Accident severity prediction using the classification algorithm for Seattle city

Coursera Capstone Project

Business understanding

- On daily basis, hundreds of thousands of people travel from A to B using road infrastructure
- One of the less fortunate side or aspect of road transportation is road accidents
- Road accidents not only cause injuries and deaths but also results in loss or damage of properties worth millions of Dollars
- Road accidents are the result of many factors
- In this project, we try to predict the severity of the accident given the road condition, weather condition, type of accident, and many more influencing factors. To effectively predict the severity, we will be using the machine learning algorithms
- The stack holders that can be benefited from an effective ML model which can predict the accident severity are:

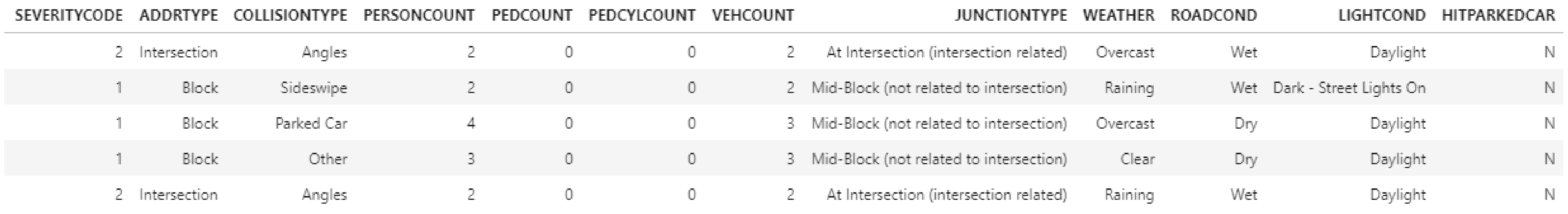
    1. People who are commuters and would like to plan their travel
    2. The government regulatory bodies who, can take precautionary steps, to avoid or to reduce the fatality of an accident

Data Understanding

- The data set used for the project is procured from the Seattle Department of Transport and consists of the vehicle accident information
- The data set is in .csv format with 38 columns and a total of 194673 entries or rows
- The label for the selected .csv data set is the column ‘SEVERITYCODE’
- Each row in the data set has given a severity code which is either ‘1’ or ‘2’ meaning ‘property damage only collision’ and ‘injury collision’, respectively
- By examining the number of 1s and 2s in the label of the given data set, we see that the data is having an imbalance i.e. there are more rows with 1s compared to 2s
- The data must be cleaned in such a way that it avoids the irrelevant attributes and increase the accuracy of the machine learning model without any bias

Data Cleaning

- Before feeding the data to a machine learning model, it should be cleaned or wrangled to avoid problems and false predictions
- The data cleaning or wrangling has been carried out in the following steps:
    1. Removing columns with higher number of missing values
    2. Removing irrelevant columns
    3. Removing columns with repeating and inconsistent data
- After data cleaning we are left with following columns:



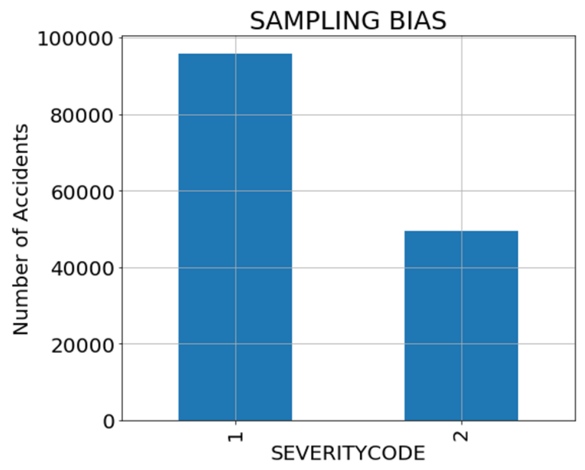
SEVERITYCODE	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	JUNCTIONTYPE	WEATHER	ROADCOND	LIGHTCOND	HITPARKEDCAR
2	Intersection	Angles	2	0	0	2	At Intersection (intersection related)	Overcast	Wet	Daylight	N
1	Block	Sideswipe	2	0	0	2	Mid-Block (not related to intersection)	Raining	Wet	Dark - Street Lights On	N
1	Block	Parked Car	4	0	0	3	Mid-Block (not related to intersection)	Overcast	Dry	Daylight	N
1	Block	Other	3	0	0	3	Mid-Block (not related to intersection)	Clear	Dry	Daylight	N
2	Intersection	Angles	2	0	0	2	At Intersection (intersection related)	Raining	Wet	Daylight	N

