# APPLIED CAPSTONE PROJECT

Model and Prediction of severity of road accidents in Seattle

## Introduction and Data Exploration

- Data file used is Data_Collisions.csv available on the Capstone Project
- This project intends to analyse and process the traffic incidents data in Seattle. The aim of the project is to predict the severity of an accident with the data given like latitude, longitude, weather conditions, junction types and others.
- The initial process is to plot the information to the nearest neighborhood and zipcode
- Pre-process the data
- Build the machine learning model
- Evaluate the model for accuracy

## Business Understanding

- This data science study is to predict the severity (1 or 2) of a vehicular accident based on already existing data for the Seattle region

- Severity of  1 indicates that there was just property damage. Severity of 2 indicates serious injury or fatality

- The occurrence of each incident is highly dependent on the location  of the accident and the environmental conditions

## Business Understanding

- Target Label : severity code (SEVERITYCODE)
- Independent variables : X (Latitude), Y (Longitude) , Light Conditions (LIGHTCOND), weather conditions(WEATHER), Road Conditions (ROADCOND)
- The project stake holders are the Seattle City corporation for implementation of safety strategies and reduction of fatalities.
- Stakeholder groups includes state, federal and local government agencies, non-governmental organisations Car and life Insurance companies, Urgent and Emergency  care and other regional authorities

## Data Understanding

- The Data Set consists of a record of all accidents. Each row corresponds to a single incident. The main features or attributes that are going to form our training set are:
- location
- Road Condition
- Weather Condition
- Junction
- Car Speeding
- No. of people/vehicles involved
- light conditions

## Data Understanding

- On Analysis, the 'speeding' feature has mostly 'na' values and is not really suitable for the training set
- 'Location' feature contains the literal address of the accident location and hence is not a suitable attribute for the feature set.
- 'Light conditions' too is a categorical value with too many categories that may impede a proper data model development
- 'Road Conditions' is a categorical value and comprises<br>
- Dry, Wet, Unknown, Ice, Snow/Slush, Other, Standing Water, Sand/Mud/Dirt, Oil

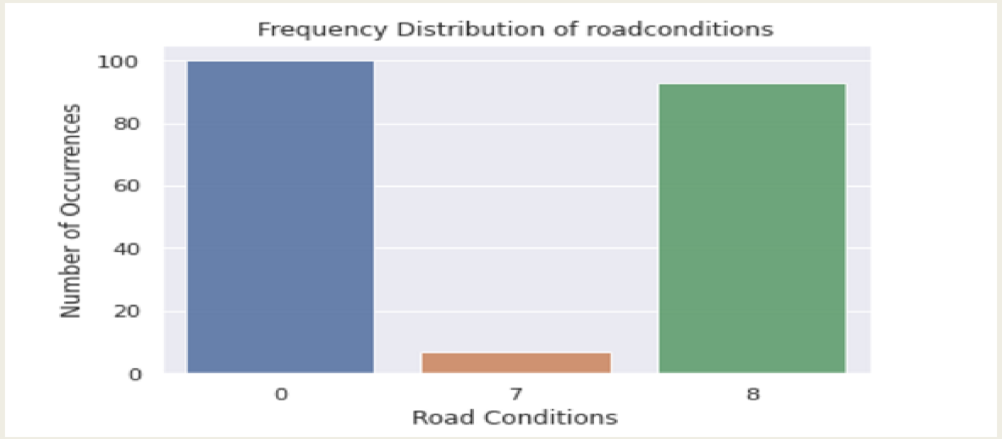
## Data Understanding

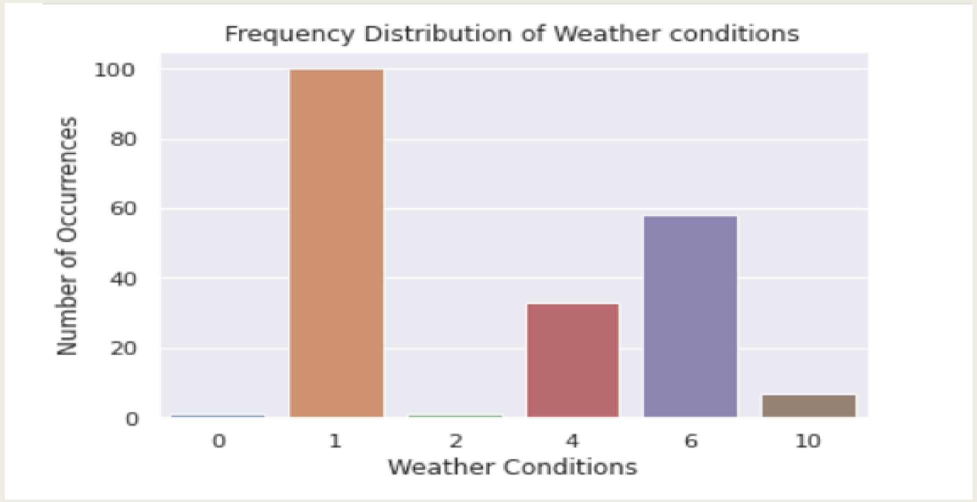
- The categorical values for 'Weather' feature are:
- Clear, Raining, Overcast, Snowing, Fog/Smog/Smoke, Sleet/Hail/Freezing Rain, Blowing Sand/Dirt, Severe Crosswind, Partly Cloudy
- 'JUNCTIONTYPE' categorical values:
- Mid-Block (not related to intersection), At Intersection (intersection related), Mid-Block (but intersection related), Driveway Junction, At Intersection (but not related to intersection), Ramp Junction,
- Hence the Features that would form a suitable feature set :
  - *[X, Y, ROADCOND, WEATHER, JUNCTIONTYPE, SEVERITYCODE]*

