Program:

**#Import Library**

import pandas as pd

import numpy as np

import warnings

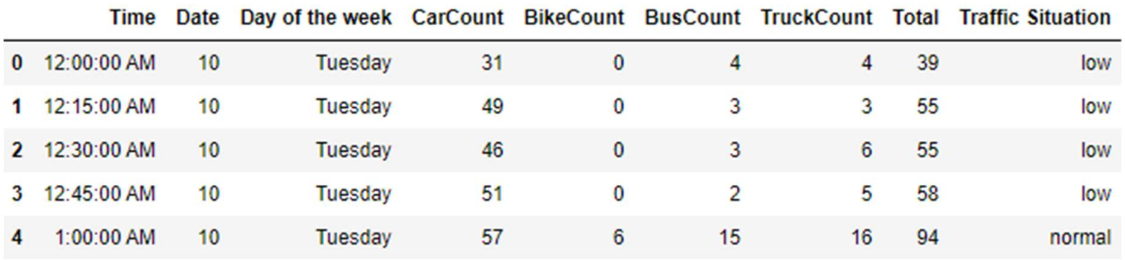
warnings.simplefilter("ignore")

**# Load the dataset**

df=pd.read_csv("D:\Sethu 45\Class prediction 04\Traffic.csv")

df.head()

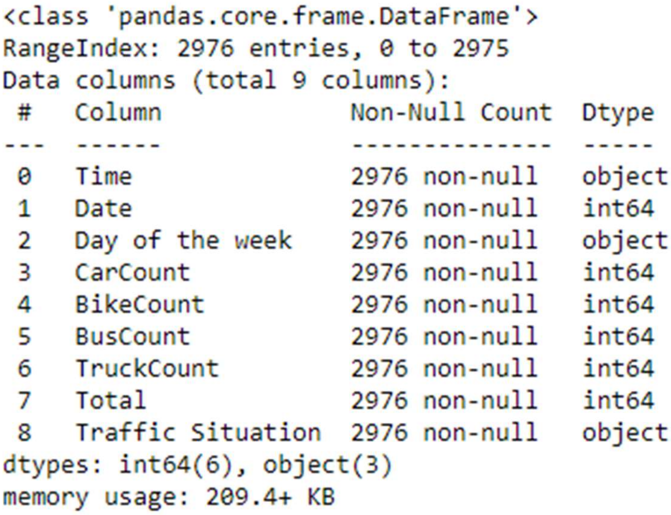
**Output:**



| | Time | Date | Day of the week | CarCount | BikeCount | BusCount | TruckCount | Total | Traffic Situation |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 12:00:00 AM | 10 | Tuesday | 31 | 0 | 4 | 4 | 39 | low |
| 1 | 12:15:00 AM | 10 | Tuesday | 49 | 0 | 3 | 3 | 55 | low |
| 2 | 12:30:00 AM | 10 | Tuesday | 46 | 0 | 3 | 6 | 55 | low |
| 3 | 12:45:00 AM | 10 | Tuesday | 51 | 0 | 2 | 5 | 58 | low |
| 4 | 1:00:00 AM | 10 | Tuesday | 57 | 6 | 15 | 16 | 94 | normal |

df.info()

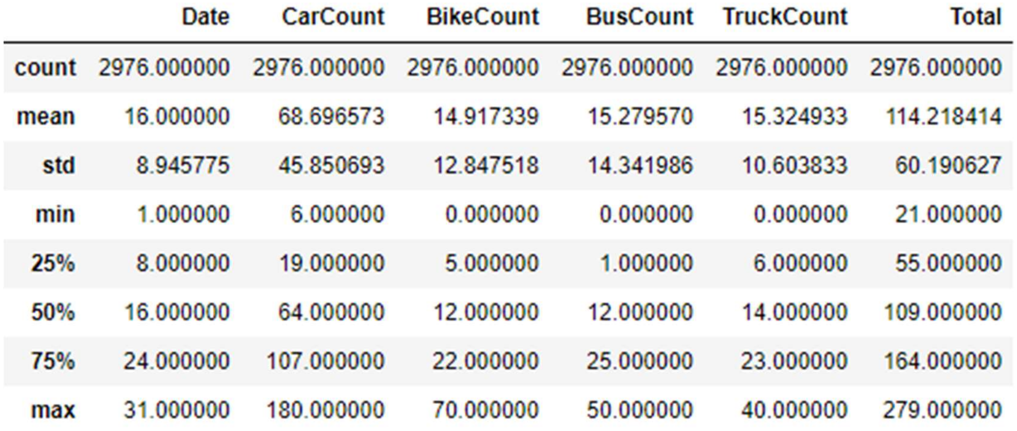
**Output:**



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2976 entries, 0 to 2975
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Time               2976 non-null   object
 1   Date               2976 non-null   int64 
 2   Day of the week    2976 non-null   object
 3   CarCount           2976 non-null   int64 
 4   BikeCount          2976 non-null   int64 
 5   BusCount           2976 non-null   int64 
 6   TruckCount         2976 non-null   int64 
 7   Total              2976 non-null   int64 
 8   Traffic Situation  2976 non-null   object
dtypes: int64(6), object(3)
memory usage: 209.4+ KB
```

df.describe()

**Output:**

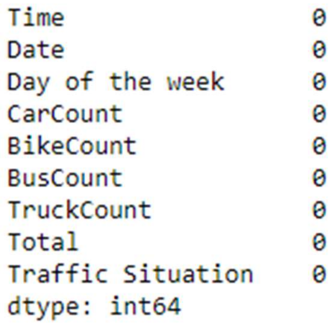


| | Date | CarCount | BikeCount | BusCount | TruckCount | Total |
|---|---|---|---|---|---|---|
| **count** | 2976.000000 | 2976.000000 | 2976.000000 | 2976.000000 | 2976.000000 | 2976.000000 |
| **mean** | 16.000000 | 68.696573 | 14.917339 | 15.279570 | 15.324933 | 114.218414 |
| **std** | 8.945775 | 45.850693 | 12.847518 | 14.341986 | 10.603833 | 60.190627 |
| **min** | 1.000000 | 6.000000 | 0.000000 | 0.000000 | 0.000000 | 21.000000 |
| **25%** | 8.000000 | 19.000000 | 5.000000 | 1.000000 | 6.000000 | 55.000000 |
| **50%** | 16.000000 | 64.000000 | 12.000000 | 12.000000 | 14.000000 | 109.000000 |
| **75%** | 24.000000 | 107.000000 | 22.000000 | 25.000000 | 23.000000 | 164.000000 |
| **max** | 31.000000 | 180.000000 | 70.000000 | 50.000000 | 40.000000 | 279.000000 |

**# Explore and preprocess the dataset**

df.isnull().sum()

**Output:**

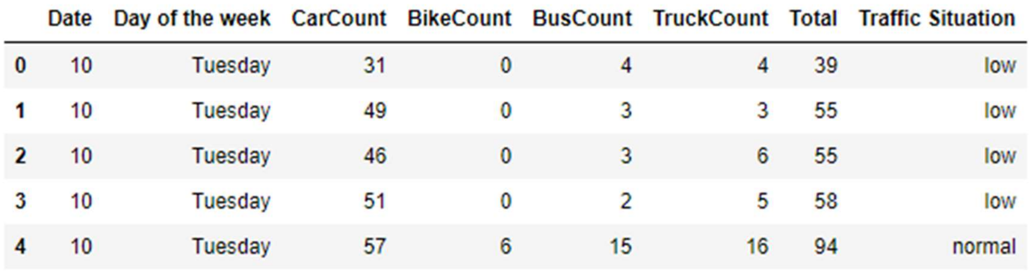