Data Formatting

- After the data is cleaned, the categorical values and numerical values are separated into two different variables to convert only the categorical attributes. After that, the unbalance in the data set must be corrected

- One Hot Encoding: it is a method in which the categorical values are automatically converted into numeric values

```
df_dummies = pd.get_dummies(df[["ADDRTYPE","COLLISIONTYPE","JUNCTIONTYPE","WEATHER","ROADCOND","LIGHTCOND","HITPARKEDCAR"]])
```



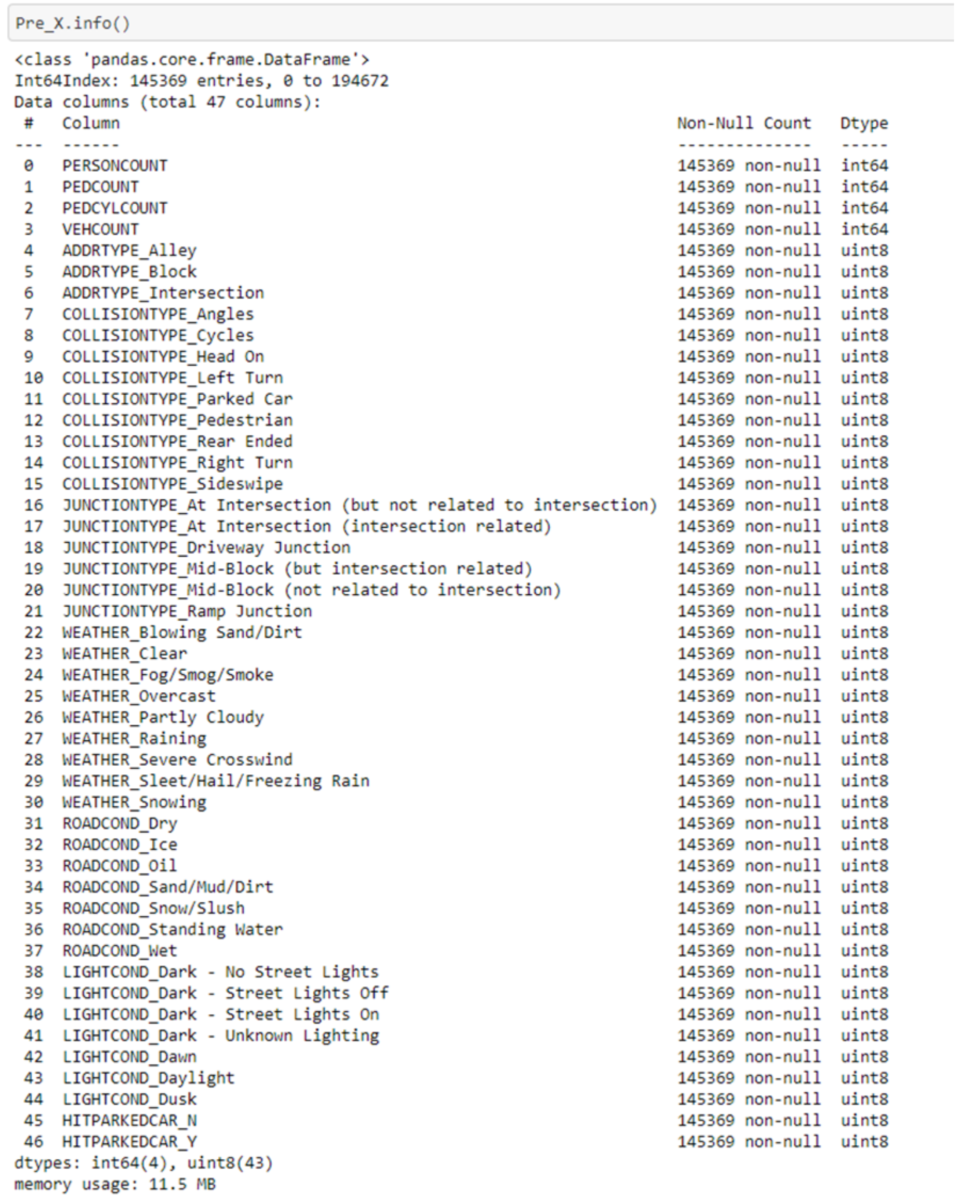
```
df['SEVERITYCODE'].value_counts()

1    95921
2    49448
Name: SEVERITYCODE, dtype: int64
```

