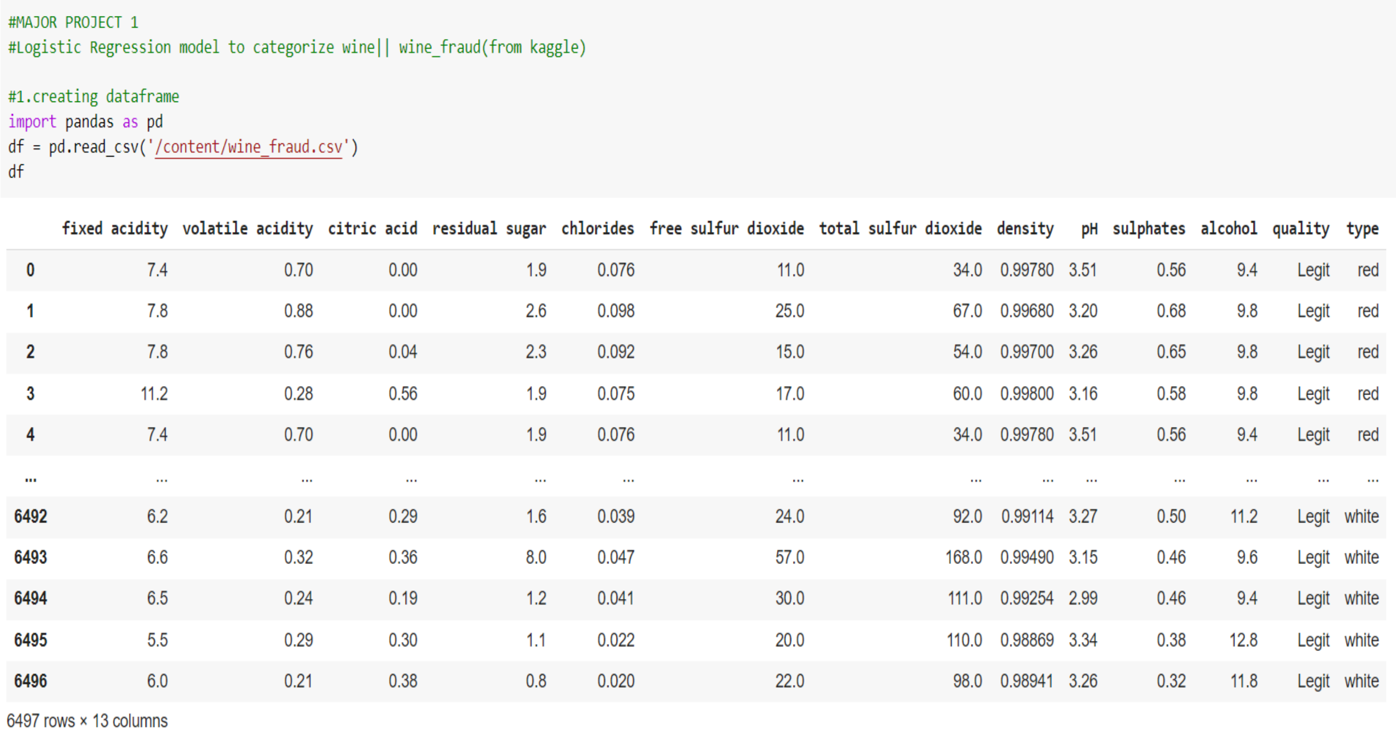
MAJOR PROJECT

Name: NIDUMUKKALA RAVI SHANKAR

Institute: IIT Bhubaneswar

Branch: MECHANICAL ENGINEERING – 2nd Year

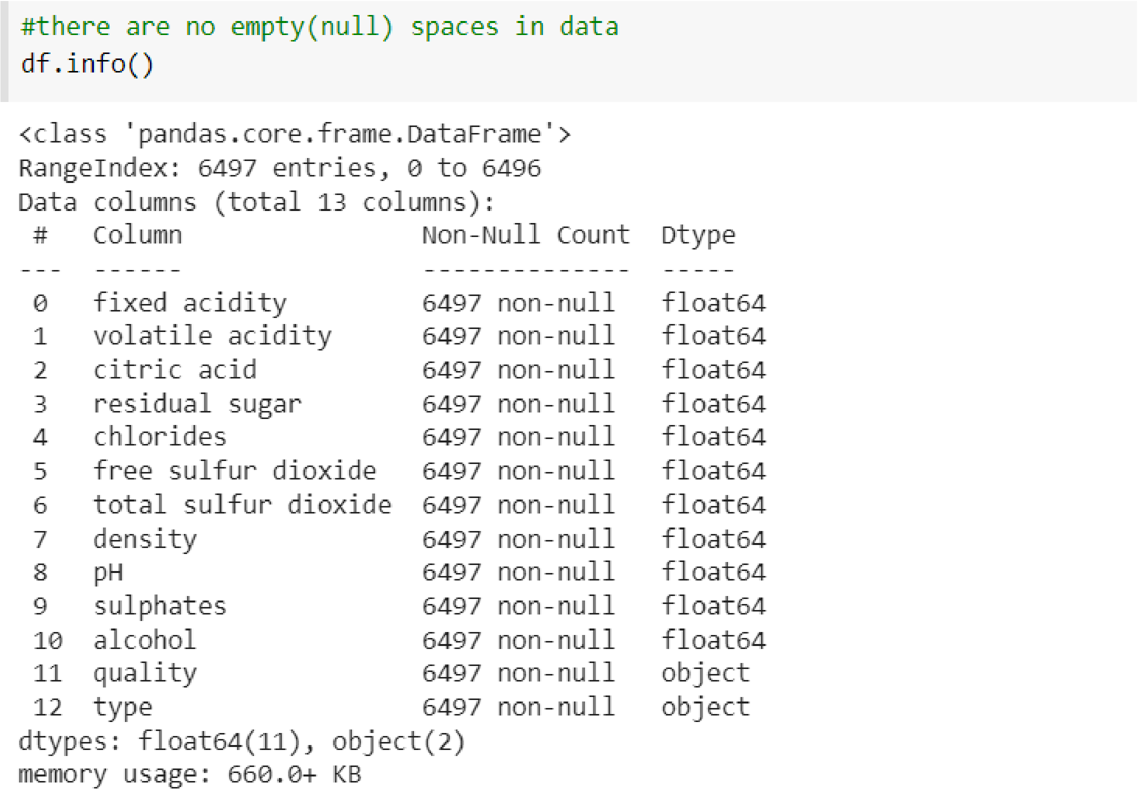
Major Project 1



```
#MAJOR PROJECT 1
#Logistic Regression model to categorize wine|| wine_fraud(from kaggle)

#1.creating dataframe
import pandas as pd
df = pd.read_csv('/content/wine_fraud.csv')
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	type
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	Legit	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	Legit	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	Legit	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	Legit	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	Legit	red
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	Legit	white
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	Legit	white
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	Legit	white
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	Legit	white
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	Legit	white

6497 rows × 13 columns



```
#there are no empty(null) spaces in data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   object 
