Coursera Capstone Project:- Predicting Accident Severity in Seattle

September 17, 2020

1. Business Problem

In USA more than **six million** car accidents occur each year and according to **NHTSA(National Highway Traffic Safety Administration)**, about **6%** of all motor vehicle accidents in the United States result in atleast one death. According to **Wikipedia** 6277 pedestrians were killed in traffic in 2018 in the US.

What if we could forecast the severity of an accident before hand? What if we predict the chances/possibility that an accident can occur depending upon road condition, weather condition, traffic at a certain place,etc? What if we can say that today there might be X percentage of chances that accident may occur.

In this project I tried to predict severity of an accident in Seattle depending upon some conditions using machine learning models. Although this project predicts severity of accident in Seattle, we can also implement this method for any city if we have data of accidents occurring in that city since 5 years. This project can be helpful to Seattle government

2. Data

In this project I have used dataset shared on Coursera. This dataset has a total of 38 columns in which 37 columns are features/attributes and one column is dependent variable or the value to be predicted. Each row/record in this dataset represents accidents happened in Seattle from 2014 to 2020. Dependent/Target variable in this dataset is **SEVERITYCODE**. It has 2 unique values which corresponds to different levels. 1 for property damage and 2 for injury. Brief explanation of each feature can be found in below images

Attribute Information

Attribute	Data type, length	Description
OBJECTID	ObjectID	ESRI unique identifier
SHAPE	Geometry	ESRI geometry field
INCKEY	Long	A unique key for the incident
COLDETKEY	Long	Secondary key for the incident
ADDRTYPE	Text, 12	Collision address type: • **Alley** • **Block** • **Intersection**
INTKEY	Double	Key that corresponds to the intersection associated with a collision

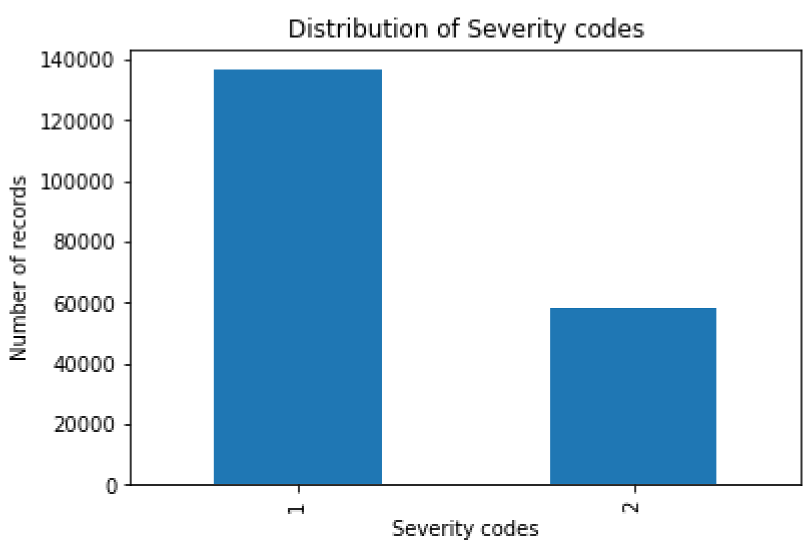
Attribute	Data type, length	Description
LOCATION	Text, 255	Description of the general location of the collision
EXCEPTRSNCODE	Text, 10	
EXCEPTRSNDESC	Text, 300	
SEVERITYCODE	Text, 100	A code that corresponds to the severity of the collision: • **3**—fatality • **2b**—serious injury • **2**—injury • **1**—prop damage • **0**—unknown
SEVERITYDESC	Text	A detailed description of the severity of the collision
COLLISIONTYPE	Text, 300	Collision type
PERSONCOUNT	Double	The total number of people involved in the collision
PEDCOUNT	Double	The number of pedestrians involved in the collision. This is entered by the state.
PEDCYLCOUNT	Double	The number of bicycles involved in the collision. This is entered by the state.
VEHCOUNT	Double	The number of vehicles involved in the collision. This is entered by the state.
INJURIES	Double	The number of total injuries in the collision. This is entered by the state.
SERIOUSINJURIES	Double	The number of serious injuries in the collision. This is entered by the state.
FATALITIES	Double	The number of fatalities in the collision. This is entered by the state.
INCDATE	Date	The date of the incident.
INCDTTM	Text, 30	The date and time of the incident.
JUNCTIONTYPE	Text, 300	Category of junction at which collision took place
SDOT_COLCODE	Text, 10	A code given to the collision by SDOT.
SDOT_COLDESC	Text, 300	A description of the collision corresponding to the collision code.
INATTENTIONIND	Text, 1	Whether or not collision was due to inattention. (Y/N)
UNDERINFL	Text, 10	Whether or not a driver involved was under the influence of drugs or alcohol.

Attribute	Data type, length	Description
WEATHER	Text, 300	A description of the weather conditions during the time of the collision.
ROADCOND	Text, 300	The condition of the road during the collision.
LIGHTCOND	Text, 300	The light conditions during the collision.
PEDROWNOTGRNT	Text, 1	Whether or not the pedestrian right of way was not granted. (Y/N)
SDOTCOLNUM	Text, 10	A number given to the collision by SDOT.
SPEEDING	Text, 1	Whether or not speeding was a factor in the collision. (Y/N)
ST_COLCODE	Text, 10	A code provided by the state that describes the collision. For more information about these codes, please see the State Collision Code Dictionary.
ST_COLDESC	Text, 300	A description that corresponds to the state's coding designation.
SEGLANEKEY	Long	A key for the lane segment in which the collision occurred.
CROSSWALKKEY	Long	A key for the crosswalk at which the collision occurred.
HITPARKEDCAR	Text, 1	Whether or not the collision involved hitting a parked car. (Y/N)

3. Methodology

(i) Data Exploration & Preparation

There are 37 features available in our dataset. I haven't used all features to predict. I have picked up only those features which has a strong correlation with target variable. Most of the attributes has missing values. Also our data is not balanced have a look at below graph which shows how unbalanced our dataset is.