- As we discussed, our data set has an unbalance in the labelled attribute 'SEVERITYCODE', which if present, can result in a bias while predicting the unknown values. The balance in the data is achieved by making the number 1s the same as 2s

- The balance is achieved by using the NearMiss package of imblearn library

Number of columns after data cleaning

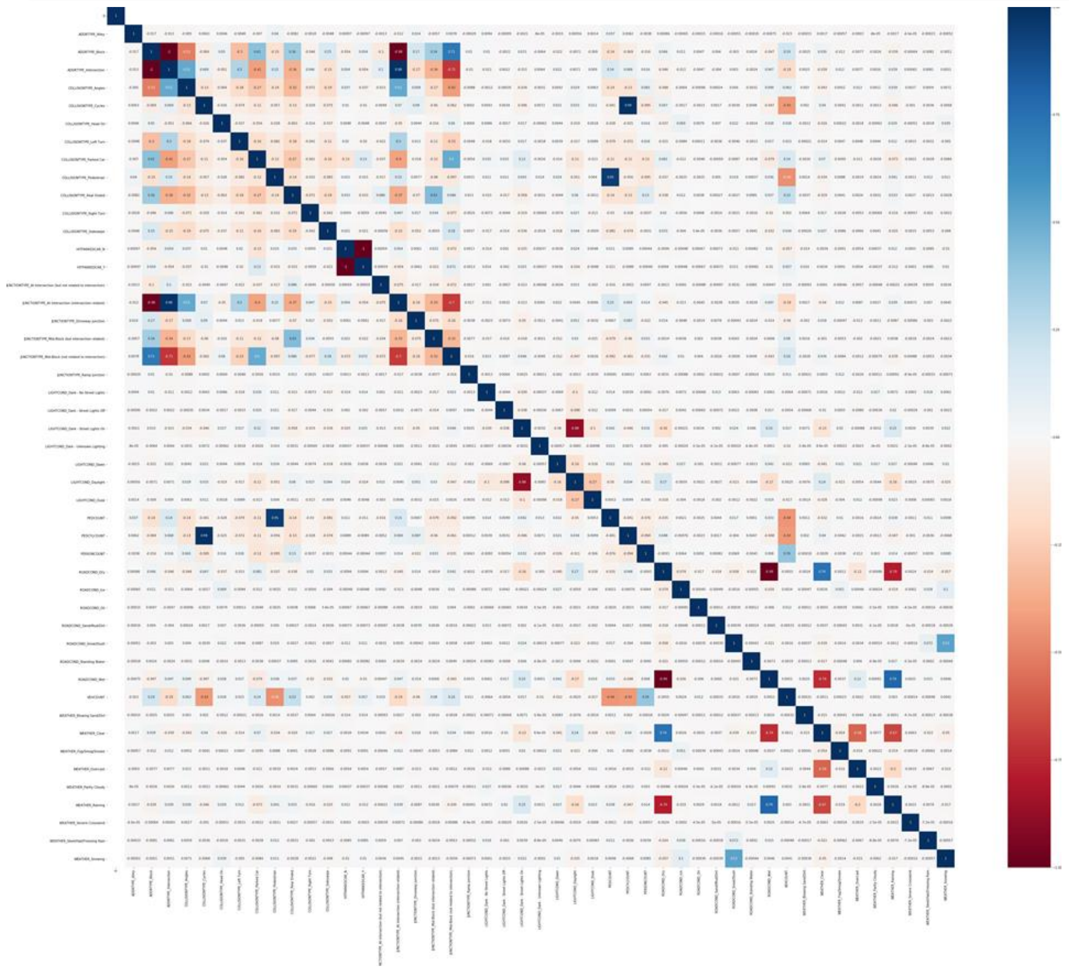
- The figure on the right shows the columns of independent variable after one hot encoding



```
Pre_X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145369 entries, 0 to 194672
Data columns (total 47 columns):
 #   Column                                                       Non-Null Count   Dtype
---  ------                                                       --------------   -----
 0   PERSONCOUNT                                                  145369 non-null  int64
 1   PEDCOUNT                                                     145369 non-null  int64
 2   PEDCYLCOUNT                                                  145369 non-null  int64
 3   VEHCOUNT                                                     145369 non-null  int64
 4   ADDRTYPE_Alley                                               145369 non-null  uint8
 5   ADDRTYPE_Block                                               145369 non-null  uint8
 6   ADDRTYPE_Intersection                                        145369 non-null  uint8
 7   COLLISIONTYPE_Angles                                         145369 non-null  uint8
 8   COLLISIONTYPE_Cycles                                         145369 non-null  uint8
 9   COLLISIONTYPE_Head On                                        145369 non-null  uint8
 10  COLLISIONTYPE_Left Turn                                      145369 non-null  uint8
 11  COLLISIONTYPE_Parked Car                                     145369 non-null  uint8
 12  COLLISIONTYPE_Pedestrian                                     145369 non-null  uint8
 13  COLLISIONTYPE_Rear Ended                                     145369 non-null  uint8
 14  COLLISIONTYPE_Right Turn                                     145369 non-null  uint8
 15  COLLISIONTYPE_Sideswipe                                      145369 non-null  uint8
 16  JUNCTIONTYPE_At Intersection (but not related to intersection)  145369 non-null  uint8
 17  JUNCTIONTYPE_At Intersection (intersection related)          145369 non-null  uint8
 18  JUNCTIONTYPE_Driveway Junction                               145369 non-null  uint8