0 D

0 Dry 1 Ice 2 Oil 3 Other 4 Sand/Mud/Dirt 5 Snow/Slush 6 Standing Water 7 Unknown 8 Wet

# Data Understanding

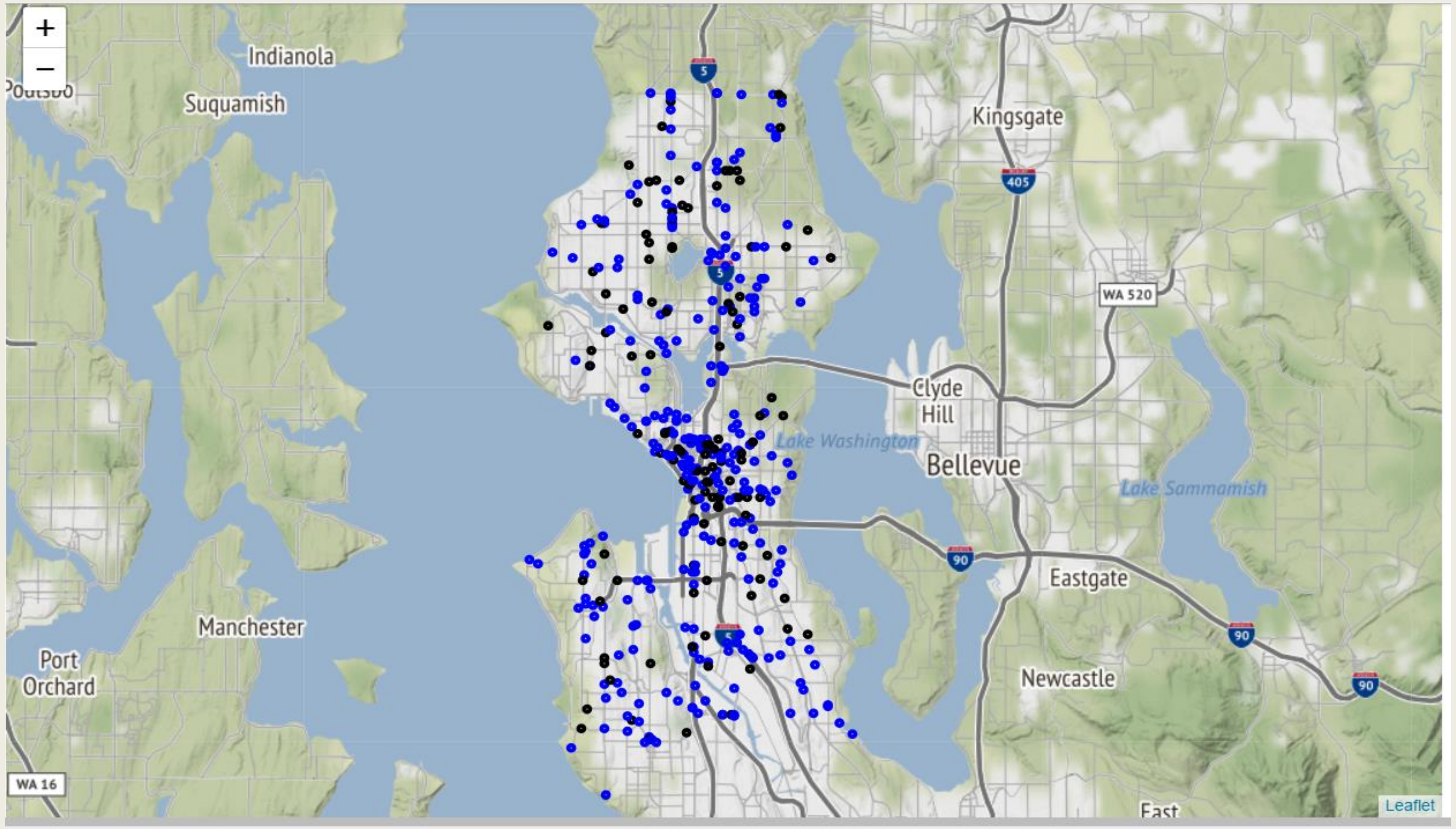




0- Dry
8 – Wet
7 - Unknown

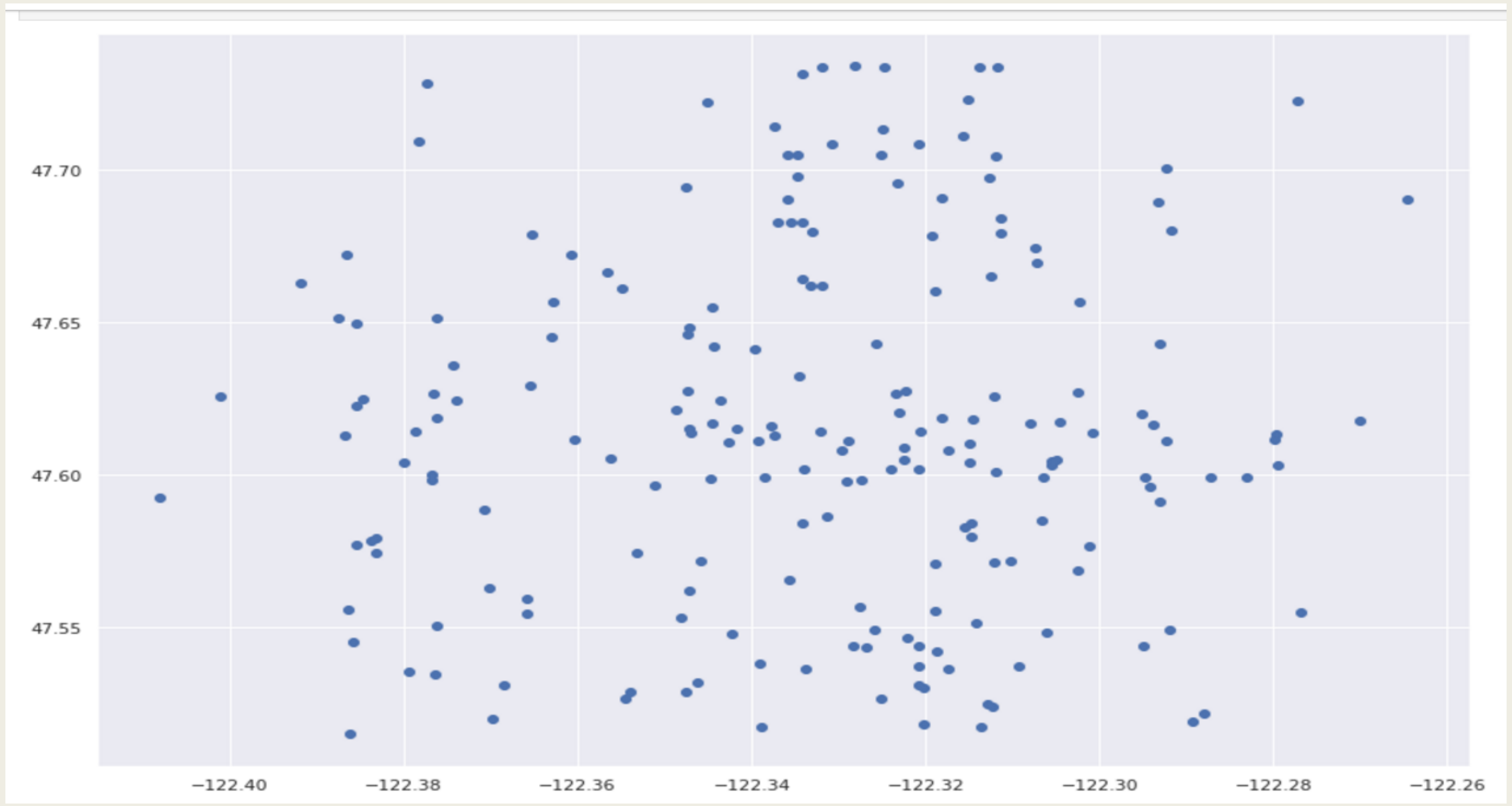
| | | | |
|---|---|---|---|
| 0 | Blowing Sand/Dirt | 6 | Raining |
| 1 | Clear | 7 | Severe Crosswind |
| 2 | Fog/Smog/Smoke | 8 | Sleet/Hail/Freezing Rain |
| 3 | Other | 9 | Snowing |
| 4 | Overcast | 10 | Unknown |
| 5 | Partly Cloudy | | |