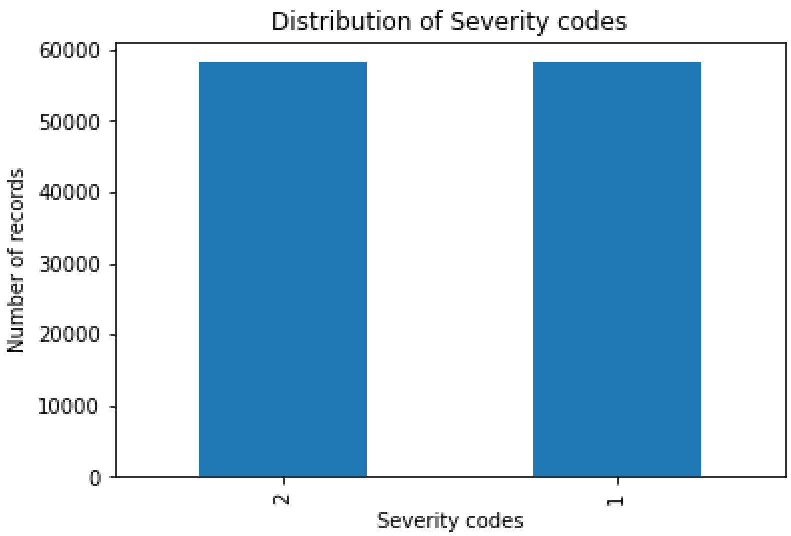


When we feed our model with this unbalanced data our model will bias towards the class which has majority of records. So dataset should always be balanced

As we can see that records with severity as 1 is much larger than records with severity as 2. There are 2 ways to balance an unbalanced dataset:-

- Undersamplng
- Oversampling

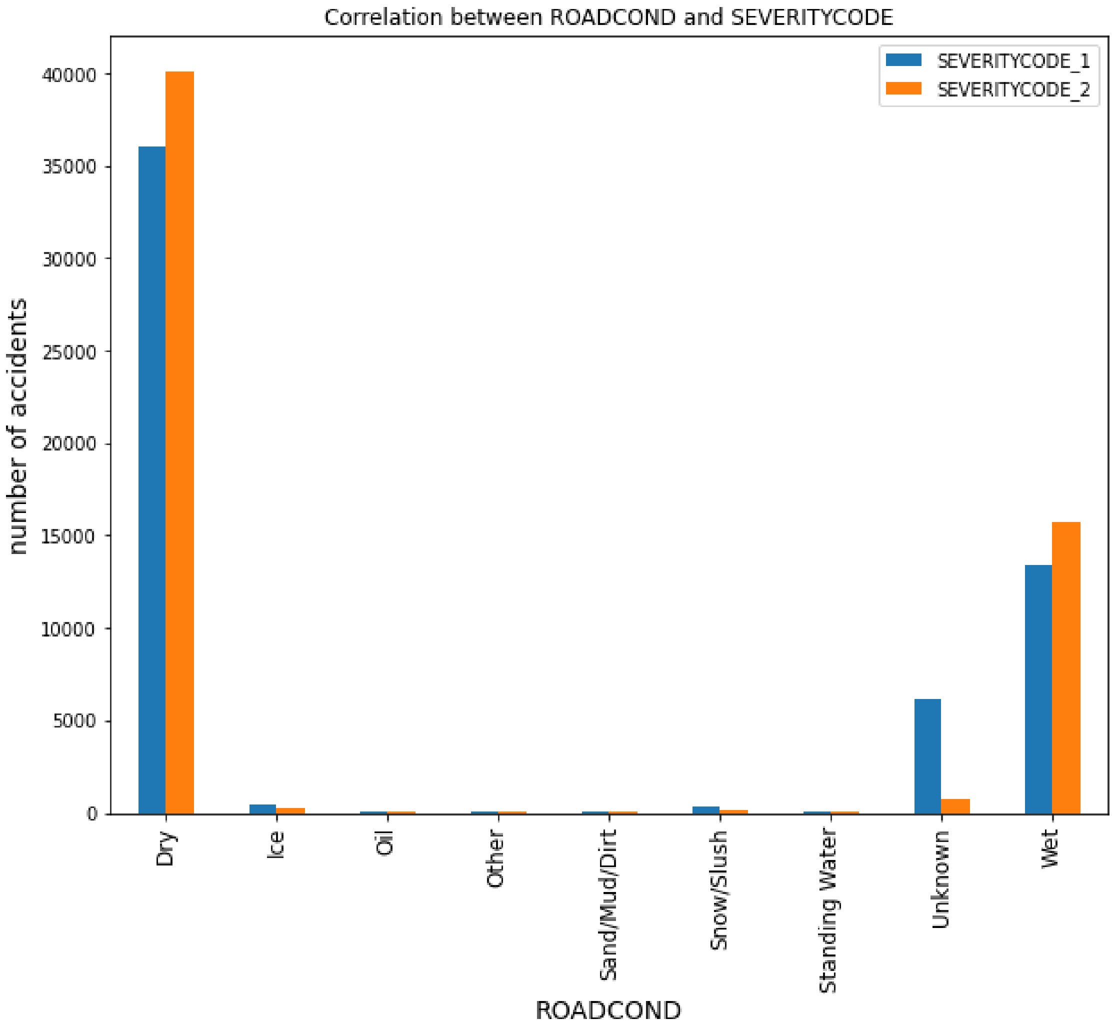
Undersampling is the process where you randomly delete some of the observations from the majority class in order to match the numbers with the minority class.
Oversampling is the process of duplicating examples from the minority class in training dataset and can result in overfiting for some models. In this case I have used random undersampling(Random undersampling means randomly deleting examples in majority class). Have a look at below graph which shows distribution of SEVERITYCODE after applying Random undersampling method