 12  type                  6497 non-null   object 
dtypes: float64(11), object(2)
memory usage: 660.0+ KB
```

```
#2.Preprocessing (finding no of unique for quality cloumn )
df.quality.unique()
```

```
array(['Legit', 'Fraud'], dtype=object)
```

```
#selecting rows which has 'Legit' in column 'Quality' and replacing this new dataframe with original dataframe (df)
df=df.loc[(df['quality'])=='Legit']
df
```

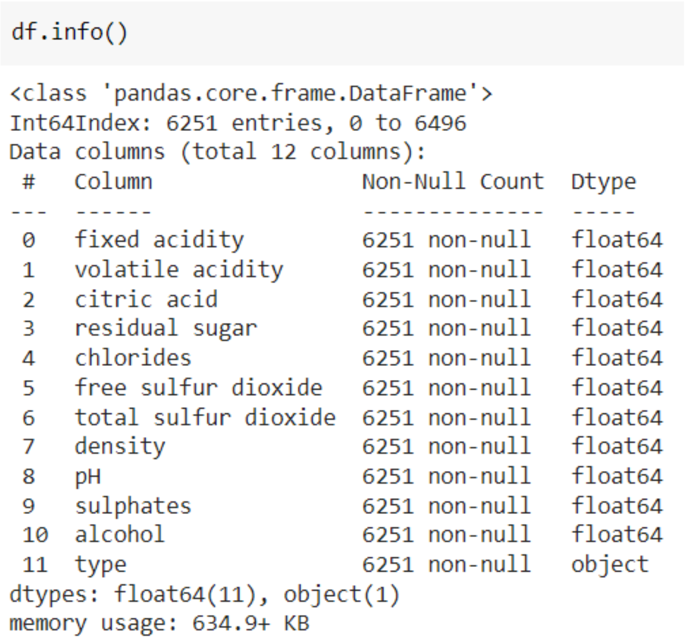
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	type
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	Legit	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	Legit	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	Legit	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	Legit	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	Legit	red
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	Legit	white
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	Legit	white
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	Legit	white
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	Legit	white
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	Legit	white

6251 rows × 13 columns

