SEATTLE CAR ACCIDENT SEVERITY PREDICTION

Business Problem:

It is estimated that road traffic accidents cost the United States' economy ~ $810 billion per year, including costs due to property damage, legal costs and associated medical bills. It is therefore important that we understand the factors influence the likelihood of a road traffic accident occurring at a given location, as well as those which influence the severity of the accidents.

The Seattle Department of Transport (SDOT) recorded all road traffic incidents in the Seattle municipal area between Jan 2004–Aug 2020. The Data has a predefined level of Severity caused by an accident. The main objective is to identify the key factors that determine the severity of accidents like Weather, Road Conditions, Light Conditions etc.

The target audience for this work will be city planners and emergency service responders:

By understanding the key factors that influence the severity of accidents, it will be possible to prevent or reduce the Severe or Fatal accidents in future by taking appropriate preventive measures.

Data:

Data Collection:

Data is obtained from all road traffic accidents recorded in the Seattle municipal area between Jan 2004–Aug 2020 by the Seattle Department of Transport (SDOT).

Data is available in Seattle Open Data portal and saved as CSV.

http://data-seattlecitygis.opendata.arcgis.com/datasets/5b5c745e0f1f48e7a53acec63a0022ab_0

The data can be read into a Pandas Data frame using the Pandas read_csv function, and the contents and data types displayed using the head and dtypes functions.

In [2]:
```
# Read the Data
df = pd.read_csv("C://Users/ManojKumar Chalamala/Downloads/Collisions.csv")
df.head()
```

Out[2]:

	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	LOCATION	...	ROADCOND	LIGHTCOND	PEDROV
0	-122.356511	47.517361	1	327920	329420	3856094	Matched	Intersection	34911.0	17TH AVE SW AND SW ROXBURY ST	...	Dry	Daylight	
1	-122.361405	47.702064	2	46200	46200	1791736	Matched	Block	NaN	HOLMAN RD NW BETWEEN 4TH AVE NW AND 3RD AVE NW	...	Wet	Dusk	
2	-122.317414	47.664028	3	1212	1212	3507861	Matched	Block	NaN	ROOSEVELT WAY NE BETWEEN NE 47TH ST AND NE 50T...	...	Dry	Dark - Street Lights On	
3	-122.318234	47.619927	4	327909	329409	EA03026	Matched	Intersection	29054.0	11TH AVE E AND E JOHN ST	...	Wet	Dark - Street Lights On	
4	-122.351724	47.560306	5	104900	104900	2671936	Matched	Block	NaN	WEST MARGINAL WAY SW BETWEEN	...	Ice	Dark - Street Lights On	

Dimensions of Data:

The Dataset contains 221738 rows (accidents) and 40 columns (attributes)

```
[3]: # Dimensions of the Dataframe

     df_shape = df.shape
     print("Dimensions of the data frame: "+str(df_shape))

     Dimensions of the data frame: (221738, 40)
```

Statistical Features of Dataset:

```
In [6]: # Explore the Statistical features of data

        df.describe().T.style.background_gradient(cmap='Set2',axis=0)

Out[6]:
```

	count	mean	std	min	25%	50%	75%	max
X	214260	-122.331	0.0300583	-122.419	-122.349	-122.33	-122.312	-122.239
Y	214260	47.6202	0.056059	47.4956	47.5771	47.616	47.6643	47.7341
OBJECTID	221738	110870	64010.4	1	55435.2	110870	166304	221738
INCKEY	221738	145007	89372.4	1001	71721.2	127358	210119	334276
COLDETKEY	221738	145237	89749.6	1001	71721.2	127358	210339	335776
INTKEY	72027	37637	52000.8	23807	28653	29973	33984	764413
PERSONCOUNT	221738	2.22674	1.4697	0	2	2	3	93
PEDCOUNT	221738	0.0380945	0.201704	0	0	0	0	6
PEDCYLCOUNT	221738	0.0273521	0.164512	0	0	0	0	2
VEHCOUNT	221738	1.72944	0.830529	0	2	2	2	15
INJURIES	221738	0.373964	0.73205	0	0	0	1	78
SERIOUSINJURIES	221738	0.0152026	0.158004	0	0	0	0	41
FATALITIES	221738	0.0017002	0.044967	0	0	0	0	5
SDOT_COLCODE	221737	13.3833	7.29829	0	11	11	14	87
SDOTCOLNUM	127205	7.97106e+06	2.61152e+06	1.00702e+06	6.00703e+06	8.03301e+06	1.0181e+07	1.3072e+07
SEGLANEKEY	221738	262.625	3252.88	0	0	0	0	525241
CROSSWALKKEY	221738	9568.04	71427.8	0	0	0	0	[illegible]