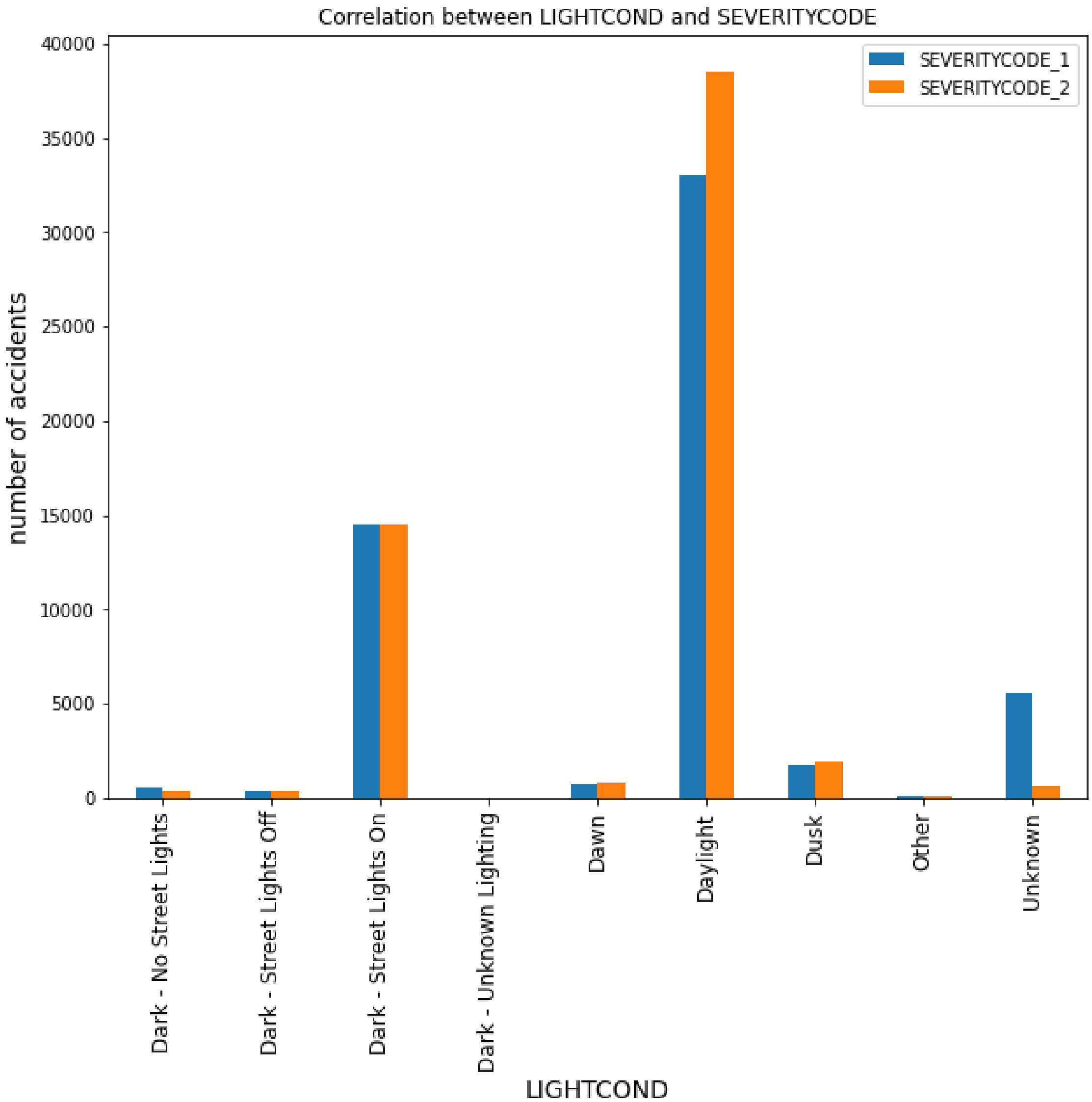


Now our data is balanced we can move further to explore correlation between independent variable and target variable. Attributes which can impact in predicting **SEVERITYCODE** are as follows:-

