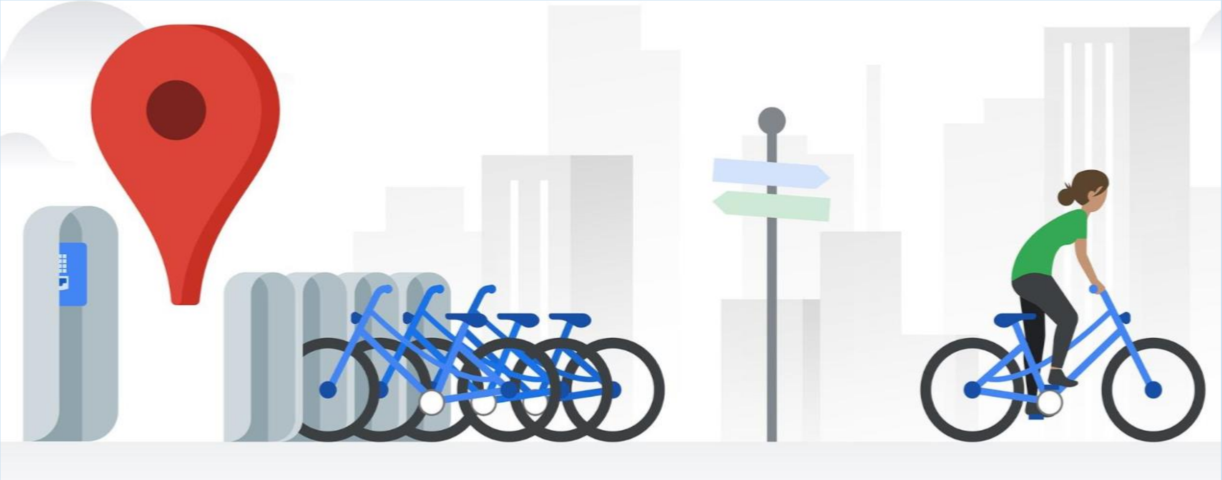
Data Analytics

ANALYSIS AND CLASSIFICATION OF BIKESHARE DATA

INTRODUCTION

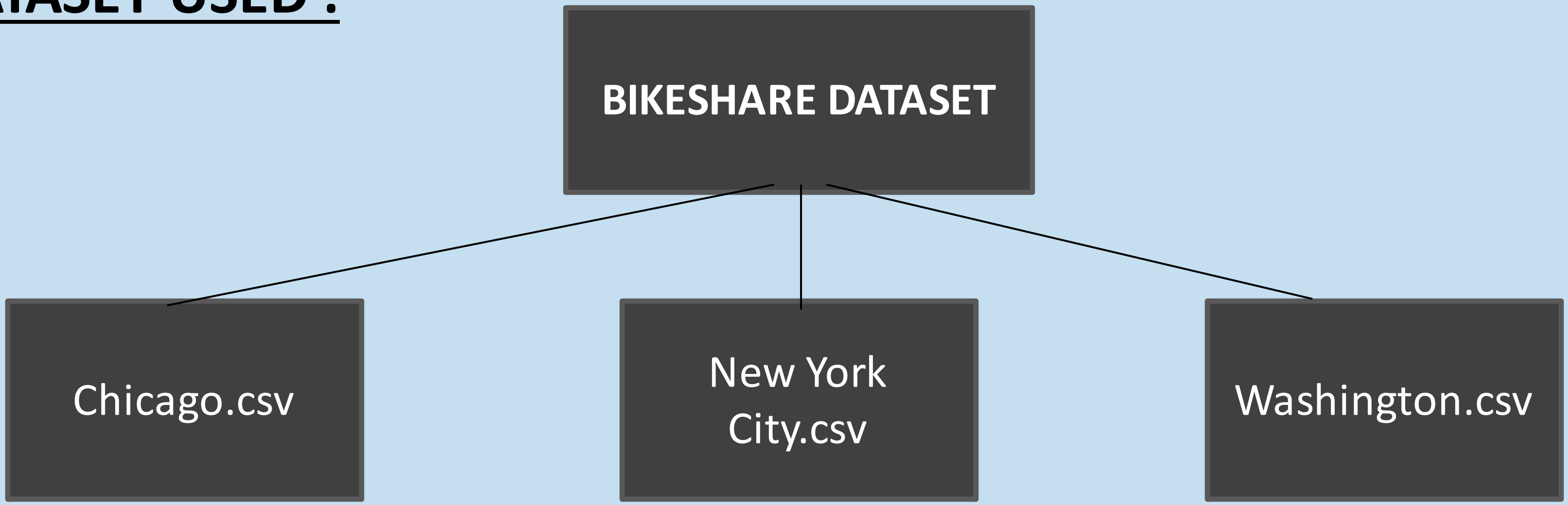
AIM : The objective here is to take the data of a Bike Sharing System and Describe, Analyse and Classify it for better understanding.