```
Time                 0
Date                 0
Day of the week      0
CarCount             0
BikeCount            0
BusCount             0
TruckCount           0
Total                0
Traffic Situation    0
dtype: int64
```

df=df.drop('Time',axis=1)

df.head()

**Output:**

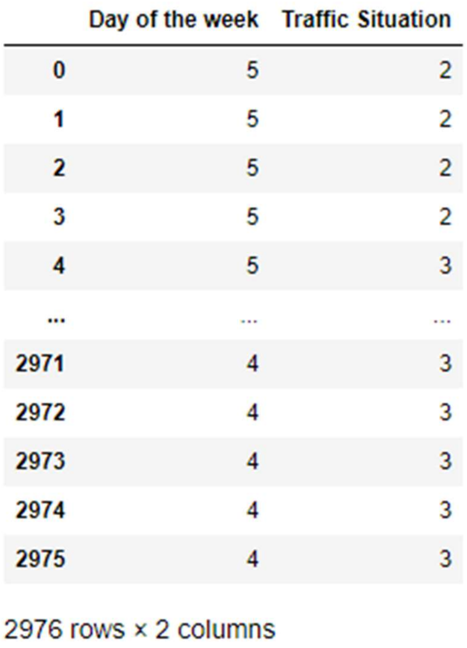


| | Date | Day of the week | CarCount | BikeCount | BusCount | TruckCount | Total | Traffic Situation |
|---|---|---|---|---|---|---|---|---|
| **0** | 10 | Tuesday | 31 | 0 | 4 | 4 | 39 | low |
| **1** | 10 | Tuesday | 49 | 0 | 3 | 3 | 55 | low |
| **2** | 10 | Tuesday | 46 | 0 | 3 | 6 | 55 | low |
| **3** | 10 | Tuesday | 51 | 0 | 2 | 5 | 58 | low |
| **4** | 10 | Tuesday | 57 | 6 | 15 | 16 | 94 | normal |

**# Encoding the categorical values**

```
from sklearn. preprocessing import LabelEncoder

label_encoder = LabelEncoder ()

categorical_columns = ['Day of the week', 'Traffic Situation']

df[categorical_columns]=df[categorical_columns].apply(label_encoder.fit_transform)

df[categorical_columns]
```

**Output:**



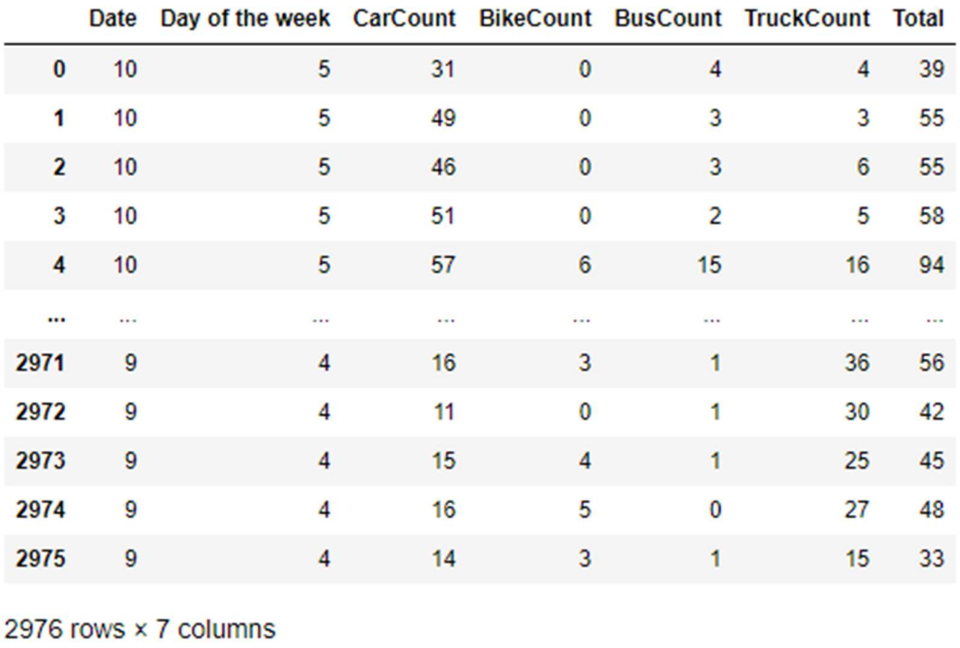
| | Day of the week | Traffic Situation |
|---|---|---|
| **0** | 5 | 2 |
| **1** | 5 | 2 |
| **2** | 5 | 2 |
| **3** | 5 | 2 |
| **4** | 5 | 3 |
| **...** | ... | ... |
| **2971** | 4 | 3 |
| **2972** | 4 | 3 |
| **2973** | 4 | 3 |
| **2974** | 4 | 3 |
| **2975** | 4 | 3 |

2976 rows × 2 columns

**#Split the dataset into features(x)**

```
x=df.iloc[:,:-1]

x
```

**Output:**



| | Date | Day of the week | CarCount | BikeCount | BusCount | TruckCount | Total |
|---|---|---|---|---|---|---|---|
| **0** | 10 | 5 | 31 | 0 | 4 | 4 | 39 |
| **1** | 10 | 5 | 49 | 0 | 3 | 3 | 55 |
| **2** | 10 | 5 | 46 | 0 | 3 | 6 | 55 |
| **3** | 10 | 5 | 51 | 0 | 2 | 5 | 58 |
| **4** | 10 | 5 | 57 | 6 | 15 | 16 | 94 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2971** | 9 | 4 | 16 | 3 | 1 | 36 | 56 |
| **2972** | 9 | 4 | 11 | 0 | 1 | 30 | 42 |
| **2973** | 9 | 4 | 15 | 4 | 1 | 25 | 45 |
| **2974** | 9 | 4 | 16 | 5 | 0 | 27 | 48 |
| **2975** | 9 | 4 | 14 | 3 | 1 | 15 | 33 |

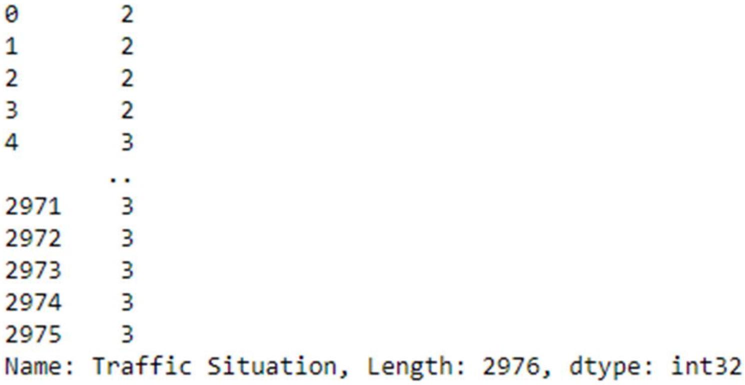
2976 rows × 7 columns

**#Split the variables into target variables(y)**

y=df.iloc[:,-1]

y

**Output:**



```
0       2
1       2
2       2
3       2
4       3
       ..
2971    3
2972    3
2973    3
2974    3
2975    3
Name: Traffic Situation, Length: 2976, dtype: int32
```

**# Further split the dataset into training and testing sets**

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=0)

**# Feature scaling the data**

from sklearn. preprocessing import StandardScaler

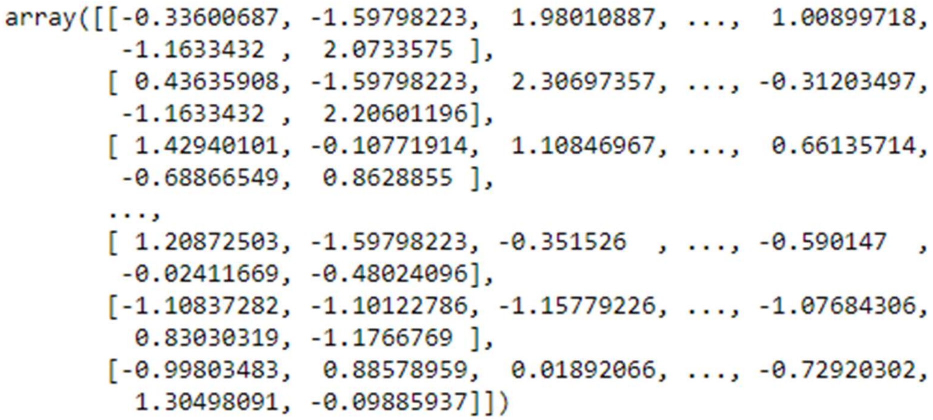
scaler = StandardScaler ()

x_train = scaler.fit_transform(x_train)

x_test = scaler. transform(x_test)

x_train

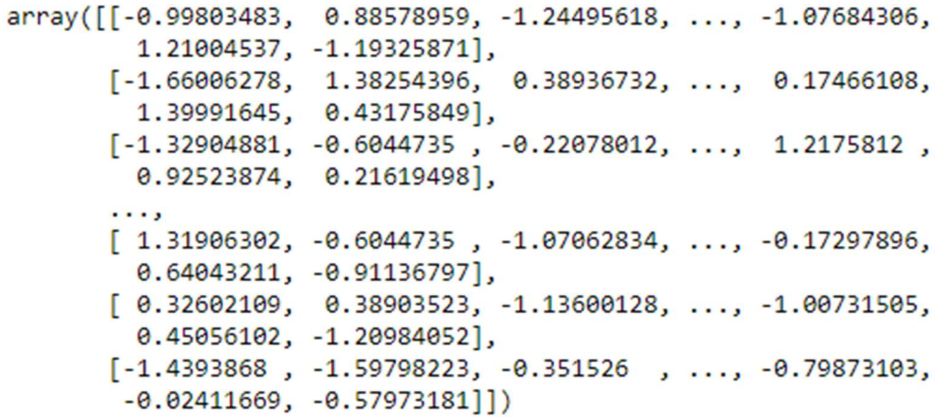
**Output:**