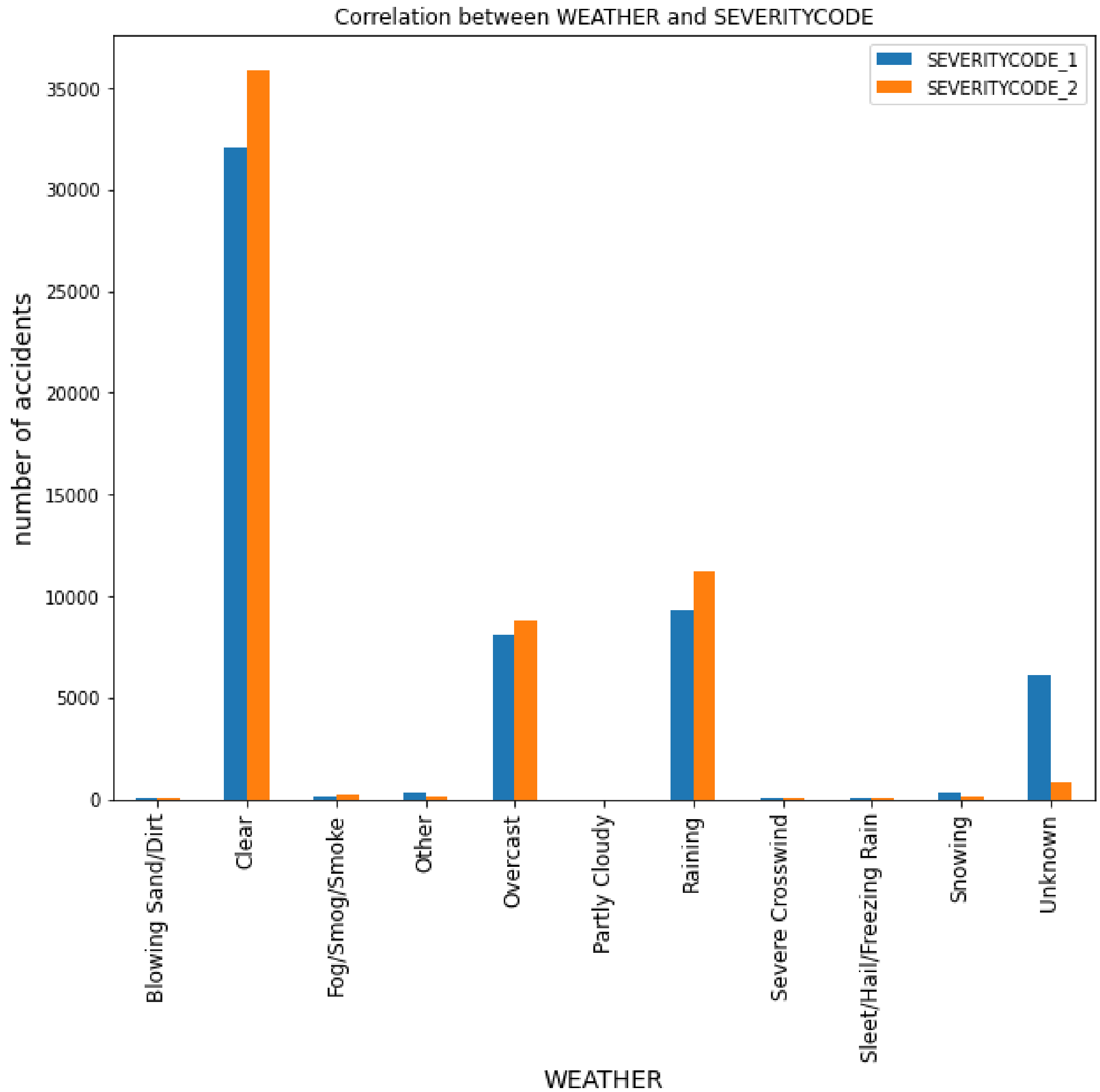
- ROADCOND
- LIGHTCOND
- WEATHER
- UNDERINFL
- ADDRTYPE

