Accident Severity

Moonkyu, Park

October, 10 , 2020

1. Introduction

1.1 Background

The total number of vehicles worldwide is increasing every year.
It mean that Car accidents can occur all the time all the way.
However there are some conditions were the probabilities of have an accident arise due multiple variables. This report has as purpose develop a model for Seattle government to predict the probabilities of have a car accident and severity, based on different conditions as weather or road conditions.

The information was provided by Seattle Police Department form 2004 to 2020.

1.2 Problem

Identify the conditions that can cause future car accidents in order to alarm the people with anticipation to be aware and drive more carefully.
In an effort to reduce the frequency of car collisions in a community, an alogorithm must be developed to predict the severity of an accdient given the current features.

It will give us THREE BIG benefits :
1. Save lives as main benefit
2. Reduce costs in damage infrastructure
3. Reduce cost from police and paramedics to attend each accident

1.3 Interest

If you want to see metadata or get more detailed information on the data set, please refer to the link below.
https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Metadata.pdf

2. Data

It comes from Seattle Police Department and recorded by Traffic Records and include Collisions at intersection or mid-block of a segment. The period information is from 2004 to May 2020.

The information is organized in a CSV File with 37 attributes and originally 194673 rows. Information is labeled and unbalanced. Additionally a document with the description of each column were given.

Due our information is labeled we know the result for each record, we have select the column **SEVERITYCODE** as Dependent variable. The possible values are:

1 Property Damage Only Collision
2 Injury Collision

The information is unbalanced by the difference in samples for each accident type. In our case there are only two types of accidents. Look at the picture below:

```
In [9]: df["SEVERITYCODE"].value_counts()

   Out[9]: 1    136485
           2     58188
           Name: SEVERITYCODE, dtype: int64
```

it's original form, this data is not fit for analysis. For one, there are many columns that we will not use for this model. Also most of the features are of type object, when they should be numerical type.
We must use label encoding to covert the features to our desired data type.

Then I will briefly introduce the overall data preprocessing process, such as handling missing values, changing data types appropriately, adjusting data to balance, and normalizing.

2.1 Missing Values & Convert to date time object

Missing Values

```
1  main_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 38 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SEVERITYCODE    194673 non-null  int64
 1   X               189339 non-null  float64
 2   Y               189339 non-null  float64
 3   OBJECTID        194673 non-null  int64
 4   INCKEY          194673 non-null  int64
 5   COLDETKEY       194673 non-null  int64
 6   REPORTNO        194673 non-null  object
 7   STATUS          194673 non-null  object
 8   ADDRTYPE        192747 non-null  object
 9   INTKEY          65070 non-null   float64
 10  LOCATION        191996 non-null  object
 11  EXCEPTRSNCODE   84811 non-null   object
 12  EXCEPTRSNDESC   5638 non-null    object
 13  SEVERITYCODE.1  194673 non-null  int64
```

Convert to date time object

```
1  main_df['Accident Time'] = pd.to_datetime(df['INCDTTM'])
2  main_df.head()
```

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	R
0	2	-122.323148	47.703140	1	1307	1307	
1	1	-122.347294	47.647172	2	52200	52200	
2	1	-122.334540	47.607871	3	26700	26700	

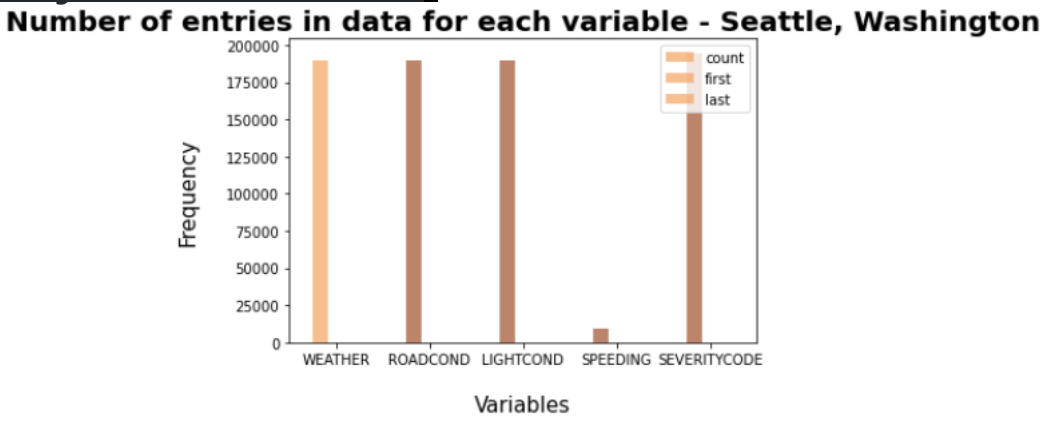
2.2 Data Visualization and Preprocessing

