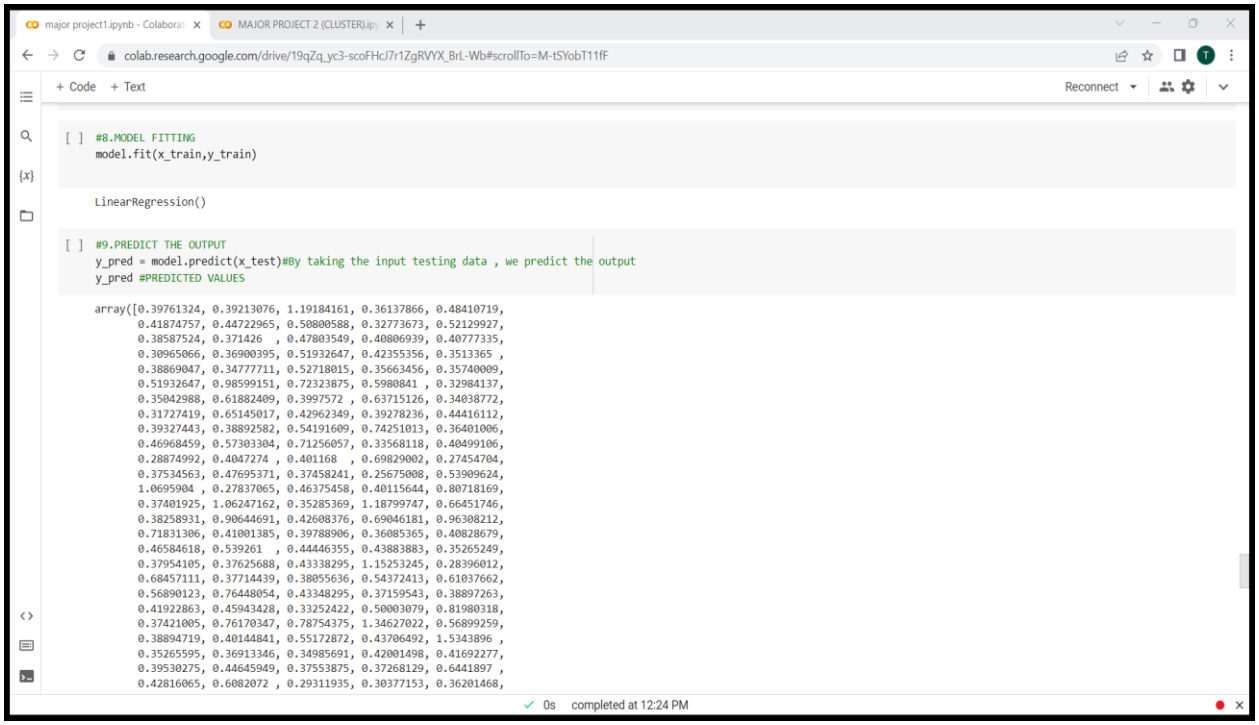
```
[ ] #divide the data into i/p and o/p
    #output - Survived
    #input - Age and Fare column
```

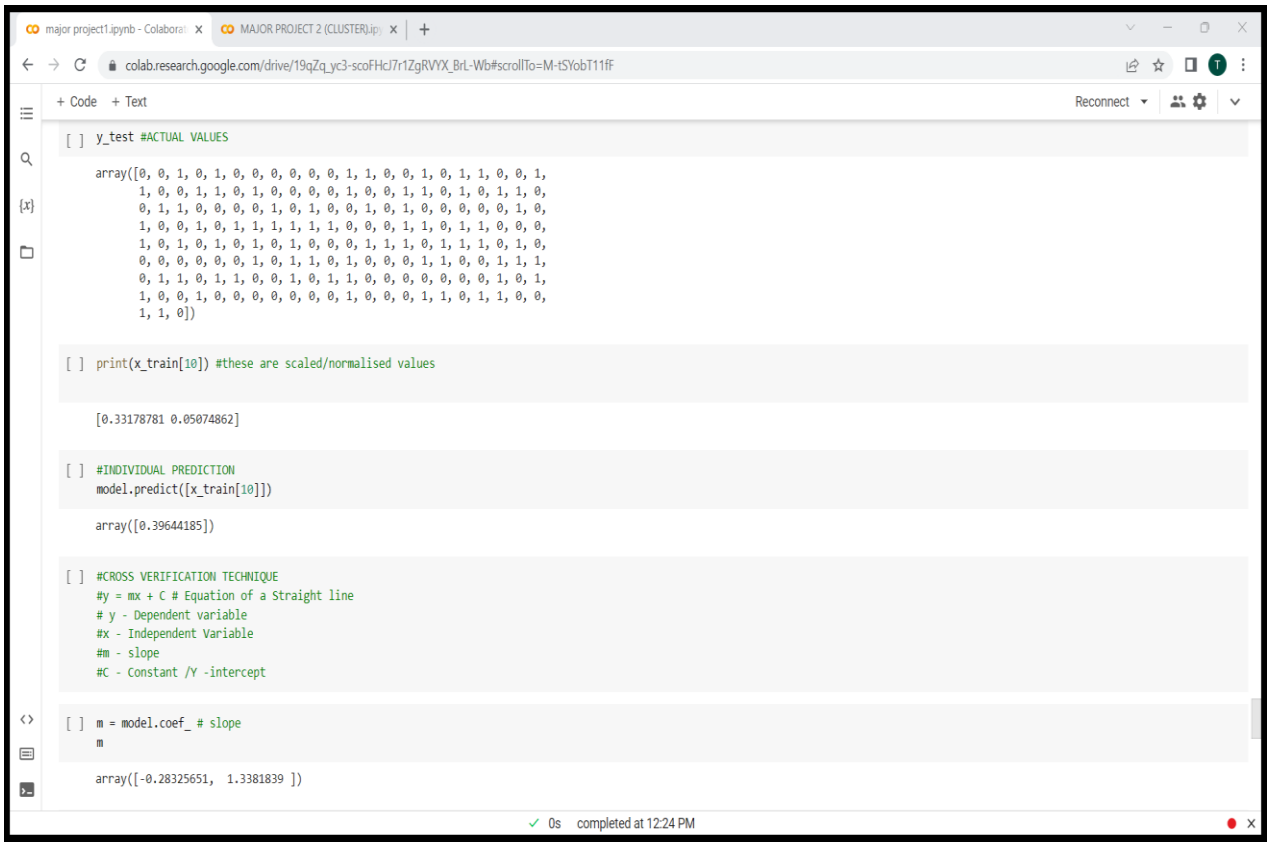
```
[ ] x = df_numeric.iloc[:,1:3].values
```