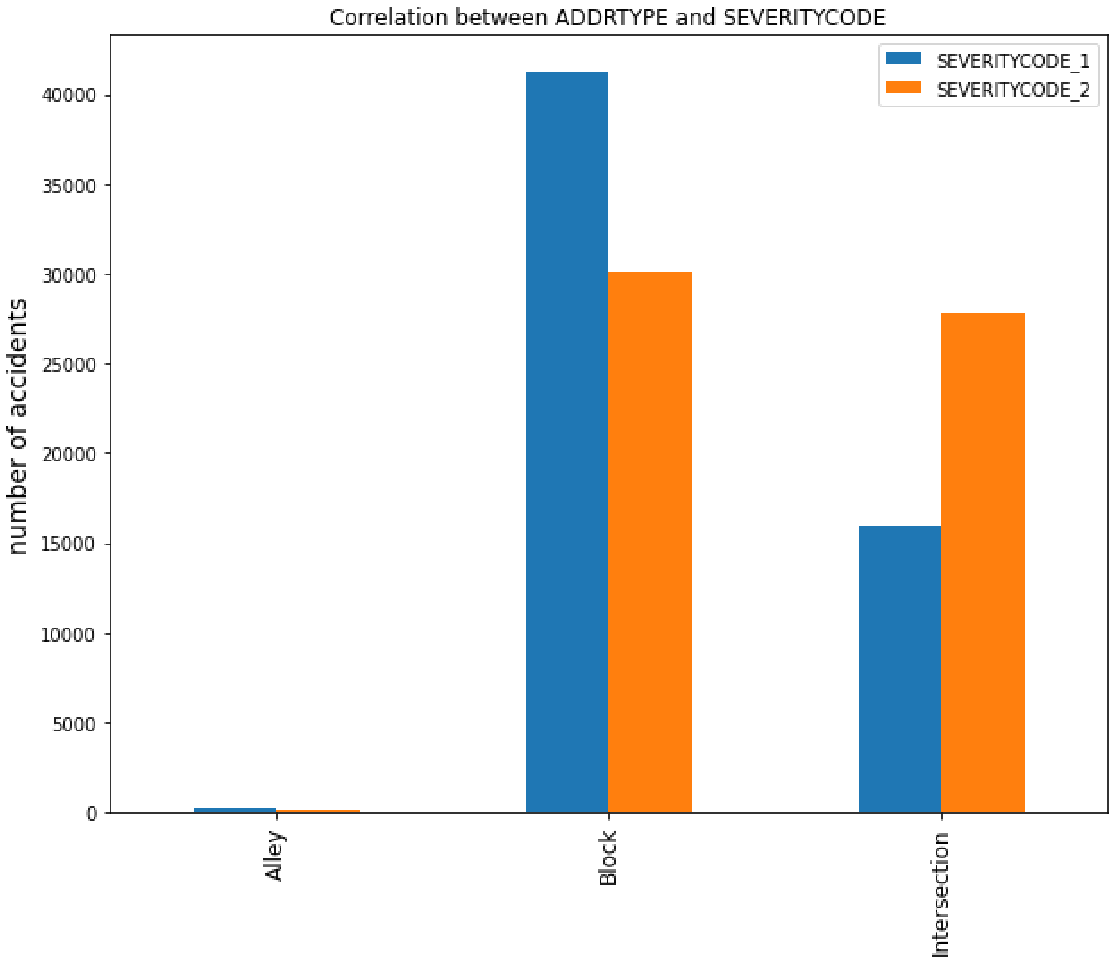
I have used **Pearson's Chi-Squared Test** to test the correlation between target and independent variable. Have a look at graphs which shows the relationship between them

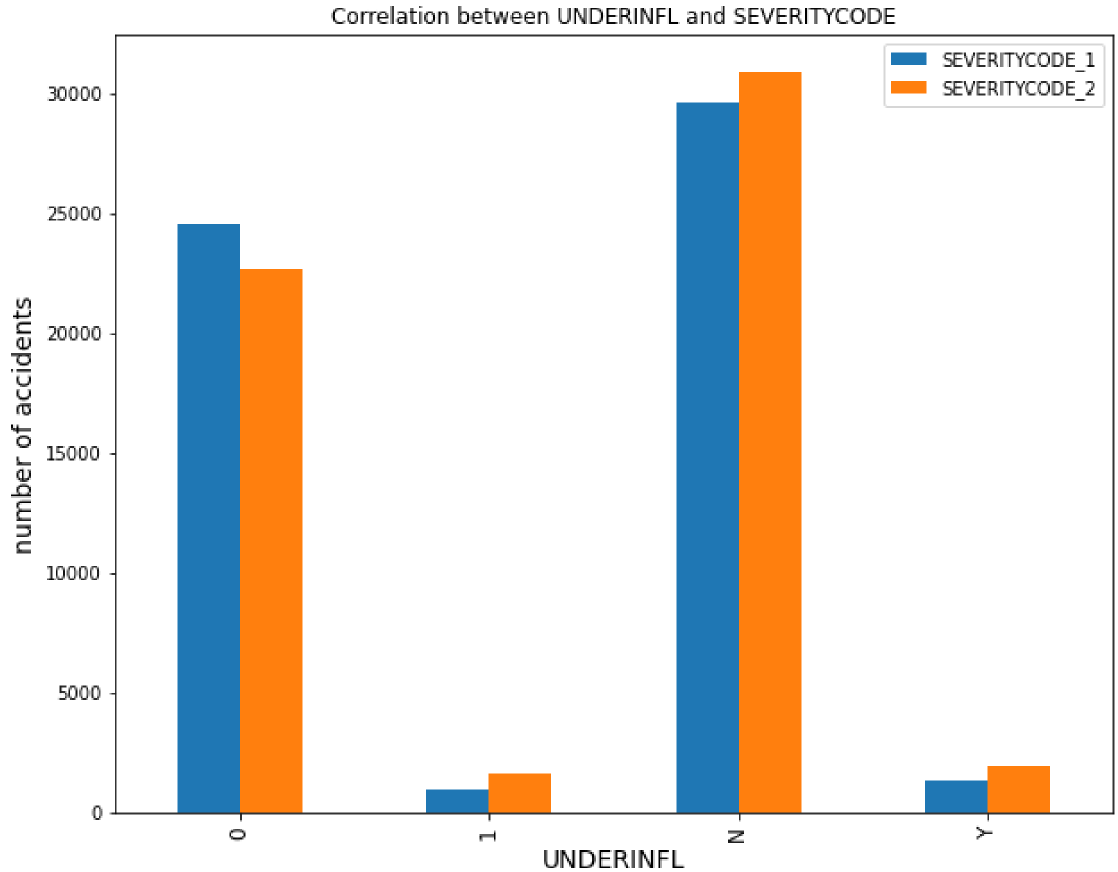

Correlation between ROADCOND and SEVERITYCODE
SEVERITYCODE_1
SEVERITYCODE_2
number of accidents
0
5000
10000
15000
20000
25000
30000
35000
40000
Dry
Ice
Oil
Other
Sand/Mud/Dirt
Snow/Slush
Standing Water
Unknown
Wet
ROADCOND