- Bicycle-sharing system allows user to rent bicycles on a very short-term basis for a particular price.
- This allows people to borrow a bike from point A and return it at point B.
- Each bike can serve several users per day.



INTRODUCTION

DATASET USED :



- All three of the data files contain the same core six columns:
  - Start Time
  - End Time
  - Trip Duration
  - Start Station
  - End Station
  - User Type
- The Chicago and New York City files also have two extra columns:
  - Birth Year
  - Gender

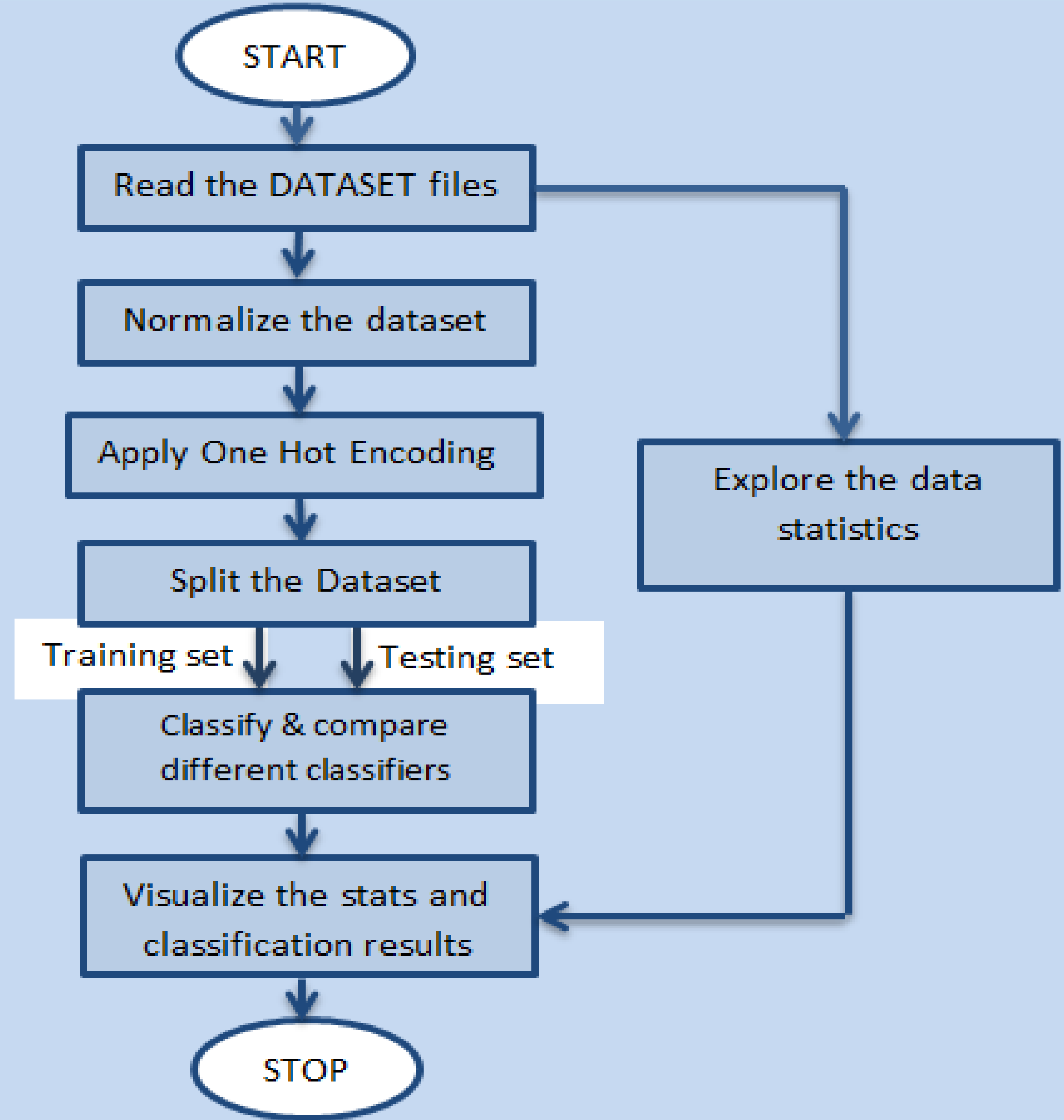
DATASET

```
In [19]: da1 = pd.read_csv('chicago.csv')
         da2 = pd.read_csv('new_york_city.csv')
         da2.head()
```

