Applied Capstone Project
Car Accident Severity

By: Alvin Wijaya

1. Introduction

Many incidents had caused traumas and deaths in many people's life, much of the news might cover incidents such as bombing, terrorist or shark attack. As scary as it seems, those incidents has nothing to compare with the seemingly overlooked accident called car crash. Car crash has consumed many lives in the world especially in the US. Taking cdc.gov statistics into account, 6 millions of it occur every year with 90 people die everyday. Hence it is imperative for us to analyze and make accurate model to predict the severity of car accidents in order to raise awareness and avoid it.

Stakeholders:

- Public Development Authority of Seattle
- Car Drivers

2. Data

The data that will be used in this model building is provided by Seattle Government from the year 2004 to 2020 which can be download here: https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv
This data is about the severity of each car accidents along with the time and conditions under which each accident occurred. The model aims to predict the severity of an accident, considering that, the variable of Severity Code was in the form of 1 (Property Damage Only) and 2 (Physical Injury).

Before going into model building it is important to make sure the data is feedable. Hence, we do some data cleaning including missing value treatment, feature selection and preprocessing.

3. Methodology

The method used in this project is:

A. Data Cleaning and Transformation
   a. Fill missing values
      Find missing values in interested variables, check whether there are any missing values. With value_counts() look into how to fill the data.
   b. Preprocessing
      This project used LabelEncoder to encode categorical data which otherwise won't be able to feed into machine learning model
   c. Resampling

Since it is detected that target variable is unbalanced (almost 2x). It is imperative to balance it so it won't mess with our prediction. Undersampling was conducted.

d. Finding Highly Correlated Features
By writing a little code one can assess whether the chosen independent variable has high correlation among themselves. This will disturb our prediction if not handled properly.

B. Exploratory Analysis
RFECV (Recursive Feature Elimination Cross Validation) was used to determine how many of our feature will be used or eliminated to gain best result. Moreover, we also want to find our which feature contributed the most to our model success.

C. Model Selection and Evaluation
Using various method, Logistic Regression, SVM, KNN, Random Forest to track their RMSE score and also the confusion matrix. See who performs the best.

4. Result and Discussion

A. Data Cleaning and Transformation

a. Missing Values

Our data consist of 38 columns as shown below with their respective categories:

```
In [8]: 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
 14  SEVERITYDESC    194673 non-null  object
 15  COLLISIONTYPE   189769 non-null  object
 16  PERSONCOUNT     194673 non-null  int64
 17  PEDCOUNT        194673 non-null  int64
 18  PEDCYLCOUNT     194673 non-null  int64
 19  VEHCOUNT        194673 non-null  int64
 20  INCDATE         194673 non-null  object
 21  INCDTTM         194673 non-null  object
 22  JUNCTIONTYPE    188344 non-null  object
 23  SDOT_COLCODE    194673 non-null  int64
 24  SDOT_COLDESC    194673 non-null  object
 25  INATTENTIONIND  194673 non-null  float64
 26  UNDERINFL       189789 non-null  object
 27  WEATHER         189592 non-null  object
 28  ROADCOND        189661 non-null  object
 29  LIGHTCOND       189503 non-null  object
 30  PEDROWNOTGRNT   4667 non-null    object
 31  SDOTCOLNUM      114936 non-null  float64
 32  SPEEDING        194673 non-null  float64
 33  ST_COLCODE      194655 non-null  object
 34  ST_COLDESC      189769 non-null  object
 35  SEGLANEKEY      194673 non-null  int64
 36  CROSSWALKKEY    194673 non-null  int64
 37  HITPARKEDCAR    194673 non-null  object
dtypes: float64(6), int64(12), object(20)
memory usage: 56.4+ MB
```