```
array([[-0.33600687, -1.59798223,  1.98010887, ...,  1.00899718,
        -1.1633432 ,  2.0733575 ],
       [ 0.43635908, -1.59798223,  2.30697357, ..., -0.31203497,
        -1.1633432 ,  2.20601196],
       [ 1.42940101, -0.10771914,  1.10846967, ...,  0.66135714,
        -0.68866549,  0.8628855 ],
       ...,
       [ 1.20872503, -1.59798223, -0.351526  , ..., -0.590147  ,
        -0.02411669, -0.48024096],
       [-1.10837282, -1.10122786, -1.15779226, ..., -1.07684306,
         0.83030319, -1.1766769 ],
       [-0.99803483,  0.88578959,  0.01892066, ..., -0.72920302,
         1.30498091, -0.09885937]])
```

x_test

**Output:**



```
array([[-0.99803483,  0.88578959, -1.24495618, ..., -1.07684306,
         1.21004537, -1.19325871],
       [-1.66006278,  1.38254396,  0.38936732, ...,  0.17466108,
         1.39991645,  0.43175849],
       [-1.32904881, -0.6044735 , -0.22078012, ...,  1.2175812 ,
         0.92523874,  0.21619498],
       ...,
       [ 1.31906302, -0.6044735 , -1.07062834, ..., -0.17297896,
         0.64043211, -0.91136797],
       [ 0.32602109,  0.38903523, -1.13600128, ..., -1.00731505,
         0.45056102, -1.20984052],
       [-1.4393868 , -1.59798223, -0.351526  , ..., -0.79873103,
        -0.02411669, -0.57973181]])
```

**#Initialise the Logistic Regression Model**

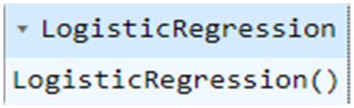
from sklearn. linear_model import LogisticRegression

model = LogisticRegression ()

**#Train the Logistic Regression Model**

model.fit(x_train,y_train)

**Output:**



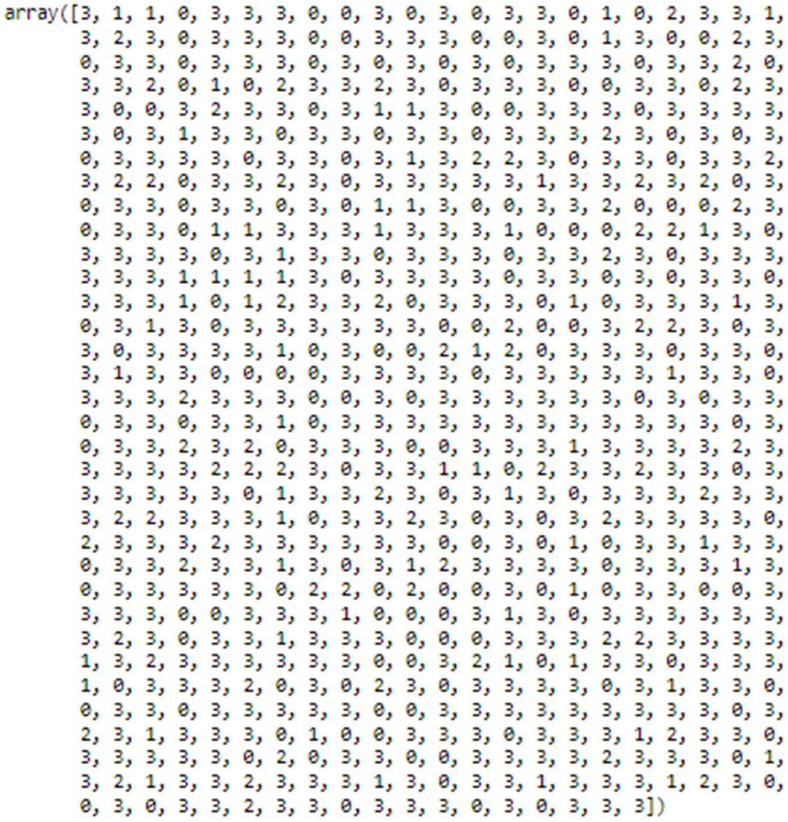
```
▾ LogisticRegression
LogisticRegression()
```

**#Make predictions**

y_pred=model. predict(x_test)

y_pred

**Output:**



```
array([3, 1, 1, 0, 3, 3, 3, 0, 0, 3, 0, 3, 0, 3, 3, 0, 1, 0, 2, 3, 3, 1,
       3, 2, 3, 0, 3, 3, 3, 0, 0, 3, 3, 3, 0, 0, 3, 0, 1, 3, 0, 0, 2, 3,
       0, 3, 3, 0, 3, 3, 3, 0, 3, 0, 3, 0, 3, 0, 3, 3, 3, 0, 3, 3, 2, 0,
       3, 3, 2, 0, 1, 0, 2, 3, 3, 2, 3, 0, 3, 3, 3, 0, 0, 3, 3, 0, 2, 3,
       3, 0, 0, 3, 2, 3, 3, 0, 3, 1, 1, 3, 0, 0, 3, 3, 3, 0, 3, 3, 3, 3,
       3, 0, 3, 1, 3, 3, 0, 3, 3, 0, 3, 3, 0, 3, 3, 3, 2, 3, 0, 3, 0, 3,
       0, 3, 3, 3, 3, 0, 3, 3, 0, 3, 1, 3, 2, 2, 3, 0, 3, 3, 0, 3, 3, 2,
       3, 2, 2, 0, 3, 3, 2, 3, 0, 3, 3, 3, 3, 3, 1, 3, 3, 2, 3, 2, 0, 3,
       0, 3, 3, 0, 3, 3, 0, 3, 0, 1, 1, 3, 0, 0, 3, 3, 2, 0, 0, 0, 2, 3,
       0, 3, 3, 0, 1, 1, 3, 3, 3, 1, 3, 3, 3, 1, 0, 0, 0, 2, 2, 1, 3, 0,
       3, 3, 3, 3, 0, 3, 1, 3, 3, 0, 3, 3, 3, 0, 3, 3, 2, 3, 0, 3, 3, 3,
       3, 3, 3, 1, 1, 1, 1, 3, 0, 3, 3, 3, 3, 0, 3, 3, 0, 3, 0, 3, 3, 0,
       3, 3, 3, 1, 0, 1, 2, 3, 3, 2, 0, 3, 3, 3, 0, 1, 0, 3, 3, 3, 1, 3,
       0, 3, 1, 3, 0, 3, 3, 3, 3, 3, 3, 0, 0, 2, 0, 0, 3, 2, 2, 3, 0, 3,
       3, 0, 3, 3, 3, 3, 1, 0, 3, 0, 0, 2, 1, 2, 0, 3, 3, 3, 0, 3, 3, 0,
       3, 1, 3, 3, 0, 0, 0, 0, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 1, 3, 3, 0,
       3, 3, 3, 2, 3, 3, 3, 0, 0, 3, 0, 3, 3, 3, 3, 3, 3, 0, 3, 0, 3, 3,
       0, 3, 3, 0, 3, 3, 1, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3,
       0, 3, 3, 2, 3, 2, 0, 3, 3, 3, 0, 0, 3, 3, 3, 1, 3, 3, 3, 3, 2, 3,
       3, 3, 3, 3, 2, 2, 2, 3, 0, 3, 3, 1, 1, 0, 2, 3, 3, 2, 3, 3, 0, 3,
       3, 3, 3, 3, 3, 0, 1, 3, 3, 2, 3, 0, 3, 1, 3, 0, 3, 3, 3, 2, 3, 3,
       3, 2, 2, 3, 3, 3, 1, 0, 3, 3, 2, 3, 0, 3, 0, 3, 2, 3, 3, 3, 3, 0,
       2, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 0, 0, 3, 0, 1, 0, 3, 3, 1, 3, 3,
       0, 3, 3, 2, 3, 3, 1, 3, 0, 3, 1, 2, 3, 3, 3, 3, 0, 3, 3, 3, 1, 3,
       0, 3, 3, 3, 3, 3, 0, 2, 2, 0, 2, 0, 0, 3, 0, 1, 0, 3, 3, 0, 0, 3,
       3, 3, 3, 0, 0, 3, 3, 3, 1, 0, 0, 0, 3, 1, 3, 0, 3, 3, 3, 3, 3, 3,
       3, 2, 3, 0, 3, 3, 1, 3, 3, 3, 0, 0, 0, 3, 3, 3, 2, 2, 3, 3, 3, 3,
       1, 3, 2, 3, 3, 3, 3, 3, 3, 0, 0, 3, 2, 1, 0, 1, 3, 3, 0, 3, 3, 3,
       1, 0, 3, 3, 3, 2, 0, 3, 0, 2, 3, 0, 3, 3, 3, 3, 0, 3, 1, 3, 3, 0,
       0, 3, 3, 0, 3, 3, 3, 3, 3, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3,
       2, 3, 1, 3, 3, 3, 0, 1, 0, 0, 3, 3, 3, 0, 3, 3, 3, 1, 2, 3, 3, 0,
       3, 3, 3, 3, 3, 0, 2, 0, 3, 3, 0, 0, 3, 3, 3, 3, 2, 3, 3, 3, 0, 1,
       3, 2, 1, 3, 3, 2, 3, 3, 3, 1, 3, 0, 3, 3, 1, 3, 3, 3, 1, 2, 3, 0,
       0, 3, 0, 3, 3, 2, 3, 3, 0, 3, 3, 3, 0, 3, 0, 3, 3, 3])
```

**#Evaluate the model**

from sklearn. metrics import accuracy_score, classification_report

accuracy = accuracy_score (y_test, y_pred) * 100

print ("Accuracy of the model is {:.2f}". format(accuracy))

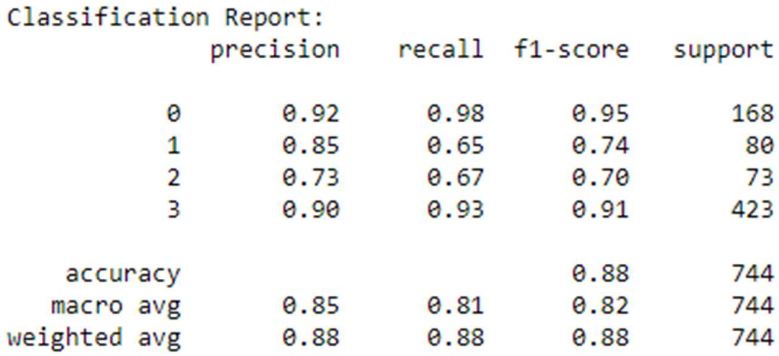
**Output:**