Out[19]:

	Unnamed: 0	Start Time	End Time	Trip Duration	Start Station	End Station	User Type	Gender	Birth Year
0	5688089	2017-06-11 14:55:05	2017-06-11 15:08:21	795	Suffolk St & Stanton St	W Broadway & Spring St	Subscriber	Male	1998.0
1	4096714	2017-05-11 15:30:11	2017-05-11 15:41:43	692	Lexington Ave & E 63 St	1 Ave & E 78 St	Subscriber	Male	1981.0
2	2173887	2017-03-29 13:26:26	2017-03-29 13:48:31	1325	1 Pl & Clinton St	Henry St & Degraw St	Subscriber	Male	1987.0
3	3945638	2017-05-08 19:47:18	2017-05-08 19:59:01	703	Barrow St & Hudson St	W 20 St & 8 Ave	Subscriber	Female	1986.0
4	6208972	2017-06-21 07:49:16	2017-06-21 07:54:46	329	1 Ave & E 44 St	E 53 St & 3 Ave	Subscriber	Male	1992.0

FLOW CHART :


START
Read the DATASET files
Normalize the dataset
Apply One Hot Encoding
Split the Dataset
Training set
Testing set
Classify & compare different classifiers
Visualize the stats and classification results
STOP
Explore the data statistics

DATA STATISTICS EXPLORATION

- Computed various Descriptive Statistics using pandas functions such as .mean(), .mode(), .sum(), .value_counts() etc.

- Statistics computed are:

➢ **Popular times of travel:**
- Most common month
- Most common day of week
- Most common hour of day

➢ **Popular stations and trip:**
- Most common start station
- Most common end station
- Most common trip from start to end

➢ **Trip duration:**
- Total travel time
- Average travel time

➢ **User info:**
- Counts of each type of user
- Counts of each gender
- Year of birth:
  Earliest ;
  Most recent ;
  Most common

Desktop/Bikeshare_Project/ × DA_project - Jupyter Notebook × +

localhost:8888/notebooks/Desktop/Bikeshare_Project/DA_project.ipynb

jupyter DA_project Last Checkpoint: a day ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
        print(df.head(5))

    while True:
        more_data = input('\nWould you like to see 5 more rows of raw data? Enter yes or no.\n')
        if more_data.lower() != 'yes':
            break
        else:
            print(df.iloc[count:count+5])
            count = count+5
```

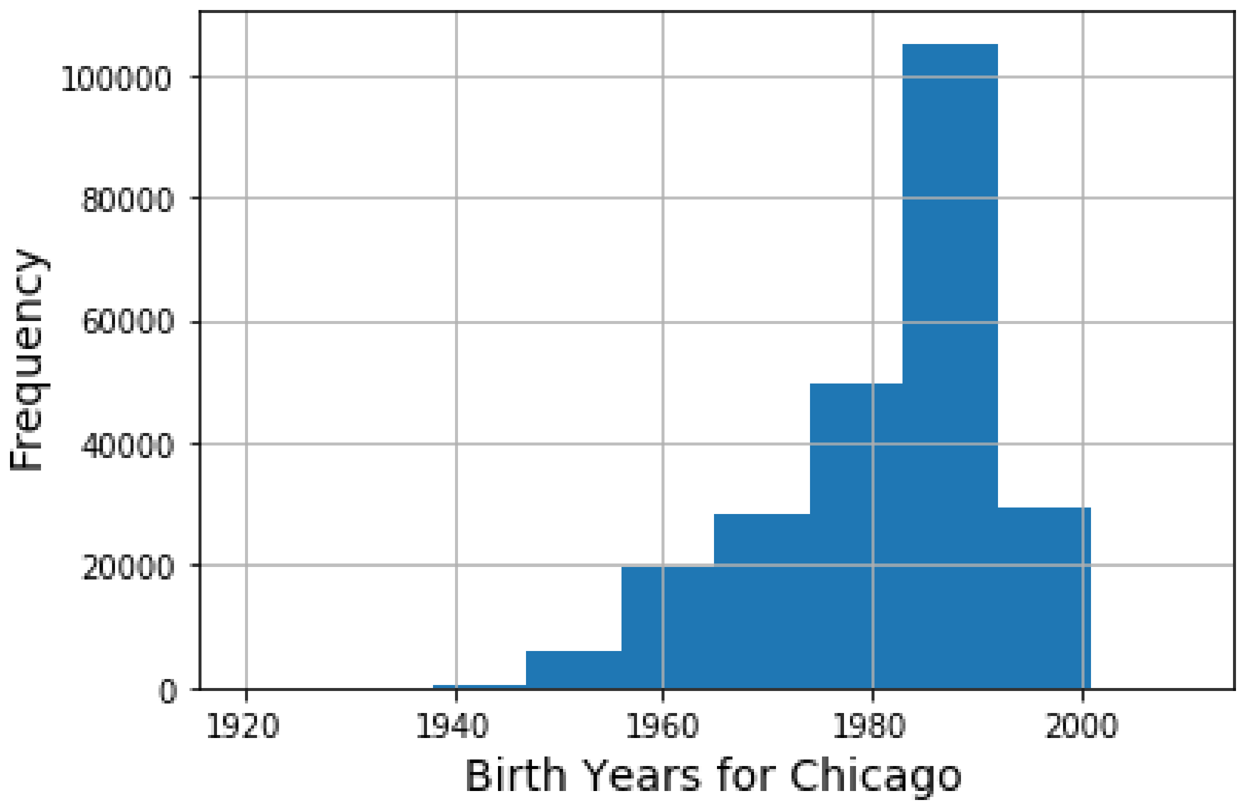
In [21]:
```
def main():
    while True:
        city, month, day = get_filters()
        df = load_data(city, month, day)