 19  JUNCTIONTYPE_Mid-Block (but intersection related)            145369 non-null  uint8
 20  JUNCTIONTYPE_Mid-Block (not related to intersection)         145369 non-null  uint8
 21  JUNCTIONTYPE_Ramp Junction                                   145369 non-null  uint8
 22  WEATHER_Blowing Sand/Dirt                                    145369 non-null  uint8
 23  WEATHER_Clear                                                145369 non-null  uint8
 24  WEATHER_Fog/Smog/Smoke                                       145369 non-null  uint8
 25  WEATHER_Overcast                                             145369 non-null  uint8
 26  WEATHER_Partly Cloudy                                        145369 non-null  uint8
 27  WEATHER_Raining                                              145369 non-null  uint8
 28  WEATHER_Severe Crosswind                                     145369 non-null  uint8
 29  WEATHER_Sleet/Hail/Freezing Rain                             145369 non-null  uint8
 30  WEATHER_Snowing                                              145369 non-null  uint8
 31  ROADCOND_Dry                                                 145369 non-null  uint8
 32  ROADCOND_Ice                                                 145369 non-null  uint8
 33  ROADCOND_Oil                                                 145369 non-null  uint8
 34  ROADCOND_Sand/Mud/Dirt                                       145369 non-null  uint8
 35  ROADCOND_Snow/Slush                                          145369 non-null  uint8
 36  ROADCOND_Standing Water                                      145369 non-null  uint8
 37  ROADCOND_Wet                                                 145369 non-null  uint8
 38  LIGHTCOND_Dark - No Street Lights                            145369 non-null  uint8
 39  LIGHTCOND_Dark - Street Lights Off                           145369 non-null  uint8
 40  LIGHTCOND_Dark - Street Lights On                            145369 non-null  uint8
 41  LIGHTCOND_Dark - Unknown Lighting                            145369 non-null  uint8
 42  LIGHTCOND_Dawn                                               145369 non-null  uint8
 43  LIGHTCOND_Daylight                                           145369 non-null  uint8
 44  LIGHTCOND_Dusk                                               145369 non-null  uint8
 45  HITPARKEDCAR_N                                               145369 non-null  uint8
 46  HITPARKEDCAR_Y                                               145369 non-null  uint8
dtypes: int64(4), uint8(43)
memory usage: 11.5 MB
```