```
Accuracy of the model is 88.44
```

**# Classification Report**

```
from sklearn. metrics import classification_report
class_report = classification_report (y_test, y_pred)
print (f"\nClassification Report:\n{class_report}")
```

**Output:**



```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       168
           1       0.85      0.65      0.74        80
           2       0.73      0.67      0.70        73
           3       0.90      0.93      0.91       423

    accuracy                           0.88       744
   macro avg       0.85      0.81      0.82       744
weighted avg       0.88      0.88      0.88       744
```

**#Predict for new data**

```
new_data = [9, 5, 6, 3, 2, 5, 45]
predictions = model. predict([new_data])
print(predictions)
```

**Output:**

```
[0]
```

**# Decoded the output of prediction**

```
decoded_predictions = label_encoder.inverse_transform(predictions)
print(decoded_predictions)
```

**Output:**

```
['heavy']
```

Program:

**#Import Library**

import pandas as pd

import numpy as np

import warnings

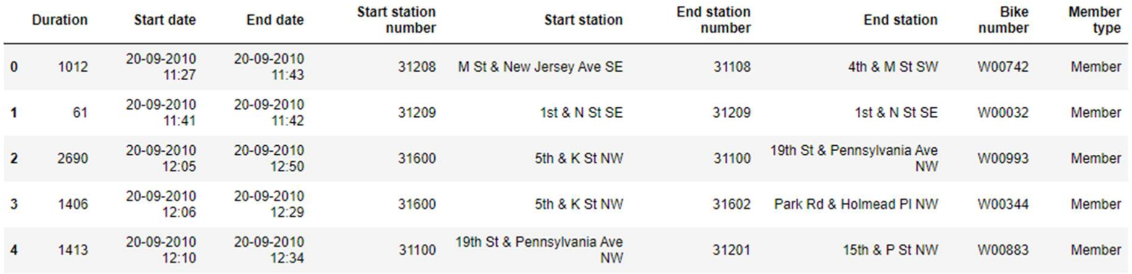
warnings.simplefilter("ignore")

**# Load the dataset**

df=pd.read_csv("D:\Sethu 45\Class prediction 04\Trip.csv")

df.head()

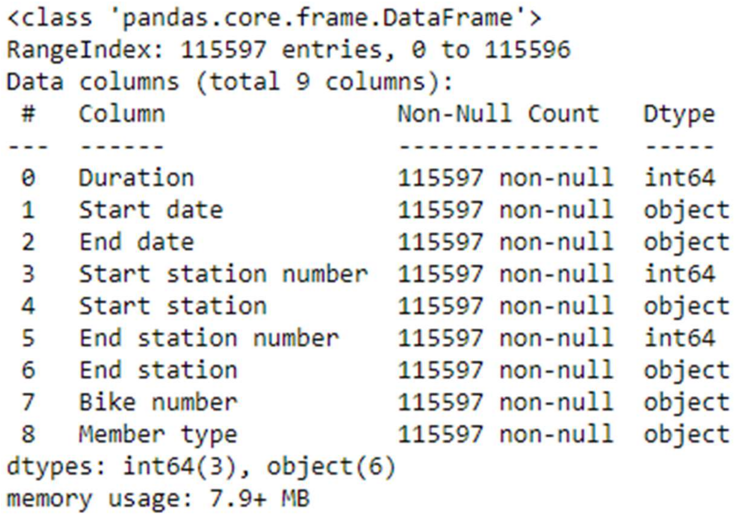
**Output:**



|  | Duration | Start date | End date | Start station number | Start station | End station number | End station | Bike number | Member type |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1012 | 20-09-2010 11:27 | 20-09-2010 11:43 | 31208 | M St & New Jersey Ave SE | 31108 | 4th & M St SW | W00742 | Member |
| **1** | 61 | 20-09-2010 11:41 | 20-09-2010 11:42 | 31209 | 1st & N St SE | 31209 | 1st & N St SE | W00032 | Member |
| **2** | 2690 | 20-09-2010 12:05 | 20-09-2010 12:50 | 31600 | 5th & K St NW | 31100 | 19th St & Pennsylvania Ave NW | W00993 | Member |
| **3** | 1406 | 20-09-2010 12:06 | 20-09-2010 12:29 | 31600 | 5th & K St NW | 31602 | Park Rd & Holmead Pl NW | W00344 | Member |
| **4** | 1413 | 20-09-2010 12:10 | 20-09-2010 12:34 | 31100 | 19th St & Pennsylvania Ave NW | 31201 | 15th & P St NW | W00883 | Member |

df.info()

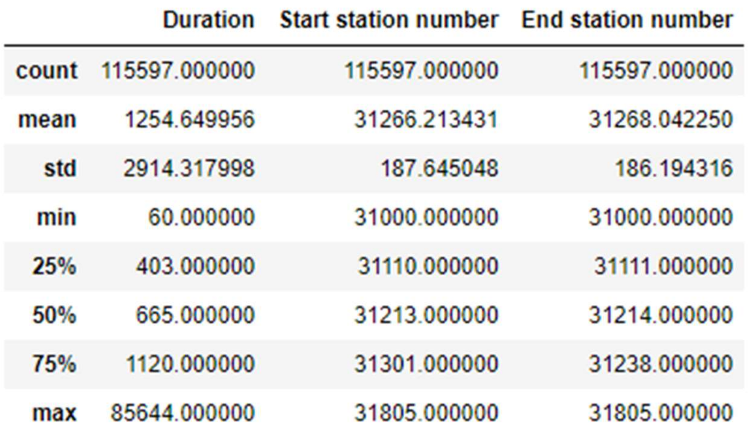
**Output:**



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115597 entries, 0 to 115596
Data columns (total 9 columns):
 #   Column                Non-Null Count   Dtype 
---  ------                --------------   ----- 
 0   Duration              115597 non-null  int64 
 1   Start date            115597 non-null  object
 2   End date              115597 non-null  object
 3   Start station number  115597 non-null  int64 
 4   Start station         115597 non-null  object
 5   End station number    115597 non-null  int64 
 6   End station           115597 non-null  object
 7   Bike number           115597 non-null  object
 8   Member type           115597 non-null  object
dtypes: int64(3), object(6)
memory usage: 7.9+ MB
```

df.describe()

