DATE : 15 9 2021

Complex Engineering Problem (CEP)

Group Members:

- Aqsa Saif (CS-18001)
- Syeda Nabiza Batool (CS-18016)

Year & Batch: TE, 2018

Subject: Machine Learning (CS-324)

Department of Computer and Information Systems

NED University of Engineering and Technology

Submitted to

Miss Hameeza Ahmed

DATE : 15 9 2021

Classification

Dataset: Bike Share Toronto Ridership

About dataset:

In this dataset, we have the bike sharing information .The Bike Share Toronto Ridership data contains anonymized trip data, including:

- Trip start day and time,
- Trip end day and time,
- Trip duration,
- Trip start station,
- Trip end station,
- User type

And we've to determined **User type** of the Rider .

Target : User type is categorical variable having two values

Casual Member

Annual Member

Features : There are 5 independent variable .

Trip start day and time,

Trip end day and time,

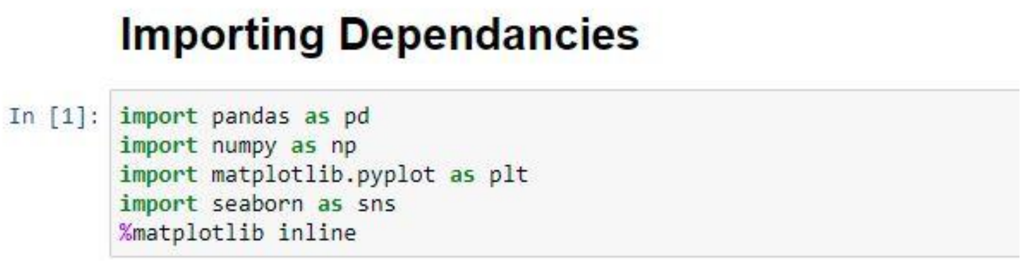
Trip duration,

Trip start station,

Trip end station

MACHINE LEARNING LIFECYCLE

Step #1 : First step in Machine learning cycle is to import all the dependencies .



Importing Dependancies

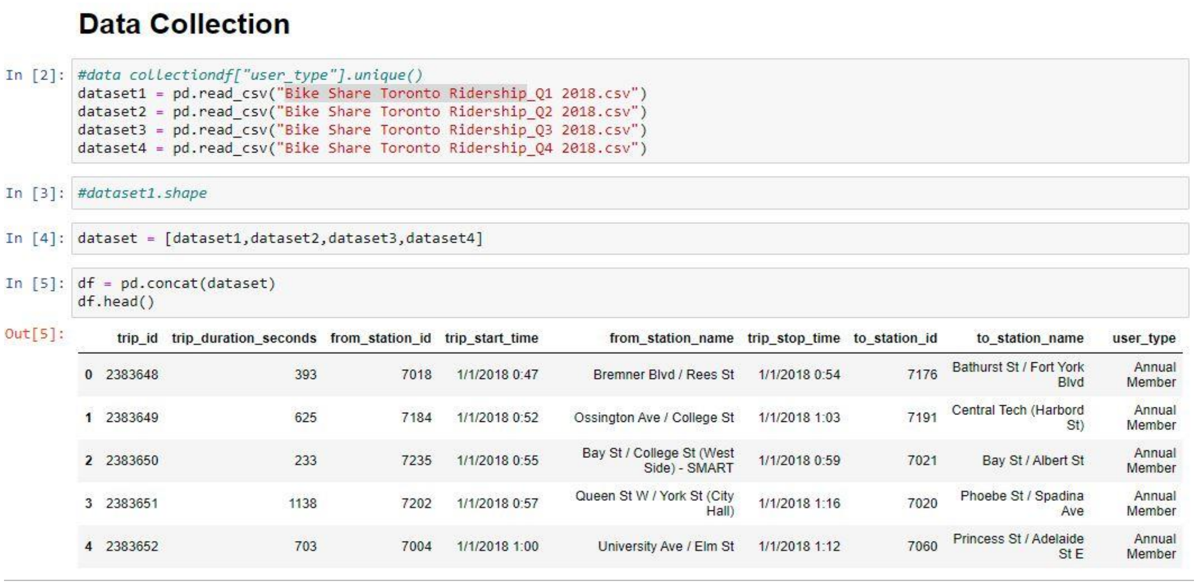
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

DATE : 15 9 2021

To complete this task , we use python programming and it's libraries NumPy, Pandas, Matplotlib, and Seaborn.

- **pandas** is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.
- **NumPy** is the fundamental package for scientific computing in Python.
- **matplotlib**. pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc .
- **Seaborn** is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Step #2 : Here we are gathering the dataset by reading csv files



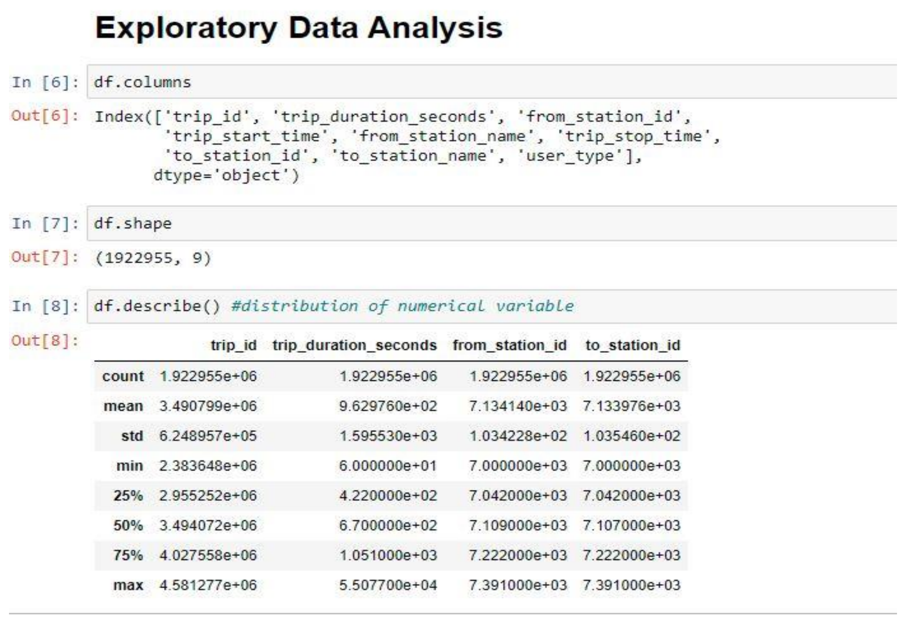
Data Collection

```
In [2]: #data collectiondf["user_type"].unique()
        dataset1 = pd.read_csv("Bike Share Toronto Ridership_Q1 2018.csv")
        dataset2 = pd.read_csv("Bike Share Toronto Ridership_Q2 2018.csv")
        dataset3 = pd.read_csv("Bike Share Toronto Ridership_Q3 2018.csv")
        dataset4 = pd.read_csv("Bike Share Toronto Ridership_Q4 2018.csv")

In [3]: #dataset1.shape

In [4]: dataset = [dataset1,dataset2,dataset3,dataset4]

In [5]: df = pd.concat(dataset)
        df.head()

Out[5]:
```

	trip_id	trip_duration_seconds	from_station_id	trip_start_time	from_station_name	trip_stop_time	to_station_id	to_station_name	user_type
0	2383648	393	7018	1/1/2018 0:47	Bremner Blvd / Rees St	1/1/2018 0:54	7176	Bathurst St / Fort York Blvd	Annual Member
1	2383649	625	7184	1/1/2018 0:52	Ossington Ave / College St	1/1/2018 1:03	7191	Central Tech (Harbord St)	Annual Member
2	2383650	233	7235	1/1/2018 0:55	Bay St / College St (West Side) - SMART	1/1/2018 0:59	7021	Bay St / Albert St	Annual Member
3	2383651	1138	7202	1/1/2018 0:57	Queen St W / York St (City Hall)	1/1/2018 1:16	7020	Phoebe St / Spadina Ave	Annual Member
4	2383652	703	7004	1/1/2018 1:00	University Ave / Elm St	1/1/2018 1:12	7060	Princess St / Adelaide St E	Annual Member

DATE : 15 9 2021

Step #3 : Exploratory data analysis (EDA) is used by to analyze and investigate data sets and summarize their main characteristics i.e : about the shape of our dataset , datatype of all the variables .