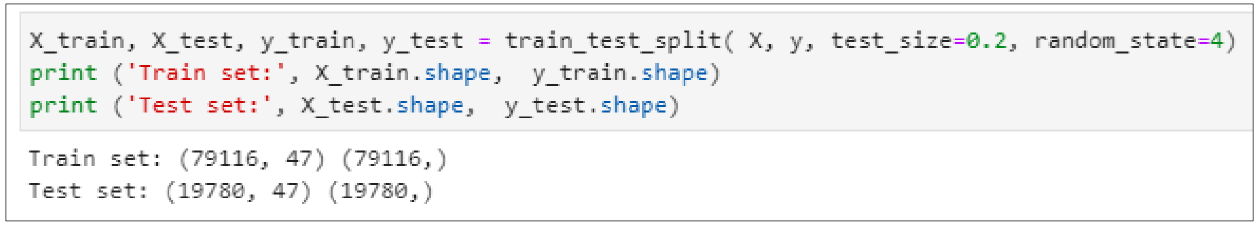
Exploratory Data Analysis



- Before creating and splitting the data, we can explore our clean and balanced data for the correlation between different variables
- The values shown are between 1 and -1. If the value is near zero, it means that the correlation is weak. Values close to 1 and -1 shows a strong positive or negative correlation, respectively
- From the correlation matrix, we can see a strong positive correlation between the 'ROADCOND_Wet' and 'WEATHERCOND_Rainy', 'ADDRTYPE_Intersection', and 'JUNCTIONTYPE_At Intersection', and 'PEDCOUNT' and 'COLLISIONTYPE_Pedestrian'
- Moreover, a strong negative correlation is observed between the attributes like 'WEATHER_Raining' and 'ROADCOND_Dry' as the given condition is contradictory

Data Normalization

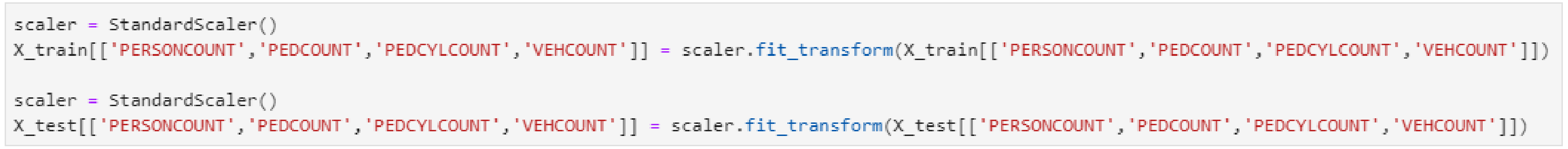
- The data should be standardized to reduce the error in the prediction because of large values in the columns. The standardization only applies to the column with the numerical value, not to the already encoded categorical values
- But before carrying out data standardization, the data set must be split into testing and training data set



```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (79116, 47) (79116,)
Test set: (19780, 47) (19780,)
```