**Output:**

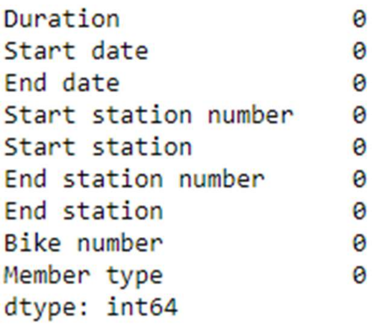


| | Duration | Start station number | End station number |
|---|---|---|---|
| **count** | 115597.000000 | 115597.000000 | 115597.000000 |
| **mean** | 1254.649956 | 31266.213431 | 31268.042250 |
| **std** | 2914.317998 | 187.645048 | 186.194316 |
| **min** | 60.000000 | 31000.000000 | 31000.000000 |
| **25%** | 403.000000 | 31110.000000 | 31111.000000 |
| **50%** | 665.000000 | 31213.000000 | 31214.000000 |
| **75%** | 1120.000000 | 31301.000000 | 31238.000000 |
| **max** | 85644.000000 | 31805.000000 | 31805.000000 |

**# Explore and preprocess the dataset**

df.isnull().sum()

**Output:**

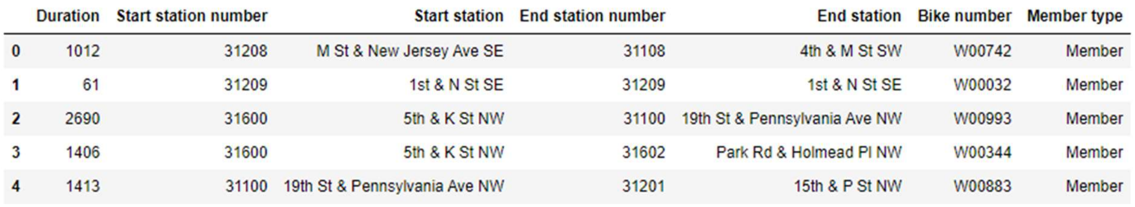


```
Duration                0
Start date              0
End date                0
Start station number    0
Start station           0
End station number      0
End station             0
Bike number             0
Member type             0
dtype: int64
```

df=df.drop(['Start date','End date'],axis=1)

df.head()

**Output:**

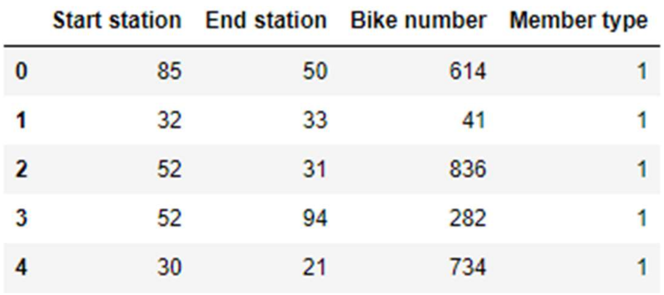


| | Duration | Start station number | Start station | End station number | End station | Bike number | Member type |
|---|---|---|---|---|---|---|---|
| **0** | 1012 | 31208 | M St & New Jersey Ave SE | 31108 | 4th & M St SW | W00742 | Member |
| **1** | 61 | 31209 | 1st & N St SE | 31209 | 1st & N St SE | W00032 | Member |
| **2** | 2690 | 31600 | 5th & K St NW | 31100 | 19th St & Pennsylvania Ave NW | W00993 | Member |
| **3** | 1406 | 31600 | 5th & K St NW | 31602 | Park Rd & Holmead Pl NW | W00344 | Member |
| **4** | 1413 | 31100 | 19th St & Pennsylvania Ave NW | 31201 | 15th & P St NW | W00883 | Member |

**# Encoding the categorical values**

```
from sklearn.preprocessing import LabelEncoder

label_encoder=LabelEncoder()

categorical_columns=['Start station','End station','Bike number','Member type']

df[categorical_columns]=df[categorical_columns].apply(label_encoder.fit_trans
form)

df[categorical_columns].head()
```

**Output:**

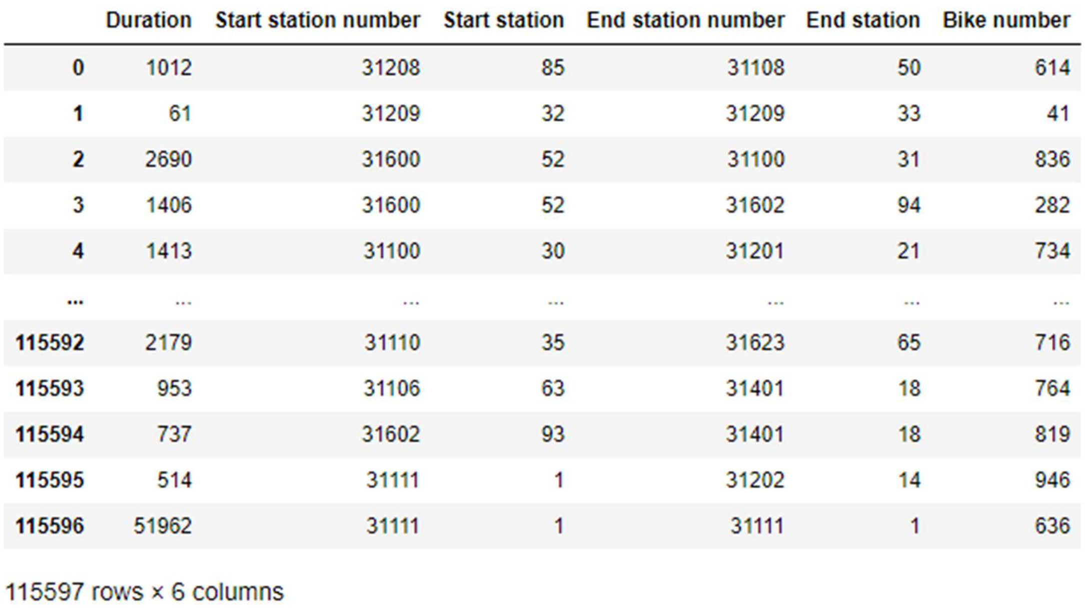


| | Start station | End station | Bike number | Member type |
|---|---|---|---|---|
| **0** | 85 | 50 | 614 | 1 |
| **1** | 32 | 33 | 41 | 1 |
| **2** | 52 | 31 | 836 | 1 |
| **3** | 52 | 94 | 282 | 1 |
| **4** | 30 | 21 | 734 | 1 |

**#Split the dataset into features(x)**

```
x=df.iloc[:,:-1]

x
```

**Output:**

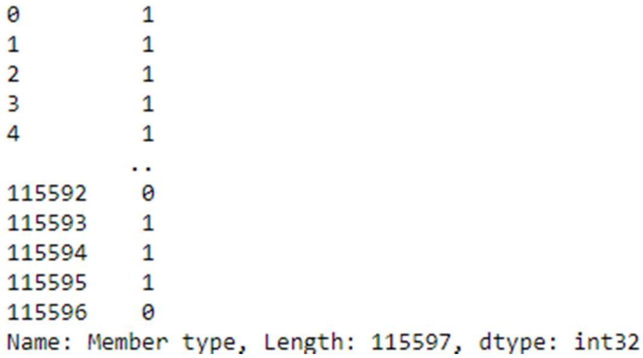


| | Duration | Start station number | Start station | End station number | End station | Bike number |
|---|---|---|---|---|---|---|
| **0** | 1012 | 31208 | 85 | 31108 | 50 | 614 |
| **1** | 61 | 31209 | 32 | 31209 | 33 | 41 |
| **2** | 2690 | 31600 | 52 | 31100 | 31 | 836 |
| **3** | 1406 | 31600 | 52 | 31602 | 94 | 282 |
| **4** | 1413 | 31100 | 30 | 31201 | 21 | 734 |
| **...** | ... | ... | ... | ... | ... | ... |
| **115592** | 2179 | 31110 | 35 | 31623 | 65 | 716 |
| **115593** | 953 | 31106 | 63 | 31401 | 18 | 764 |
| **115594** | 737 | 31602 | 93 | 31401 | 18 | 819 |
| **115595** | 514 | 31111 | 1 | 31202 | 14 | 946 |
| **115596** | 51962 | 31111 | 1 | 31111 | 1 | 636 |