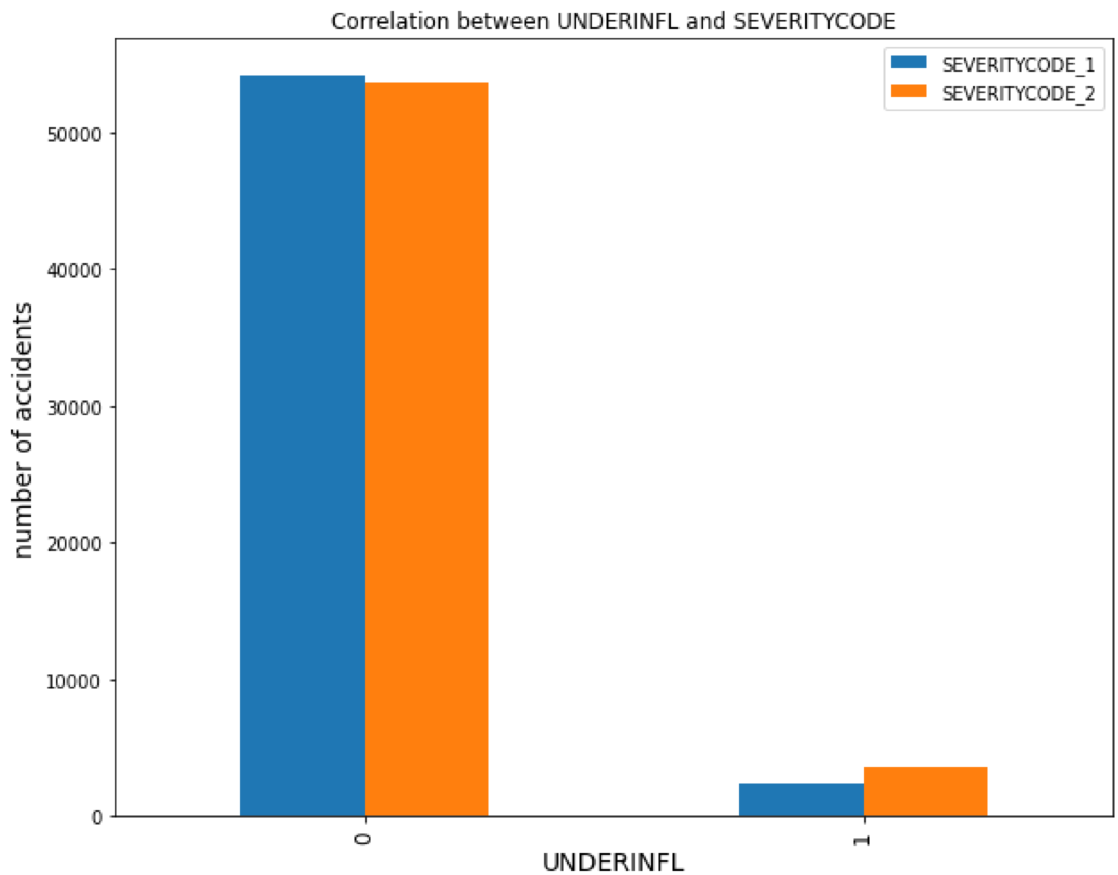
Correlation between LIGHTCOND and SEVERITYCODE
SEVERITYCODE_1
SEVERITYCODE_2
number of accidents
40000
35000
30000
25000
20000
15000
10000
5000
0
Dark - No Street Lights
Dark - Street Lights Off
Dark - Street Lights On
Dark - Unknown Lighting
Dawn
Daylight
Dusk
Other
Unknown
LIGHTCOND


Correlation between WEATHER and SEVERITYCODE
SEVERITYCODE_1
SEVERITYCODE_2
number of accidents
0
5000
10000
15000
20000
25000
30000
35000
Blowing Sand/Dirt
Clear
Fog/Smog/Smoke
Other
Overcast
Partly Cloudy
Raining
Severe Crosswind
Sleet/Hail/Freezing Rain
Snowing
Unknown
WEATHER


Correlation between ADDRTYPE and SEVERITYCODE
SEVERITYCODE_1
SEVERITYCODE_2
number of accidents
0
5000
10000
15000
20000
25000
30000
35000
40000
Alley
Block
Intersection
ADDRTYPE



As we can see here that **UNDERINFL** attribute has 4 values (N,0,Y,1) in which N is equal to 0 and y is equal to 1. I have replaced 'N' with '0' ad 'Y' with '1'.



Our dataset contains alot NAN(Not A Number) values. Sklearn models cannot handle NAN values. I have dropped records/rows which have NAN values

In [10]:
```
balanced_df = balanced_df.dropna()
balanced_df.isnull().values.any()
```

Out[10]:
```
False
```

I have seperated our dataset into two parts, one contains indepenpent variables and other contains only target variable. Also Sklearn models always expect labels and target values to be a numpy array.