```
#removing the column 'Quality' from dataframe
df=df.drop(['quality'],axis=1)
df
```

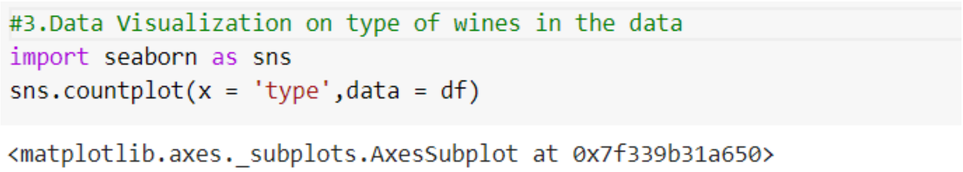
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	type
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	red
...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	white
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	white
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	white
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	white
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	white

6251 rows × 12 columns



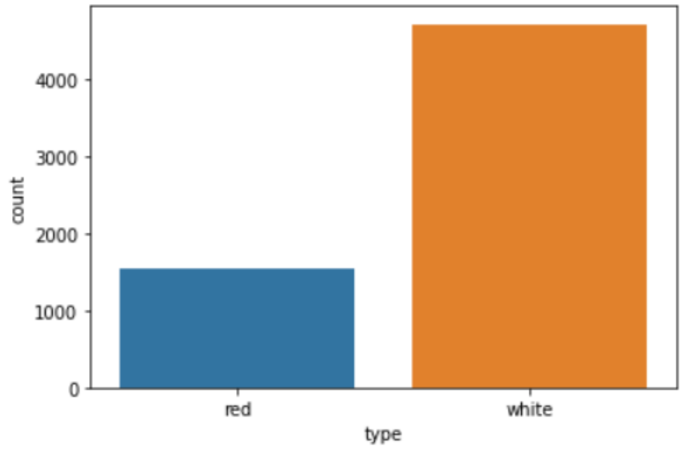
```
df.info()
```

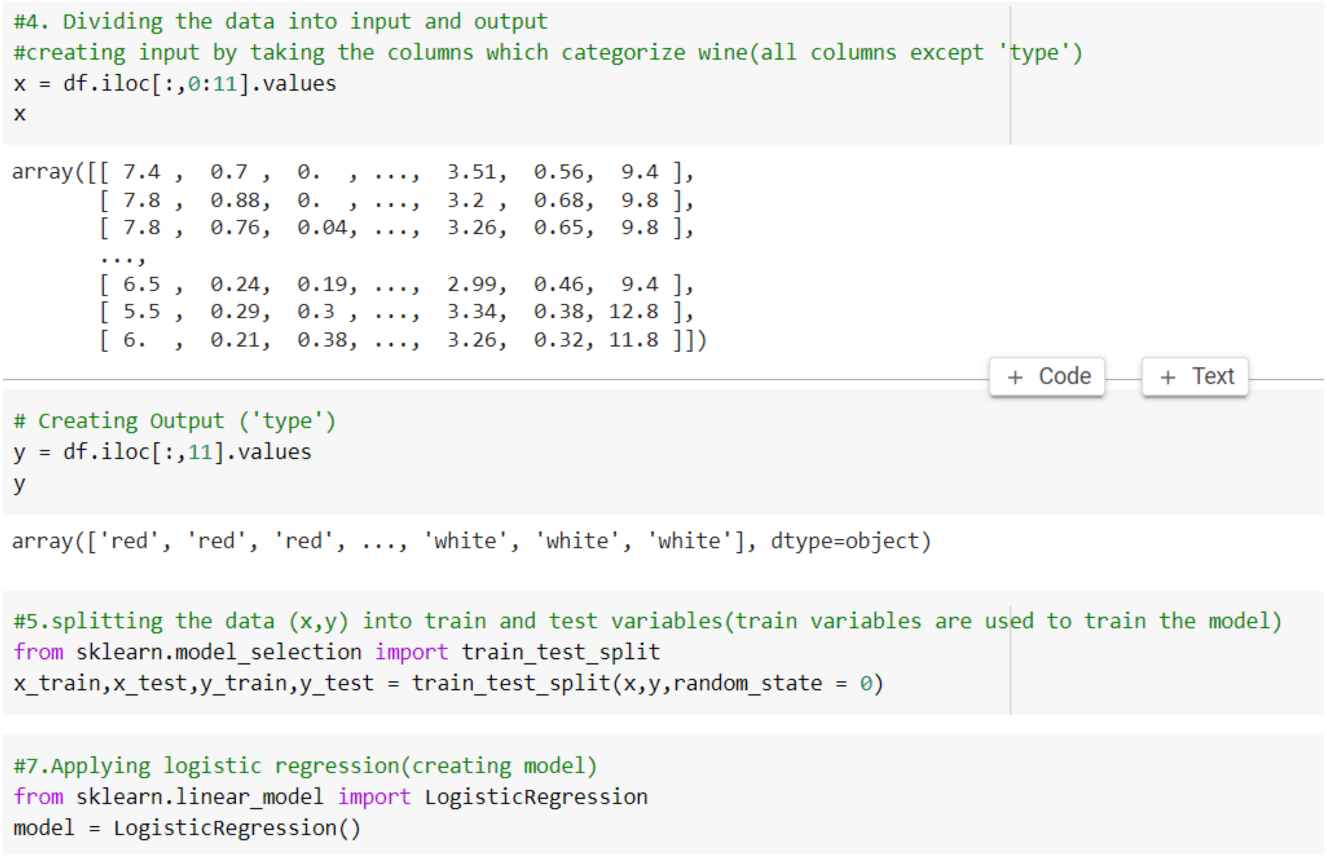
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6251 entries, 0 to 6496
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6251 non-null   float64
 1   volatile acidity      6251 non-null   float64
 2   citric acid           6251 non-null   float64
 3   residual sugar        6251 non-null   float64
 4   chlorides             6251 non-null   float64
 5   free sulfur dioxide   6251 non-null   float64
 6   total sulfur dioxide  6251 non-null   float64
 7   density               6251 non-null   float64
 8   pH                    6251 non-null   float64
 9   sulphates             6251 non-null   float64
 10  alcohol               6251 non-null   float64
 11  type                  6251 non-null   object 
dtypes: float64(11), object(1)
memory usage: 634.9+ KB
```



```
#3.Data Visualization on type of wines in the data
import seaborn as sns
sns.countplot(x = 'type',data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f339b31a650>
```





```
#4. Dividing the data into input and output
#creating input by taking the columns which categorize wine(all columns except 'type')
x = df.iloc[:,0:11].values
x
```