115597 rows × 6 columns

```
#Split the variables into target variables(y)
y=df.iloc[:,-1]
y
```

**Output:**

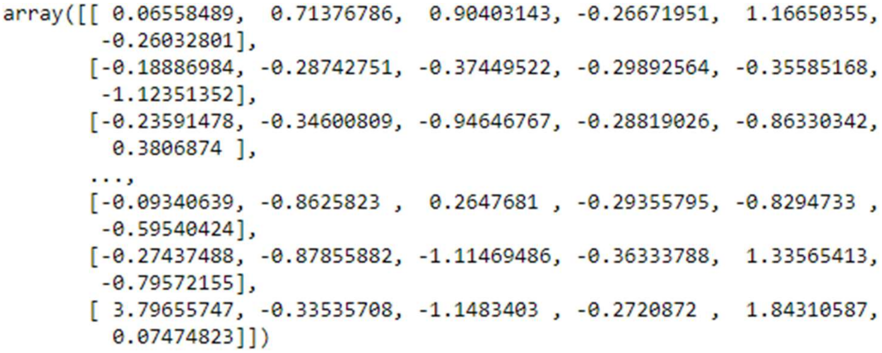


```
0         1
1         1
2         1
3         1
4         1
         ..
115592    0
115593    1
115594    1
115595    1
115596    0
Name: Member type, Length: 115597, dtype: int32
```

```
# Further split the dataset into training and testing sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=365)
```

```
# Feature scaling the data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
x_train
```

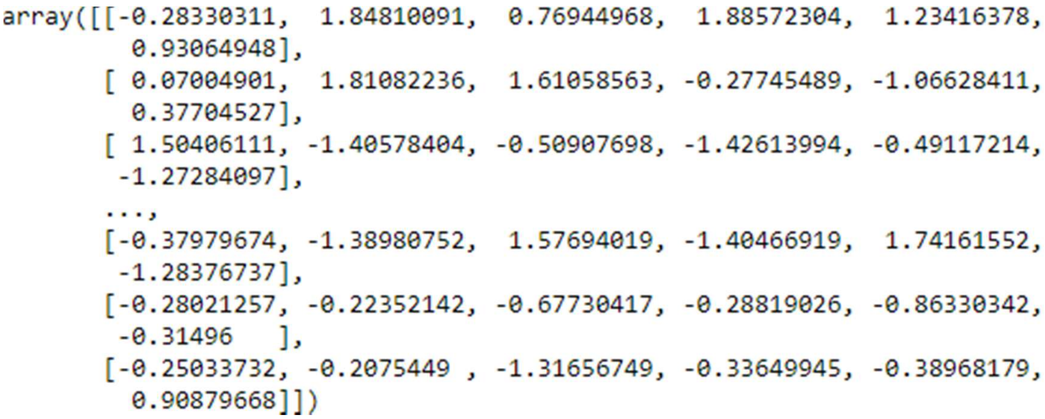
**Output:**



```
array([[ 0.06558489,  0.71376786,  0.90403143, -0.26671951,  1.16650355,
        -0.26032801],
       [-0.18886984, -0.28742751, -0.37449522, -0.29892564, -0.35585168,
        -1.12351352],
       [-0.23591478, -0.34600809, -0.94646767, -0.28819026, -0.86330342,
         0.3806874 ],
       ...,
       [-0.09340639, -0.8625823 ,  0.2647681 , -0.29355795, -0.8294733 ,
        -0.59540424],
       [-0.27437488, -0.87855882, -1.11469486, -0.36333788,  1.33565413,
        -0.79572155],
       [ 3.79655747, -0.33535708, -1.1483403 , -0.2720872 ,  1.84310587,
         0.07474823]])
```

x_test

**Output:**



```
array([[-0.28330311,  1.84810091,  0.76944968,  1.88572304,  1.23416378,
         0.93064948],
       [ 0.07004901,  1.81082236,  1.61058563, -0.27745489, -1.06628411,
         0.37704527],
       [ 1.50406111, -1.40578404, -0.50907698, -1.42613994, -0.49117214,
        -1.27284097],
       ...,
       [-0.37979674, -1.38980752,  1.57694019, -1.40466919,  1.74161552,
        -1.28376737],
       [-0.28021257, -0.22352142, -0.67730417, -0.28819026, -0.86330342,
        -0.31496   ],
       [-0.25033732, -0.2075449 , -1.31656749, -0.33649945, -0.38968179,
         0.90879668]])
```

**#Initialise the Bernoulli Naïve Bayes Model**

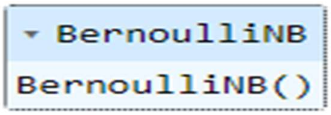
from sklearn.naive_bayes import BernoulliNB

model = BernoulliNB ()

**#Train the Bernoulli Naïve Bayes Model**

model.fit(x_train,y_train)

**Output:**



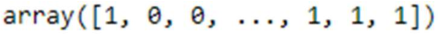
```
▾ BernoulliNB
BernoulliNB()
```

**#Make predictions**

y_pred=model.predict(x_test)

y_pred

**Output:**



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

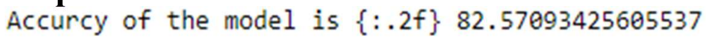
**#Evaluate the model**

from sklearn.metrics import accuracy_score,classification_report

accuracy=accuracy_score(y_test,y_pred)*100

```
print("Accurcy of the model is {:.2f}", format(accuracy))
```

**Output:**

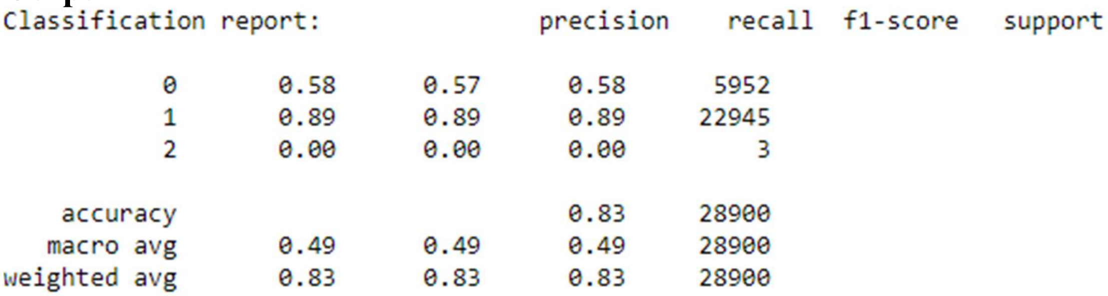


```
Accurcy of the model is {:.2f} 82.57093425605537
```

**# Classification Report**

```
from  sklearn.metrics import classification_report

class_report=classification_report(y_test,y_pred)

print(f"Classification report: {class_report}")
```

**Output:**

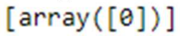


```
Classification report:               precision    recall  f1-score   support

           0       0.58      0.57      0.58      5952
           1       0.89      0.89      0.89     22945
           2       0.00      0.00      0.00         3

    accuracy                           0.83     28900
   macro avg       0.49      0.49      0.49     28900
weighted avg       0.83      0.83      0.83     28900
```

**#Predict for new data**

```
new_data=[737,31110,63,31623,18,636]

predictions=model.predict([new_data])

print([predictions])
```

**Output:**

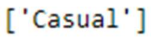


```
[array([0])]
```

**# Decoded the output of prediction**

```
decoded_predictions=label_encoder.inverse_transform([predictions])

print(decoded_predictions)
```