In [11]:
```
X = balanced_df[['ADDRTYPE', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND']].values
X[0:5]
```

Out[11]:
```
array([['Block', '0', 'Clear', 'Dry', 'Unknown'],
       ['Intersection', '0', 'Raining', 'Wet', 'Dark - Street Lights On'],
       ['Intersection', '0', 'Raining', 'Wet', 'Daylight'],
       ['Block', '0', 'Raining', 'Wet', 'Dark - Street Lights On'],
       ['Block', '0', 'Clear', 'Dry', 'Daylight']], dtype=object)
```

In [12]:
```
y = balanced_df["SEVERITYCODE"].values
y[:5]
```

Out[12]:
```
array([1, 1, 1, 1, 1], dtype=int64)
```

As we can see that there are many categorical variables in our dataset. Sklearn models do not handle categorical variables. We can convert categorical features into numerical values using `sklearn.preprocessing.LabelEncoder`. I have used below code to convert into numerical values

In [13]:
```
from sklearn import preprocessing
le_addrtype = preprocessing.LabelEncoder()
le_addrtype.fit(balanced_df["ADDRTYPE"].unique().tolist())
X[:,0] = le_addrtype.transform(X[:,0])


le_weather = preprocessing.LabelEncoder()
le_weather.fit(balanced_df["WEATHER"].unique().tolist())
X[:,2] = le_weather.transform(X[:,2])


le_roadcond = preprocessing.LabelEncoder()
le_roadcond.fit(balanced_df["ROADCOND"].unique().tolist())
X[:,3] = le_roadcond.transform(X[:,3])



le_lightcond = preprocessing.LabelEncoder()
le_lightcond.fit(balanced_df["LIGHTCOND"].unique().tolist())
X[:,4] = le_lightcond.transform(X[:,4])

X[0:5]
```

Out[13]:
```
array([[1, '0', 1, 0, 8],
       [2, '0', 6, 8, 2],
       [2, '0', 6, 8, 5],
       [1, '0', 6, 8, 2],
       [1, '0', 1, 0, 5]], dtype=object)
```

1.4.2 (ii) Data Modelling

As we can see that our data is well prepared we can now use **Scikit-learn** algorithms to predict the severity.Here I have used 4 classification models:-

- K-Nearest Neighbors
- Decision Trees
- Logistic Regression
- Support Vector Machine(SVM)

I have reserved a part of dataset to test each model for picking up the best. I have further divided training set in to training and cross validation set which can be used to select best hyper parameter for each model.

```
In [14]: from sklearn.model_selection import train_test_split
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import f1_score
```

```
In [15]: X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
```

```
In [16]: X_train, X_cv, y_train, y_cv = train_test_split( X_train, y_train, test_size=0.2, random_state=3)
```

```
In [17]: print("training size:- ",X_train.shape)
         print("cross validation size:- ",X_cv.shape)
         print("test size:- ",X_test.shape)

         training size:-  (72051, 5)
         cross validation size:-  (18013, 5)
         test size:-  (22517, 5)
```

I have used training and cross validation set for each model to find the best hyper parameter for each model and then test set to detect the best classifier among others

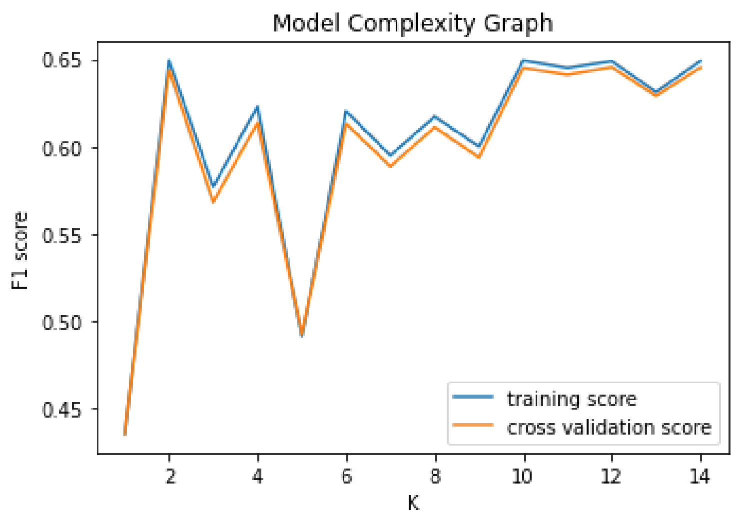
K-Nearest Neighbor

To calculate best K for KNN classifier I have iterated over all possible k values until `k=15`, computed score for each k and then chosen best.

```
In [21]: ks=15
         training_f1_scores = np.zeros(ks-1)
         cv_f1_scores = np.zeros(ks-1)
```

```
In [22]: for n in range(1,ks):
             neigh = KNeighborsClassifier(n_neighbors = n)
             neigh.fit(X_train,y_train)
             y_train_hat = neigh.predict(X_train)
             y_cv_hat = neigh.predict(X_cv)
             training_f1_scores[n-1] = f1_score(y_train,y_train_hat)
             cv_f1_scores[n-1] = f1_score(y_cv,y_cv_hat)
```

In [29]:
```
plt.plot(range(1,ks),training_f1_scores,label="training score")
plt.plot(range(1,ks),cv_f1_scores,label="cross validation score")
plt.xlabel("K")
plt.ylabel("F1 score")
plt.legend()
plt.title("Model Complexity Graph")
plt.show()
```



At k = 11 our model performs best

Decision Tree

In [30]:
```
from sklearn import tree
```

In [31]:
```
decision_tree_clf = tree.DecisionTreeClassifier(criterion="entropy")
decision_tree_clf.fit(X_train,y_train)
y_cv_hat = decision_tree_clf.predict(X_cv)
y_cv_hat[:5]
```

Out[31]:
```
array([2, 1, 2, 1, 1], dtype=int64)
```

In [32]:
```
print("f1 score:- ",f1_score(y_cv,y_cv_hat))
```

```
f1 score:-  0.632002476013618
```