## Data Understanding – A bird’s eye view of the plot of incidents in the sample size



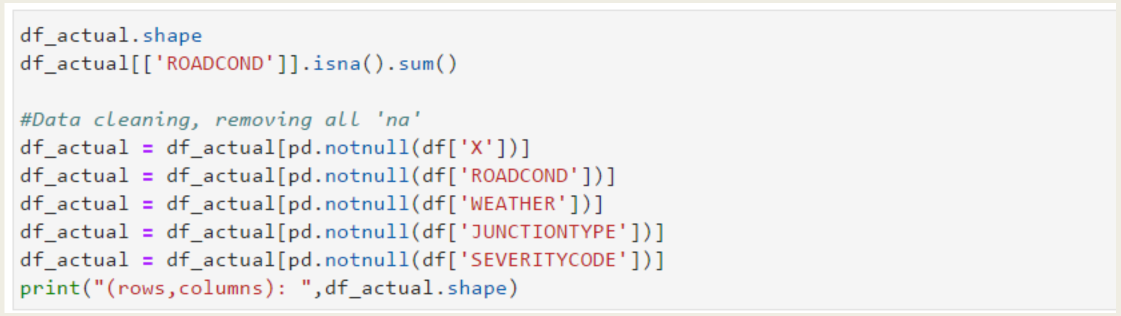
# Methodology

- In the above choropleth map we can see that all the black circles correspond to incidents of severity code 2 and all blue dots correspond to severity 1. Since we have the Longitudes and Latitudes as numeric values,we can plot a scatter plot indicating the same on a graph. Since the target label in in case is a categorical variable with discrete values, we will develop a model based on **classification techniques.**