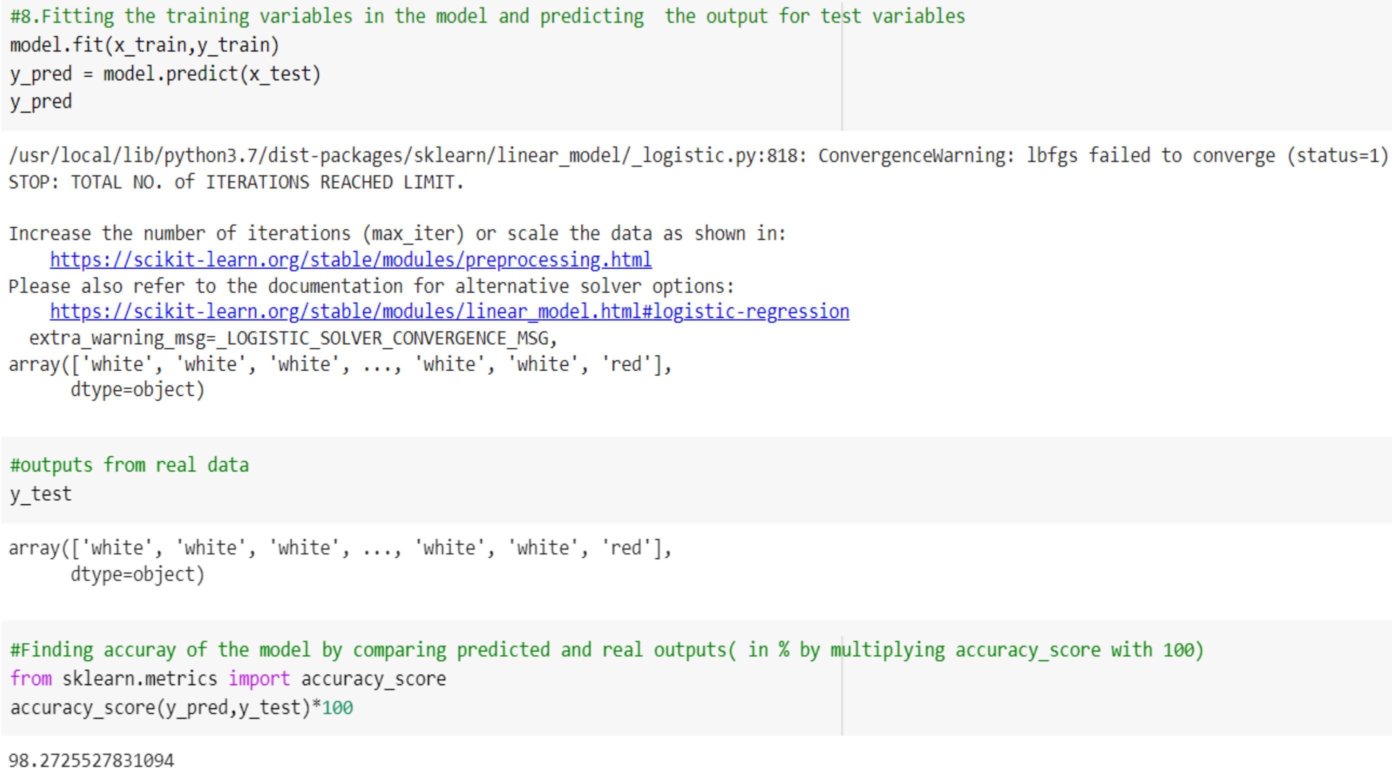
```
array([[ 7.4 ,  0.7 ,  0.  , ...,  3.51,  0.56,  9.4 ],
       [ 7.8 ,  0.88,  0.  , ...,  3.2 ,  0.68,  9.8 ],
       [ 7.8 ,  0.76,  0.04, ...,  3.26,  0.65,  9.8 ],
       ...,
       [ 6.5 ,  0.24,  0.19, ...,  2.99,  0.46,  9.4 ],
       [ 5.5 ,  0.29,  0.3 , ...,  3.34,  0.38, 12.8 ],
       [ 6.  ,  0.21,  0.38, ...,  3.26,  0.32, 11.8 ]])
```

```
# Creating Output ('type')
y = df.iloc[:,11].values
y
```

```
array(['red', 'red', 'red', ..., 'white', 'white', 'white'], dtype=object)
```

```
#5.splitting the data (x,y) into train and test variables(train variables are used to train the model)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
```

```
#7.Applying logistic regression(creating model)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```