Here is the null values of each columns:

```
In [9]: main_df.isnull().sum()
```

```
Out[9]: SEVERITYCODE           0
        X                   5334
        Y                   5334
        OBJECTID               0
        INCKEY                 0
        COLDETKEY              0
        REPORTNO               0
        STATUS                 0
        ADDRTYPE            1926
        INTKEY            129603
        LOCATION            2677
        EXCEPTRSNCODE     109862
        EXCEPTRSNDESC     189035
        SEVERITYCODE.1         0
        SEVERITYDESC           0
        COLLISIONTYPE       4904
        PERSONCOUNT            0
        PEDCOUNT               0
        PEDCYLCOUNT            0
        VEHCOUNT               0
        INCDATE                0
        INCDTTM                0
        JUNCTIONTYPE        6329
        SDOT_COLCODE           0
        SDOT_COLDESC           0
        INATTENTIONIND         0
        UNDERINFL           4884
        WEATHER             5081
        ROADCOND            5012
        LIGHTCOND           5170
        PEDROWNOTGRNT     190006
        SDOTCOLNUM         79737
        SPEEDING               0
        ST_COLCODE            18
        ST_COLDESC          4904
        SEGLANEKEY             0
        CROSSWALKKEY           0
        HITPARKEDCAR           0
        dtype: int64
```

I am particularly interested in several variables so I choose several of them to use as my model independent variable. It consist of: 'SEVERITYCODE','JUNCTIONTYPE','PEDCOUNT','ROADCOND','LIGHTCOND','SPEEDING','INATTENTIONIND','UNDERINFL','WEATHER','ADDRTYPE'

After selecting the feature I perform a null value replacement as shown below.

Replace Null Values of Selected Feature

```
In [94]: df['UNDERINFL']=df['UNDERINFL'].fillna('0')
         df['SPEEDING']=df['SPEEDING'].fillna('0')
         df['ADDRTYPE']=df['ADDRTYPE'].fillna('Unknown')
         df['WEATHER']=df['WEATHER'].fillna('Unknown')
         df['ROADCOND']=df['ROADCOND'].fillna('Unknown')
         df['LIGHTCOND']=df['LIGHTCOND'].fillna('Unknown')
         df['JUNCTIONTYPE']=df['LIGHTCOND'].fillna('Unknown')
         df['INATTENTIONIND']=df['INATTENTIONIND'].fillna('0')
```

Replace Values for supposedly numerical categories

```
In [95]: df['UNDERINFL'].replace({'0': 0, '1':1, 'Y':1, 'N':0}, inplace=True)
         df['INATTENTIONIND'].replace({'0': 0,'Y':1}, inplace=True)
         df['SPEEDING'].replace({'0': 0,'Y':1}, inplace=True)
```

b. Encode Categorical Columns

As we know, there are two types of data: Categorical and Numerical. Only numerical data can fit into model. Hence, the categorical columns below must be transformed into numerical by using sklearn preprocessing Label Encoder.

Label encode the categorical columns

In [98]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['JUNCTIONTYPE']=label_encoder.fit_transform(df['JUNCTIONTYPE'])
df['WEATHER']=label_encoder.fit_transform(df['WEATHER'])
df['ADDRTYPE']=label_encoder.fit_transform(df['ADDRTYPE'])
df['LIGHTCOND']=label_encoder.fit_transform(df['LIGHTCOND'])
df['ROADCOND']=label_encoder.fit_transform(df['ROADCOND'])
```

In [99]:
```
df.head()
```

Out[99]:

	SEVERITYCODE	JUNCTIONTYPE	PEDCOUNT	ROADCOND	LIGHTCOND	SPEEDING	INATTENTIONIND	UNDERINFL	WEATHER	ADDRTYPE
0	2	5	0	8	5	0	0	0	4	2
1	1	2	0	8	2	0	0	0	6	1
2	1	5	0	0	5	0	0	0	4	1
3	1	5	0	0	5	0	0	0	1	1
4	2	5	0	8	5	0	0	0	6	2

c. Resampling Target Variable

The problem with this dataset is the imbalance found in the target variable (SEVERITYCODE), as shown below:

In [84]:
```
target=df['SEVERITYCODE']
target.value_counts()
```

Out[84]:
```
1    136485
2     58188
Name: SEVERITYCODE, dtype: int64
```

This will create bias in our model so we have to avoid it. One of the technique I used is Undersampling.

:
```
from sklearn.utils import resample
df_severity1=df[df.SEVERITYCODE==1]
df_severity2=df[df.SEVERITYCODE==2]
df_severity1_sample = resample(df_severity1,
                               replace=False,
                               n_samples=58188,
                               random_state=123)
df_balanced=pd.concat([df_severity1_sample,df_severity2])
df_balanced.SEVERITYCODE.value_counts()
```

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

d. Highly correlated features

Highly correlated features in our independent variable will undermine our model prediction. Here is the code that I used to check whether there is collinearity above 0.8.

In [101]:
```
correlated_features = set()
correlation_matrix = df_balanced.drop('SEVERITYCODE', axis=1).corr()