Report.pdf - Adobe Acrobat Reader DC

Data Preprocessing:

Remove the data with unknown information in the Target variable

The predefined target variable in the data determines the Car Accident Severity. However, there are few rows in the data frame with 'SEVERITYCODE = 0' which means an accident with "Unknown" Severity. We cannot use these accident data with unknown information to predict the Car Accident severity. So, these rows should be dropped.

```
In [8]: # Identify the rows with SeverityDESC = Unknown
        df['SEVERITYDESC'].value_counts()

Out[8]: Property Damage Only Collision    137776
        Injury Collision                   58842
        Unknown                            21657
        Serious Injury Collision            3111
        Fatality Collision                   352
        Name: SEVERITYDESC, dtype: int64
```

```
In [9]: # Remove the Unknown Accident Severity rows
        Unknown = df['SEVERITYDESC'] == 'Unknown'
        df.drop(df.index[Unknown], inplace=True)

        # Reset index of the data frame
        df.reset_index(inplace=True)
```

:

Relabel the Target Variable

The Target Variable "SEVERITYCODE" is having values (0, 1, 2, 2b, 3). It contains categorical values. So, this must be converted into numerical format. We have already dropped the rows with code value "0". So, we are left with (1, 2, 2b, 3)

Relabel the codes from (1, 2, 2b, 3) to (1, 2, 3, 4).

```
In [10]: # Values before Conversion
         print(df["SEVERITYCODE"].value_counts())

         # Convert the target variable value from (1. 2, 2b, 3) to (1, 2, 3, 4) by changing 2b to 3 and 3 to 4

         for i in range(0,len(df["SEVERITYCODE"])):
             if df["SEVERITYDESC"][i] == 'Serious Injury Collision':
                 df["SEVERITYCODE"][i] = 3
             if df["SEVERITYDESC"][i] == 'Fatality Collision':
                 df["SEVERITYCODE"][i] = 4

         # Converted values
         df["SEVERITYCODE"].value_counts()

         1     137776
         2      58842
         2b      3111
         3        352
         Name: SEVERITYCODE, dtype: int64

Out[10]: 1    137776
         2     58842
         3      3111
         4       352
         Name: SEVERITYCODE, dtype: int64
```

Remove Columns with unnecessary information:

Columns containing descriptions and identification numbers that would not help in the classification are dropped from the dataset to reduce the complexity and dimensionality of the dataset.

```
In [11]: df = df.drop(['OBJECTID','INCKEY','LOCATION','COLDETKEY','REPORTNO','STATUS','INTKEY','EXCEPTRSNCODE',
                       'EXCEPTRSNDESC','SEVERITYDESC','INCDATE','SDOT_COLCODE','SDOT_COLDESC','SDOTCOLNUM','ST_COLCODE',
                       'ST_COLDESC','SEGLANEKEY','CROSSWALKKEY','INCDTTM'],axis=1)
         df.columns

Out[11]: Index(['index', 'X', 'Y', 'ADDRTYPE', 'SEVERITYCODE', 'COLLISIONTYPE',
                'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'INJURIES',
                'SERIOUSINJURIES', 'FATALITIES', 'JUNCTIONTYPE', 'INATTENTIONIND',
                'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND', 'PEDROWNOTGRNT',
                'SPEEDING', 'HITPARKEDCAR'],
               dtype='object')
```

Finding missing values and handling them:

Empty boxes, ‘Unknown’ and ‘Other’ were values considered as missing values. These were replaced with NA to make the dataset uniform.