```
#8.Fitting the training variables in the model and predicting  the output for test variables
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
array(['white', 'white', 'white', ..., 'white', 'white', 'red'],
      dtype=object)
```

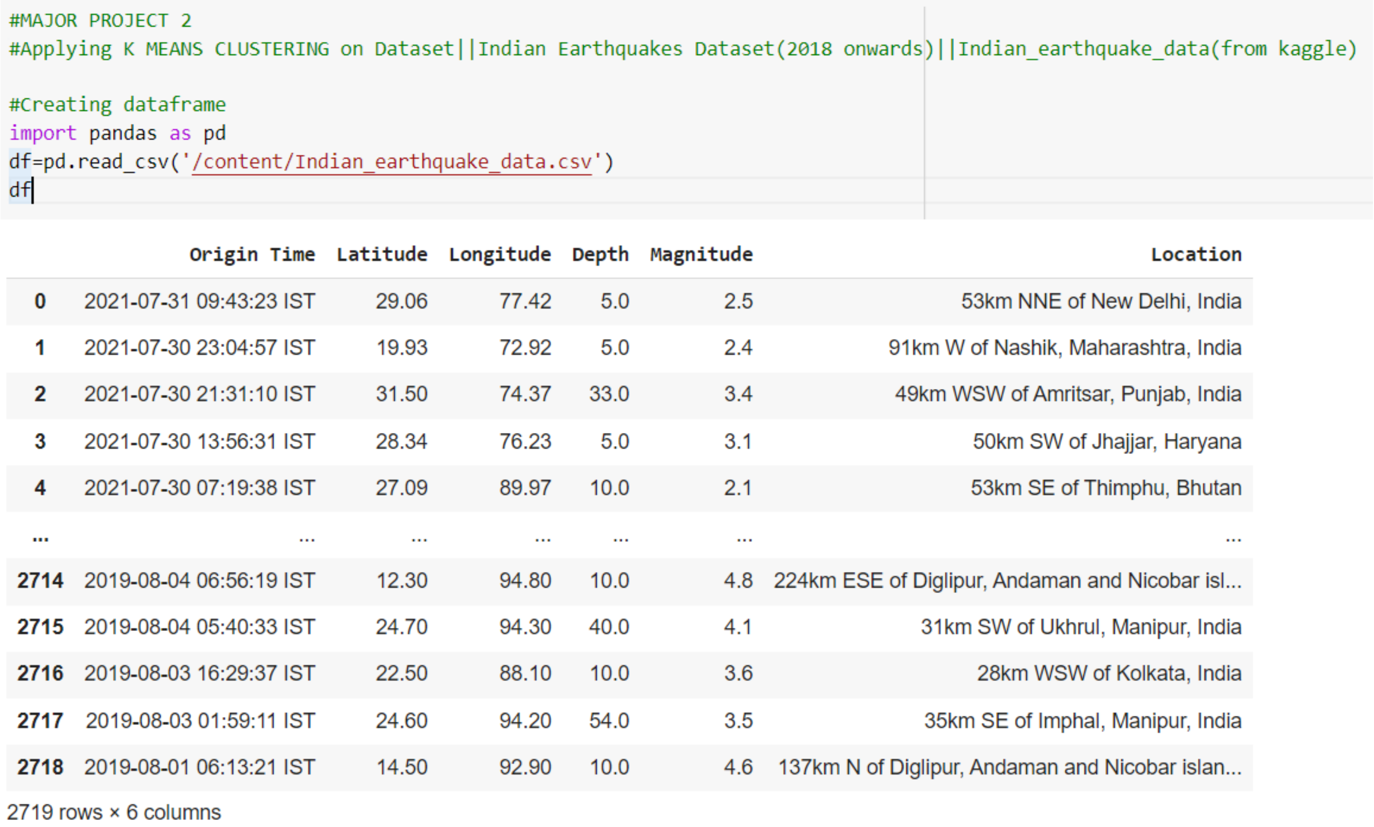
```
#outputs from real data
y_test
```

