Capstone Project Report

Business Understanding

The Seattle government has undertaken a project to minimize the number of accidents caused and the severity of accidents. For the project they need insight from the data about the major factors that contribute in the accidents.

The government is aware of human factors such as micro-sleep, drugs and alcohol that are causing accidents which may or may not be prevented but they are interested in non-human factors which would surely minimize the accidents and accidents severity if the causes are fixed. These major factors include of Weather, Light conditions and Road conditions.

Therefore, these major factors can be studied and a pattern can be found out to accomplish the goal.¶

Data exlporation

The data was collected by the Seattle Police Department and Accident Traffic Records Department from 2004 to present.The data consists of 37 independent variables and 194,673 rows. The dependent variable, “SEVERITYCODE”.

In [1]:

```python
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('C:/Users/ACHAL SHAH/Desktop/Data-Collisions.csv')
df.head()
```

Out[1]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADC
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	

5 rows × 38 columns

In [2]:

```
df.dtypes
```

Out[2]:

```
SEVERITYCODE        int64
X                 float64
Y                 float64
OBJECTID            int64
INCKEY              int64
COLDETKEY           int64
REPORTNO           object
STATUS             object
ADDRTYPE           object
INTKEY            float64
LOCATION           object
EXCEPTRSNCODE      object
EXCEPTRSNDESC      object
SEVERITYCODE.1      int64
SEVERITYDESC       object
COLLISIONTYPE      object
PERSONCOUNT         int64
PEDCOUNT            int64
PEDCYLCOUNT         int64
VEHCOUNT            int64
INCDATE            object
INCDTTM            object
JUNCTIONTYPE       object
SDOT_COLCODE        int64
SDOT_COLDESC       object
INATTENTIONIND     object
UNDERINFL          object
WEATHER            object
ROADCOND           object
LIGHTCOND          object
PEDROWNOTGRNT      object
SDOTCOLNUM        float64
SPEEDING           object
ST_COLCODE         object
ST_COLDESC         object
SEGLANEKEY          int64
CROSSWALKKEY        int64
HITPARKEDCAR       object
dtype: object
```

Furthermore, because of the existence of null values in some records, the data needs to be pre-processed before any further processing. The data set in the original form is not ready for data analysis. In order to prepare the data, first, we need to drop the non-relevant columns. In addition, most of the features are of object data types that need to be converted into numerical data types.After analyzing the data set, I have decided to focus on only four features, severity, weather conditions, road conditions, and light conditions, among others.

In [3]:

```
pre_df=df[["SEVERITYCODE","WEATHER","ROADCOND","LIGHTCOND"]]

# Convert object columns to category
pre_df["WEATHER"] = pre_df["WEATHER"].astype('category')
pre_df["ROADCOND"] = pre_df["ROADCOND"].astype('category')
pre_df["LIGHTCOND"] = pre_df["LIGHTCOND"].astype('category')

# Create new column for analysis
pre_df["WEATHER_CAT"] = pre_df["WEATHER"].cat.codes
pre_df["ROADCOND_CAT"] = pre_df["ROADCOND"].cat.codes
pre_df["LIGHTCOND_CAT"] = pre_df["LIGHTCOND"].cat.codes

pre_df.dtypes
```

Out[3]:

```
SEVERITYCODE        int64
WEATHER          category
ROADCOND         category
LIGHTCOND        category
WEATHER_CAT          int8
ROADCOND_CAT         int8
LIGHTCOND_CAT        int8
dtype: object
```

Value count of targeted variables - severity, weather, light and road conditions.

In [4]:

```
pre_df["SEVERITYCODE"].value_counts()
```

Out[4]:

```
1    136485
2     58188
Name: SEVERITYCODE, dtype: int64
```

In [5]:

```
pre_df["WEATHER"].value_counts()
```

Out[5]:

```
Clear                       111135
Raining                      33145
Overcast                     27714
Unknown                      15091
Snowing                        907
Other                          832
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               56
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [6]:

```
pre_df["ROADCOND"].value_counts()
```

Out[6]:

```
Dry               124510
Wet                47474
Unknown            15078
Ice                 1209
Snow/Slush          1004
Other                132
Standing Water       115
Sand/Mud/Dirt         75
Oil                   64
Name: ROADCOND, dtype: int64
```

In [7]:

```
pre_df["LIGHTCOND"].value_counts()
```

Out[7]:

```
Daylight                    116137
Dark - Street Lights On      48507
Unknown                      13473
Dusk                          5902
Dawn                          2502
Dark - No Street Lights       1537
Dark - Street Lights Off      1199
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

In [8]:

```
from sklearn.utils import resample

pre_df_maj = pre_df[pre_df.SEVERITYCODE==1]
pre_df_min = pre_df[pre_df.SEVERITYCODE==2]

pre_df_maj_dsample = resample(pre_df_maj,
                              replace=False,
                              n_samples=58188,
                              random_state=123)

balanced_df = pd.concat([pre_df_maj_dsample, pre_df_min])

balanced_df.SEVERITYCODE.value_counts()
```

Out[8]:

```
2    58188
1    58188
Name: SEVERITYCODE, dtype: int64
```

In [9]:

```
X = np.asarray(balanced_df[['WEATHER_CAT', 'ROADCOND_CAT', 'LIGHTCOND_CAT']])
X[0:5]
```

Out[9]:

```
array([[ 6,  8,  2],
       [ 1,  0,  5],
       [10,  7,  8],
       [ 1,  0,  5],
       [ 1,  0,  5]], dtype=int8)
```

In [10]:

```
y = np.asarray(balanced_df['SEVERITYCODE'])
y [0:5]
```

Out[10]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [11]:

```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

Out[11]:

