NAME- PRIYADARSINI MOHARANA

COLLEGE- KIIT UNIVERSITY

BRANCH- COMPUTER SCIENCE & ENGINEERING

YEAR- 2ND

```
#MAJOR PROJECT-1

#Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR

#DATASET-mcdonalds ice cream
#URL-/content/mcdonalds_dataset.csv.zip

#CREATE DATAFRAME
import pandas as pd
df=pd.read_csv('/content/mcdonalds_dataset.csv.zip')
df
```

	lat	lon	alt	is_broken	is_active	dot	state	city	street	country	last_checked
0	-73.988281	40.718830	0	False	True	working	NY	New York	114 Delancey St	USA	Checked 142 minutes ago
1	-74.005090	40.728794	0	False	True	working	NY	New York	208 Varick St	USA	Checked 142 minutes ago
2	-73.993408	40.729197	0	False	True	working	NY	New York	724 Broadway	USA	Checked 142 minutes ago
3	-73.985855	40.726555	0	False	True	working	NY	New York	102 1st Ave	USA	Checked 142 minutes ago
4	-73.991692	40.691383	0	True	True	broken	NY	Brooklyn	82 Court St	USA	Checked 142 minutes ago
...	...	...	...	...	...	...	...	...	...	...	...
16666	13.475643	52.514265	0	False	False	inactive	NaN	Berlin	Frankfurter Allee 117	DE	Checked 31 minutes ago
16667	13.429812	54.076239	0	False	False	inactive	NaN	Greifswald	Anklamer Landstr. 1	DE	Checked 31 minutes ago
16668	8.787059	53.100934	0	False	False	inactive	NaN	Bremen	Waller Heerstr. 101	DE	Checked 31 minutes ago
16669	11.409059	53.628227	0	False	False	inactive	NaN	Schwerin	Marienplatz 5-7	DE	Checked 31 minutes ago
16670	11.405999	53.903701	0	False	False	inactive	NaN	Wismar	Zierower Landstr. 3	DE	Checked 31 minutes ago

16671 rows × 11 columns

```
df.shape
#ROWS-16671
#COLUMNS-11
```

```
(16671, 11)
```

```
df.info()
```

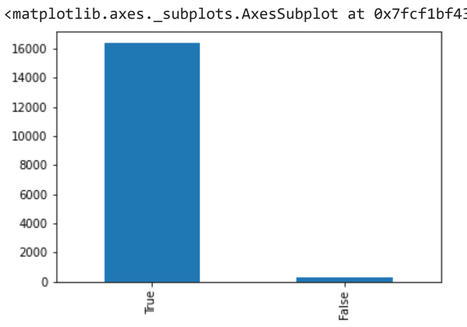
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16671 entries, 0 to 16670
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   lat           16671 non-null  float64
 1   lon           16671 non-null  float64
 2   alt           16671 non-null  int64
 3   is_broken     16671 non-null  bool
 4   is_active     16671 non-null  bool
 5   dot           16671 non-null  object
 6   state         12725 non-null  object
 7   city          16663 non-null  object
 8   street        16671 non-null  object
 9   country       16671 non-null  object
 10  last_checked  16671 non-null  object
dtypes: bool(2), float64(2), int64(1), object(6)
memory usage: 1.2+ MB
```

```
#I just want to know how many  true and false values in 'is_active' column are there
df['is_active'].value_counts()
```

```
True     16352
False      319
Name: is_active, dtype: int64
```

```
#VISULAISATION
df['is_active'].value_counts().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcf1bf43910>
```




```
#Divide the data into input and output
#Input:
x = df.iloc[:,0:2].values
x
```

```
array([[-73.988281  ,  40.71883   ],
       [-74.00509   ,  40.728794  ],
       [-73.993408  ,  40.729197  ],
       ...,
       [  8.78705919,  53.10093429],
       [ 11.4090589 ,  53.6282266 ],
       [ 11.40599889,  53.90370139]])
```