        time_stats(df)
        station_stats(df)
        trip_duration_stats(df)
        user_stats(df)

        display_data(df)

        restart = input('\nWould you like to restart? Enter yes or no.\n')
        if restart.lower() != 'yes':
            break
```

In [*]:
```
if __name__ == "__main__":
    main()
```

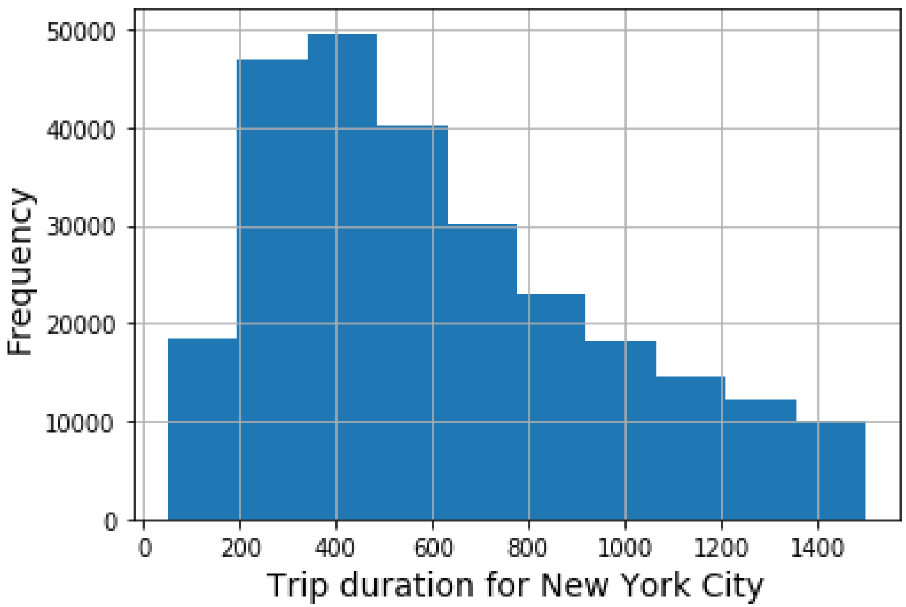
```
Hello! Let's explore some US bikeshare data!