Exploratory Data Analysis

```
In [6]: df.columns
```

```
Out[6]: Index(['trip_id', 'trip_duration_seconds', 'from_station_id',
               'trip_start_time', 'from_station_name', 'trip_stop_time',
               'to_station_id', 'to_station_name', 'user_type'],
              dtype='object')
```

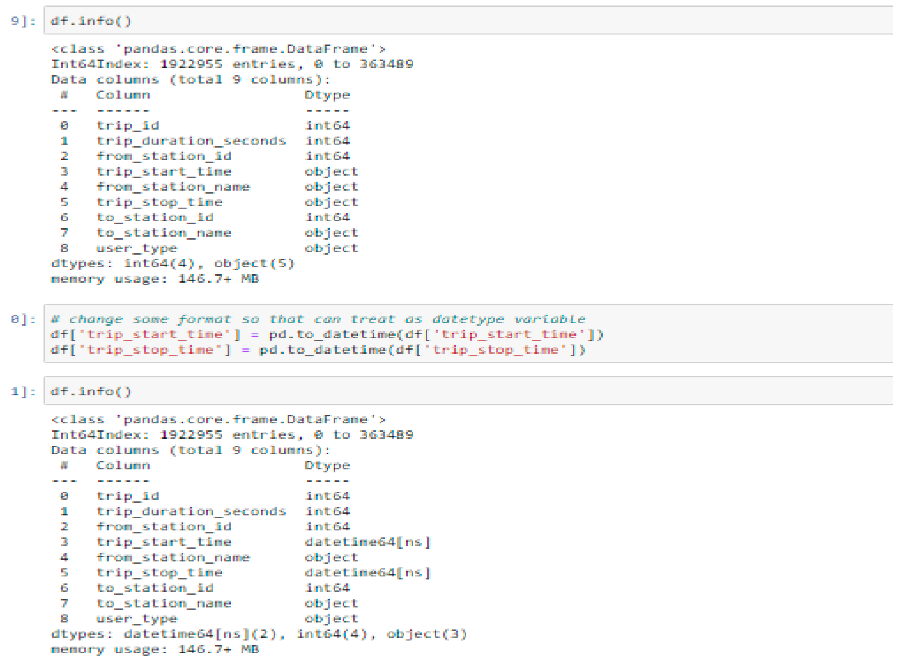
```
In [7]: df.shape
```

```
Out[7]: (1922955, 9)
```

```
In [8]: df.describe() #distribution of numerical variable
```

Out[8]:

	trip_id	trip_duration_seconds	from_station_id	to_station_id
count	1.922955e+06	1.922955e+06	1.922955e+06	1.922955e+06
mean	3.490799e+06	9.629760e+02	7.134140e+03	7.133976e+03
std	6.248957e+05	1.595530e+03	1.034228e+02	1.035460e+02
min	2.383648e+06	6.000000e+01	7.000000e+03	7.000000e+03
25%	2.955252e+06	4.220000e+02	7.042000e+03	7.042000e+03
50%	3.494072e+06	6.700000e+02	7.109000e+03	7.107000e+03
75%	4.027558e+06	1.051000e+03	7.222000e+03	7.222000e+03
max	4.581277e+06	5.507700e+04	7.391000e+03	7.391000e+03



```
9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1922955 entries, 0 to 363489
Data columns (total 9 columns):
 #   Column                 Dtype
---  ------                 -----
 0   trip_id                int64
 1   trip_duration_seconds  int64
 2   from_station_id        int64
 3   trip_start_time        object
 4   from_station_name      object
 5   trip_stop_time         object
 6   to_station_id          int64
 7   to_station_name        object
 8   user_type              object
dtypes: int64(4), object(5)
memory usage: 146.7+ MB
```

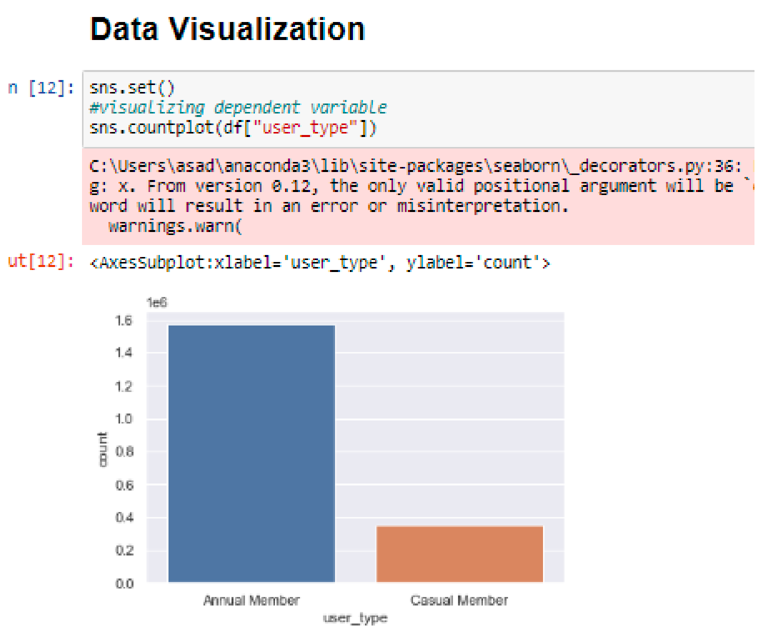
```
0]: # change some format so that can treat as datetype variable
    df['trip_start_time'] = pd.to_datetime(df['trip_start_time'])
    df['trip_stop_time'] = pd.to_datetime(df['trip_stop_time'])
```

```
1]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1922955 entries, 0 to 363489
Data columns (total 9 columns):
 #   Column                 Dtype
---  ------                 -----
 0   trip_id                int64
 1   trip_duration_seconds  int64
 2   from_station_id        int64
 3   trip_start_time        datetime64[ns]
 4   from_station_name      object
 5   trip_stop_time         datetime64[ns]
 6   to_station_id          int64
 7   to_station_name        object
 8   user_type              object
dtypes: datetime64[ns](2), int64(4), object(3)
memory usage: 146.7+ MB
```

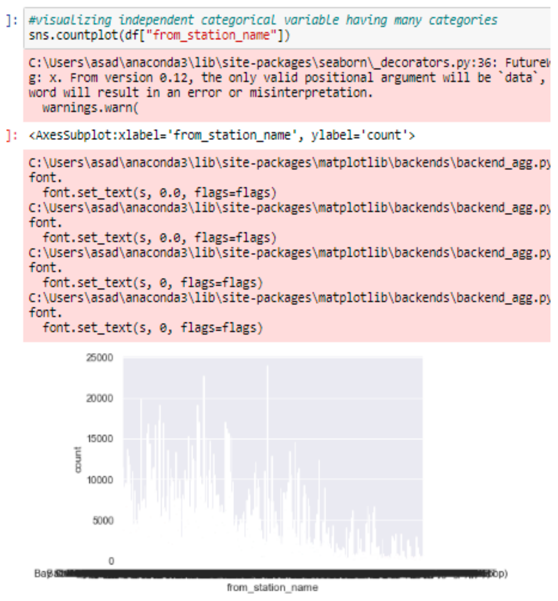
DATE : 15 9 2021

Step# 4 : Data visualization is the process of translating large data sets and metrics into charts, graphs and other visuals. The resulting visual representation of data makes it easier to identify outliers, and new insights about the information represented in the data.With the help of data visualization, we can see how the data looks like.

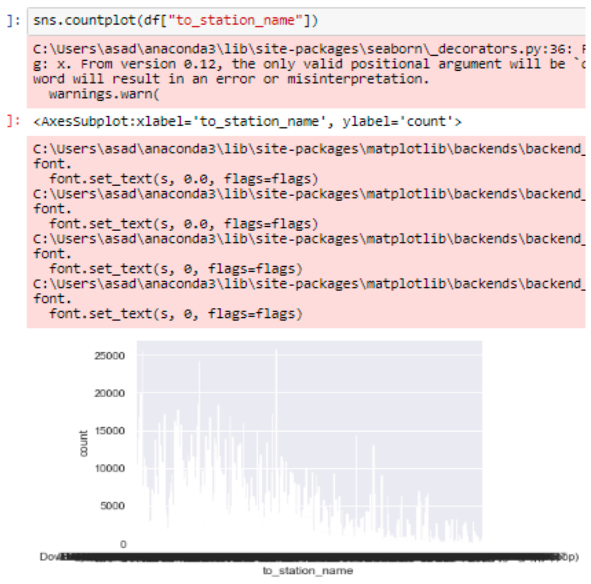


Data Visualization

```
n [12]: sns.set()
        #visualizing dependent variable
        sns.countplot(df["user_type"])

        C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
        g: x. From version 0.12, the only valid positional argument will be `
        word will result in an error or misinterpretation.
          warnings.warn(

ut[12]: <AxesSubplot:xlabel='user_type', ylabel='count'>
```