Support Vector Machine

In [18]:
```
from sklearn import svm
```

In [19]:
```
svm_clf_rbf = svm.SVC(kernel='rbf')
svm_clf_linear = svm.SVC(kernel='linear')
svm_clf_poly = svm.SVC(kernel='poly')
svm_clf_sigmoid = svm.SVC(kernel='sigmoid')
```

In [20]:
```
svm_clf_rbf.fit(X_train, y_train)
svm_clf_linear.fit(X_train, y_train)
svm_clf_poly.fit(X_train, y_train)
svm_clf_sigmoid.fit(X_train, y_train)


y_cv_hat_rbf = svm_clf_rbf.predict(X_cv)
y_cv_hat_linear = svm_clf_linear.predict(X_cv)
y_cv_hat_poly = svm_clf_poly.predict(X_cv)
y_cv_hat_sigmoid = svm_clf_sigmoid.predict(X_cv)
```

In [21]:
```
print("f1 score of svm with kernel as rbf:- ",f1_score(y_cv,y_cv_hat_rbf))
print("f1 score of svm with kernel as linear:- ",f1_score(y_cv,y_cv_hat_linear))
print("f1 score of svm with kernel as poly:- ",f1_score(y_cv,y_cv_hat_poly))
print("f1 score of svm with kernel as sigmoid:- ",f1_score(y_cv,y_cv_hat_sigmoid))
```

```
f1 score of svm with kernel as rbf:-  0.6374668570263105
f1 score of svm with kernel as linear:-  0.6318351284175642
f1 score of svm with kernel as poly:-  0.6381197357139254
f1 score of svm with kernel as sigmoid:-  0.5065723548482168
```

Here SVM classifier with kernel as Polynomial gives best result

Logistic Regression

In [22]:
```
from sklearn.linear_model import LogisticRegression
```

In [23]:
```
log_reg = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
```

In [24]:
```
y_cv_hat = log_reg.predict(X_cv)
print("f1 score:- ",f1_score(y_cv,y_cv_hat))
```

```
f1 score:-  0.6306074043518614
```

4. Result

In this phase I tried to find out which model best fits our data using testing set

```
In [26]: from sklearn.metrics import jaccard_score
         from sklearn.metrics import log_loss
```

```
In [28]: neigh = KNeighborsClassifier(n_neighbors = 11)
         neigh.fit(X_train,y_train)
         y_test_hat = neigh.predict(X_test)
         print("f1 score for KNN Classifier:- ", f1_score(y_test,y_test_hat))
         print("jacard score for KNN Classifier:- ",jaccard_score(y_test,y_test_hat))

         f1 score for KNN Classifier:-  0.6416072125794201
         jacard score for KNN Classifier:-  0.4723281944081526
```

```
In [34]: y_test_hat = decision_tree_clf.predict(X_test)
         print("f1 score for Decision Tree Classifier:- ", f1_score(y_test,y_test_hat))
         print("jacard score for Decision Tree Classifier:- ",jaccard_score(y_test,y_test_hat))

         f1 score for Decision Tree Classifier:-  0.6372875776653087
         jacard score for Decision Tree Classifier:-  0.4676610906455704
```

```
In [35]: y_test_hat = svm_clf_poly.predict(X_test)
         print("f1 score for SVM Classifier:- ", f1_score(y_test,y_test_hat))
         print("jacard score SVM Classifier:- ",jaccard_score(y_test,y_test_hat))

         f1 score for SVM Classifier:-  0.6422535211267605
         jacard score SVM Classifier:-  0.4730290456431535
```

```
In [37]: y_test_hat = log_reg.predict(X_test)
         print("f1 score for Logistic Regression:- ", f1_score(y_test,y_test_hat))
         print("jacard score Logistic Regression:- ",jaccard_score(y_test,y_test_hat))

         from sklearn.metrics import log_loss
         print("log loss:- ",log_loss(y_test,y_test_hat))

         f1 score for Logistic Regression:-  0.637160133273004
         jacard score Logistic Regression:-  0.4675238440178679
         log loss:-  16.96376830391694
```

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.472	0.641	NA
Decision Tree	0.467	0.637	NA
SVM	0.473	0.642	NA
LogisticRegression	0.467	0.637	16.96

As we can see that SVM classifier with kernel as polynomial gives highest results among other algorithms. Overall only 63.9 percent(average) of variance is predicted by these models. This is actually low score. It can be due to ignoring other features such as **INATTENTIONIND**, **INCDTTM**, **INCDATE** and **SPEEDING**. Features such as **INATTENTIONIND** and **SPEEDING** was ignored because it has many NAN values nearly 75 percent of it are NAN values and there is no alternative value which can be replaced with NAN values. Other features such as **INCDTTM** and **INCDATE** are not included because of simplicity. There can be a relationship between **SEVERITY** and **INCDTTM**. But inorder to use this feature we have to process and categorize these in to 5 different categories(Morning, Afternoon, Evening , Night and Midnight).**INCDATE** can be categorized into 4 seasons(Summer, Fall, Winter, Spring).

Another reason could be that our dataset has only 2 labels. If we could have more labels then our models would have learnt more.

5. Conclusion

As stated earlier in introduction part; can we predict an accident before hand, given some features? Answer is yes, we can. If more data is given and we put little more effort on data wrangling, we can get good results and we can use these models to predict the possibility of an accident on a given day. This could very helpful to government and also some private sectors like Google maps.