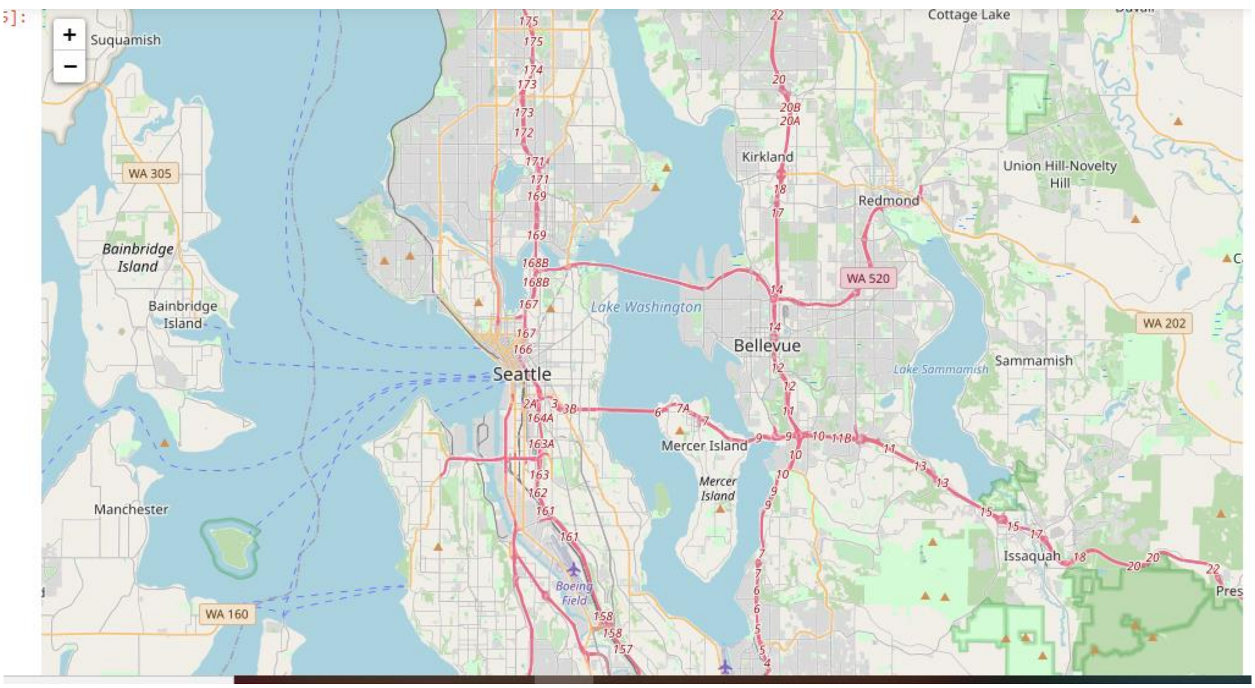
```
df.replace(r'^\s*$', np.nan, regex=True)
df.replace("Unknown", np.nan, inplace = True)
df.replace("Other", np.nan, inplace = True)
```

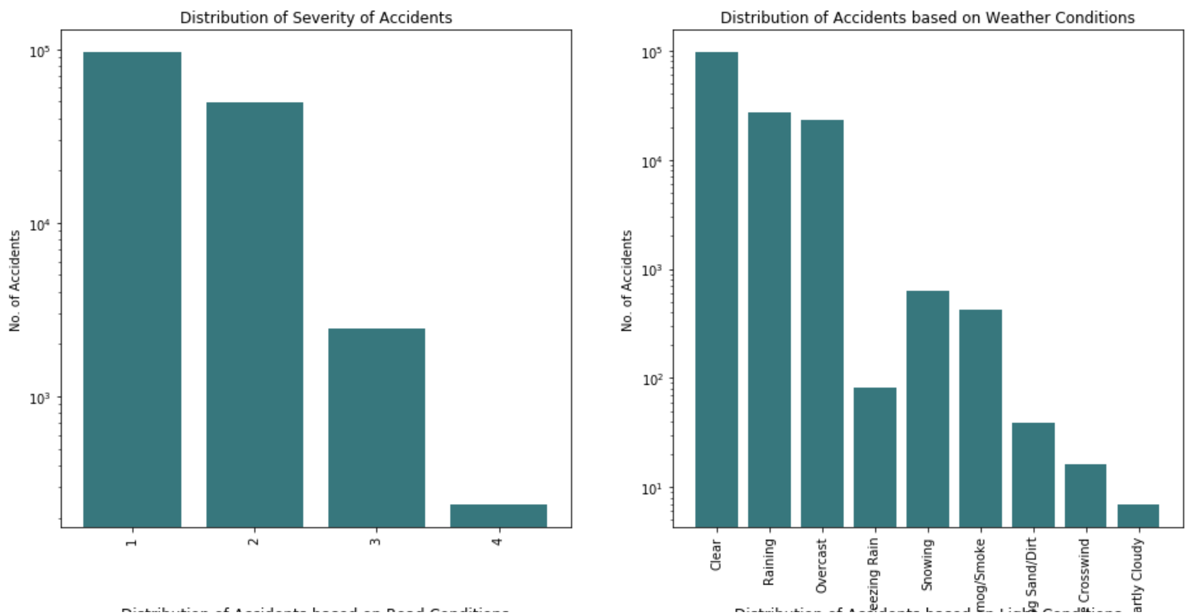
remove columns with more than 20% values missing