- We can use StandardScaler() from the preprocessing module of the Scikit Learn library. First, we create a StandardScaler() object and then apply .fit_transform method on the numerical columns of our testing and training data set separately



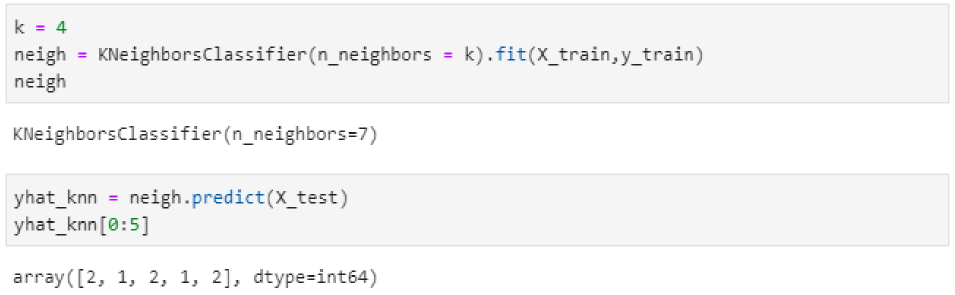
```
scaler = StandardScaler()
X_train[['PERSONCOUNT','PEDCOUNT','PEDCYLCOUNT','VEHCOUNT']] = scaler.fit_transform(X_train[['PERSONCOUNT','PEDCOUNT','PEDCYLCOUNT','VEHCOUNT']])

scaler = StandardScaler()
X_test[['PERSONCOUNT','PEDCOUNT','PEDCYLCOUNT','VEHCOUNT']] = scaler.fit_transform(X_test[['PERSONCOUNT','PEDCOUNT','PEDCYLCOUNT','VEHCOUNT']])
```

Modeling

- Our goal is to predict the severity of an accident in binary classes i.e. the severity could be either 1 or 2
- This clearly shows that this problem can be solved by using Classification machine learning algorithms as the target attribute is a categorical variable
- Classification is a supervised learning approach in which the label of our data is categorized into a discrete set of categories or classes. Classification determines a class label for an unlabelled case
- The following classification models are used :
    1. KNN model
    2. Decision Tree model
    3. SVM Model
    4. Logistic regression model

KNN Model

- k-nearest neighbour is an algorithm that takes several labelled points and uses them to label other points based on the similarity to other cases
- Initially, a value of k = 4 is assumed and the distance of unknown cases is calculated. Then k-observations in the training data that are nearest to the unknown data point is selected. Afterward, the response of the unknown data point using the most popular response value from the k-nearest neighbours is predicted

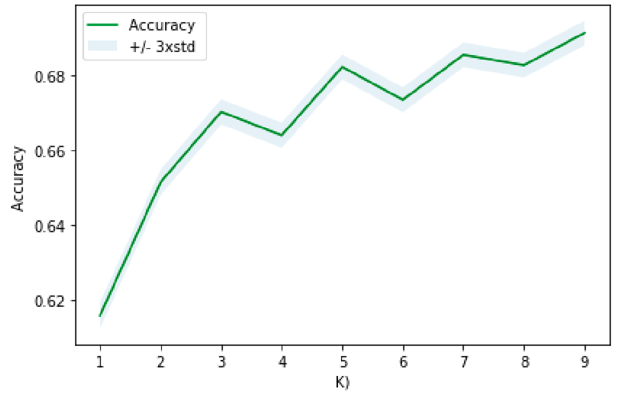


```
k = 4
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
KNeighborsClassifier(n_neighbors=7)
```