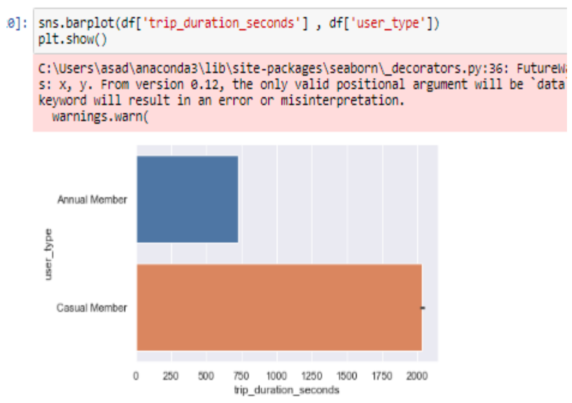


```
]: #visualizing independent categorical variable having many categories
   sns.countplot(df["from_station_name"])

   C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
   g: x. From version 0.12, the only valid positional argument will be `data`,
   word will result in an error or misinterpretation.
     warnings.warn(

]: <AxesSubplot:xlabel='from_station_name', ylabel='count'>

   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py
   font.
     font.set_text(s, 0.0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py
   font.
     font.set_text(s, 0.0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py
   font.
     font.set_text(s, 0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py
   font.
     font.set_text(s, 0, flags=flags)
```



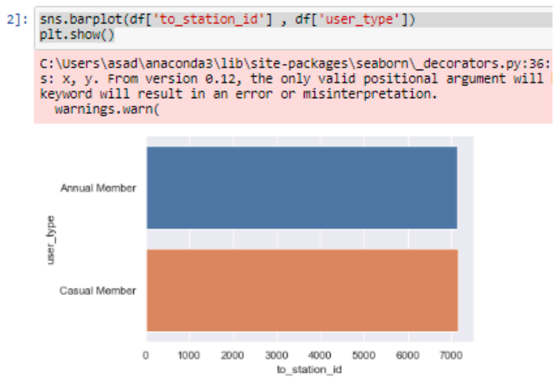
```
]: sns.countplot(df["to_station_name"])

   C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36: F
   g: x. From version 0.12, the only valid positional argument will be `c
   word will result in an error or misinterpretation.
     warnings.warn(

]: <AxesSubplot:xlabel='to_station_name', ylabel='count'>

   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_
   font.
     font.set_text(s, 0.0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_
   font.
     font.set_text(s, 0.0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_
   font.
     font.set_text(s, 0, flags=flags)
   C:\Users\asad\anaconda3\lib\site-packages\matplotlib\backends\backend_
   font.
     font.set_text(s, 0, flags=flags)
```

DATE : 15 9 2021



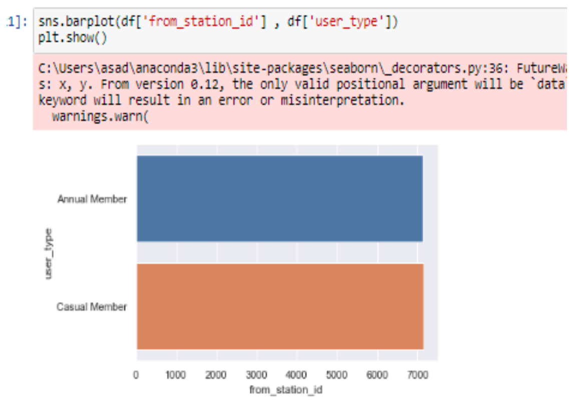
```
0]: sns.barplot(df['trip_duration_seconds'] , df['user_type'])
    plt.show()
```

```
C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
s: x, y. From version 0.12, the only valid positional argument will be `data
keyword will result in an error or misinterpretation.
  warnings.warn(
```



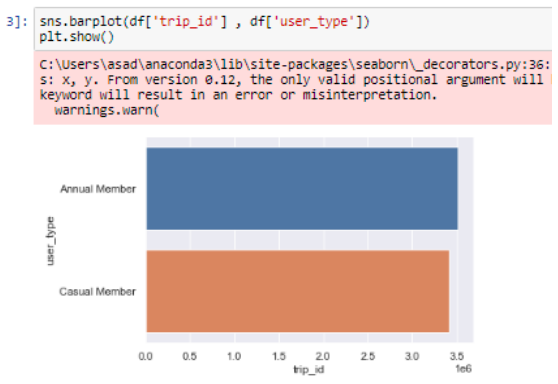
```
2]: sns.barplot(df['to_station_id'] , df['user_type'])
    plt.show()
```

```
C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
s: x, y. From version 0.12, the only valid positional argument will
keyword will result in an error or misinterpretation.
  warnings.warn(
```



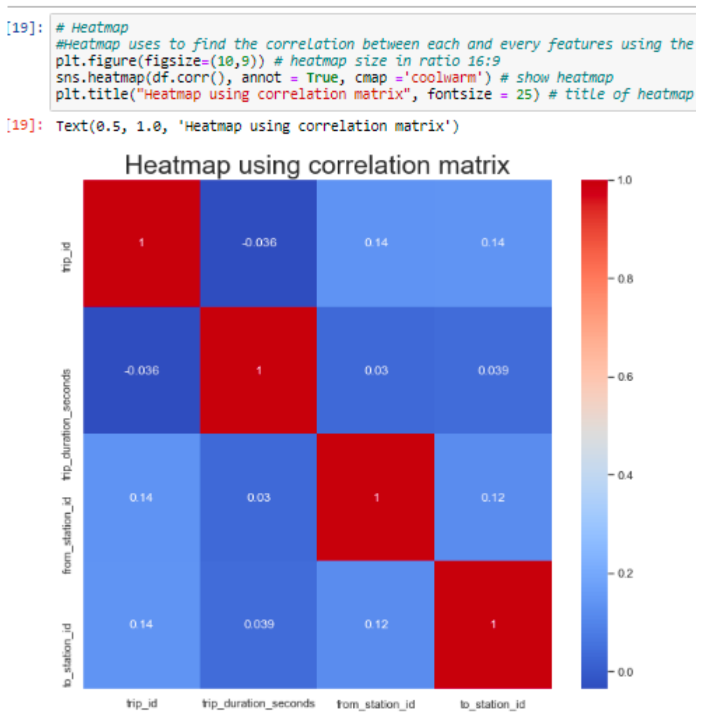
```
1]: sns.barplot(df['from_station_id'] , df['user_type'])
    plt.show()
```

```
C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
s: x, y. From version 0.12, the only valid positional argument will be `data
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
3]: sns.barplot(df['trip_id'] , df['user_type'])
    plt.show()
```

```
C:\Users\asad\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
s: x, y. From version 0.12, the only valid positional argument will
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
[19]: # Heatmap
      #Heatmap uses to find the correlation between each and every features using the
      plt.figure(figsize=(10,9)) # heatmap size in ratio 16:9
      sns.heatmap(df.corr(), annot = True, cmap ='coolwarm') # show heatmap
      plt.title("Heatmap using correlation matrix", fontsize = 25) # title of heatmap
```

```
[19]: Text(0.5, 1.0, 'Heatmap using correlation matrix')
```

