Seattle Car Accident Prediction

Venugopal T

September 24th 2020

Problem Statement

There are many accidents happening in and around Seattle region and Seattle Department of Transportation(SDOT)
Has approached to understand the cause of the car accidents and want the best solution which can avoid accidents and casualities

SDOT has gathered various parameters based on the previous car accidents and
want to understand the impact of various factors through Machine Learning and come up with the best model which can predict the factor which is the cause for these accidents and take measure to avoid these

Data Description

- We have used the SDOT data from Feb 2004 till May 2020
- The data consists of 194673 rows and 38 columns
- Some of the attributes which are being considered are SEVERITYCODE, WEATHER, LIGHTCOND, ROADCOND
- Accidents involving parked cars
- There are 2 types of SEVERITY CODES which cause major accidents
- Class1 and Class2
- They are in the ratio of 30:70 respectively
- Some of the Variables are categorical type which are transformed into numeric data type

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

5 rows × 38 columns

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	INTKEY	SEVERITYCODE.1	PERSONCOUNT	PEDCOUNT
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	65070.000000	194673.000000	194673.000000	194673.000000
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	37558.450576	1.298901	2.444427	0.037139
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	51745.990273	0.457778	1.345929	0.198150
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	23807.000000	1.000000	0.000000	0.000000
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	28667.000000	1.000000	2.000000	0.000000
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	29973.000000	1.000000	2.000000	0.000000
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	33973.000000	2.000000	3.000000	0.000000
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	757580.000000	2.000000	81.000000	6.000000

Packages and Functions used for analysis

```
import numpy as np
import pandas as pd
import itertools
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from sklearn import preprocessing, svm, metrics, ensemble, tree
from sklearn.preprocessing import OneHotEncoder, RobustScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

Exploratory Data Analysis

The Data-Collisions CSV provided SDOT was used for analysis of data

We used the read_csv on pandas package
And DataFrame for reading the data

There were 194673 rows and 38 columns

We split the data through the existing train_test_split function to analyze the data

And have split the data into Training : Test = 70:30

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

```python
df = pd.read_csv("https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv")
df.head()
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Columns (33) have mixed type
s. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

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

5 rows × 38 columns

```python
print ('Training set:', X_train.shape,  y_train.shape)
print ('Testing set:', X_test.shape,  y_test.shape)
```

```
Training set: (81463, 3) (81463,)
Testing set: (34913, 3) (34913,)
```

Machine Learning Models used for analysis

- K-Nearest Neighbors
- Decision Tree
- Linear Regression
- Random Forest Classification

- K-Nearest Neighbors

K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
k = 25
```

```
#Train Model & Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh

Kyhat = neigh.predict(X_test)
Kyhat[0:5]
```

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

K-Nearest Neighbor Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, Kyhat)
```

```
11]:  0.5237017729785467
```

```
# F1-Score
f1_score(y_test, Kyhat, average='macro')
```

```
12]:  0.5196155093297656
```

• Decision Tree

Decision Tree

```python
# Building the Decision Tree
from sklearn.tree import DecisionTreeClassifier
colDataTree = DecisionTreeClassifier(criterion="entropy", max_depth = 7)
colDataTree
colDataTree.fit(X_train,y_train)
```

```
48]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=7,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

```python
# Train Model & Predict
dtyhat = colDataTree.predict(X_test)
print (dtyhat [0:5])
print (y_test [0:5])
```

```
[1 2 2 2 2]
[1 1 2 1 1]
```

Decision Tree Evaluation

```python
# Jaccard Similarity Score
jaccard_similarity_score(y_test, dtyhat)
```

```
3]:  0.5626843869045914
```

```python
# F1-Score
f1_score(y_test, dtyhat, average='macro')
```

```
4]:  0.5385207275454998
```

- Linear Regression

Logistic Regression

```
# Building the linear regression Model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
lr = LogisticRegression(C=0.03, solver='liblinear').fit(X_train,y_train)
lr
```

```
50]: LogisticRegression(C=0.03, class_weight=None, dual=False, fit_intercept=True,
                intercept_scaling=1, max_iter=100, multi_class='warn',
                n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                tol=0.0001, verbose=0, warm_start=False)
```