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

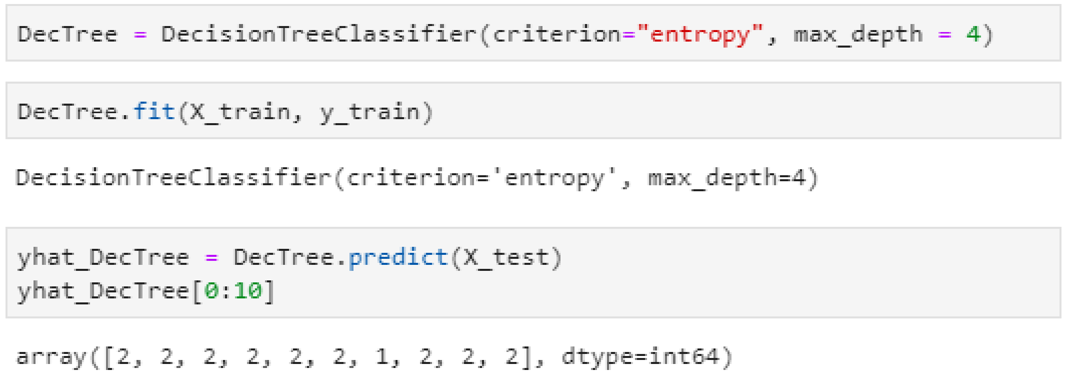
```
array([2, 1, 2, 1, 2], dtype=int64)
```



- Now, to calculate the optimum value of k, we can loop different values of k using for loop and then visualize the value of k vs. the accuracy score. The visualization is shown below. After the evaluation, we select k = 7 for our model

Decision Tree Model

- The basic intuition behind a decision tree is to map out all the possible decision paths in the form of a tree
- First, the attributes are considered and chosen one after another, then the significance of the attributes in the splitting of data is calculated
- Thereafter, the data based on the value of the best attribute is split. Finally, this process is repeated until all the attributes are split into the leaf nodes
- The aim here is to increase the accuracy as much as possible and keep the entropy smallest



```
DecTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
```

```
DecTree.fit(X_train, y_train)
```

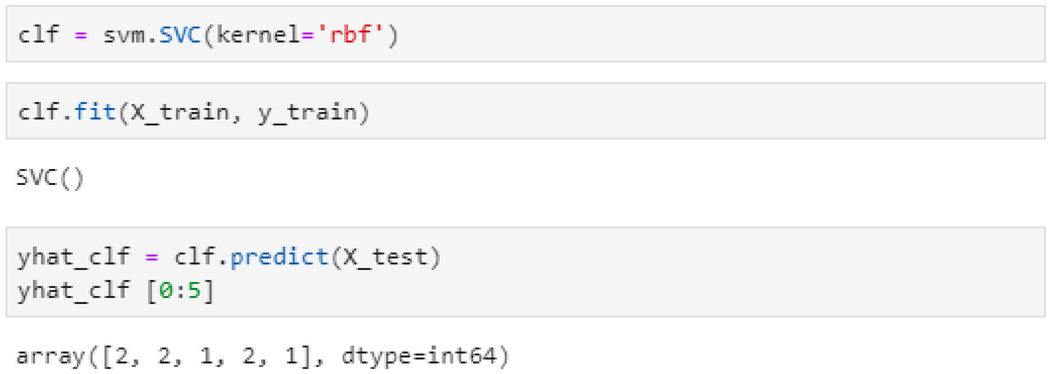
```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
yhat_DecTree = DecTree.predict(X_test)
yhat_DecTree[0:10]
```

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

SVM Model

- The support vector machine algorithm classifies cases by finding a separator
- SVM works by mapping data to a high-dimensional feature space in such a way that data points can be classified even when the data is not linearly separable in 2D space
- When plotting the data in multi-dimensional space, the category or class or the data points are separated using a separator represented as a hyperplane. The mapping of the data in higher dimensions is called kernelling. When creating an SVM object clf, RBF (Random Basis Function) type of kernel is been selected



```
clf = svm.SVC(kernel='rbf')
```

```
clf.fit(X_train, y_train)
```