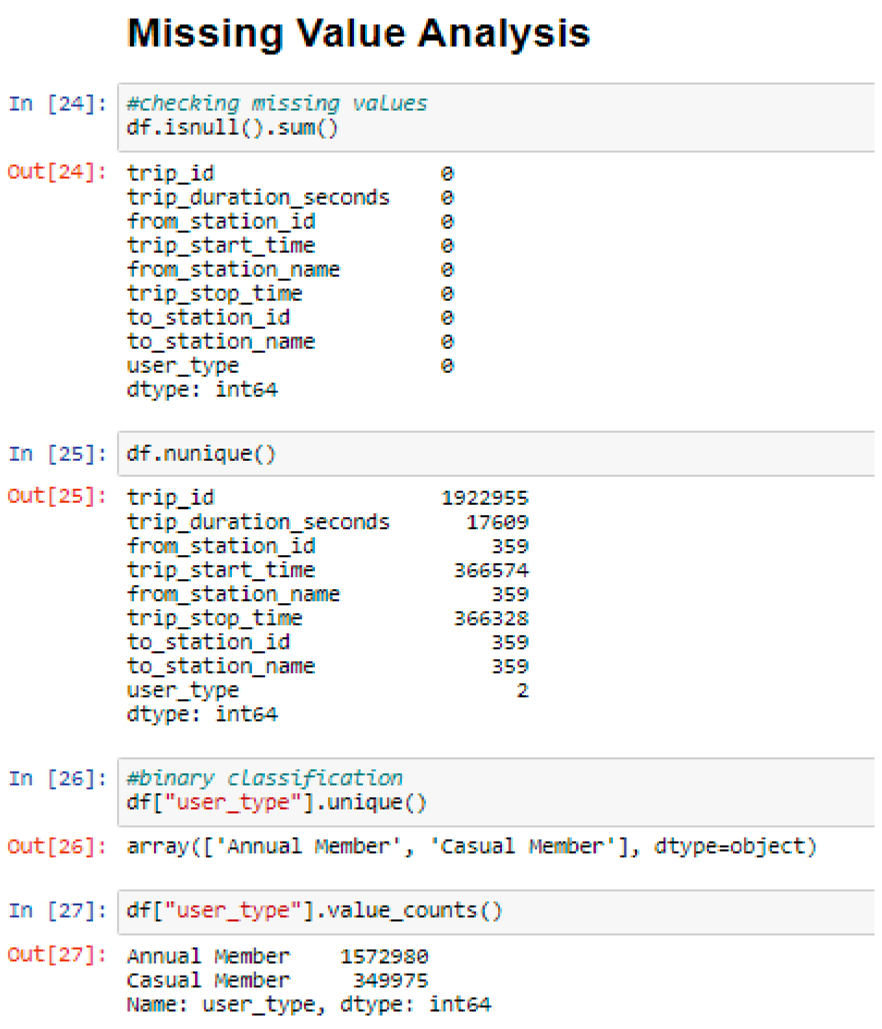
DATE : 15 9 2021

Step #5 : Data Cleaning is to identify and remove errors in order to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making .

Convert Data Types.

Take Care of Missing Values.

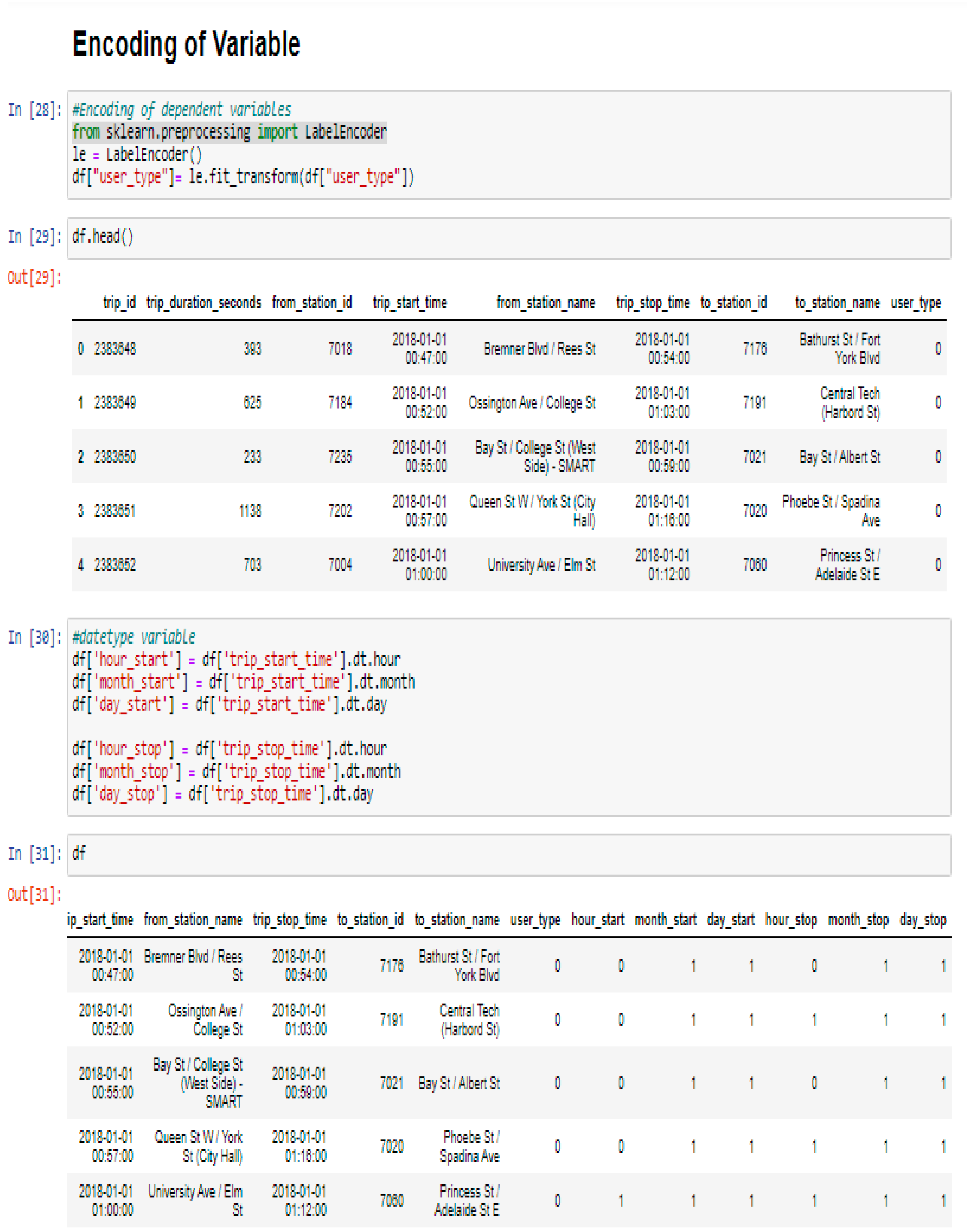
Remove Irrelevant Values.



Missing Value Analysis

```
In [24]: #checking missing values
         df.isnull().sum()

Out[24]: trip_id                    0
         trip_duration_seconds      0
         from_station_id            0
         trip_start_time            0
         from_station_name          0
         trip_stop_time             0
         to_station_id              0
         to_station_name            0
         user_type                  0
         dtype: int64

In [25]: df.nunique()

Out[25]: trip_id                    1922955
         trip_duration_seconds        17609
         from_station_id                359
         trip_start_time             366574
         from_station_name              359
         trip_stop_time              366328
         to_station_id                  359
         to_station_name                359
         user_type                        2
         dtype: int64

In [26]: #binary classification
         df["user_type"].unique()

Out[26]: array(['Annual Member', 'Casual Member'], dtype=object)

In [27]: df["user_type"].value_counts()

Out[27]: Annual Member    1572980
         Casual Member     349975
         Name: user_type, dtype: int64
```

DATE : 15 9 2021



Encoding of Variable

In [28]:
```python
#Encoding of dependent variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["user_type"]= le.fit_transform(df["user_type"])
```

In [29]:
```python
df.head()
```

Out[29]:

	trip_id	trip_duration_seconds	from_station_id	trip_start_time	from_station_name	trip_stop_time	to_station_id	to_station_name	user_type
0	2383648	393	7018	2018-01-01 00:47:00	Bremner Blvd / Rees St	2018-01-01 00:54:00	7176	Bathurst St / Fort York Blvd	0
1	2383649	625	7184	2018-01-01 00:52:00	Ossington Ave / College St	2018-01-01 01:03:00	7191	Central Tech (Harbord St)	0
2	2383650	233	7235	2018-01-01 00:55:00	Bay St / College St (West Side) - SMART	2018-01-01 00:59:00	7021	Bay St / Albert St	0
3	2383651	1138	7202	2018-01-01 00:57:00	Queen St W / York St (City Hall)	2018-01-01 01:16:00	7020	Phoebe St / Spadina Ave	0
4	2383652	703	7004	2018-01-01 01:00:00	University Ave / Elm St	2018-01-01 01:12:00	7060	Princess St / Adelaide St E	0

In [30]:
```python
#datetype variable
df['hour_start'] = df['trip_start_time'].dt.hour
df['month_start'] = df['trip_start_time'].dt.month
df['day_start'] = df['trip_start_time'].dt.day

df['hour_stop'] = df['trip_stop_time'].dt.hour
df['month_stop'] = df['trip_stop_time'].dt.month
df['day_stop'] = df['trip_stop_time'].dt.day
```