Please enter the name of the city you want data for: Chicago, New York City, Washington
```

VISUALIZATION

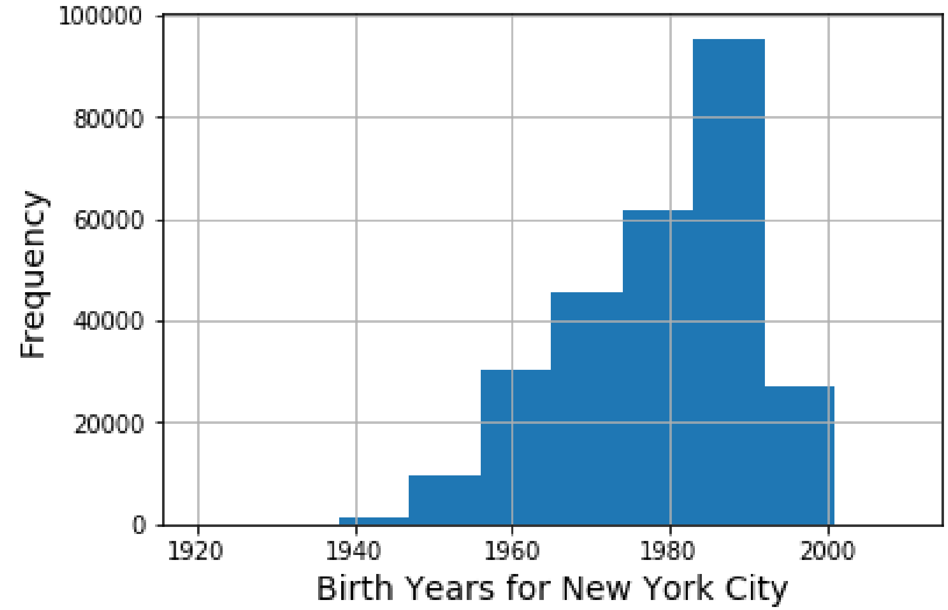
```
In [28]: ## Most common birth year for Chicago ##
         plt.xlabel('Birth Years for Chicago', fontsize=14)
         plt.ylabel('Frequency', fontsize=14)
         da1['Birth Year'].hist(range = [1920,2010])