for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.8:
            colname = correlation_matrix.columns[i]
            correlated_features.add(colname)
```

In [102]:
```
correlated_features
```

Out[102]:
```
set()
```

As we can see, there are none of them. It's safe to proceed.

B. Exploratory Analysis

a. RFECV (Recursive Feature Elimination Cross Validation)

To further optimize feature selection, RFECV is used. It is basically a model that can show and eliminate the least important feature.

In [124]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV

X = df_balanced.drop('SEVERITYCODE', axis=1)
target = df_balanced['SEVERITYCODE']

rfc = RandomForestClassifier(random_state=101)
rfecv = RFECV(estimator=rfc, step=1, cv=StratifiedKFold(10), scoring='accuracy')
rfecv.fit(X, target)
```

Out[124]:
```
RFECV(cv=StratifiedKFold(n_splits=10, random_state=None, shuffle=False),
      estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                       class_weight=None, criterion='gini',
                                       max_depth=None, max_features='auto',
                                       max_leaf_nodes=None, max_samples=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=101,
                                       verbose=0, warm_start=False),
      min_features_to_select=1, n_jobs=None, scoring='accuracy', step=1,
      verbose=0)
```

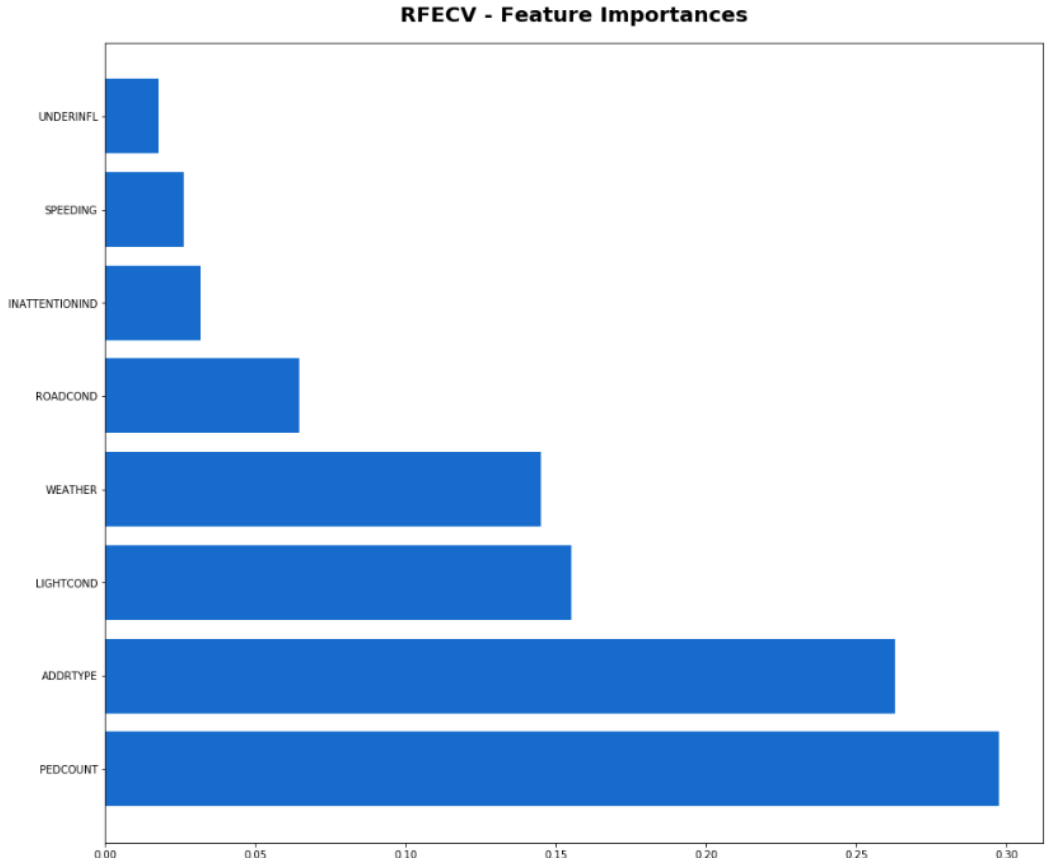
In [125]:
```
print('Optimal number of features: {}'.format(rfecv.n_features_))
```

```
Optimal number of features: 9
```