**Output:**



```
['Casual']
```

**PROGRAM**

**# Import Libraries**

import pandas as pd

import numpy as np

import warnings

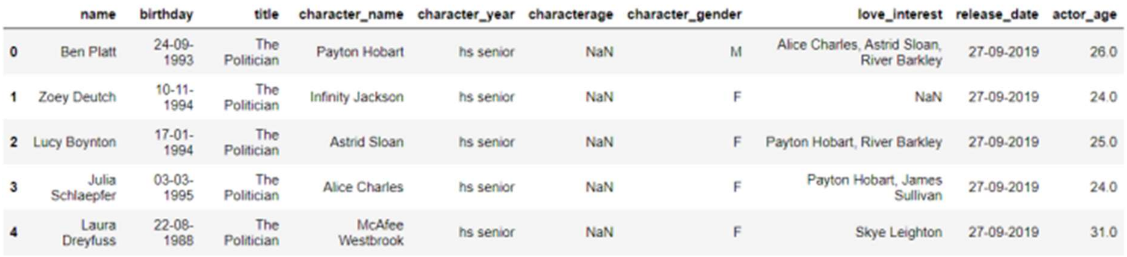
warnings. Simplefilter ("ignore")

**# Load the Dataset**

df = pd. read_csv("C:\\Users\\Documents\\age.csv")

df. head ()

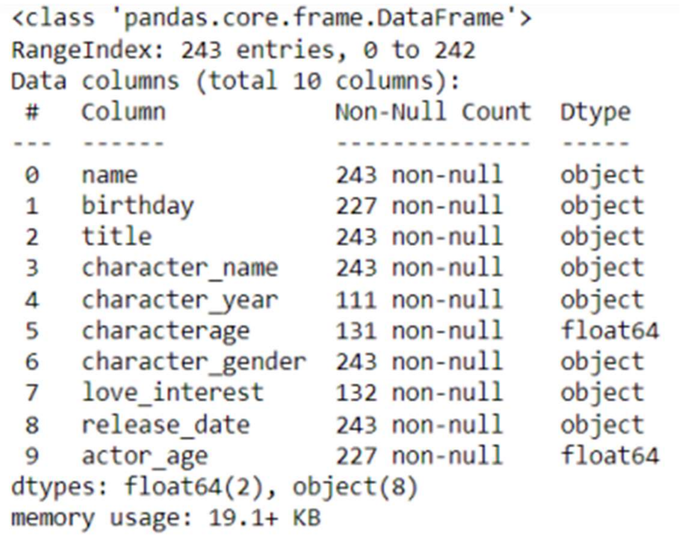
**Output**



| | name | birthday | title | character_name | character_year | characterage | character_gender | love_interest | release_date | actor_age |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Ben Platt | 24-09-1993 | The Politician | Payton Hobart | hs senior | NaN | M | Alice Charles, Astrid Sloan, River Barkley | 27-09-2019 | 26.0 |
| 1 | Zoey Deutch | 10-11-1994 | The Politician | Infinity Jackson | hs senior | NaN | F | NaN | 27-09-2019 | 24.0 |
| 2 | Lucy Boynton | 17-01-1994 | The Politician | Astrid Sloan | hs senior | NaN | F | Payton Hobart, River Barkley | 27-09-2019 | 25.0 |
| 3 | Julia Schlaepfer | 03-03-1995 | The Politician | Alice Charles | hs senior | NaN | F | Payton Hobart, James Sullivan | 27-09-2019 | 24.0 |
| 4 | Laura Dreyfuss | 22-08-1988 | The Politician | McAfee Westbrook | hs senior | NaN | F | Skye Leighton | 27-09-2019 | 31.0 |

df.info ()

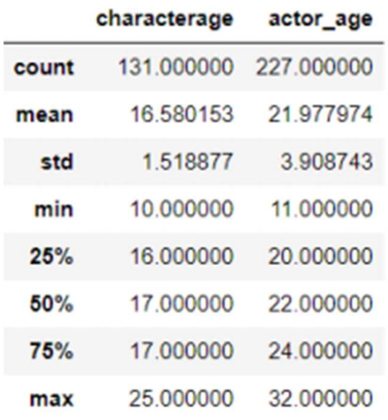
**Output**



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              243 non-null    object
 1   birthday          227 non-null    object
 2   title             243 non-null    object
 3   character_name    243 non-null    object
 4   character_year    111 non-null    object
 5   characterage      131 non-null    float64
 6   character_gender  243 non-null    object
 7   love_interest     132 non-null    object
 8   release_date      243 non-null    object
 9   actor_age         227 non-null    float64
dtypes: float64(2), object(8)
memory usage: 19.1+ KB
```

df. describe ()

**Output**

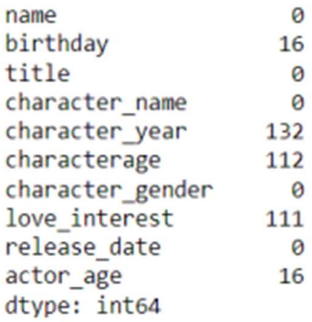


| | characterage | actor_age |
|---|---|---|
| **count** | 131.000000 | 227.000000 |
| **mean** | 16.580153 | 21.977974 |
| **std** | 1.518877 | 3.908743 |
| **min** | 10.000000 | 11.000000 |
| **25%** | 16.000000 | 20.000000 |
| **50%** | 17.000000 | 22.000000 |
| **75%** | 17.000000 | 24.000000 |
| **max** | 25.000000 | 32.000000 |

**# Explore and preprocess the data**

df. isnull (). sum ()

**Output**



```
name                 0
birthday            16
title                0
character_name       0
character_year     132
characterage       112
character_gender     0
love_interest      111
release_date         0
actor_age           16
dtype: int64
```

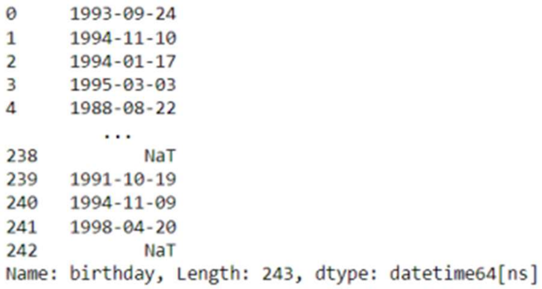
**# Preprocessing**

**# Extract year from birthday**

df['birthday'] = pd.to_datetime(df['birthday'])

df['birthday']

**Output**

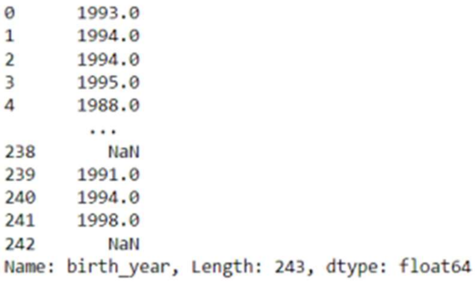


```
0      1993-09-24
1      1994-11-10
2      1994-01-17
3      1995-03-03
4      1988-08-22
          ...
238           NaT
239    1991-10-19
240    1994-11-09
241    1998-04-20
242           NaT
Name: birthday, Length: 243, dtype: datetime64[ns]
```

df['birth_year'] = df['birthday'].dt. year

df['birth_year']

**Output**