```
#Output:
y = df.iloc[:,4].values
y
```

```
array([ True,  True,  True, ..., False, False, False])
```

```
#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)
```

```
(16671, 2)
(12503, 2)
(4168, 2)
```

```
print(y.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(16671,)
(12503,)
(4168,)
```

```
#NORMALISATION or SCALING
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)



#Apply Classifier/Regressor in the dataset
#Here I apply LogisticRegression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()


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

```
LogisticRegression()
```

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

```
array([ True,  True,  True, ...,  True,  True,  True])
```

```
y_test
```

```
array([ True,  True,  True, ...,  True,  True,  True])
```

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

```
98.24856046065258
```

```
#Individual Prediction
a = scaler.transform([[13.475643,52.514265]])
```

```
model.predict(a)
```

```
array([ True])
```

```
a
```

```
array([[0.99436863, 0.72699608]])
```

```
#End
```

#Major Project 2

#Choose any dataset of your choice and apply K Means Clustering

#Dataset-women-world-cup
#url-/content/womens-world-cup.csv

#Create data frame
import pandas as pd
df=pd.read_csv('/content/womens-world-cup.csv')
df

	id	squad	year	players	age	possesion	matches_played	starts	min_playing_time	minutes_played_90s	...	non_penalty_goals	penalty_kicks_made	penalty_kicks_attempted	yellow_cards	red_cards	goals_per_90	assists_per_90	goals_plus_assists_per_90	goals_minus_penalty_kicks_per_90	goals_plus_assists_minus_penalty_kicks_per_90
0	1	Argentina	2019	18	26.8	34.7	3	33	270	3.0	...	1	1	1	3.0	0.0	0.67	0.33	1.00	0.33	0.67
1	2	Australia	2019	18	25.4	61.3	4	44	390	4.3	...	8	0	1	2.0	0.0	1.85	0.92	2.77	1.85	2.77
2	3	Brazil	2019	18	29.7	51.5	4	44	390	4.3	...	5	2	3	7.0	0.0	1.62	0.69	2.31	1.15	1.85
3	4	Cameroon	2019	20	27.7	36.0	4	44	360	4.0	...	3	0	0	6.0	0.0	0.75	0.75	1.50	0.75	1.50
4	5	Canada	2019	16	27.0	63.0	4	44	360	4.0	...	4	0	1	2.0	0.0	1.00	0.75	1.75	1.00	1.75
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
131	132	New Zealand	1991	15	25.5	NaN	3	33	240	2.7	...	1	0	0	NaN	NaN	0.37	0.37	0.75	0.37	0.75
132	133	Nigeria	1991	17	18.2	NaN	3	33	240	2.7	...	0	0	0	NaN	NaN	0.00	0.00	0.00	0.00	0.00
133	134	Norway	1991	15	24.1	NaN	6	66	500	5.6	...	12	1	1	NaN	NaN	2.34	1.08	3.42	2.16	3.24
134	135	Sweden	1991	18	25.4	NaN	6	66	480	5.3	...	16	1	1	NaN	NaN	3.19	2.06	5.25	3.00	5.06
135	136	USA	1991	16	23.0	NaN	6	66	480	5.3	...	24	1	1	NaN	NaN	4.69	2.06	6.75	4.50	6.56

136 rows × 22 columns

df.shape
#Here 136 rows and 22 columns

(136, 22)