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x12c755f50>
```



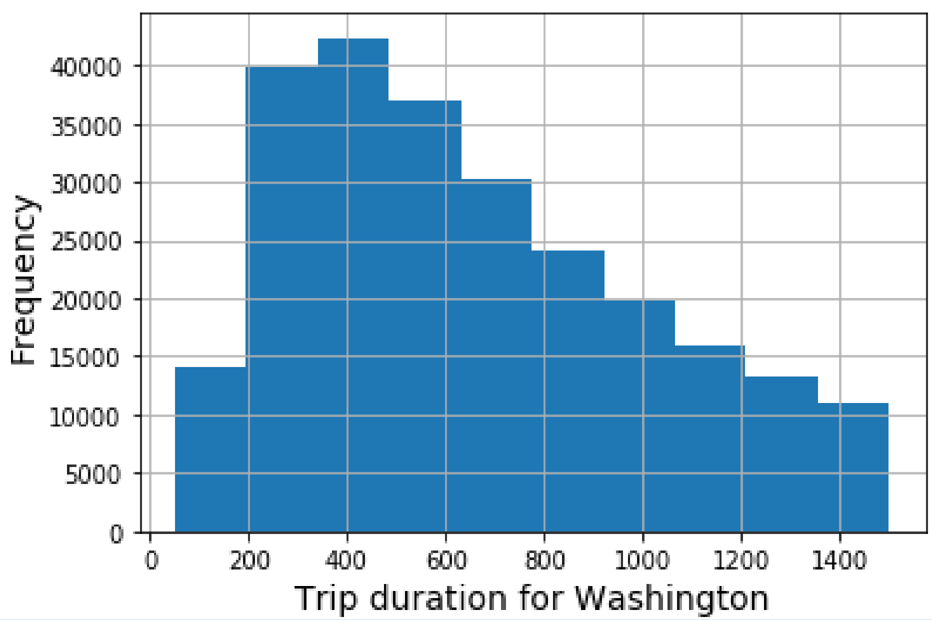
<matplotlib.axes._subplots.AxesSubplot at 0x13c635e10>



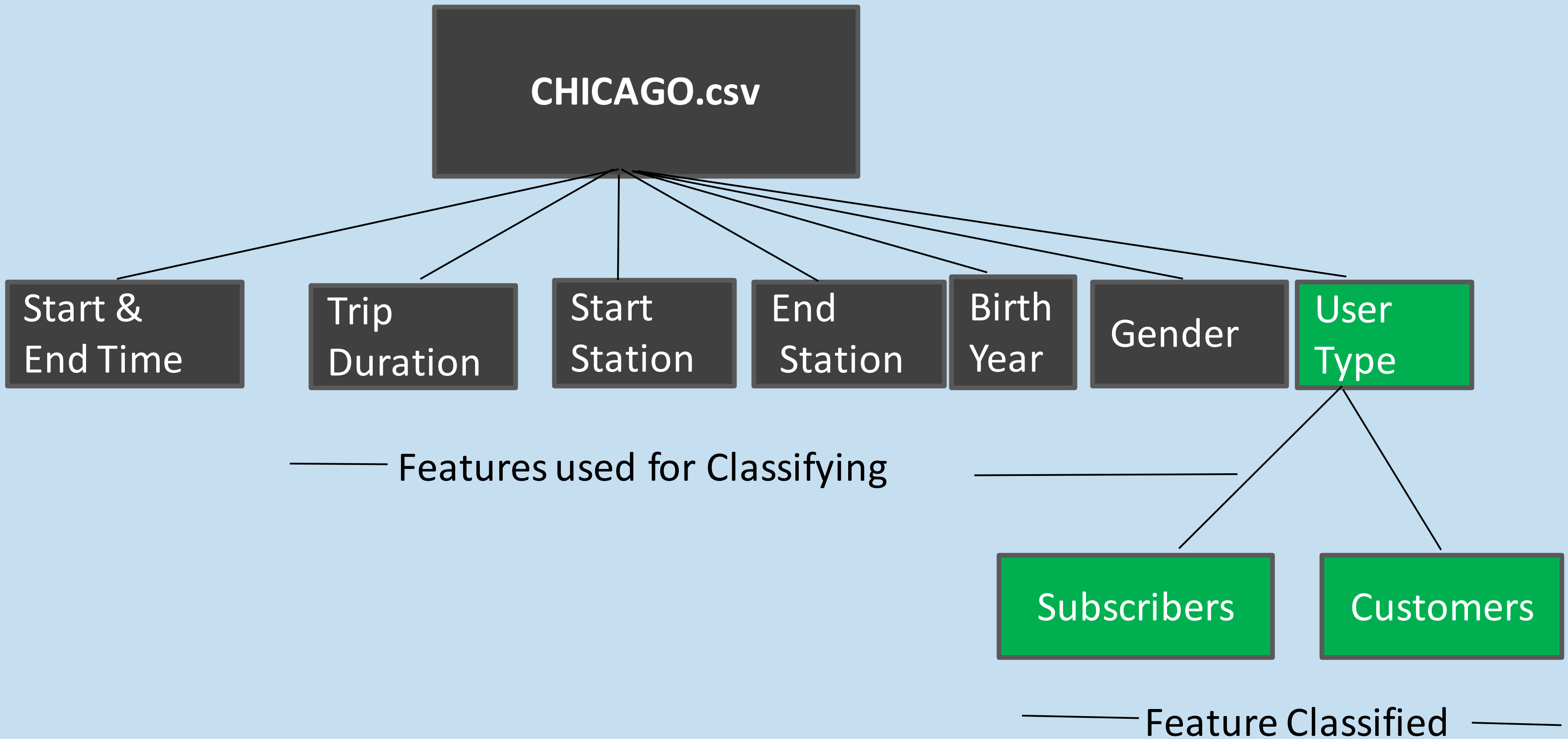
<matplotlib.axes._subplots.AxesSubplot at 0x141e18f50>



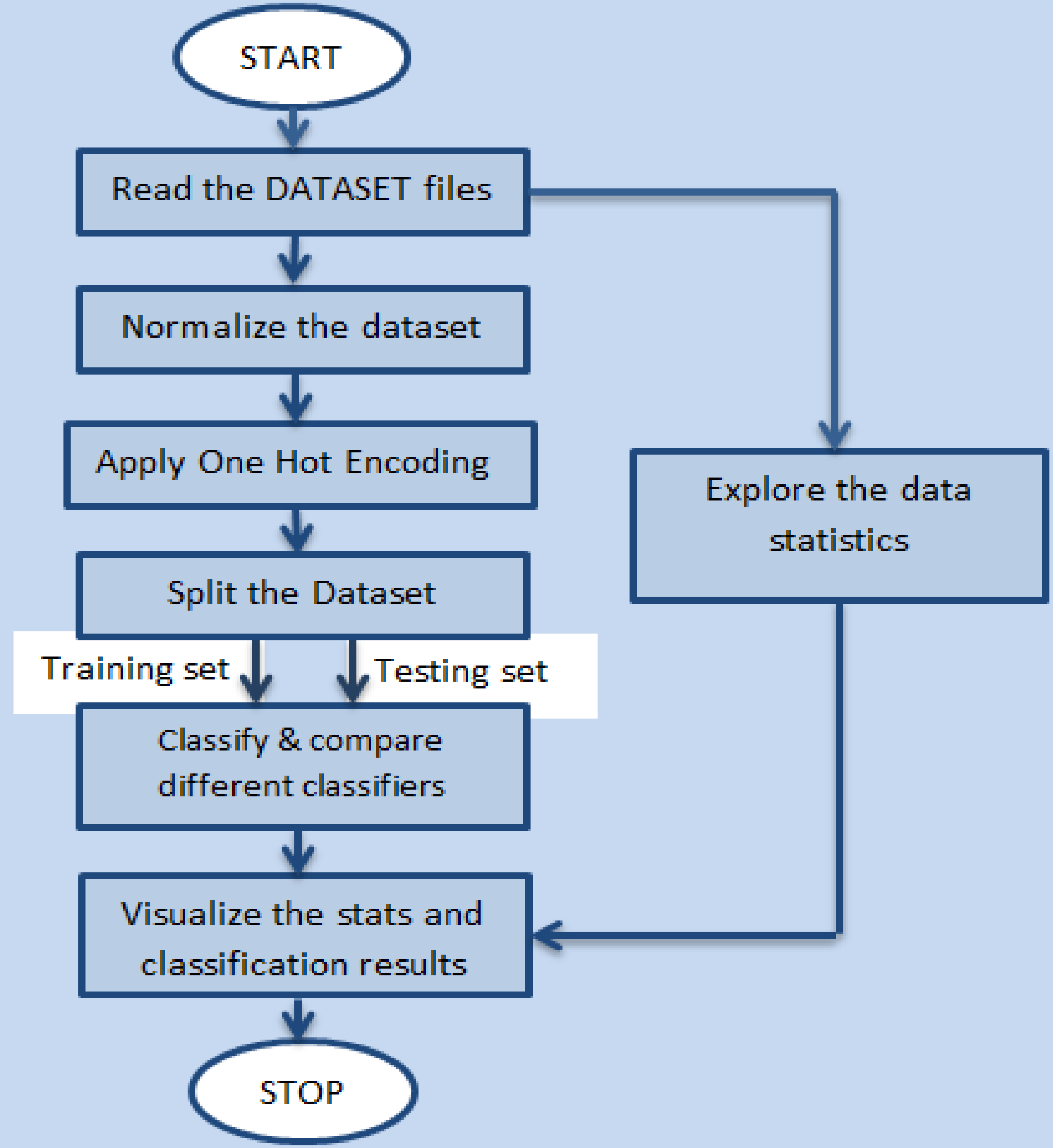
<matplotlib.axes._subplots.AxesSubplot at 0x13b47e950



CLASSIFICATION OF BIKESHARE DATA



FLOW CHART :


START
Read the DATASET files
Normalize the dataset
Apply One Hot Encoding
Split the Dataset
Training set
Testing set
Classify & compare different classifiers
Visualize the stats and classification results
STOP
Explore the data statistics

DATA NORMALIZATION

- Normalization often refers to rescaling, to make sure all features are on similar scale
- It makes algorithms run faster
- Normalization used: Min-Max Scalar

Trip Duration	Start Station	End Station	Gender	Birth Year
925	Marshfield Ave & Cortland St	Sheffield Ave & Wellington Ave	Female	1960
368	Michigan Ave & Washington St	Franklin St & Quincy St	Male	1990
2012	Mies van der Rohe Way & Chicago Ave	State St & 19th St	Male	1969
311	University Library (NU)	Sheridan Rd & Noyes St (NU)	Female	1989
223	Damen Ave & Leland Ave	Western Ave & Leland Ave	Female	1989
...	...	...	...	...
706	Clarendon Ave & Junior Ter	Halsted St & Diversey Pkwy	Male	1961
560	Franklin St & Quincy St	Michigan Ave & Washington St	Male	1983
469	Emerald Ave & 31st St	Normal Ave & Archer Ave	Female	1989
635	Clark St & Elm St	Sedgwick St & Webster Ave	Female	1993
387	Clark St & Lake St	Field Blvd & South Water St	Female	1983

Data before Normalization

Trip Duration	Start Station	End Station	Gender	Birth Year
0.010135	Marshfield Ave & Cortland St	Sheffield Ave & Wellington Ave	Female	0.517241
0.003609	Michigan Ave & Washington St	Franklin St & Quincy St	Male	0.775862
0.022871	Mies van der Rohe Way & Chicago Ave	State St & 19th St	Male	0.594828
0.002941	University Library (NU)	Sheridan Rd & Noyes St (NU)	Female	0.767241
0.001910	Damen Ave & Leland Ave	Western Ave & Leland Ave	Female	0.767241
...	...	...	...	...
0.007569	Clarendon Ave & Junior Ter	Halsted St & Diversey Pkwy	Male	0.525862
0.005858	Franklin St & Quincy St	Michigan Ave & Washington St	Male	0.715517
0.004792	Emerald Ave & 31st St	Normal Ave & Archer Ave	Female	0.767241
0.006737	Clark St & Elm St	Sedgwick St & Webster Ave	Female	0.801724
0.003831	Clark St & Lake St	Field Blvd & South Water St	Female	0.715517

Data after Normalization

ONE HOT ENCODING

- One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.
- In this a new column for each category is created & is assigned a value of 1/0 based on it is there or not. (extra variables k/a dummy variables)