```
# Train Model & Predictor
lryhat = lr.predict(X_test)
print(lryhat)

yhat_prob = lr.predict_proba(X_test)
print(yhat_prob)
```

```
[2 1 2 ... 2 2 1]
[[0.40364293 0.59635707]
 [0.53529771 0.46470229]
 [0.46743605 0.53256395]
 ...
 [0.46293233 0.53706767]
 [0.46743605 0.53256395]
 [0.67878612 0.32121388]]
```

Logistic Regression Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, lryhat)
```

```
5]:  0.523501274596855
```

```
# F1-Score
f1_score(y_test, lryhat, average='macro')
```

```
6]:  0.509857271706865
```

```
# LogLoss
yhat_prob = lr.predict_proba(X_test)
log_loss(y_test, yhat_prob)
```

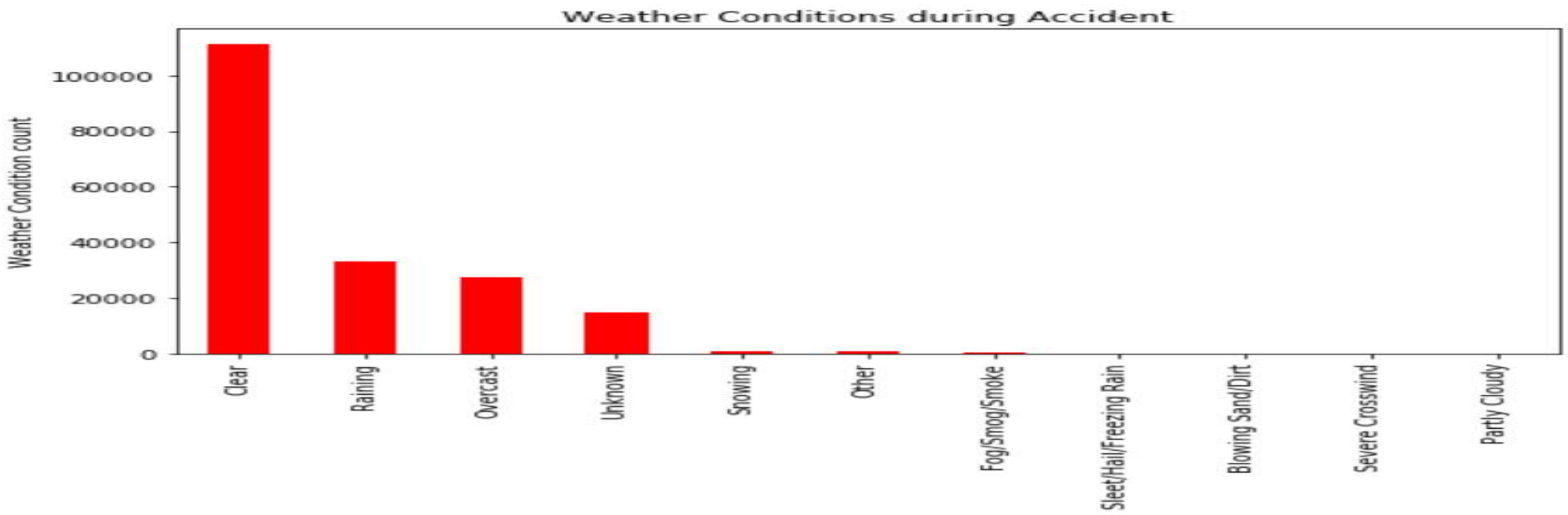
```
7]:  0.6855290309651024
```

- Random Forest Classification

```
Classification of different models Random Forest Classification
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

- Weather Count

	Weather Condition Counts	Weather condition Percentage
Clear	111135	58.617980
Raining	33145	17.482278
Overcast	27714	14.617705
Unknown	15091	7.959724
Snowing	907	0.478396
Other	832	0.438837
Fog/Smog/Smoke	569	0.300118
Sleet/Hail/Freezing Rain	113	0.059602
Blowing Sand/Dirt	56	0.029537
Severe Crosswind	25	0.013186
Partly Cloudy	5	0.002637



CONCLUSION

Model	Jaccard Score	F1-Score	Logloss
KNN	0.5512563468298398	0.5046608485281039	0
Decision Tree	0.5633381070173155	0.5387154848944434	0
Logistic Regression	0.5237338888165604	0.5097244174539972	0.6855086422628527
Random Forest	0.5617497721650827	0.5097244174539972	0

- By looking the above evaluation parameters
  - Jaccard Score is being measured for different models
  - F1 Score shows that Decision Tree has highest prediction
  - Logloss is only measured for Linear Regression

- Through Decision Tree Model we can get to the root cause of the entire data and is mostly dependent on only 3 factors WEATHER, ROAD_COND, LIGHT_COND

- This can be used by Police, Health Authorities and Civil Society to take enough measures so as to avoid Car Accident Severity