```
array(['white', 'white', 'white', ..., 'white', 'white', 'red'],
      dtype=object)
```

```
#Finding accuracy of the model by comparing predicted and real outputs( in % by multiplying accuracy_score with 100)
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100
```

```
98.2725527831094
```

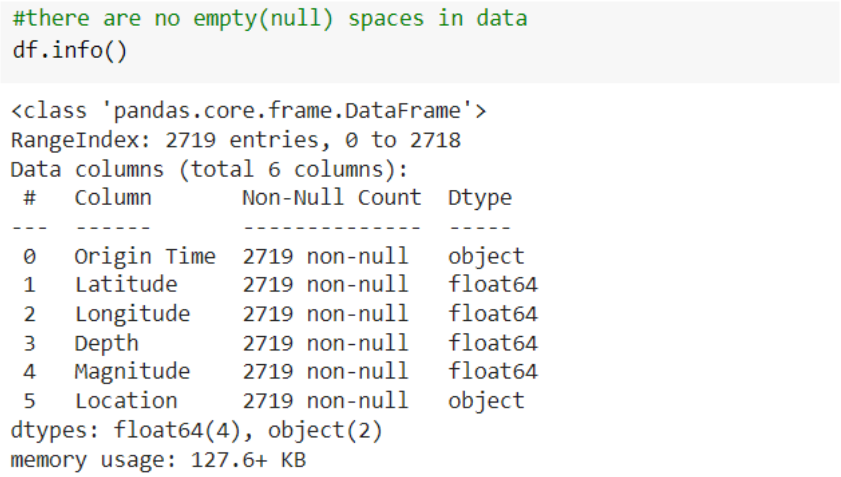
Major project 2



```
#MAJOR PROJECT 2
#Applying K MEANS CLUSTERING on Dataset||Indian Earthquakes Dataset(2018 onwards)||Indian_earthquake_data(from kaggle)

#Creating dataframe
import pandas as pd
df=pd.read_csv('/content/Indian_earthquake_data.csv')
df
```

	Origin Time	Latitude	Longitude	Depth	Magnitude	Location
0	2021-07-31 09:43:23 IST	29.06	77.42	5.0	2.5	53km NNE of New Delhi, India
1	2021-07-30 23:04:57 IST	19.93	72.92	5.0	2.4	91km W of Nashik, Maharashtra, India
2	2021-07-30 21:31:10 IST	31.50	74.37	33.0	3.4	49km WSW of Amritsar, Punjab, India
3	2021-07-30 13:56:31 IST	28.34	76.23	5.0	3.1	50km SW of Jhajjar, Haryana
4	2021-07-30 07:19:38 IST	27.09	89.97	10.0	2.1	53km SE of Thimphu, Bhutan
...	...	...	...	...	...	...
2714	2019-08-04 06:56:19 IST	12.30	94.80	10.0	4.8	224km ESE of Diglipur, Andaman and Nicobar isl...
2715	2019-08-04 05:40:33 IST	24.70	94.30	40.0	4.1	31km SW of Ukhrul, Manipur, India
2716	2019-08-03 16:29:37 IST	22.50	88.10	10.0	3.6	28km WSW of Kolkata, India
2717	2019-08-03 01:59:11 IST	24.60	94.20	54.0	3.5	35km SE of Imphal, Manipur, India
2718	2019-08-01 06:13:21 IST	14.50	92.90	10.0	4.6	137km N of Diglipur, Andaman and Nicobar islan...

2719 rows × 6 columns

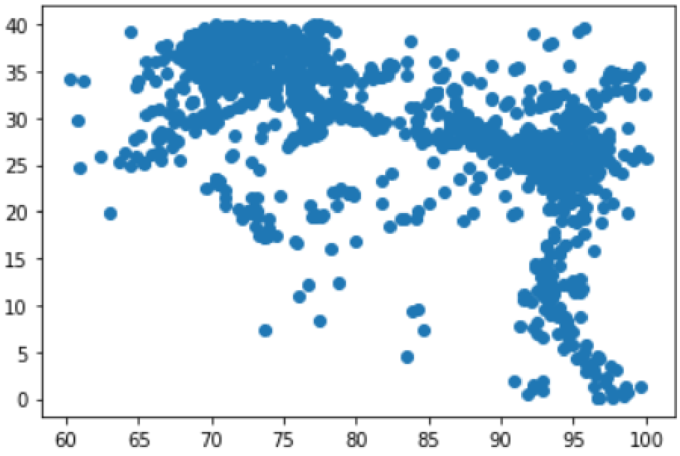