```
0      1993.0
1      1994.0
2      1994.0
3      1995.0
4      1988.0
        ...
238       NaN
239    1991.0
240    1994.0
241    1998.0
242       NaN
Name: birth_year, Length: 243, dtype: float64
```

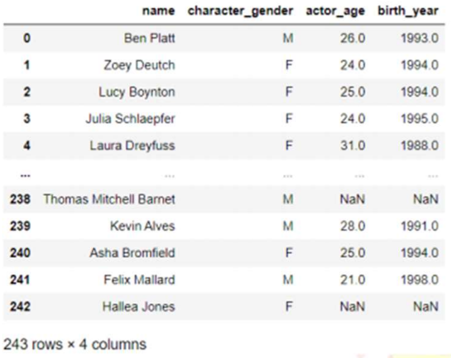
**# Drop the null values**

df = df. drop (['birthday', 'title','character_name', 'character_year', 'characterage', 'love_interest', 'release_date'],

axis=1)

df

**Output**



| | name | character_gender | actor_age | birth_year |
|---|---|---|---|---|
| **0** | Ben Platt | M | 26.0 | 1993.0 |
| **1** | Zoey Deutch | F | 24.0 | 1994.0 |
| **2** | Lucy Boynton | F | 25.0 | 1994.0 |
| **3** | Julia Schlaepfer | F | 24.0 | 1995.0 |
| **4** | Laura Dreyfuss | F | 31.0 | 1988.0 |
| **...** | ... | ... | ... | ... |
| **238** | Thomas Mitchell Barnet | M | NaN | NaN |
| **239** | Kevin Alves | M | 28.0 | 1991.0 |
| **240** | Asha Bromfield | F | 25.0 | 1994.0 |
| **241** | Felix Mallard | M | 21.0 | 1998.0 |
| **242** | Hallea Jones | F | NaN | NaN |

243 rows × 4 columns

**# Encoding the categorical values**

from sklearn. preprocessing import LabelEncoder

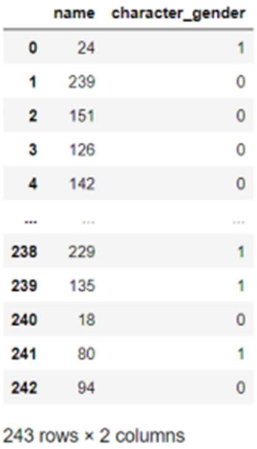
label_encoder = LabelEncoder ()

categorical_columns = ['name', 'character_gender']

df[categorical_columns] = df[categorical_columns].
apply(label_encoder.fit_transform)

df[categorical_columns]

**Output**



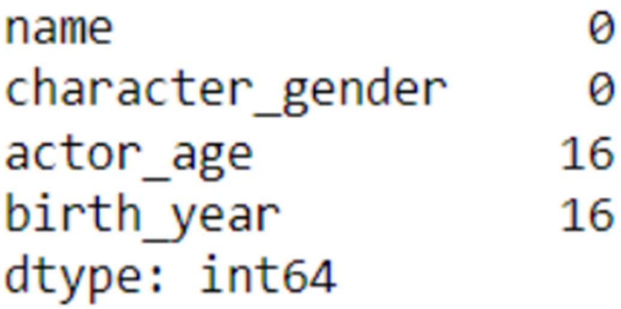
| | name | character_gender |
|---|---|---|
| 0 | 24 | 1 |
| 1 | 239 | 0 |
| 2 | 151 | 0 |
| 3 | 126 | 0 |
| 4 | 142 | 0 |
| ... | ... | ... |
| 238 | 229 | 1 |
| 239 | 135 | 1 |
| 240 | 18 | 0 |
| 241 | 80 | 1 |
| 242 | 94 | 0 |

243 rows × 2 columns

**# Explore the null values in the data**

df. isnull (). sum ()

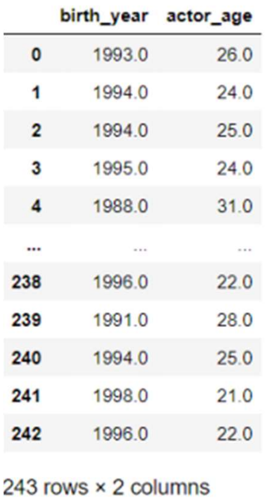
**Output**



```
name                 0
character_gender     0
actor_age           16
birth_year          16
dtype: int64
```

```
# Replace the null values

from sklearn. impute import SimpleImputer

numeric_columns = ['birth_year','actor_age']

df[numeric_columns] = SimpleImputer(strategy='most_frequent').
fit_transform(df[numeric_columns])

df[numeric_columns]
```

**Output**

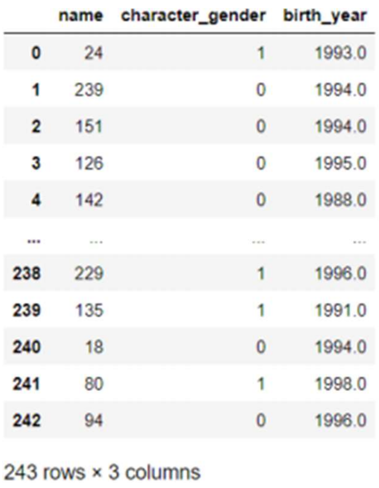


| | birth_year | actor_age |
|---|---|---|
| 0 | 1993.0 | 26.0 |
| 1 | 1994.0 | 24.0 |
| 2 | 1994.0 | 25.0 |
| 3 | 1995.0 | 24.0 |
| 4 | 1988.0 | 31.0 |
| ... | ... | ... |
| 238 | 1996.0 | 22.0 |
| 239 | 1991.0 | 28.0 |
| 240 | 1994.0 | 25.0 |
| 241 | 1998.0 | 21.0 |
| 242 | 1996.0 | 22.0 |

243 rows × 2 columns

```
# Split the data into features (x)

x = df[['name', 'character_gender', 'birth_year']]

x
```

**Output**



| | name | character_gender | birth_year |
|---|---|---|---|
| 0 | 24 | 1 | 1993.0 |
| 1 | 239 | 0 | 1994.0 |
| 2 | 151 | 0 | 1994.0 |
| 3 | 126 | 0 | 1995.0 |
| 4 | 142 | 0 | 1988.0 |
| ... | ... | ... | ... |
| 238 | 229 | 1 | 1996.0 |
| 239 | 135 | 1 | 1991.0 |
| 240 | 18 | 0 | 1994.0 |
| 241 | 80 | 1 | 1998.0 |
| 242 | 94 | 0 | 1996.0 |

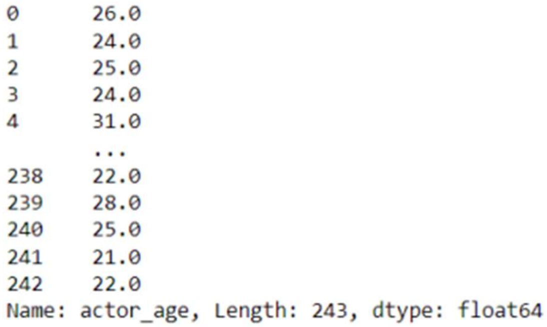
243 rows × 3 columns

**# Split the data into target variable (y)**

y = df['actor_age']

y

**Output**



```
0      26.0
1      24.0
2      25.0
3      24.0
4      31.0
       ...
238    22.0
239    28.0
240    25.0
241    21.0
242    22.0
Name: actor_age, Length: 243, dtype: float64
```

**# Further split the dataset into training and testing sets**

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=0.2, random_state=42)

**# Intialize the Linear Regression model**

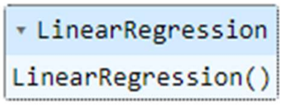
from sklearn. linear_model import LinearRegression

model = LinearRegression ()

**# Train the Linear Regression model**

model.fit (x_train, y_train)

**Output**



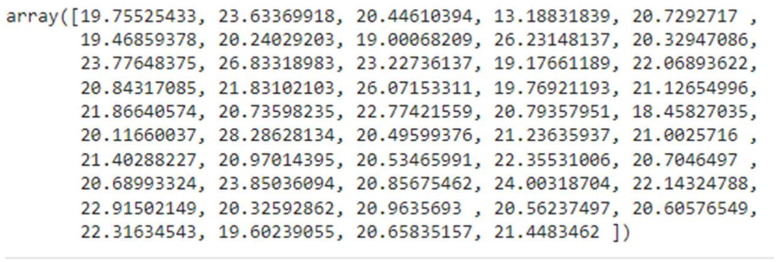
```
▾ LinearRegression
LinearRegression()
```

**# Make predictions**

y_pred = model. predict(x_test)

y_pred

**Output**



```
array([19.75525433, 23.63369918, 20.44610394, 13.18831839, 20.7292717 ,
       19.46859378, 20.24029203, 19.00068209, 26.23148137, 20.32947086,
       23.77648375, 26.83318983, 23.22736137, 19.17661189, 22.06893622,
       20.84317085, 21.83102103, 26.07153311, 19.76921193, 21.12654996,
       21.86640574, 20.73598235, 22.77421559, 20.79357951, 18.45827035,
       20.11660037, 28.28628134, 20.49599376, 21.23635937, 21.0025716 ,
       21.40288227, 20.97014395, 20.53465991, 22.35531006, 20.7046497 ,
       20.68993324, 23.85036094, 20.85675462, 24.00318704, 22.14324788,
       22.91502149, 20.32592862, 20.9635693 , 20.56237497, 20.60576549,
       22.31634543, 19.60239055, 20.65835157, 21.4483462 ])
```

```
# Evaluate the model
from sklearn. metrics import mean_squared_error
mse = mean_squared_error (y_test, y_pred)
print (f"Mean Squared Error: {mse}")
```

**Output**

```
Mean Squared Error: 8.89497933910581
```

```
# Predict the new data
new_data = pd. DataFrame ({"name": [142], "character_gender":
[1],'characterage':[17], "b_year": [1996.0]})
predicted_age = model. predict(new_data)
print (f"Predicted Actor Age: {predicted_age [0]}")
```

**Output**

```
Predicted Actor Age: 21.037446484786074
```