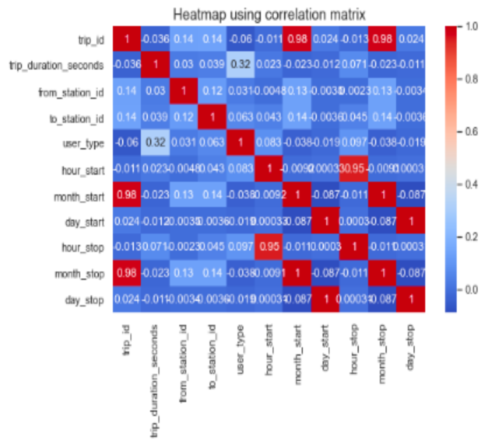
In [31]:
```python
df
```

Out[31]:

ip_start_time	from_station_name	trip_stop_time	to_station_id	to_station_name	user_type	hour_start	month_start	day_start	hour_stop	month_stop	day_stop
2018-01-01 00:47:00	Bremner Blvd / Rees St	2018-01-01 00:54:00	7176	Bathurst St / Fort York Blvd	0	0	1	1	0	1	1
2018-01-01 00:52:00	Ossington Ave / College St	2018-01-01 01:03:00	7191	Central Tech (Harbord St)	0	0	1	1	1	1	1
2018-01-01 00:55:00	Bay St / College St (West Side) - SMART	2018-01-01 00:59:00	7021	Bay St / Albert St	0	0	1	1	0	1	1
2018-01-01 00:57:00	Queen St W / York St (City Hall)	2018-01-01 01:16:00	7020	Phoebe St / Spadina Ave	0	0	1	1	1	1	1
2018-01-01 01:00:00	University Ave / Elm St	2018-01-01 01:12:00	7060	Princess St / Adelaide St E	0	1	1	1	1	1	1

DATE : 15 9 2021

```
In [32]: plt.figure(figsize=(8,5)) # heatmap size in ratio 16:9
         sns.heatmap(df.corr(), annot = True, cmap ='coolwarm') # show heatmap
         plt.title("Heatmap using correlation matrix", fontsize = 15) # title of heatmap

Out[32]: Text(0.5, 1.0, 'Heatmap using correlation matrix')
```



	trip_id	trip_duration_seconds	from_station_id	to_station_id	user_type	hour_start	month_start	day_start	hour_stop	month_stop	day_stop
trip_id	1	-0.036	0.14	0.14	-0.06	-0.011	0.98	0.024	-0.013	0.98	0.024
trip_duration_seconds	-0.036	1	0.03	0.039	0.32	0.023	-0.023	-0.012	0.071	-0.023	-0.011
from_station_id	0.14	0.03	1	0.12	0.031	-0.0048	0.13	-0.0035	-0.0023	0.13	-0.0034
to_station_id	0.14	0.039	0.12	1	0.063	0.043	0.14	-0.0036	0.045	0.14	-0.0036
user_type	-0.06	0.32	0.031	0.063	1	0.083	-0.038	-0.019	0.097	-0.038	-0.019
hour_start	-0.011	0.023	-0.0048	0.043	0.083	1	-0.0092	0.00033	0.95	-0.0091	0.0003
month_start	0.98	-0.023	0.13	0.14	-0.038	-0.0092	1	-0.087	-0.011	1	-0.087
day_start	0.024	-0.012	-0.0035	-0.0036	-0.019	0.00033	-0.087	1	0.0003	-0.087	1
hour_stop	-0.013	0.071	-0.0023	0.045	0.097	0.95	-0.011	0.0003	1	-0.011	0.0003
month_stop	0.98	-0.023	0.13	0.14	-0.038	-0.0091	1	-0.087	-0.011	1	-0.087
day_stop	0.024	-0.011	-0.0034	-0.0036	-0.019	0.00034	-0.087	1	0.00034	-0.087	1

```
In [33]: df=df.drop([ "hour_start","day_start" , "month_start" , "month_stop"],axis=1) #removing highly corelated indepen features

In [34]: df

Out[34]:
```

	trip_id	trip_duration_seconds	from_station_id	trip_start_time	from_station_name	trip_stop_time	to_station_id	to_station_name	user_type	hour_
0	2383648	393	7018	2018-01-01 00:47:00	Bremner Blvd / Rees St	2018-01-01 00:54:00	7176	Bathurst St / Fort York Blvd	0	
1	2383649	625	7184	2018-01-01 00:52:00	Ossington Ave / College St	2018-01-01 01:03:00	7191	Central Tech (Harbord St)	0	
2	2383650	233	7235	2018-01-01 00:55:00	Bay St / College St (West Side) - SMART	2018-01-01 00:59:00	7021	Bay St / Albert St	0	
3	2383651	1138	7202	2018-01-01 00:57:00	Queen St W / York St (City Hall)	2018-01-01 01:16:00	7020	Phoebe St / Spadina Ave	0	
4	2383652	703	7004	2018-01-01 01:00:00	University Ave / Elm St	2018-01-01 01:12:00	7060	Princess St / Adelaide St E	0	

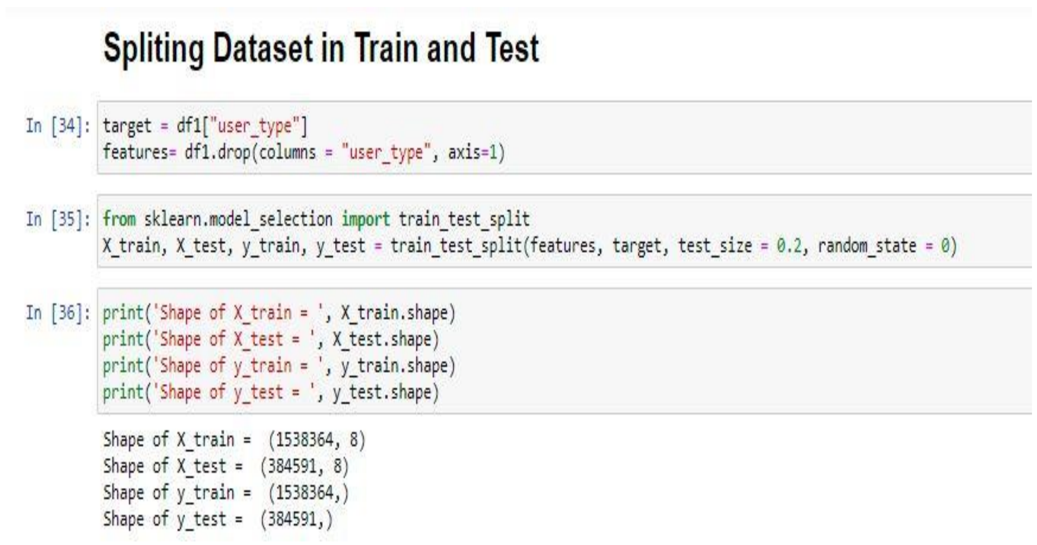
```
In [35]: #encoding categorical variable with multiple categories
         from_map=df["from_station_name"].value_counts().to_dict()

In [36]: df["from_station_name"]=df["from_station_name"].map(from_map)

In [37]: to_map=df["to_station_name"].value_counts().to_dict()

In [38]: df["to_station_name"]=df["to_station_name"].map(to_map)

In [39]: df1=df.drop(["trip_start_time" , "trip_stop_time"] , axis=1)

In [40]: df1

Out[40]:
```

	trip_id	trip_duration_seconds	from_station_id	from_station_name	to_station_id	to_station_name	user_type	hour_stop	day_stop
0	2383648	393	7018	11498	7176	10543	0	0	1
1	2383649	625	7184	3767	7191	4406	0	1	1
2	2383650	233	7235	9029	7021	13911	0	0	1
3	2383651	1138	7202	9586	7020	16878	0	1	1
4	2383652	703	7004	6692	7060	19986	0	1	1
...	...	...	...	...	...	...	...	...	...
363485	4581273	379	7088	1541	7091	2021	0	23	31
363486	4581274	306	7030	19184	7031	6144	0	23	31
363487	4581275	340	7020	15707	7000	13123	0	23	31
363488	4581276	1466	7014	10307	7269	10329	0	0	1
363489	4581277	333	7299	6135	7013	10922	0	0	1

1922955 rows × 9 columns

DATE : 15 9 2021

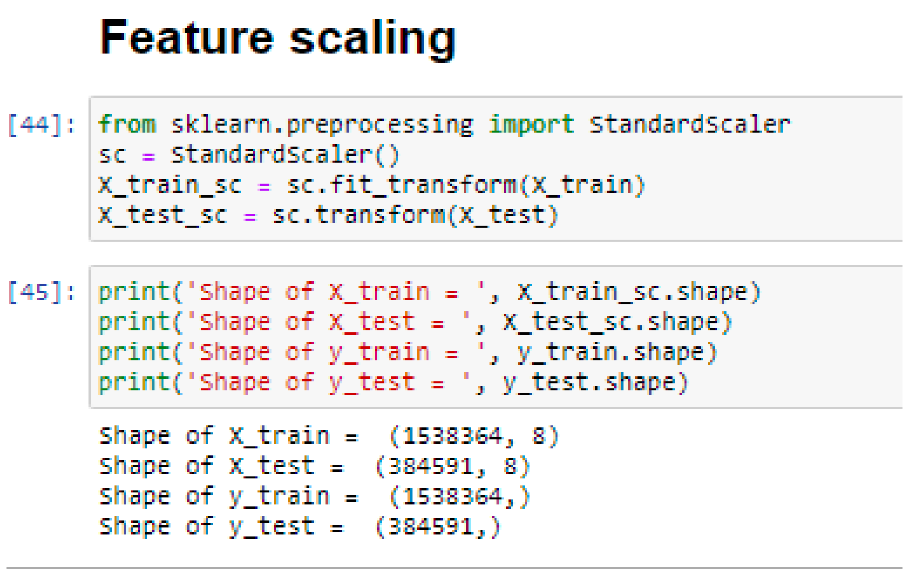
1. Step# 6: Split the data into training and testing (80:20)