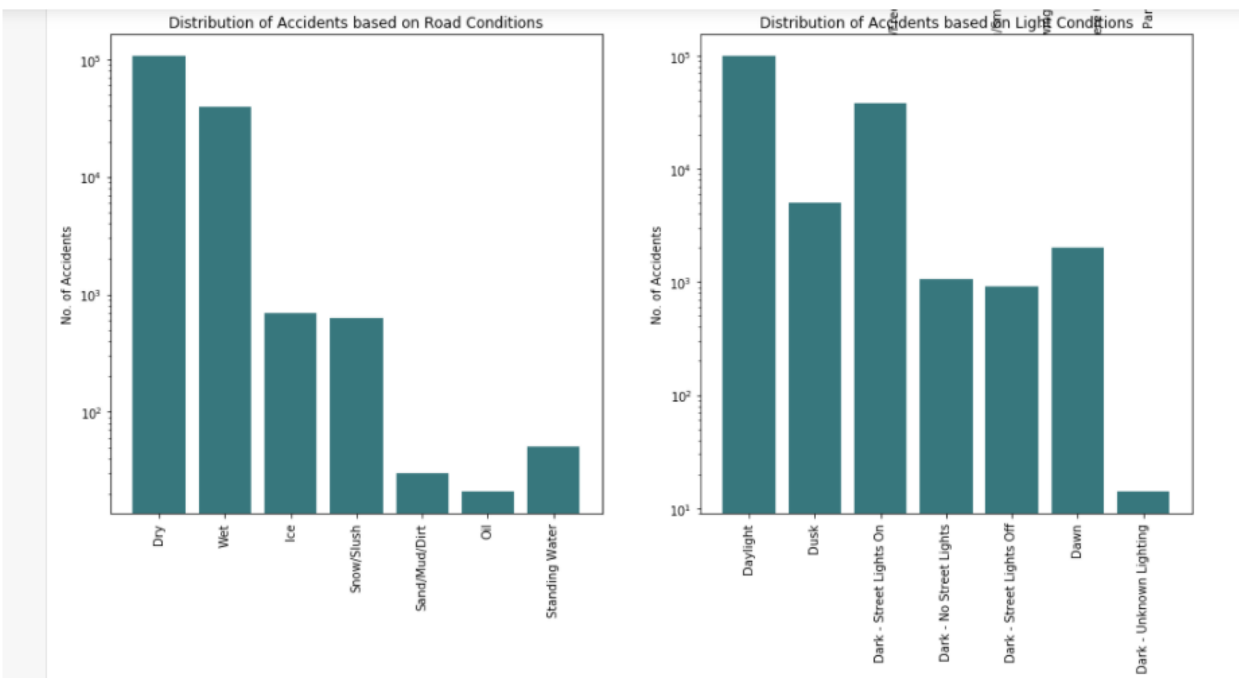
```
#removing columns with more than 20% values missing (INATTENTIONIND,PEDROWNOTGRNT,SPEEDING)
df = df.drop(["INATTENTIONIND","PEDROWNOTGRNT","SPEEDING"],axis=1)

#removing rows for columns with less than 20% values missing (X, Y,COLLISIONTYPE,JUNCTIONTYPE,
                                                      #UNDERINFL,WEATHER,ROADCOND,LIGHTCOND)
df.dropna(subset=["X","Y","COLLISIONTYPE","JUNCTIONTYPE","UNDERINFL","WEATHER","ROADCOND","LIGHTCOND"],
          axis=0, inplace=True)
```