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 136 entries, 0 to 135
Data columns (total 22 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            136 non-null    int64
 1   squad                                         136 non-null    object
 2   year                                          136 non-null    int64
 3   players                                       136 non-null    int64
 4   age                                           136 non-null    float64
 5   possesion                                     96 non-null     float64
 6   matches_played                                136 non-null    int64
 7   starts                                        136 non-null    int64
 8   min_playing_time                              136 non-null    int64
 9   minutes_played_90s                            136 non-null    float64
 10  goals                                         136 non-null    int64
 11  assists                                       136 non-null    int64
 12  non_penalty_goals                             136 non-null    int64
 13  penalty_kicks_made                            136 non-null    int64
 14  penalty_kicks_attempted                       136 non-null    int64
 15  yellow_cards                                  62 non-null     float64
 16  red_cards                                     62 non-null     float64
 17  goals_per_90                                  136 non-null    float64
 18  assists_per_90                                136 non-null    float64
 19  goals_plus_assists_per_90                     136 non-null    float64
 20  goals_minus_penalty_kicks_per_90              136 non-null    float64
 21  goals_plus_assists_minus_penalty_kicks_per_90 136 non-null    float64
dtypes: float64(10), int64(11), object(1)
memory usage: 23.5+ KB
```

#I want to create a dataframe having numeric values only
df_numeric = df.select_dtypes(include = ['float64','int64'])
df_numeric

	id	year	players	age	possesion	matches_played	starts	min_playing_time	minutes_played_90s	goals	...	non_penalty_goals	penalty_kicks_made	penalty_kicks_attempted	yellow_cards	red_cards	goals_per_90	assists_per_90	goals_plus_assists_per_90	goals_minus_penalty
0	1	2019	18	26.8	34.7	3	33	270	3.0	2	...	1	1	1	3.0	0.0	0.67	0.33	1.00	
1	2	2019	18	25.4	61.3	4	44	390	4.3	8	...	8	0	1	2.0	0.0	1.85	0.92	2.77	
2	3	2019	18	29.7	51.5	4	44	390	4.3	7	...	5	2	3	7.0	0.0	1.62	0.69	2.31	
3	4	2019	20	27.7	36.0	4	44	360	4.0	3	...	3	0	0	6.0	0.0	0.75	0.75	1.50	
4	5	2019	16	27.0	63.0	4	44	360	4.0	4	...	4	0	1	2.0	0.0	1.00	0.75	1.75	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
131	132	1991	15	25.5	NaN	3	33	240	2.7	1	...	1	0	0	NaN	NaN	0.37	0.37	0.75	
132	133	1991	17	18.2	NaN	3	33	240	2.7	0	...	0	0	0	NaN	NaN	0.00	0.00	0.00	
133	134	1991	15	24.1	NaN	6	66	500	5.6	13	...	12	1	1	NaN	NaN	2.34	1.08	3.42	
134	135	1991	18	25.4	NaN	6	66	480	5.3	17	...	16	1	1	NaN	NaN	3.19	2.06	5.25	
135	136	1991	16	23.0	NaN	6	66	480	5.3	25	...	24	1	1	NaN	NaN	4.69	2.06	6.75	

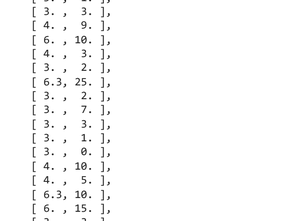
136 rows × 21 columns

#Input data-'minutes_played_90s' and 'goals'
#Divide the data ino input
#Slicing of input columns
x=df_numeric.iloc[:,8:10].values
x