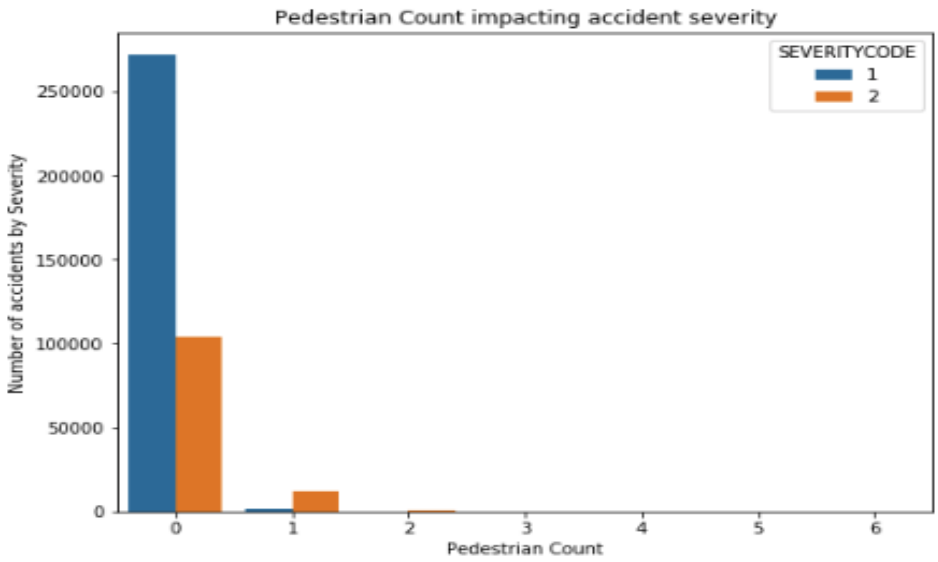
After running the code with Random Forest Classifier as estimator, it is found that the best combination is 9 features (the same with the beginning). Moreover,

we can plot each feature importance as well.

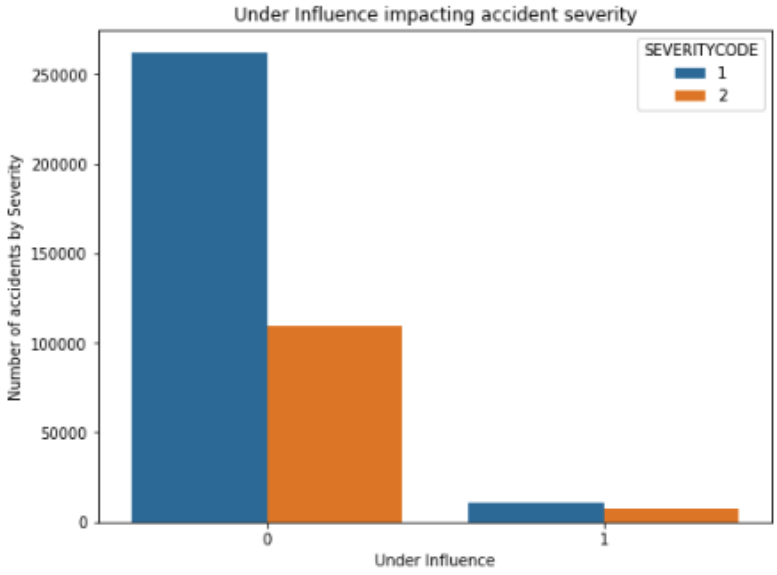
In [119]:
```
dset = pd.DataFrame()
dset['attr'] = X.columns
dset['importance'] = rfecv.estimator_.feature_importances_

dset = dset.sort_values(by='importance', ascending=False)


plt.figure(figsize=(16, 14))
plt.barh(y=dset['attr'], width=dset['importance'], color='#1976D2')
plt.title('RFECV - Feature Importances', fontsize=20, fontweight='bold', pad=20)
plt.xlabel('Importance', fontsize=14, labelpad=20)
plt.show()
```