```
#there are no empty(null) spaces in data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2719 entries, 0 to 2718
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Origin Time  2719 non-null   object 
 1   Latitude     2719 non-null   float64
 2   Longitude    2719 non-null   float64
 3   Depth        2719 non-null   float64
 4   Magnitude    2719 non-null   float64
 5   Location     2719 non-null   object 
dtypes: float64(4), object(2)
memory usage: 127.6+ KB
```

```
#Data Visualization on points where earthquakes occured
import matplotlib.pyplot as plt
plt.scatter(df['Longitude'],df['Latitude'])#representing as shown in map
```

```
<matplotlib.collections.PathCollection at 0x7fbebd282650>
```



```
#Dividing data into input and output
#creating input(Longitude,Latitude)
x=df.iloc[:,[2,1]].values
x
```

```
array([[77.42, 29.06],
       [72.92, 19.93],
       [74.37, 31.5 ],
       ...,
       [88.1 , 22.5 ],
       [94.2 , 24.6 ],
       [92.9 , 14.5 ]])
```

```
#2719 rows(points), 6 columns
df.shape
```

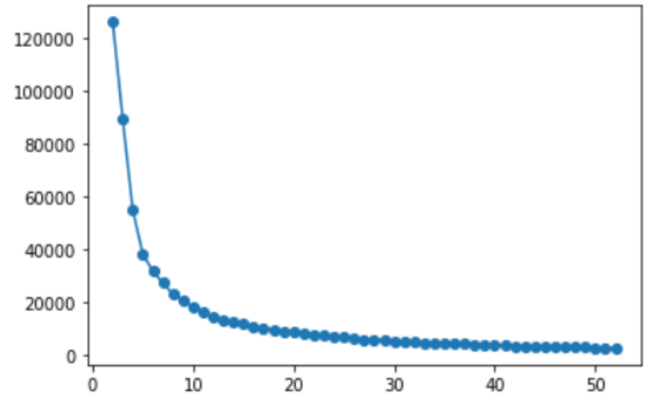
```
(2719, 6)
```

```
#finding sqaure root of 2719
import numpy as np
np.sqrt(2719)
```

```
52.14403129793476
```

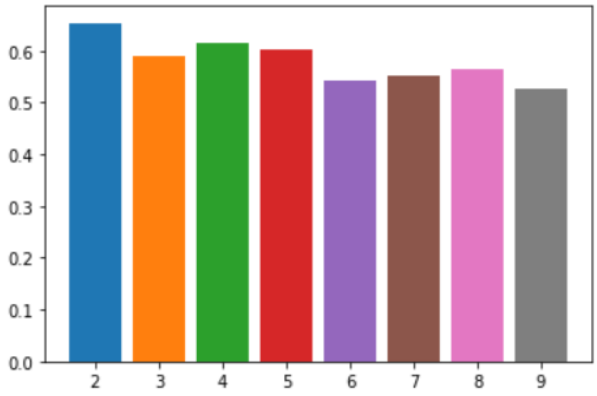
```
#no of clusters should be in the range of 2 to 53

from sklearn.cluster import KMeans
k=range(2,53)
s=[]
for i in k :
  model_demo=KMeans(n_clusters=i,random_state=0)
  model_demo.fit(x)
  s.append(model_demo.inertia_)
plt.scatter(k,s)
plt.plot(k,s)
#from ELBOW METHOD we can say that prominent point can be in range (2,10)
```

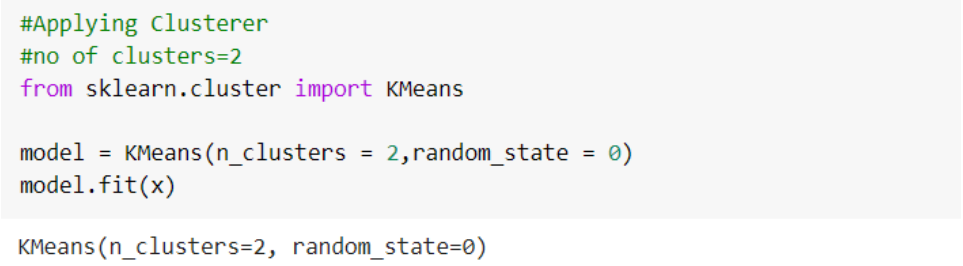
```
[<matplotlib.lines.Line2D at 0x7fbebd211a50>]
```