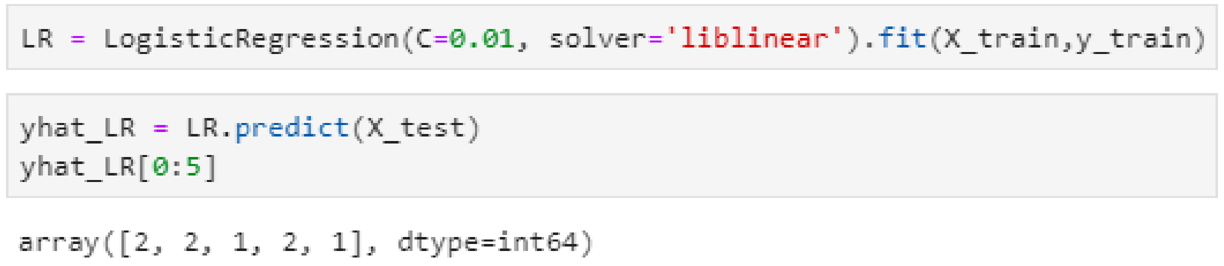
```
SVC()
```

```
yhat_clf = clf.predict(X_test)
yhat_clf [0:5]
```

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

Logistic Regression Model

- Logistic regression can be used to determine the probability of the outcome and to understand the impact of the feature
- The probability is predicted by fitting the data using a sigmoid or logistic function. After fitting the data, the model output *yhat* and the actual output of severity are compared, and the error value is recorded
- The aim here is to reduce the cost i.e. maximizing the accuracy by constantly changing the value of theta. In our instance of logistic regression, the value of C (indicates inverse of regularization strength) is taken as 0.01 (must be a positive float). The numerical optimizer used is ‘liblinear’



```
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
```

```
yhat_LR = LR.predict(X_test)
yhat_LR[0:5]
```

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

Model Evaluation

- For the evaluation of the models, we have selected 3 types of evaluation criteria
- All the implemented method gives a score between 0 and 1 where 0 being least accurate and 1 being very accurate
- The following methods are used :
    1. Accuracy Scores
    2. Jaccard Similarity Scores
    3. F1 Score

Result and Conclusion

- The following table shows the above-mentioned accuracy scores for 4 of our selected models

Algorithm	Jaccard	F1-score	Train set Accuracy	Test set Accuracy
KNN	0.4884	0.6832	0.7004	0.6853
Decision Tree	0.4173	0.6679	0.6922	0.6861
SVM	0.4943	0.6981	0.7125	0.7023
Logistic Regression	0.4736	0.6815	0.6960	0.6860

- Here, we can see that no algorithm has resulted in an accuracy of more than 71%. Considering the result, we can conclude that most model has shown the score in very close range of approx. +- 2%. The Support Vector Machine provides the highest accuracy scores in all the categories. Thus, for this project SVM classifiers proves to be the relatively most accurate model to predict the severity of the accident given different attributes or conditions
- Moreover, it has also been observed that the time required to fit the model and predicting the output, Logistic regression, and Decision tree took the least time
- The maximum test accuracy of the SNM model is 70% which can be a result of underfitting and can be improved using more data. Due to an imbalance in the selected data set, many observations are dropped, so in the future, a more balanced data set can be used

Future scopes

- Many important columns such as ‘UNDERINFL’ and ‘INATTENTIONIND’ which shows whether the driver is under influence of drugs and driver inattention respectively, has been dropped due to data inconstancy and missing values
- These could have been good factors in predicting accident severity. In the future, we can select more consistent data
- Moreover, we only had a binary label in our data set (1s and 2s), in the future, we can have data with multiclass labels to make machine learning model more challenging