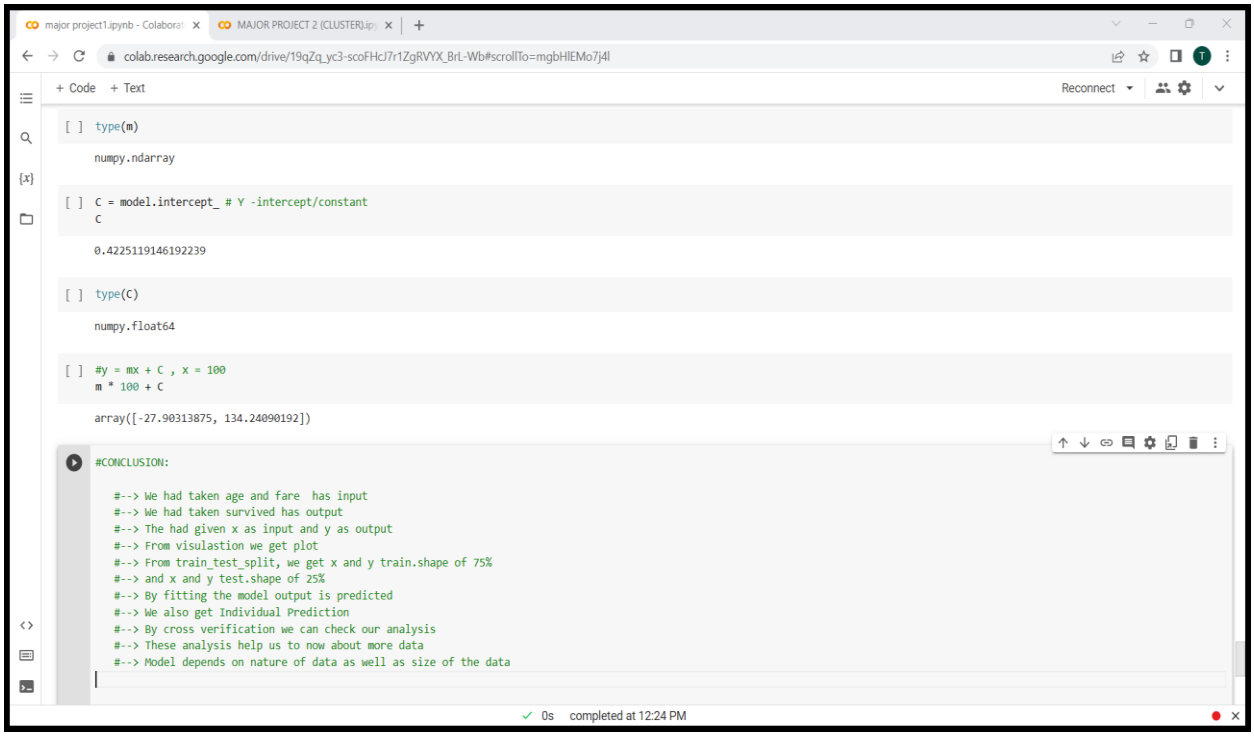
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
[ ] x = df_numeric.iloc[:,1:3].values
x
array([[22. , 7.25],
[38. , 71.2833],
[26. , 7.925],
...,
[19. , 30.],
[26. , 30.],
[32. , 7.75]])
[ ] y = df_numeric.iloc[:,3].values
y
array([0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
[ ] #5.TRAIN and TEST VARIABLES
#sklearn.model_selection - package , train_test_split - library
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
[ ] print(x.shape)
print(x_train.shape)
print(x_test.shape)
(714, 2)
(535, 2)
(179, 2)
[ ] print(y.shape)
print(y_train.shape)
print(y_test.shape)
(714,)
(535,)
(179,)
[ ] #6.SCALING or NORMALISATION -DONE ONLY FOR INPUTS
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
[ ] #7.RUN a CLASSIFIER/REGRESSOR/CLUSTERER
from sklearn.linear_model import LinearRegression
model = LinearRegression()
[ ] #8.MODEL FITTING
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
[ ] #8.MODEL FITTING
model.fit(x_train,y_train)
LinearRegression()
[ ] #9.PREDICT THE OUTPUT
y_pred = model.predict(x_test)#By taking the input testing data , we predict the output
y_pred #PREDICTED VALUES
array([0.39761324, 0.39213076, 1.19184161, 0.36137866, 0.48410719,
0.41874757, 0.44722965, 0.50800588, 0.32773673, 0.52129927,
0.38587524, 0.371426 , 0.47803549, 0.40806939, 0.40777335,
0.30965066, 0.36900395, 0.51932647, 0.42355356, 0.3513365 ,
0.38869047, 0.34777711, 0.52718015, 0.35663456, 0.35740009,
0.51932647, 0.98599151, 0.72323875, 0.5980841 , 0.32984137,
0.35042988, 0.61882409, 0.3997572 , 0.63715126, 0.34038772,
0.31727419, 0.65145017, 0.42962349, 0.39278236, 0.44416112,
0.39327443, 0.38892582, 0.54191609, 0.74251013, 0.36401006,
0.46968459, 0.57303304, 0.71256057, 0.33568118, 0.40499106,
0.28874992, 0.4047274 , 0.401168 , 0.69829002, 0.27454704,
0.37534563, 0.47695371, 0.37458241, 0.25675008, 0.53909624,
1.0695904 , 0.27837065, 0.46375458, 0.40115644, 0.80718169,
0.37401925, 1.06247162, 0.35285369, 1.18799747, 0.66451746,
0.38258931, 0.90644691, 0.42608376, 0.69046181, 0.96308212,
0.71831306, 0.41001385, 0.39788906, 0.36085365, 0.40828679,
0.46584618, 0.539261 , 0.44446355, 0.43883883, 0.35265249,
0.37954105, 0.37625688, 0.43338295, 1.15253245, 0.28396012,
0.68457111, 0.37714439, 0.38055636, 0.54372413, 0.61037662,
0.56890123, 0.76448054, 0.43348295, 0.37159543, 0.38897263,
0.41922863, 0.45943428, 0.33252422, 0.50003079, 0.81980318,
0.37421005, 0.76170347, 0.78754375, 1.34627022, 0.56899259,
0.38894719, 0.40144841, 0.55172872, 0.43706492, 1.5343896 ,
0.35265595, 0.36913346, 0.34985691, 0.42001498, 0.41692277,
0.39530275, 0.44645949, 0.37553875, 0.37268129, 0.6441897 ,
0.42816065, 0.6082072 , 0.29311935, 0.30377153, 0.36201468,
0s completed at 12:24 PM