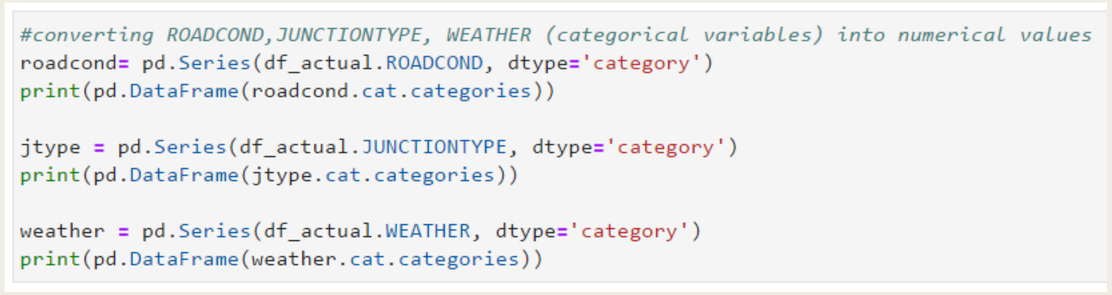
# Methodology

- Data Cleaning



```
df_actual.shape
df_actual[['ROADCOND']].isna().sum()

#Data cleaning, removing all 'na'
df_actual = df_actual[pd.notnull(df['X'])]
df_actual = df_actual[pd.notnull(df['ROADCOND'])]
df_actual = df_actual[pd.notnull(df['WEATHER'])]
df_actual = df_actual[pd.notnull(df['JUNCTIONTYPE'])]
df_actual = df_actual[pd.notnull(df['SEVERITYCODE'])]
print("(rows,columns): ",df_actual.shape)
```

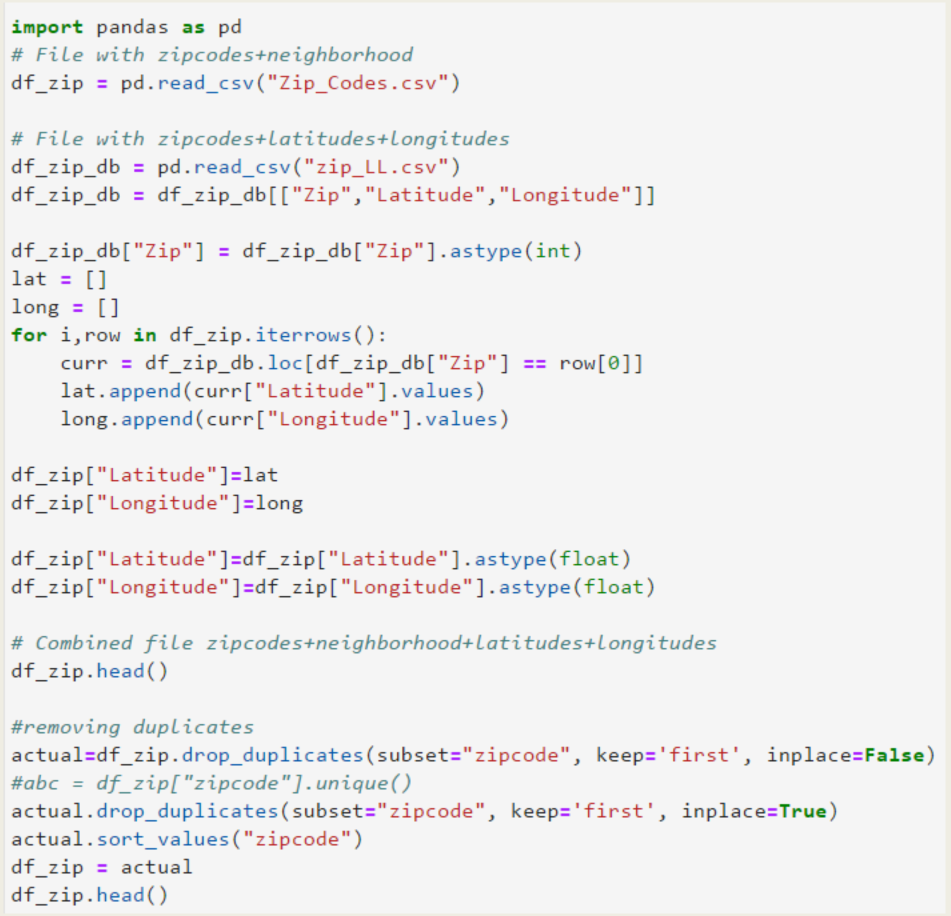
- Converting column values to Categorical Values



```
#converting ROADCOND,JUNCTIONTYPE, WEATHER (categorical variables) into numerical values
roadcond= pd.Series(df_actual.ROADCOND, dtype='category')
print(pd.DataFrame(roadcond.cat.categories))

jtype = pd.Series(df_actual.JUNCTIONTYPE, dtype='category')
print(pd.DataFrame(jtype.cat.categories))

weather = pd.Series(df_actual.WEATHER, dtype='category')
print(pd.DataFrame(weather.cat.categories))
```