Spliting Dataset in Train and Test

```
In [34]: target = df1["user_type"]
         features= df1.drop(columns = "user_type", axis=1)

In [35]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, random_state = 0)

In [36]: print('Shape of X_train = ', X_train.shape)
         print('Shape of X_test = ', X_test.shape)
         print('Shape of y_train = ', y_train.shape)
         print('Shape of y_test = ', y_test.shape)

         Shape of X_train =  (1538364, 8)
         Shape of X_test =  (384591, 8)
         Shape of y_train =  (1538364,)
         Shape of y_test =  (384591,)
```

Step #7 : Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step

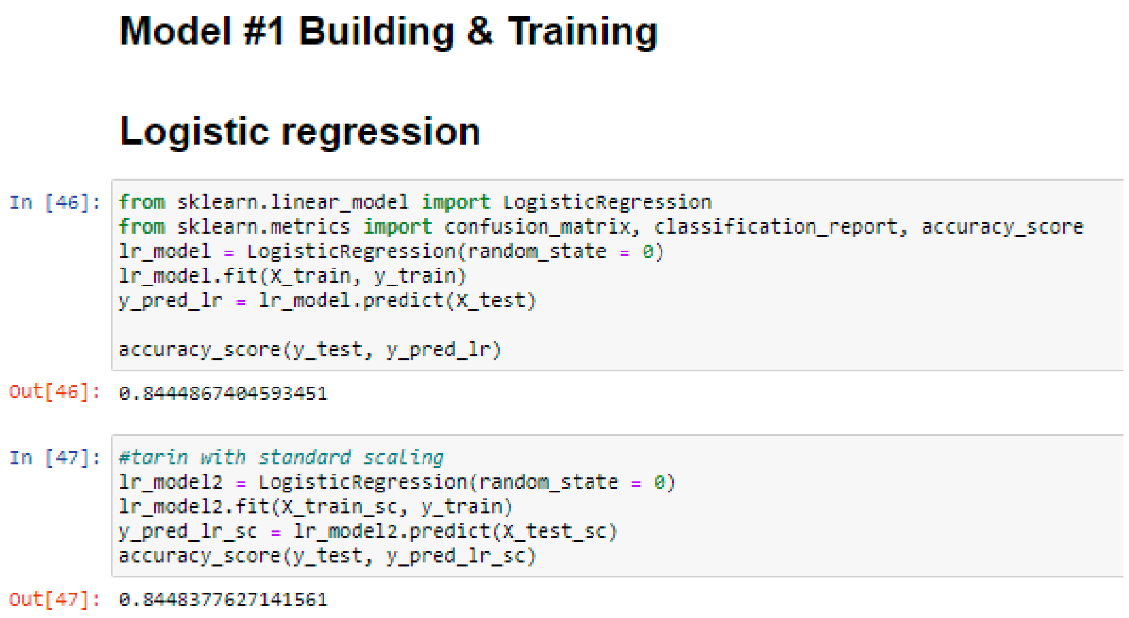


Feature scaling

```
[44]: from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X_train_sc = sc.fit_transform(X_train)
      X_test_sc = sc.transform(X_test)

[45]: print('Shape of X_train = ', X_train_sc.shape)
      print('Shape of X_test = ', X_test_sc.shape)
      print('Shape of y_train = ', y_train.shape)
      print('Shape of y_test = ', y_test.shape)

      Shape of X_train =  (1538364, 8)
      Shape of X_test =  (384591, 8)
      Shape of y_train =  (1538364,)
      Shape of y_test =  (384591,)
```

DATE : 15 9 2021

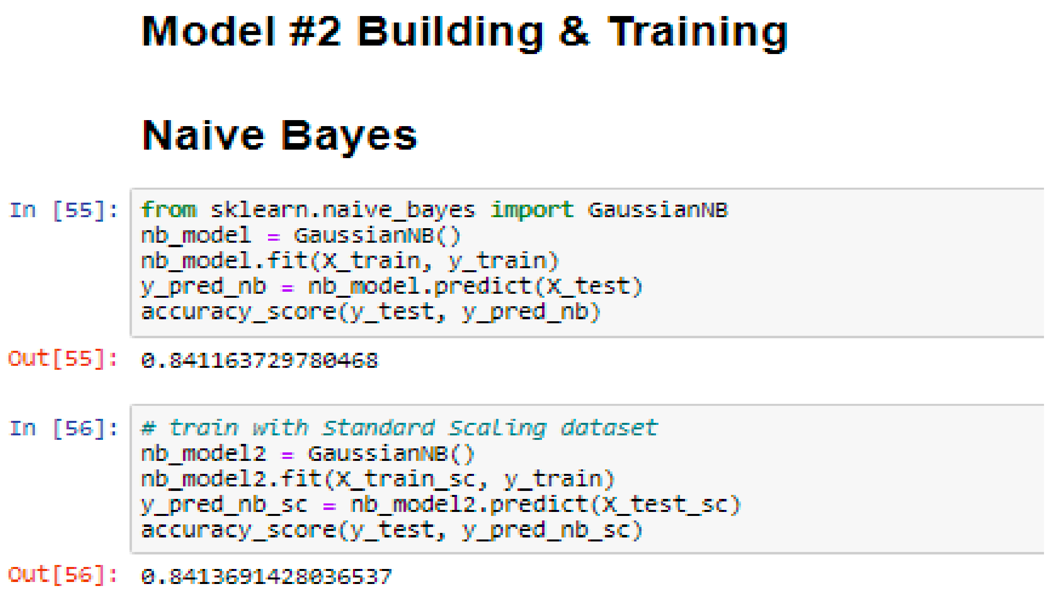
Step #8 : Model is built by learning and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfill its purpose..



Model #1 Building & Training

Logistic regression

```
In [46]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
         lr_model = LogisticRegression(random_state = 0)
         lr_model.fit(X_train, y_train)
         y_pred_lr = lr_model.predict(X_test)

         accuracy_score(y_test, y_pred_lr)

Out[46]: 0.8444867404593451

In [47]: #tarin with standard scaling
         lr_model2 = LogisticRegression(random_state = 0)
         lr_model2.fit(X_train_sc, y_train)
         y_pred_lr_sc = lr_model2.predict(X_test_sc)
         accuracy_score(y_test, y_pred_lr_sc)

Out[47]: 0.8448377627141561
```

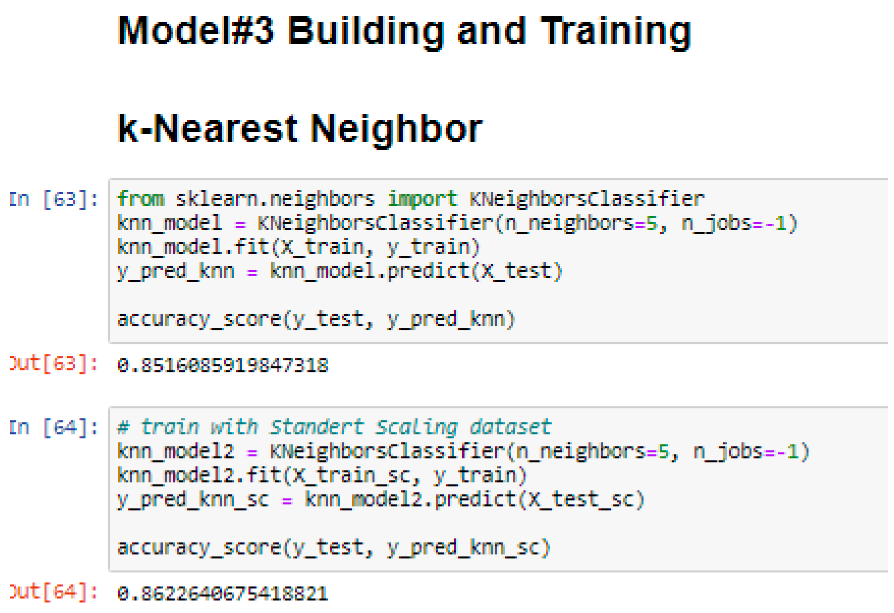


Model #2 Building & Training

Naive Bayes

```
In [55]: from sklearn.naive_bayes import GaussianNB
         nb_model = GaussianNB()
         nb_model.fit(X_train, y_train)
         y_pred_nb = nb_model.predict(X_test)
         accuracy_score(y_test, y_pred_nb)

Out[55]: 0.841163729780468

In [56]: # train with Standard Scaling dataset
         nb_model2 = GaussianNB()
         nb_model2.fit(X_train_sc, y_train)
         y_pred_nb_sc = nb_model2.predict(X_test_sc)
         accuracy_score(y_test, y_pred_nb_sc)

Out[56]: 0.8413691428036537
```