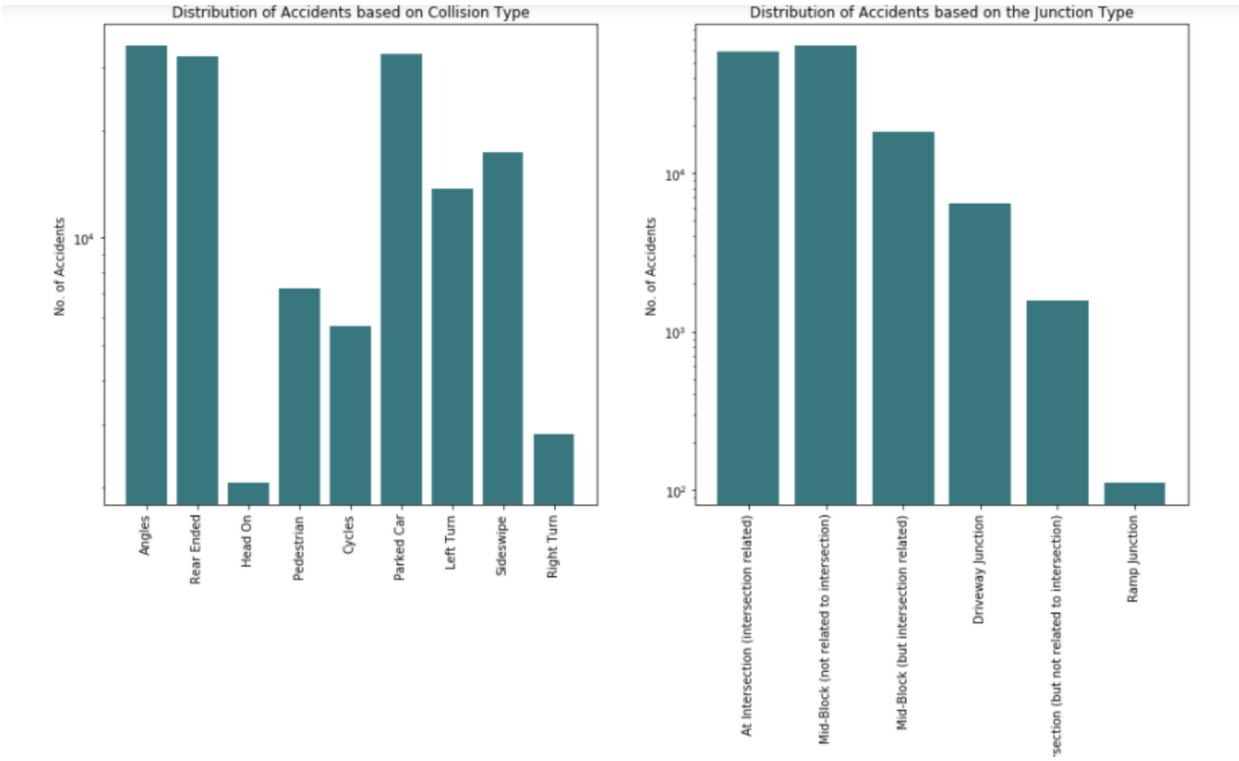
Visualizing the Data:

Seattle City Map of Accidents:




Distribution of Severity of Accidents
No. of Accidents
10⁵
10⁴
10³
1
2
3
4
Distribution of Accidents based on Weather Conditions
No. of Accidents
10⁵
10⁴
10³
10²
10¹
Clear
Raining
Overcast
eezing Rain
Snowing
mog/Smoke
g Sand/Dirt
Crosswind
artly Cloudy