# Methodology

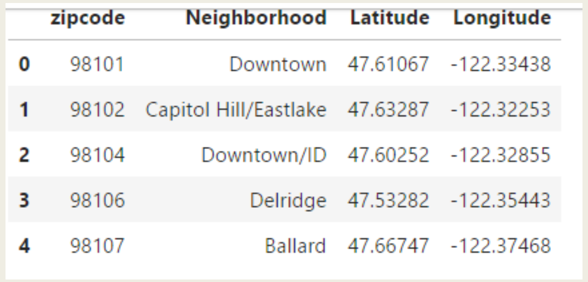
- Data scraping from the web to gain information table on Seattle Neighborhoods and their zipcodes



```python
import pandas as pd
# File with zipcodes+neighborhood
df_zip = pd.read_csv("Zip_Codes.csv")

# File with zipcodes+latitudes+longitudes
df_zip_db = pd.read_csv("zip_LL.csv")
df_zip_db = df_zip_db[["Zip","Latitude","Longitude"]]

df_zip_db["Zip"] = df_zip_db["Zip"].astype(int)
lat = []
long = []
for i,row in df_zip.iterrows():
    curr = df_zip_db.loc[df_zip_db["Zip"] == row[0]]
    lat.append(curr["Latitude"].values)
    long.append(curr["Longitude"].values)

df_zip["Latitude"]=lat
df_zip["Longitude"]=long

df_zip["Latitude"]=df_zip["Latitude"].astype(float)
df_zip["Longitude"]=df_zip["Longitude"].astype(float)

# Combined file zipcodes+neighborhood+latitudes+longitudes
df_zip.head()

#removing duplicates
actual=df_zip.drop_duplicates(subset="zipcode", keep='first', inplace=False)
#abc = df_zip["zipcode"].unique()
actual.drop_duplicates(subset="zipcode", keep='first', inplace=True)
actual.sort_values("zipcode")
df_zip = actual
df_zip.head()
```

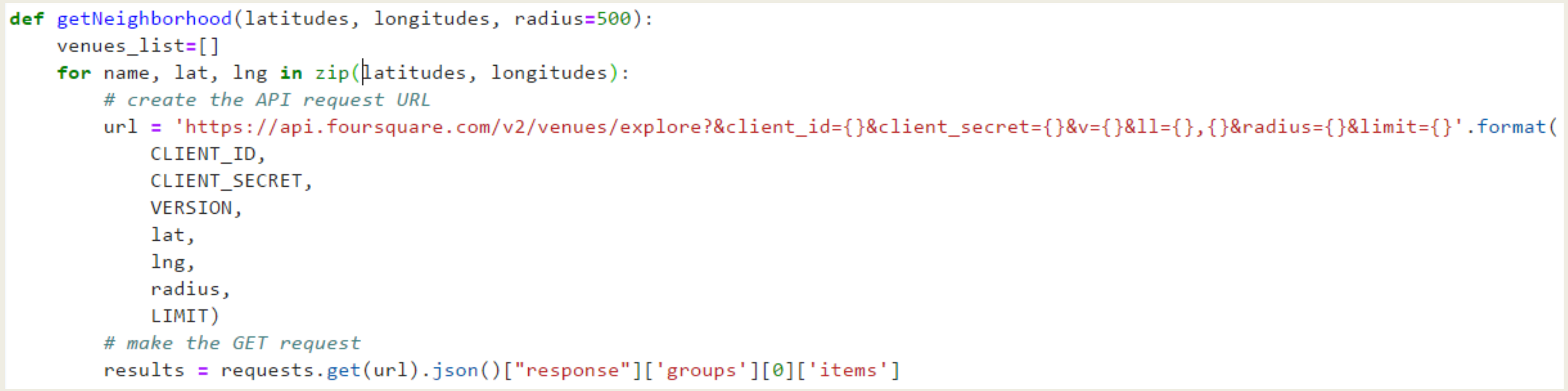
# Methodology

- The Zipcode file:



| | zipcode | Neighborhood | Latitude | Longitude |
|---|---|---|---|---|
| 0 | 98101 | Downtown | 47.61067 | -122.33438 |
| 1 | 98102 | Capitol Hill/Eastlake | 47.63287 | -122.32253 |
| 2 | 98104 | Downtown/ID | 47.60252 | -122.32855 |
| 3 | 98106 | Delridge | 47.53282 | -122.35443 |
| 4 | 98107 | Ballard | 47.66747 | -122.37468 |