```
## PERFORMING ONE HOT ENCODING ON THE FEATURES
X_final = pd.get_dummies(X_normalized)
X_final = X_final.fillna(0.0)
X_final = X_final
X_final
```

Trip Duration	Birth Year	Start Station_2112 W Peterson Ave	Start Station_63rd St Beach	Start Station_900 W Harrison St	Start Station_Aberdeen St & Jackson Blvd	Start Station_Aberdeen St & Monroe St	Start Station_Ada St & Washington Blvd	Start Station_Adler Planetarium	Start Station_Albany (Kedzie) Ave & Montrose Ave	...	End Station_Wolcott Ave & Polk St
0.010135	0.517241	0	0	0	0	0	0	0	0	...	0
0.003609	0.775862	0	0	0	0	0	0	0	0	...	0
0.022871	0.594828	0	0	0	0	0	0	0	0	...	0
0.002941	0.767241	0	0	0	0	0	0	0	0	...	0
0.001910	0.767241	0	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...
0.007569	0.525862	0	0	0	0	0	0	0	0	...	0
0.005858	0.715517	0	0	0	0	0	0	0	0	...	0
0.004792	0.767241	0	0	0	0	0	0	0	0	...	0
0.006737	0.801724	0	0	0	0	0	0	0	0	...	0
0.003831	0.715517	0	0	0	0	0	0	0	0	...	0

Data after One-Hot Encoding

CLASSIFICATION RESULTS