major project1.ipynb - Colaborat
MAJOR PROJECT 2 (CLUSTER).ipy
colab.research.google.com/drive/19qZq_yc3-scoFHcJ7r1ZgRVYX_BrL-Wb#scrollTo=M-tSYobT11fF
+ Code
+ Text
Reconnect
[ ] y_test #ACTUAL VALUES
array([0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 1, 0])
[ ] print(x_train[10]) #these are scaled/normalised values
[0.33178781 0.05074862]
[ ] #INDIVIDUAL PREDICTION
model.predict([x_train[10]])
array([0.39644185])
[ ] #CROSS VERIFICATION TECHNIQUE
#y = mx + C # Equation of a Straight line
y - Dependent variable
#x - Independent Variable
#m - slope
#C - Constant /Y -intercept
[ ] m = model.coef_ # slope
m
array([-0.28325651, 1.3381839])
0s completed at 12:24 PM



Conclusion:

Linear Regression is the process of finding a line that best fits the data points available on the plot, so that we can use it to predict output values for inputs that are not present in the data set we have, with the belief that those outputs would fall on the line. Performance (and error rates) depends on various factors including the how clean and consistent the data is. There are different ways of improving the performance (i.e., generalizability) of the model. However, each one has its own pros and cons, which makes the choice of methods application-dependent.

GITHUBLINK:

https://raw.githubusercontent.com/thejaswini82/PROJECTS/main/LOGISTIC_AND_LINEAR_REGRESSION.ipynb

THANK YOU