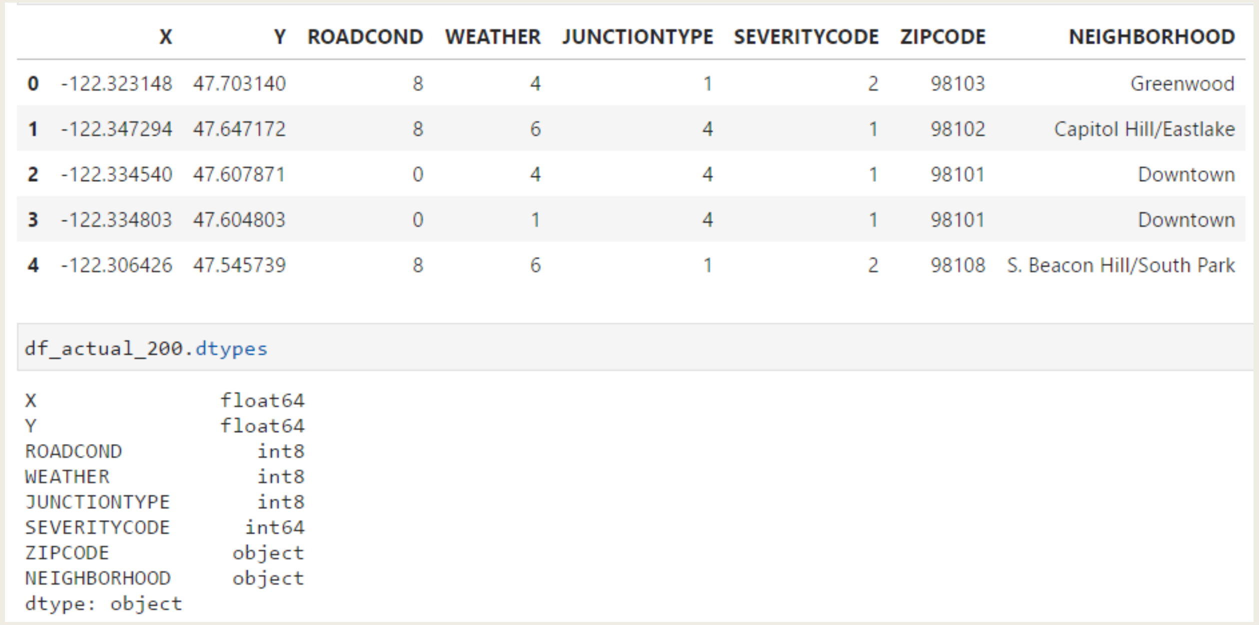
- Using FourSquareAPI, The sample data was segmented into neighborhoods



```
def getNeighborhood(latitudes, longitudes, radius=500):
    venues_list=[]
    for name, lat, lng in zip(latitudes, longitudes):
        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)
        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']
```

## Methodology

- The final dataset that is to be used to model is retrieved along with the zipcodes and the neighborhoods



| | X | Y | ROADCOND | WEATHER | JUNCTIONTYPE | SEVERITYCODE | ZIPCODE | NEIGHBORHOOD |
|---|---|---|---|---|---|---|---|---|
| **0** | -122.323148 | 47.703140 | 8 | 4 | 1 | 2 | 98103 | Greenwood |
| **1** | -122.347294 | 47.647172 | 8 | 6 | 4 | 1 | 98102 | Capitol Hill/Eastlake |
| **2** | -122.334540 | 47.607871 | 0 | 4 | 4 | 1 | 98101 | Downtown |
| **3** | -122.334803 | 47.604803 | 0 | 1 | 4 | 1 | 98101 | Downtown |
| **4** | -122.306426 | 47.545739 | 8 | 6 | 1 | 2 | 98108 | S. Beacon Hill/South Park |

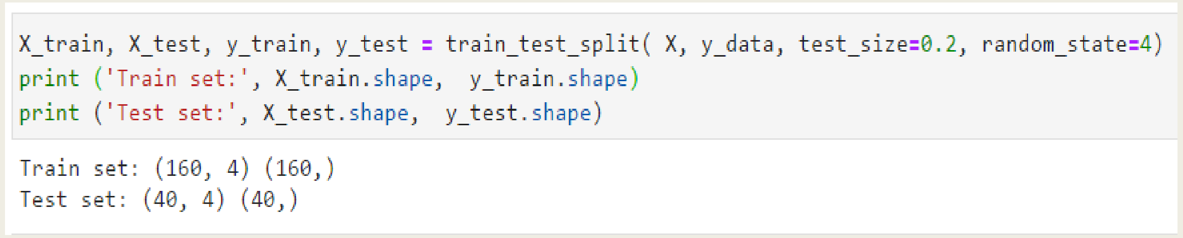
```
df_actual_200.dtypes
```

```
X                float64
Y                float64
ROADCOND            int8
WEATHER             int8
JUNCTIONTYPE        int8
SEVERITYCODE       int64
ZIPCODE           object
NEIGHBORHOOD      object
dtype: object
```

# Train and Test

■ After normalization, the data set is split into training set and test set where the target label is the severity code