```
#Applying SILHOUETTE SCORE METHOD to find prominent point in range (2,10)
from sklearn.metrics import silhouette_score
k = range(2,10)
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))
#from SILHOUETTE SCORE METHOD we conclude 2 clusters would be appropriate
```

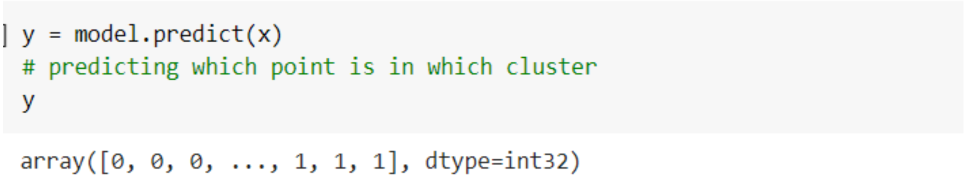
```
2 Clusters ,Score = 0.6539436830056814
3 Clusters ,Score = 0.5909525390590425
4 Clusters ,Score = 0.6147081891189834
5 Clusters ,Score = 0.6027082160971342
6 Clusters ,Score = 0.5437930418817251
7 Clusters ,Score = 0.5509742903479136
8 Clusters ,Score = 0.56612125756764
9 Clusters ,Score = 0.5271225395705129
```





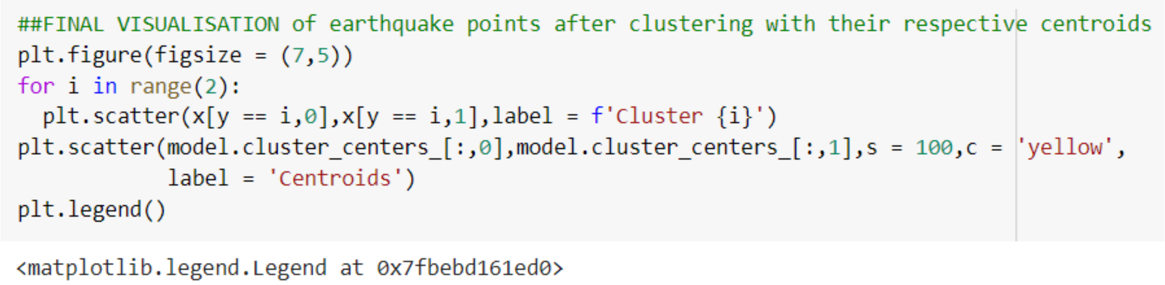
```
#Applying Clusterer
#no of clusters=2
from sklearn.cluster import KMeans

model = KMeans(n_clusters = 2,random_state = 0)
model.fit(x)
```

```
KMeans(n_clusters=2, random_state=0)
```



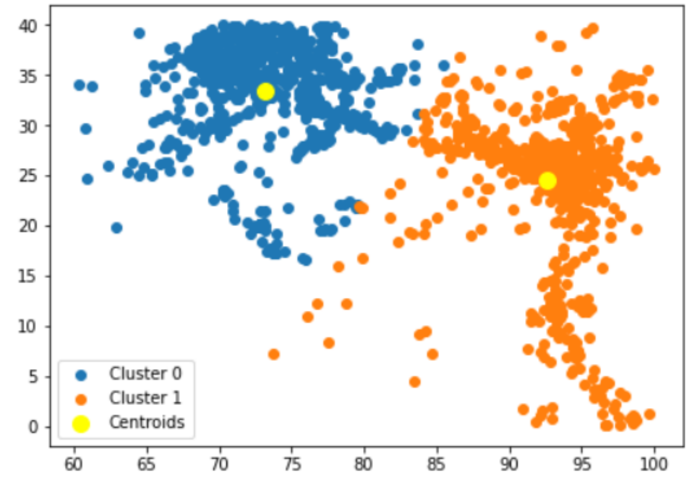
```
] y = model.predict(x)
# predicting which point is in which cluster
y
```

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



```
##FINAL VISUALISATION of earthquake points after clustering with their respective centroids
plt.figure(figsize = (7,5))
for i in range(2):
  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 = 100,c = 'yellow',
            label = 'Centroids')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fbebd161ed0>
```



Github account link-

https://github.com/ravishankar2003/RINEX