DATE : 15 9 2021



Model#3 Building and Training

k-Nearest Neighbor

```
In [63]: from sklearn.neighbors import KNeighborsClassifier
         knn_model = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
         knn_model.fit(X_train, y_train)
         y_pred_knn = knn_model.predict(X_test)

         accuracy_score(y_test, y_pred_knn)

Out[63]: 0.8516085919847318
```

```
In [64]: # train with Standert Scaling dataset
         knn_model2 = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
         knn_model2.fit(X_train_sc, y_train)
         y_pred_knn_sc = knn_model2.predict(X_test_sc)

         accuracy_score(y_test, y_pred_knn_sc)

Out[64]: 0.8622640675418821
```

Step # 9 : It is the phase that is decided whether the model performs better. We evaluate the model on the basis of confusion matrix , classification report , ROC curve .

The **recall** rate specifies true positive rate which means number of instances classified as correctly . Recall also gives a measure of how accurately our model is able to identify the relevant data.

Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. Precision of our model is also good which shows that our system is not at great risk because of False positives.

ROC Curve : Classifiers that give curves closer to the top-left corner indicate a better performance.

Model # 3 KNN has highest accuracy among all other that is 86% .

On basis on accuracy , F1 score , precision , AUC KNN is best

On basis of recall LR and NB are good

DATE : 15 9 2021

Model #1 Evaluation

```
In [48]: ################### Generalize model ############
```

```
In [49]: y_pred_lr_sc_train = lr_model2.predict(X_train_sc)
         print("Accuracy Score on Train data" ,accuracy_score(y_train, y_pred_lr_sc_train)
         print("Accuracy Score on Test data" , accuracy_score(y_test, y_pred_lr_sc))

         Accuracy Score on Train data 0.8439179543983089
         Accuracy Score on Test data 0.8448377627141561
```

confusion matrix

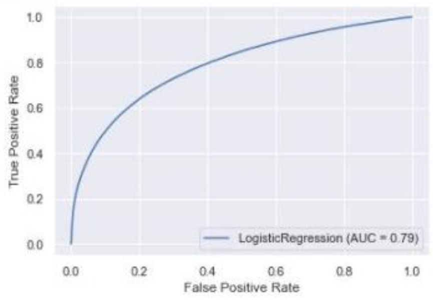
```
In [50]: cm_lr = confusion_matrix(y_test, y_pred_lr_sc)
         print('Confussion matrix = \n', cm_lr)

         Confussion matrix =
          [[312025   3203]
          [ 56471  12892]]
```

Classification report

```
In [51]: # Clasification Report
         target_names=['User_type 0' , 'User_type 1']
         cr_lr = classification_report(y_test, y_pred_lr_sc , target_names=target_names)
         print("Classification report >>> \n", cr_lr)

         Classification report >>>
                        precision    recall  f1-score   support

          User_type 0       0.85      0.99      0.91    315228
          User_type 1       0.80      0.19      0.30     69363

             accuracy                           0.84    384591
            macro avg       0.82      0.59      0.61    384591
         weighted avg       0.84      0.84      0.80    384591
```

Roc Curve

```
In [47]: from sklearn import metrics
         fpr , tpr , threshold= metrics.roc_curve(y_test, y_pred_lr_sc , pos_label=2)
         metrics.plot_roc_curve(lr_model2 , X_test_sc , y_test)
         plt.show()

         C:\Users\asad\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:811: Un
         e, true positive value should be meaningless
           warnings.warn("No positive samples in y_true, "
```



DATE : 15 9 2021

Model#2 Evaluation

```
[57]: #################### Generalize ####################
```

```
[58]: y_pred_nb_sc_train = nb_model2.predict(X_train_sc)
      print("Accuracy Score on Train data" ,accuracy_score(y_train, y_pred_nb_sc_train))
      print("Accuracy Score on Test data" , accuracy_score(y_test, y_pred_nb_sc))
```

```
Accuracy Score on Train data 0.8401639663954694
Accuracy Score on Test data 0.8413691428036537
```

Confusion Matrix

```
[59]: cm_nb = confusion_matrix(y_test, y_pred_nb_sc)
      print('Confussion matrix = \n', cm_nb)
```

```
Confussion matrix =
 [[313074   2154]
 [ 58854  10509]]
```

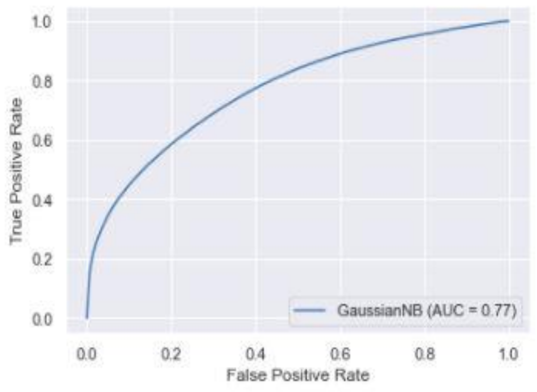
Clasification Report

```
[60]: #Clasification Report
      target_names=['User_type 0' , 'User_type 1']
      cr_NB = classification_report(y_test, y_pred_nb_sc , target_names=target_names)
      print("Classification report >>> \n", cr_NB)
```

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

 User_type 0       0.84      0.99      0.91    315228
 User_type 1       0.83      0.15      0.26     69363

    accuracy                           0.84    384591
   macro avg       0.84      0.57      0.58    384591
weighted avg       0.84      0.84      0.79    384591
```

ROC Curve

```
In [59]: from sklearn import metrics
         fpr , tpr , threshold= metrics.roc_curve(y_test, y_pred_nb_sc , pos_label=2)
         metrics.plot_roc_curve(nb_model2 , X_test_sc , y_test)
         plt.show()
```



DATE : 15 9 2021

Model#3 Evaluation

```
[65]: ##################### overfitting perfom good at training but not testing  ###############
```

```
[66]: y_pred_knn_sc_train = knn_model2.predict(X_train_sc)
      print("Accuracy Score on Train data" ,accuracy_score(y_train, y_pred_knn_sc_train))
      print("Accuracy Score on Test data" , accuracy_score(y_test, y_pred_knn_sc))
```

```
Accuracy Score on Train data 0.900532643769615
Accuracy Score on Test data 0.8622640675418821
```

Confusion matrix

```
[67]: cm_KNN = confusion_matrix(y_test, y_pred_knn_sc)
      print('Confussion matrix = \n', cm_KNN)
```

```
Confussion matrix =
 [[296888  18340]
 [ 34632  34731]]
```

Classification Report

```
[68]: # Clasification Report
      target_names=['User_type 0' , 'User_type 1']
      cr_KNN = classification_report(y_test, y_pred_knn_sc , target_names=target_names)
      print("Classification report >>> \n", cr_KNN)
```

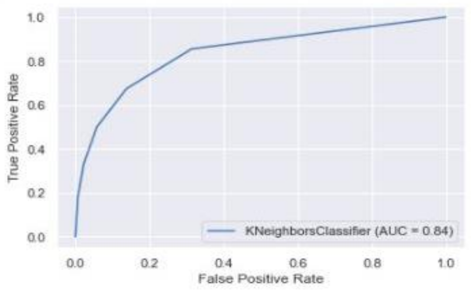
```
Classification report >>>
               precision    recall  f1-score   support

 User_type 0       0.90      0.94      0.92    315228
 User_type 1       0.65      0.50      0.57     69363

    accuracy                           0.86    384591
   macro avg       0.77      0.72      0.74    384591
weighted avg       0.85      0.86      0.85    384591
```

Roc Curve

```
In [67]: from sklearn import metrics
         fpr , tpr , threshold= metrics.roc_curve(y_test, y_pred_knn_sc , pos_label=2)
```