```
array([[ 3. ,  3. ],
       [ 6. , 12. ],
       [ 3. ,  1. ],
       [ 3. ,  3. ],
       [ 4. ,  9. ],
       [ 6. , 10. ],
       [ 4. ,  3. ],
       [ 3. ,  2. ],
       [ 6.3, 25. ],
       [ 3. ,  2. ],
       [ 3. ,  7. ],
       [ 3. ,  3. ],
       [ 3. ,  1. ],
       [ 3. ,  0. ],
       [ 4. , 10. ],
       [ 4. ,  5. ],
       [ 6.3, 10. ],
       [ 6. , 15. ],
       [ 3. ,  3. ],
       [ 6.2, 16. ],
       [ 3. ,  3. ],
       [ 6.3, 19. ],
       [ 3. ,  1. ],
       [ 4. , 11. ],
       [ 3. ,  1. ],
       [ 3. ,  3. ],
       [ 3. ,  1. ],
       [ 3. ,  4. ],
       [ 3. ,  1. ],
       [ 4.2,  8. ],
       [ 6. , 15. ],
       [ 4. , 10. ],
       [ 4. ,  7. ],
       [ 6.3, 17. ],
       [ 3. ,  3. ],
       [ 3. ,  3. ],
       [ 3. ,  5. ],
       [ 6.3, 11. ],
       [ 4. ,  7. ],
       [ 4. ,  6. ],
       [ 6. , 13. ],
       [ 4. ,  2. ],
       [ 3. ,  5. ],
       [ 6. , 23. ],
       [ 4.3,  6. ],
       [ 6. , 15. ],
       [ 2.7,  1. ],
       [ 3.6, 10. ],
       [ 3.6,  2. ],
       [ 3.8,  7. ],
       [ 5.6, 13. ],
       [ 3.8,  8. ],
       [ 2.7,  0. ],
       [ 2.7,  1. ],
       [ 2.7,  0. ],
       [ 5.6, 13. ],
       [ 5.3, 17. ],
       [ 5.3, 25. ]])
```

#Visualisation before clustering
import matplotlib.pyplot as plt
plt.scatter(df_numeric['goals'],df_numeric['minutes_played_90s'])

<matplotlib.collections.PathCollection at 0x7fb88d0b87d0>



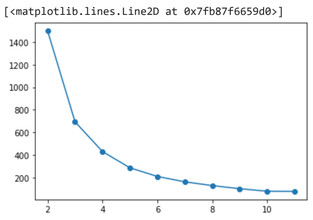
import numpy as np
np.sqrt(136)
#136 is the total number of points
#No of cluster-k

11.661903789690601

#Find out the value of k
#Two methods are-
#1.Elbow method
from sklearn.cluster import KMeans
k=range(2,12)
#Here k range is in b/w 2 and 12
sse=[]

```
for i in k:
  model_demo=KMeans(n_clusters=i,random_state=0)
  model_demo.fit(x)
  sse.append(model_demo.inertia_)
plt.scatter(k,sse)
plt.plot(k,sse)
```

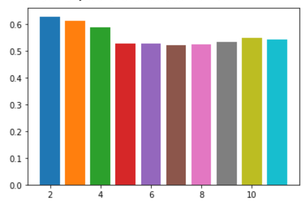
[<matplotlib.lines.Line2D at 0x7fb87f6659d0>]



#2.Silhouette score method to find out k value
from sklearn.metrics import silhouette_score
k=range(2,12)
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=0.6272354597096452
    3 Clusters ,Score=0.6098248740215009
    4 Clusters ,Score=0.5882779526243221
    5 Clusters ,Score=0.5267801634835054
    6 Clusters ,Score=0.5262100155208476
    7 Clusters ,Score=0.5210185765506407
    8 Clusters ,Score=0.5221903963947018
    9 Clusters ,Score=0.5332144053977441
    10 Clusters ,Score=0.5473552798282356
    11 Clusters ,Score=0.5408406521024972
```




```
#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)
```

```
#Output prediction
y=model.predict(x)
y
```

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

```
y.size
```

```
    136
```

```
x[y==0,1]
#the value of input when cluster 0 is selcted and column index 1 is selected
```

```
    array([13., 10., 10.,  9., 11., 12., 25.,  9., 10.,  9., 20.,  9.,  9.,
            9., 14.,  9., 10., 12., 10., 12.,  9., 16., 20., 12., 12.,  9.,
           10., 25., 10., 10., 15., 16., 19., 11., 15., 10., 17., 11., 13.,
           23., 15., 10., 13., 13., 17., 25.])
```

```
np.unique(y,return_counts=True)
```

```
    (array([0, 1], dtype=int32), array([46, 90]))
```

```
#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 0x7fb87cd2d710>
```



```
#End
```


Github Account Link: https://github.com/21051235/Major-Projects