```
In [8]: ## DIVIDING DATA INTO TRAINING SET AND TEST SET

        X_train, X_test, y_train, y_test = train_test_split(X_final, Y_final, test_size=0.2)
```

DECISION TREE :

```
Decision Tree
Training accuracy is:   1.0
Testing accuracy is:   0.73
f beta score for train data is:   1.0
f beta score for test data is:   0.7601880877742947
```

DECISION TREE WITH HYPER-PARAMETERS:

```
In [38]: ## INTRODUCING HYPERPARAMETERS FOR DECISION TREE TO REDUCE OVER FITTING

         clf_of = tree.DecisionTreeClassifier(max_depth = 2, min_samples_leaf=20)
         clf_of.fit(X_train, y_train)
         y_predict_test_of = clf_of.predict(X_test)
         predictions_train_of = clf_of.predict(X_train)
```

```
Decision Tree with hyperparameters
Training accuracy is:   0.8222778473091364
Testing accuracy is:   0.79
f beta score for train data is:   0.8163595215001618
f beta score for test data is:   0.7783641160949868
```

CLASSIFICATION RESULTS

SVM:

```
SVM
Training accuracy is:   0.8986232790988736
Testing accuracy is:   0.785
f beta score for train data is:   0.8892438764643237
f beta score for test data is:   0.7879656160458453
```

GAUSSIAN NB & RANDOM FOREST :

```
GaussianNB
Training accuracy is:   0.7634543178973717
Testing accuracy is:   0.605
f beta score for train data is:   0.8931599773883551
f beta score for test data is:   0.6797235023041475
RandomForestClassifier
Training accuracy is:   0.9874843554443054
Testing accuracy is:   0.77
f beta score for train data is:   0.9844054580896686
f beta score for test data is:   0.7729805013927578
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_esti
mators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

CONCLUSION

Classifiers:	DECISION TREE	DECISION TREE WITH HYPER-PARAMETERS	SVM	GAUSSIAN NB	RANDOM FOREST
TRAINING SET ACCURACY	1	0.82	0.89	0.76	0.98
TEST SET ACCURACY	0.73	0.79	0.78	0.6	0.77
TRAINING SET F1 SCORE	1	0.81	0.88	0.89	0.98
TEST SET F1 SCORE	0.76	0.77	0.78	0.67	0.77