Data visualization and pre-processing

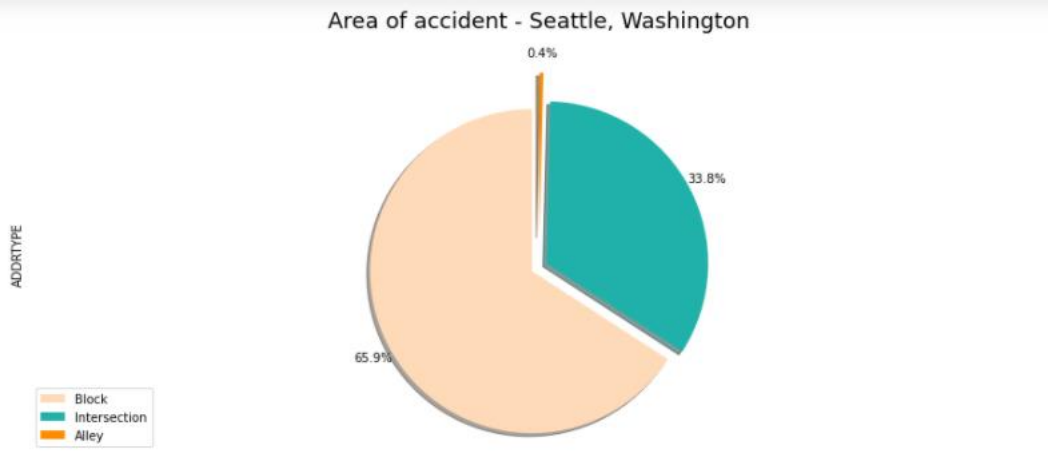
```
: 1 main_df["SEVERITYCODE"].value_counts()
```

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

2.3 Plotting counts of selected variables



2.4 Area type of each accident



2.5 Making new dataframe with only variables and unique keys

```
selected_columns=main_df[["X","Y","INCKEY","INATTENTIONIND","UNDERINFL","WEATHER","ROADCOND","LIGHTCON
feature_df=selected_columns.copy()
feature_df.dropna(axis=0,how='any',inplace=True)
feature_stats=feature_df.describe()

np.count_nonzero(feature_df['UNDERINFL'])
```

2.6 Light Condition

In [113]:
```
lightcondsize = feature_df ["LIGHTCOND"].size

featureinlightcond = feature_df ['LIGHTCOND'] == 'Unknown'

lightcond = feature_df['LIGHTCOND']
lightcond = lightcond.values
lightcond = lightcond[featureinlightcond]

lightcond[0:9036]=0
lightcond[9036:13417]=1
lightcond[13417:13961]=2

feature_df.loc [feature_df.LIGHTCOND == "Unknown", 'LIGHTCOND'] = lightcond

feature_df["LIGHTCOND"]=feature_df["LIGHTCOND"].astype(int)
```

2.7 Weather Condition

In [115]:
```
weathersize = feature_df ["WEATHER"].size

featureinweather = feature_df ['WEATHER'] == 'Unknown'

weather = feature_df['WEATHER']
weather = weather.values
weather = weather[featureinweather]

weather[0:10151]=0
weather[10151:12683]=1
weather[12683:12742]=2
weather[12742:15864]=3

feature_df.loc[feature_df.WEATHER == "Unknown", 'WEATHER'] = weather
feature_df["WEATHER"]=feature_df["WEATHER"].astype(int)

```

2.8 Converting remaining to int

In [116]:
```
feature_df["SPEEDING"]=feature_df["SPEEDING"].astype(int)
feature_df["INATTENTIONIND"]=feature_df["INATTENTIONIND"].astype(int)
feature_df["UNDERINFL"]=feature_df["UNDERINFL"].astype(int)