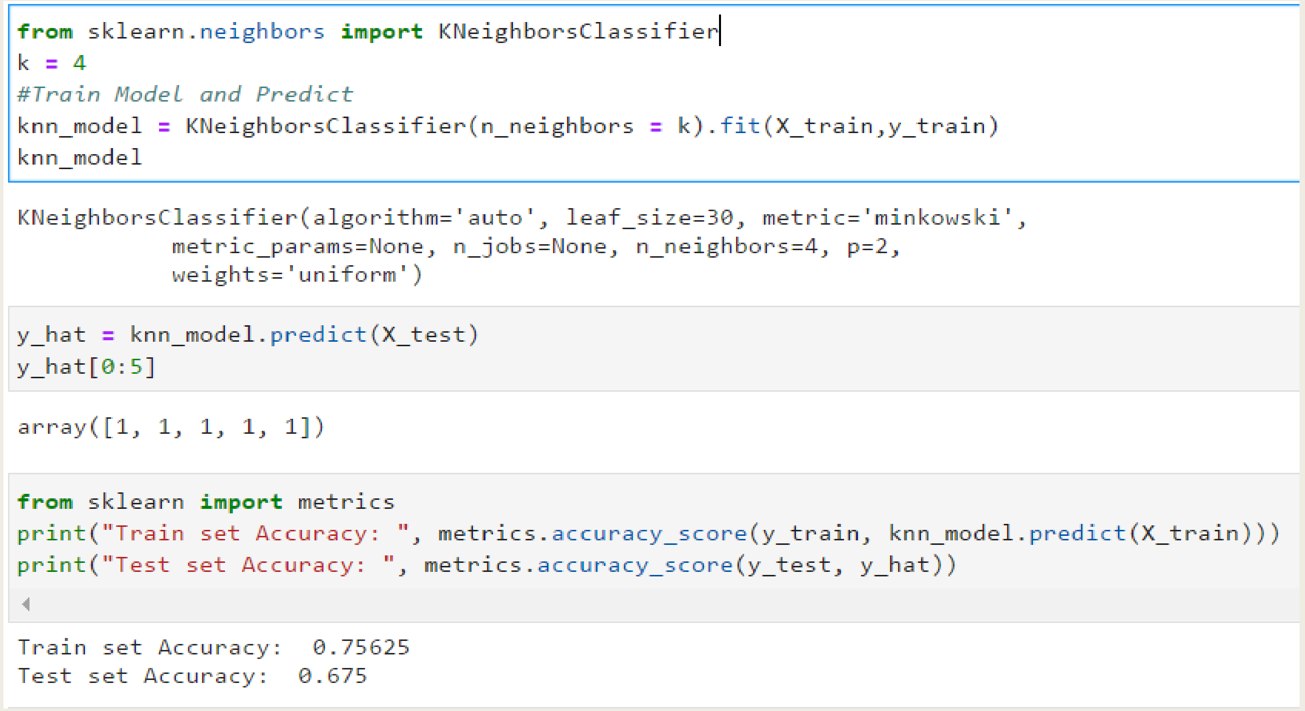
- X_data = df_actual_200[['ZIPCODE','ROADCOND','WEATHER','JUNCTIONTYPE']].values
- y_data = df_actual_200['SEVERITYCODE']
- X = preprocessing.StandardScaler().fit(X_data).transform(X_data.astype(float))
- X[0:5]
- X_train, X_test, y_train, y_test = train_test_split( X, y_data, test_size=0.2, random_state=4)
- print ('Train set:', X_train.shape,  y_train.shape)
- print ('Test set:', X_test.shape,  y_test.shape)



```
X_train, X_test, y_train, y_test = train_test_split( X, y_data, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (160, 4) (160,)
Test set: (40, 4) (40,)
```

## K-Nearest Neighbor classification Algorithm



```
from sklearn.neighbors import KNeighborsClassifier
k = 4
#Train Model and Predict
knn_model = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
knn_model
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=4, p=2,
           weights='uniform')
```

```
y_hat = knn_model.predict(X_test)
y_hat[0:5]
```

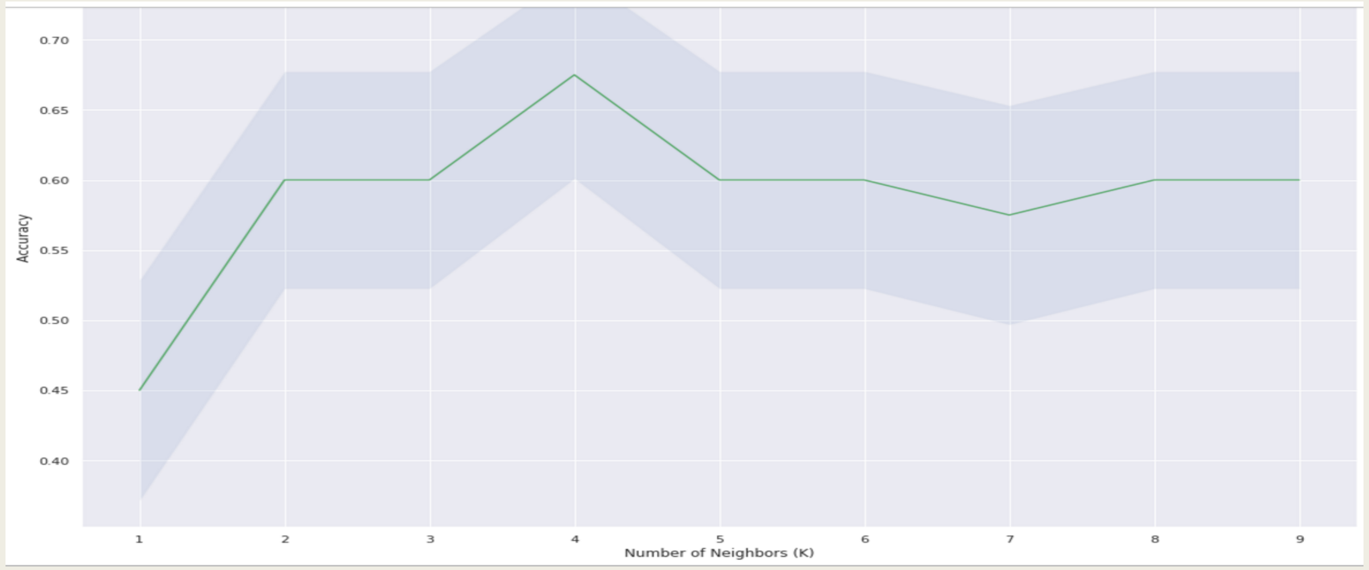
```
array([1, 1, 1, 1, 1])
```

```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, knn_model.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, y_hat))
```

```
Train set Accuracy:  0.75625
Test set Accuracy:  0.675
```