```
array([[ 1.15236718,  1.52797946, -1.21648407],
       [-0.67488   , -0.67084969,  0.42978835],
       [ 2.61416492,  1.25312582,  2.07606076],
       [-0.67488   , -0.67084969,  0.42978835],
       [-0.67488   , -0.67084969,  0.42978835]])
```

Methodology

Once I have load data into Pandas Dataframe, used ‘dtypes’ attribute to check the feature names and their data types. Then I have selected the most important features to predict the severity of accidents in Seattle. Among all the features, the road, weather and light conditions features have the most influence in the accuracy of the predictions.

Also, as I mentioned earlier, “SEVERITYCODE” is the target variable.I have run a value count on road (‘ROADCOND’) and weather condition (‘WEATHER’) to get ideas of the different road and weather conditions. I also have run a value count on light condition (’LIGHTCOND’), to see the breakdowns of accidents occurring during the different light conditions.

After balancing SEVERITYCODE feature, and standardizing the input feature, the data has been ready for building machine learning models.

After importing necessary packages and splitting preprocessed data into test and train sets, for each machine learning model, I have built and evaluated the model and shown the results as follow:

Model and Evaluation

In [12]:

```
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss

#Train and Test Sets

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)
print ('Train set rows:', X_train.shape[0])
print ('Test set rows:', X_test.shape[0])
```

```
Train set rows: 81463
Test set rows: 34913
```

K Nearst Neigbours

In [13]:

```
from sklearn.neighbors import KNeighborsClassifier
k = 14
knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)

knn_y_pred = knn.predict(X_test)
knn_y_pred[0:5]
```

Out[13]:

```
array([2, 2, 1, 1, 2], dtype=int64)
```

In [14]:

```
jaccard_score(y_test, knn_y_pred)
```

Out[14]:

```
0.311108117816091 93
```

In [15]:

```
f1_score(y_test, knn_y_pred, average='macro')
```

Out[15]:

```
0.5484494712246419
```

Decision Tree

In [16]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth = 7)

dt.fit(X_train,y_train)
```

Out[16]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [17]:

```
dt_y_pred = dt.predict(X_test)
jaccard_score(y_test, dt_y_pred)
```

Out[17]:

```
0.2873687679487783
```

In [18]:

```
f1_score(y_test, dt_y_pred, average='macro')
```

Out[18]:

```
0.5450597937389444
```

Linear Regression

In [19]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=6, solver='liblinear').fit(X_train,y_train)

LR_y_pred = LR.predict(X_test)
LR_y_prob = LR.predict_proba(X_test)

LR_y_prob = LR.predict_proba(X_test)
log_loss(y_test, LR_y_prob)
```

Out[19]:

```
0.6849535383198887
```

In [20]:

```
jaccard_score(y_test, LR_y_pred)
```

Out[20]:

```
0.2720073907879108
```

In [21]:

```
f1_score(y_test, LR_y_pred, average='macro')
```

Out[21]:

```
0.511602093963383
```

Model Accuracy

In [22]:

```
from sklearn.metrics import accuracy_score
```

In [23]:

```
print("KNN Accuracy: ", accuracy_score(y_test, knn_y_pred))
```

```
KNN Accuracy:  0.5605361899578953
```

In [24]:

```
print("Decision Tree Accuracy: ", accuracy_score(y_test, dt_y_pred))
```

```
Decision Tree Accuracy:  0.5664365709048206
```

In [25]:

```
print("LR Accuracy: ", accuracy_score(y_test, LR_y_pred))
```

```
LR Accuracy:  0.5260218256809784
```

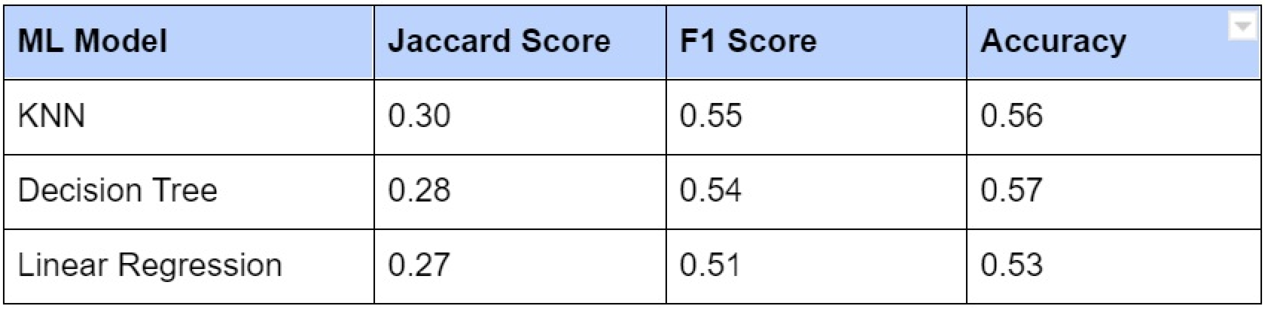
Result and Evaluation

In [26]:

```
from IPython.display import Image

Image("C:/Users/ACHAL SHAH/Desktop/capstone image.png")
```

Out[26]:



ML Model	Jaccard Score	F1 Score	Accuracy
KNN	0.30	0.55	0.56
Decision Tree	0.28	0.54	0.57
Linear Regression	0.27	0.51	0.53

Based on the above table, KNN is the best model to predict car accident severity.

Conclusion

Based on the above, about 30-50% accidents and accidents severity are due to the cause of weather, light and road conditions. Apart from that human error and other factors that are not considered in the data may lead to the accidents.

Thank you!

In [ ]:

```
```