```

2.9 Test/Train split

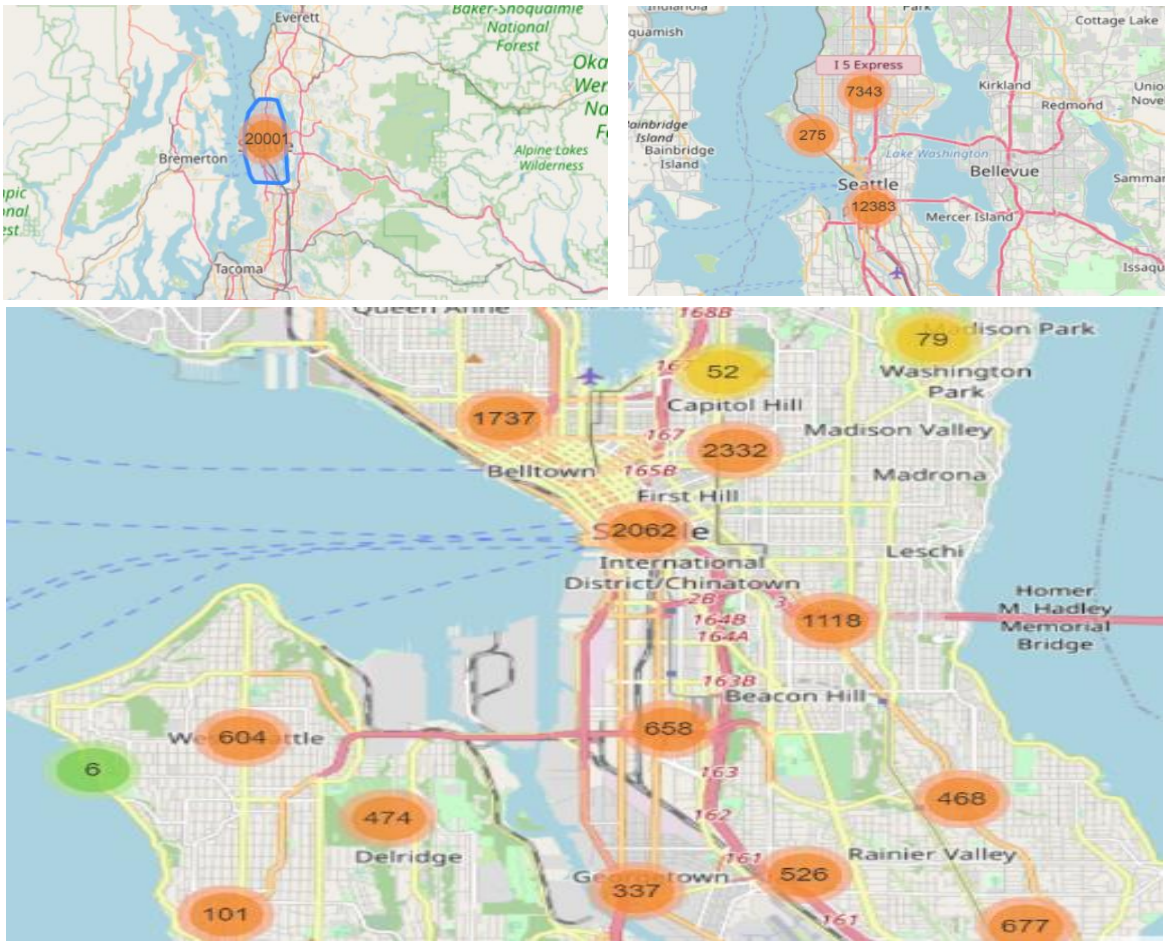
In [118]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)

```

```
Train set: (128916, 6) (128916, 1)
```

2.10 Balance the Data

In [119]:
```
os = SMOTE (random_state=0)
os_data_X, os_data_y= os.fit_sample(X_train, y_train)
```

2.11 Folium Map



3.Modeling

3.1 Methodology

For implementing the solution, I have used Github as a repository and running Jupyter Notebook to preprocess data and build Machine Learning models. Regarding coding, I have used Python and its popular packages such as Pandas, NumPy and Sklearn.

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 following features have the most influence in the accuracy of the predictions:

__“WEATHER”,__

__“ROADCOND”__

__“LIGHTCOND”__

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. The results can be seen below:

3.2 KNN

K Nearest Neighbor(KNN)

In [127]:
```
k=17
knn=KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)

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

Out[127]: array([1, 1, 1, 1, 1], dtype=int64)

KNN Evaluation

In [128]:
```
jaccard_score(y_test,knn_y_pred)
```

Out[128]: 0.6604929023558935

In [129]:
```
f1_score(y_test,knn_y_pred,average='macro')
```

Out[129]: 0.4826329321775803

3.3 Decision Tree

Decision Tree

In [130]:
```
dt=DecisionTreeClassifier(criterion="entropy", max_depth=7)

dt.fit(X_train,y_train)
```

Out[130]: DecisionTreeClassifier(criterion='entropy', max_depth=7)

In [131]:
```
dt_y_pred=dt.predict(X_test)
```

Decision Tree Evaluation

In [132]:
```
jaccard_score(y_test,dt_y_pred)
```

Out[132]: 0.695744333931192

In [133]:
```
f1_score(y_test,dt_y_pred,average='macro')
```

Out[133]: 0.4134833497173752

3.4 Logistic Regression

Logistic Regression

In [134]:
```
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[134]: 0.6108418122540358

Logistic Regression Evaluation

In [135]:
```
jaccard_score(y_test,LR_y_pred)
```

Out[135]: 0.6955742601140374

In [136]:
```
f1_score(y_test,LR_y_pred,average='macro')
```

Out[136]: 0.4105858170254336

4 Evaluation

ML Model	Jaccard Score	F1 Score
KNN	0.660	0.482
Decision Tree	0.695	0.413
Logistic Regression	0.695	0.410

5 Conclusion

Based on the dataset provided for this capstone from weather, road, and light conditions pointing to certain classes, we can conclude that particular conditions have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2). Thanks all the readers.