Distribution of Accidents based on Road Conditions
No. of Accidents
10⁵
10⁴
10³
10²
Dry
Wet
Ice
Snow/Slush
Sand/Mud/Dirt
Oil
Standing Water
Distribution of Accidents based on Light Conditions
No. of Accidents
10⁵
10⁴
10³
10²
10¹
Daylight
Dusk
Dark - Street Lights On
Dark - No Street Lights
Dark - Street Lights Off
Dawn
Dark - Unknown Lighting



Most accidents (75.6%) occurred in clear or overcast (i.e. dry) weather conditions. The remaining 24.4% took place either in severe conditions (such as severe winds) or during periods of precipitation (rain, snow, fog, etc).

Road conditions at the time of each accident. Clearly the road conditions are related to the prevailing weather at the time (e.g. if there is rain, the roads are likely to be wet), however conditions are not wholly determined by the weather. For instance, 61 accidents occurred on roads where oil was present.

The light conditions at the time of each accident. 62.6% accidents occurred during daylight hours, while 26.2% of accidents occurred at nighttime in areas with streetlights (i.e. urban areas). The remaining 11.2% of accidents include those which happened at dawn/dusk, or on roads with no/faulty streetlights.

Balancing the Dataset:

As is clear from the histogram of severity code shown above most accidents involve either no injuries, or minor injuries only. Only a small number of accidents involve serious injuries or fatalities. If we train a classification model on these data, the model will be biased. To fix this issue we need to resample the data.

The Dataset contains 4 target variables i.e., Severity code: 1, 2, 3, and 4

Property Damage Only Collision   Severity Code 1
Injury Collision   Severity Code 2
Serious Injury Collision   Severity Code 3
Fatality Collision   Severity Code 4

Combined the Severity codes 2, 3 and 4 to one Severity. i.e., making it "0" which is completely related to Fatal or Injury. Code 1 represents Property damage.

```
In [23]: MyData['SEVERITYCODE'] = [1 if b=="1" else 0 for b in MyData.SEVERITYCODE]

         MyData['SEVERITYCODE'].value_counts()

Out[23]: 1    95913
         0    52258
         Name: SEVERITYCODE, dtype: int64
```

Down sample the Severity code 1 to match the number of Samples in Severity code 0

```
In [24]: # shuffling and creating a balanced dataset
         MyData= MyData.sample(frac=1,random_state=0,replace=False)

         # 1 - Put all severity code 2 class in a separate dataset.
         df_scode2 = MyData.loc[MyData['SEVERITYCODE'] == 0]

         # 2 - Randomly select 58188 observations from the severity code 1(majority class)
         df_scode1 = MyData.loc[MyData['SEVERITYCODE'] == 1].sample(n=52258,random_state=42)

         # 3 - concatenating datasets to get balanced dataset
         MyData_balanced = pd.concat([df_scode1,df_scode2])
         MyData_balanced = MyData_balanced.sample(frac=1,random_state=0,replace=False)

         #checking if dataset balanced
         print(MyData_balanced['SEVERITYCODE'].value_counts())
         MyData_balanced.info()

         1    52258
         0    52258
         Name: SEVERITYCODE, dtype: int64
```

Methodology:

Among all the features, the following features have the most influence in the accuracy of the predictions: WEATHER, ROADCOND, LIGHTCOND.

Encoding Categorical columns and creating dummy Variables

Machine Learning model should be trained only on numerical data. So, convert all Categorical Columns to numerical format by creating Dummy Variables.

Encoding Categorical columns and creating dummies

```
In [25]:
         Feature = MyData_balanced.iloc[:,1:]

         #Encoding Categorical Features - Training Dataset
         Feature = pd.get_dummies(data=Feature, columns=['ADDRTYPE','COLLISIONTYPE','JUNCTIONTYPE','WEATHER',
                                                          'ROADCOND','LIGHTCOND','UNDERINFL','HITPARKEDCAR'])
         del Feature["SEVERITYCODE"]
```

Normalize the Data and Split into Test and Train Set:

Normalizing and Feature Scaling

```
[27]: from sklearn import preprocessing
      X = preprocessing.StandardScaler().fit(Feature).transform(Feature)

      #Binarise SEVERITY code
      Y = MyData_balanced["SEVERITYCODE"]
```