From Random Forest feature importance we can conclude that the Number of Pedestrian, Address Type, Light Condition account for much of the contribution, with the least comes from Under Influence. To demonstrate the findings, I will look into the most contributive and the least, PEDCOUNT and UNDERINFL respectively.



In the pedestrian count over severity code graph, we can clearly saw that as the pedestrian count adds up, the severity is also more leaning towards 2. Hence it is a high contribution feature.



In the Under Influence over Severity Code graph, driver under no influence score higher than under influence. Moreover, for 0 and 1 Under Influence, the percentage difference between number of accidents by severity is not contrast. As a result the algorithm will have a hard time classifying which to which. Hence, it has a low contribution value.

C. Model Selection and Evaluation

Since this is a classification problem, this report use several model to evaluate which one has the best performance. It consist of SVM Classifier, Gradient Booster

Classifier, Random Forest, KNN and the classic Logistic Regression. It will be evaluate mainly based on RMSE and Confusion Matrix.

a. Model Selection
Here is the result of the model when fed with the data. Before running each model, Stratified K-Fold with Cross Validation of 10 was run.

	Model	Performance
0	GradientBooster	0.729598
1	RandomForest	0.728077
2	KNN	0.665315
3	DecisionTree	0.727815
4	Logistic	0.726506

From the result it is evident that Gradient Booster has a slight advantage over the other model. Hence, we will choose Gradient Booster Classifier.

b. Model Evaluation
Other than the performance using RMSE, Precision, Recall and F1 Score are also calculated. KNN is left out because it has the lowest score.

	Model	Precision	Recall	F1
0	GradientBooster	0.723819	0.993333	0.837425
1	RandomForest	0.724735	0.987039	0.835790
2	DecisionTree	0.724447	0.987310	0.835696
3	Logistic	0.724381	0.984496	0.834642

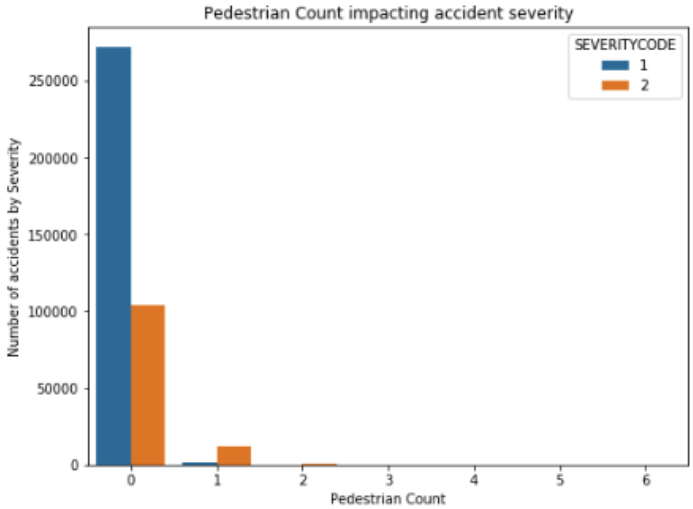
From this data we can conclude that GradientBooster performed better than any other on all aspects.