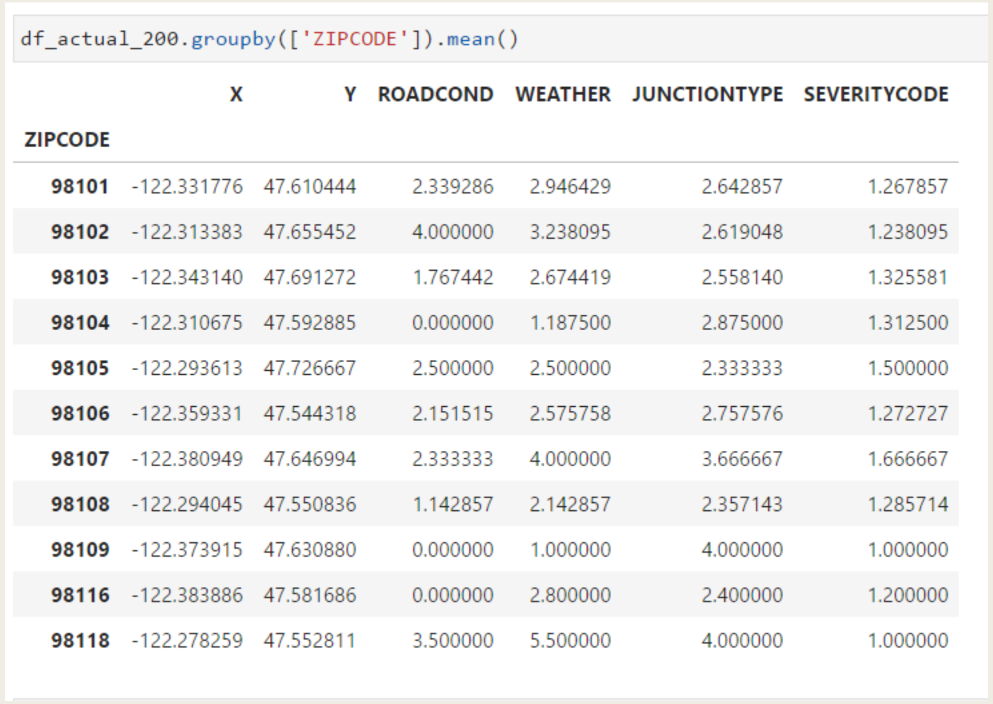
## Results

- Using the K-Nearest Neighbour Algorithm, we are able to develop a training set consisting of x_data = location(zipcode) and weather conditions and y_data = severity code
- This model gives us an accuracy score of 0.72 for the training set and 0.67 for the test set
- The best case of accuracy in prediction is obtained when the no. of neighbours considered, k=4



# Discussion

- When we see the data grouped by zipcode, we can see that certain zipcodes have a higher severity code than the others
- These observations are useful to the previously mentioned stakeholders including the Traffic police to avoid future motor accidents



```
df_actual_200.groupby(['ZIPCODE']).mean()
```

| ZIPCODE | X | Y | ROADCOND | WEATHER | JUNCTIONTYPE | SEVERITYCODE |
|---|---|---|---|---|---|---|
| **98101** | -122.331776 | 47.610444 | 2.339286 | 2.946429 | 2.642857 | 1.267857 |
| **98102** | -122.313383 | 47.655452 | 4.000000 | 3.238095 | 2.619048 | 1.238095 |
| **98103** | -122.343140 | 47.691272 | 1.767442 | 2.674419 | 2.558140 | 1.325581 |
| **98104** | -122.310675 | 47.592885 | 0.000000 | 1.187500 | 2.875000 | 1.312500 |
| **98105** | -122.293613 | 47.726667 | 2.500000 | 2.500000 | 2.333333 | 1.500000 |
| **98106** | -122.359331 | 47.544318 | 2.151515 | 2.575758 | 2.757576 | 1.272727 |
| **98107** | -122.380949 | 47.646994 | 2.333333 | 4.000000 | 3.666667 | 1.666667 |
| **98108** | -122.294045 | 47.550836 | 1.142857 | 2.142857 | 2.357143 | 1.285714 |
| **98109** | -122.373915 | 47.630880 | 0.000000 | 1.000000 | 4.000000 | 1.000000 |
| **98116** | -122.383886 | 47.581686 | 0.000000 | 2.800000 | 2.400000 | 1.200000 |
| **98118** | -122.278259 | 47.552811 | 3.500000 | 5.500000 | 4.000000 | 1.000000 |

## Conclusion

- Data science equips us with a system to predict an outcome before it happens
- This is especially very useful in the case of road accidents
- Using the data of environment conditions and locations of previous accidents we can now say with a 70% certainty that dry road conditions and mid-block junctions related to intersections cause a higher probability of accidents than any other conditions