```
C:\Users\asad\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:811: Und
e, true positive value should be meaningless
  warnings.warn("No positive samples in y_true, "
```

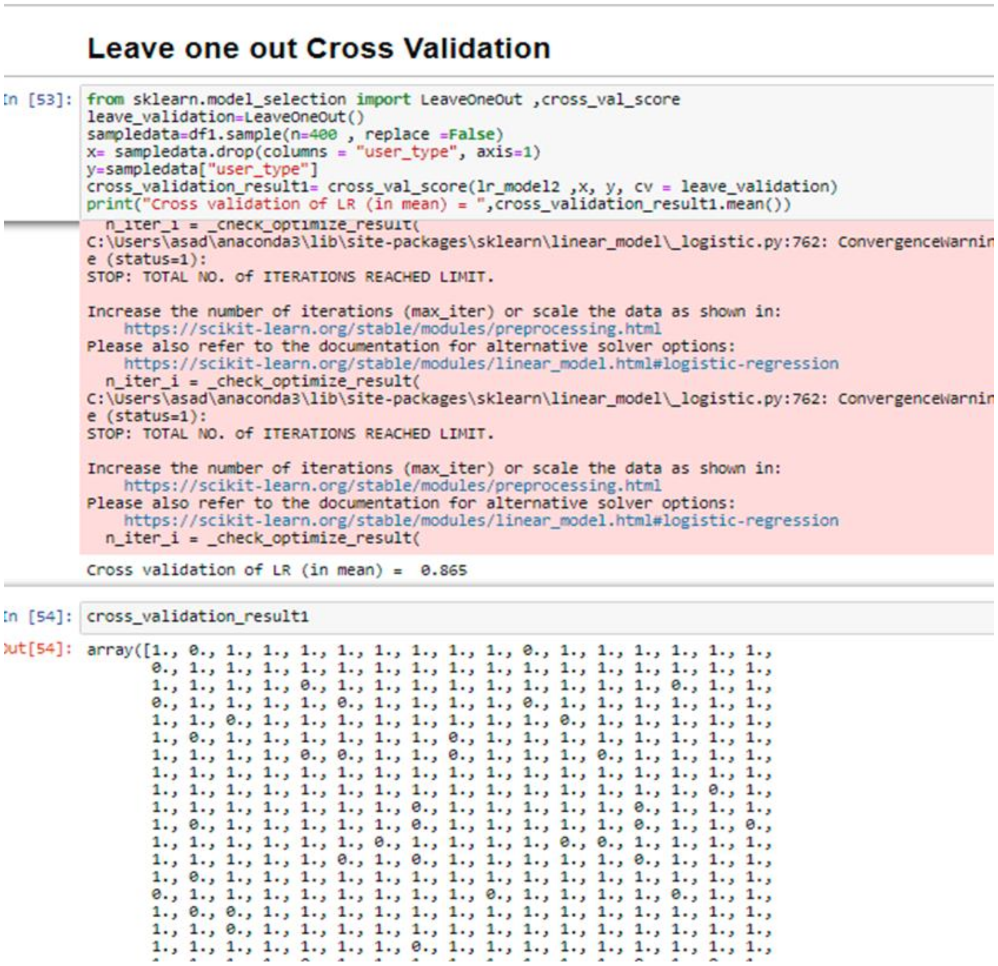
```
In [68]: metrics.plot_roc_curve(knn_model2 , X_test_sc , y_test)
         plt.show()
```



DATE : 15 9 2021

Step# 10 Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

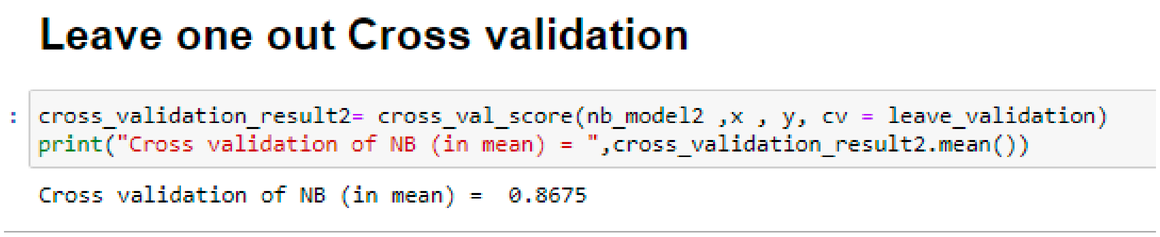
Logistic regression



Leave one out Cross Validation

```
In [53]: from sklearn.model_selection import LeaveOneOut ,cross_val_score
         leave_validation=LeaveOneOut()
         sampledata=df1.sample(n=400 , replace =False)
         x= sampledata.drop(columns = "user_type", axis=1)
         y=sampledata["user_type"]
         cross_validation_result1= cross_val_score(lr_model2 ,x, y, cv = leave_validation)
         print("Cross validation of LR (in mean) = ",cross_validation_result1.mean())

           n_iter_i = _check_optimize_result(
         C:\Users\asad\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarnin
         e (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
           n_iter_i = _check_optimize_result(
         C:\Users\asad\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarnin
         e (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
           n_iter_i = _check_optimize_result(

         Cross validation of LR (in mean) =  0.865

In [54]: cross_validation_result1

Out[54]: array([1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 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., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
                0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
                1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
                1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
                1., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1.,
                1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 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., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
                1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0.,
                1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1.,
                1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
                1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
                0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1.,
                1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 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., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
```

Naïve bayes

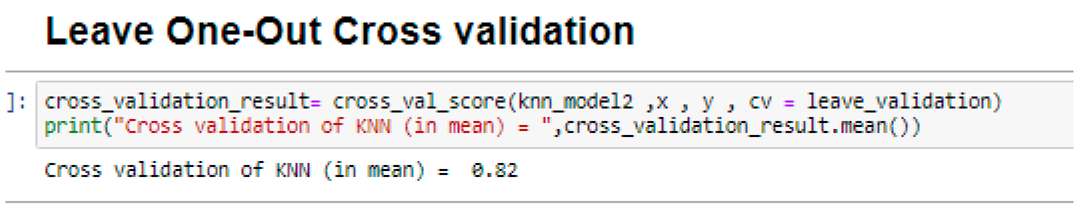


Leave one out Cross validation

```
: cross_validation_result2= cross_val_score(nb_model2 ,x , y, cv = leave_validation)
  print("Cross validation of NB (in mean) = ",cross_validation_result2.mean())

  Cross validation of NB (in mean) =  0.8675
```

DATE : 15 9 2021

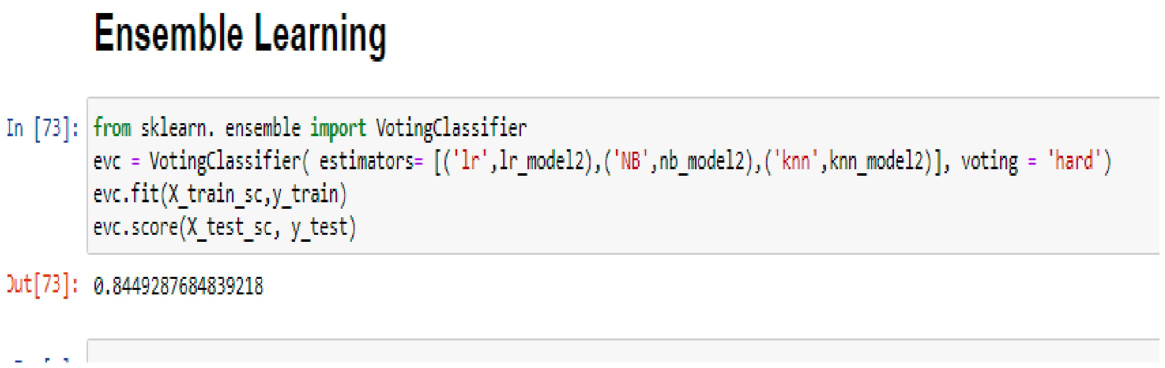
K Nearest Neighbor



Leave One-Out Cross validation

```
]: cross_validation_result= cross_val_score(knn_model2 ,x , y , cv = leave_validation)
   print("Cross validation of KNN (in mean) = ",cross_validation_result.mean())

   Cross validation of KNN (in mean) =  0.82
```

Ensemble learning is technique to ensemble the models so improve their performance if they are not working good individually



Ensemble Learning

```
In [73]: from sklearn. ensemble import VotingClassifier
         evc = VotingClassifier( estimators= [('lr',lr_model2),('NB',nb_model2),('knn',knn_model2)], voting = 'hard')
         evc.fit(X_train_sc,y_train)
         evc.score(X_test_sc, y_test)

Out[73]: 0.8449287684839218
```

=