5. Recommendations

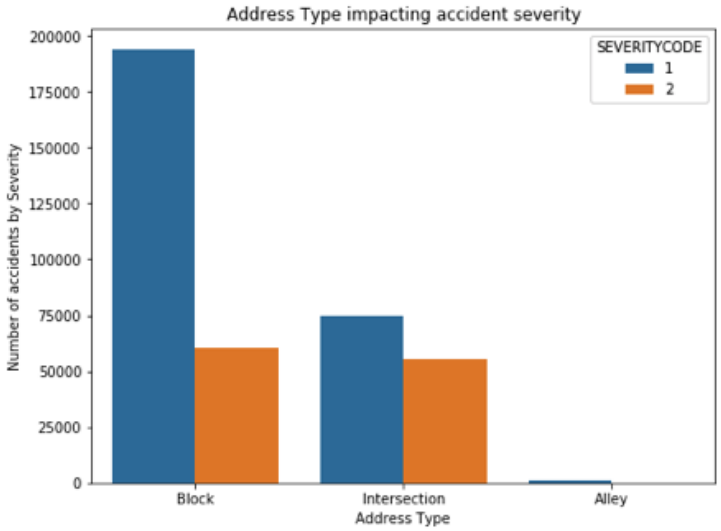
From Exploratory Data Analysis RFECV graph, we can see that three most important factor in influencing car accident severity is Pedestrian Count, Address Type, Light Condition.

- Pedestrian Count
  Our data shows that as number of pedestrian increase, so does the severity. This may be a sign for the government to increase awareness in crowded public spaces, limit vehicle access during the peak hour of the day or even

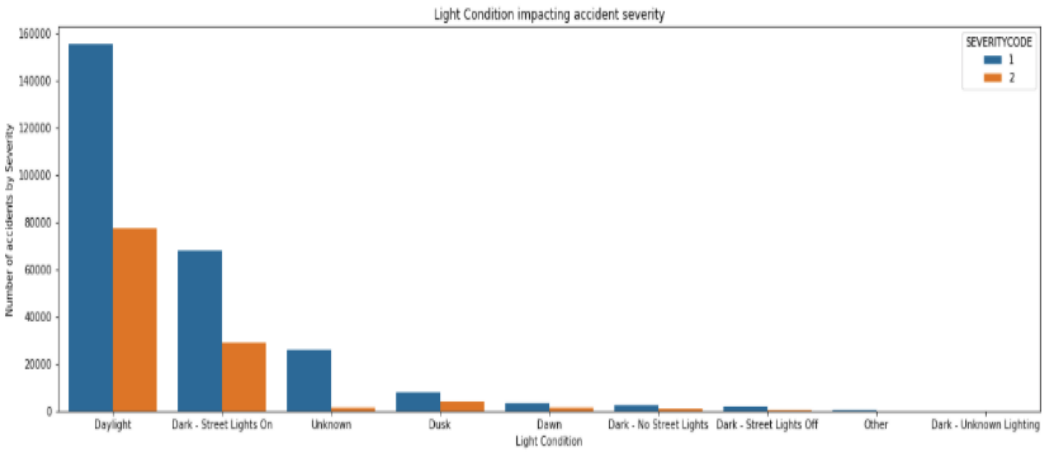
assign a traffic officer to conduct the heavy traffic and make sure everything stays in order.



- Address Type
  Our data shows many of the accident occur in Block and Intersection with the former as the highest. To avoid this, government may try to reanalyze the traffic structure of both block and intersection. Perhaps, they can set more speedometer, CCTV, and more clear traffic sign.



- Light Condition
  Light condition is the third most contributing factors in our model. However, contrary to our common sense, highest accident number for both 1 & 2 are recorded in daylight condition followed by dark – street lights on. After considering other factors this might make sense because most people are out during daylight (more pedestrian more severity). While daylight seems pretty unimprovable parameter. We can still work on the dark – street lights on by setting more lights in high activities road and places during the night such as clubbing area, night market, etc.



6. Conclusion

In conclusion, we have explored the Seattle government traffic data and identified most contributing factor: Pedestrian Count, Address Type and Light Condition. Also, we make recommendations for it. Data cleaning, feature selection, model selection and evaluation is also performed and resulted in Gradient Booster Classifier as our winner.