Split Train and Test Set

```
[28]: # We split X and Y into train and test subsets
      from sklearn.model_selection import train_test_split
      X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
      print ('Train set:', X_train.shape,  Y_train.shape)
      print ('Test set:', X_test.shape,  Y_test.shape)

      Train set: (73161, 54) (73161,)
      Test set: (31355, 54) (31355,)
```

Model and Evaluation:

We now finally have a clean, balanced, and standardized dataset for the Seattle area. Categorical variables have been converted to numerical variables using standard data processing techniques. We are finally ready to begin building and testing models for predicting *SEVERITYCODE* from our chosen feature set.

The four models which will be built, tested, and compared are:

1. Decision Tree
2. Random forest
3. Logistic Regression
4. Support Vector Machine (SVM)

Decision Tree:

This model will build a decision tree by splitting and branching the data on all the possible values of every attribute in the dataset to determine the most predictive features in the dataset. The decision tree will then be used to predict the severity of an accident in the test dataset based on the values of those predictive features.

```
Accuracy of Decision Tree model:
Train set Accuracy:  1.0
Test set Accuracy:  1.0
Jaccard index: 1.00
F1-score: 1.00
R2-score: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     15609
           1       1.00      1.00      1.00     15746

    accuracy                           1.00     31355
   macro avg       1.00      1.00      1.00     31355
weighted avg       1.00      1.00      1.00     31355
```

Random Forest:

This is an ensemble algorithm which combines more than one algorithm of same or different kind for classifying objects tree-based learning algorithm.

RFC is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. Used for both classification and regression

A hyper parameter RFT was used to determine the best choices for the above mentioned parameters.

```
Best Hyperparameter RFT :  {'criterion': 'gini', 'n_estimators': 75, 'random_state': 0}
[[15600     9]
 [    0 15746]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     15609
           1       1.00      1.00      1.00     15746

    accuracy                           1.00     31355
   macro avg       1.00      1.00      1.00     31355
weighted avg       1.00      1.00      1.00     31355


0.9997129644394833
```

Logistic Regression:

Logistic Regression is useful when the observed dependent variable, y, is categorical. It produces a formula that predicts the probability of the class label as a function of the independent variables.

```
Accuracy of Logistic Regression model:
Train set Accuracy:  0.9999863315154249
Test set Accuracy:  0.9998724286397703
Jaccard index: 1.00
F1-score: 1.00
R2-score: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     15609
           1       1.00      1.00      1.00     15746

    accuracy                           1.00     31355
   macro avg       1.00      1.00      1.00     31355
weighted avg       1.00      1.00      1.00     31355
```

Support Vector Machine (SVM)

The target variable SEVERITYCODE is not binary in this dataset, and therefore is not suited to logistic regression techniques. Instead, SVM will be used to map the training data to a multi-dimensional space (allowing hyperplanes to be fit which cleanly separate accidents with different severity codes), and then these hyperplanes will be used to predict the SEVERITYCODE of accidents in the test dataset, given the values of its independent variables.

```
Accuracy of SVM model:
Train set Accuracy:  1.0
Test set Accuracy:  1.0
Jaccard index: 1.00
F1-score: 1.00
R2-score: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     15609
           1       1.00      1.00      1.00     15746

    accuracy                           1.00     31355
   macro avg       1.00      1.00      1.00     31355
weighted avg       1.00      1.00      1.00     31355
```

Results:

Model	Precision	recall	f1-score	Jaccard Index
Decision Tree	1.00	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00	1.00
Logistic Regression	1.00	1.00	1.00	1.00
SVM	1.00	1.00	1.00	1.00

Conclusion:

The accuracy of the classifiers is excellent, i.e. 100%. This means that the model has trained well and fits the training data and performs well on the testing set as well as the training set. We can conclude that this model can accurately predict the severity of car accidents in Seattle

Future Work:

- In future, the model could be improved to predict the accident severity on a continuum running from 1–4, rather than simply predicting a binary accident severity of 0 (minor) or 1 (major).

- In future, it may be worth revisiting this work and modelling the accident data in five-year chunks, to see if the features which best predict accident severity have changed over time.