REPORT:ROAD ACCIDENT SEVERITY PREDICTOR

This Report is a part of the peer graded assignment of the final course:Applied Data Science Capstone of the IBM Data Science Professional Certificate Course.

We will be following the CRISP-DM(Cross-Industry Standard Process for Data Mining) Approach to solve the problem and build a predictor model.

BUSINESS UNDERSTANDING:

Oftentimes, while travelling from one place to another we encounter accidents on the road, sometimes severe, sometimes fatal, sometimes not so severe. What if we knew in advance the severity of accidents beforehand and avoid travelling when the probability of accidents is more.

This project is useful for anyone and everyone: If you're travelling from 1 city to another, your daily commute to your workplace and back home, including your everyday and other travels. Knowing in advance the severity of an accident will help you save you and your time by avoiding taking that route. Moreover, it is useful for the government as well to check what conditions lead to more severe accidents and how to reduce it.

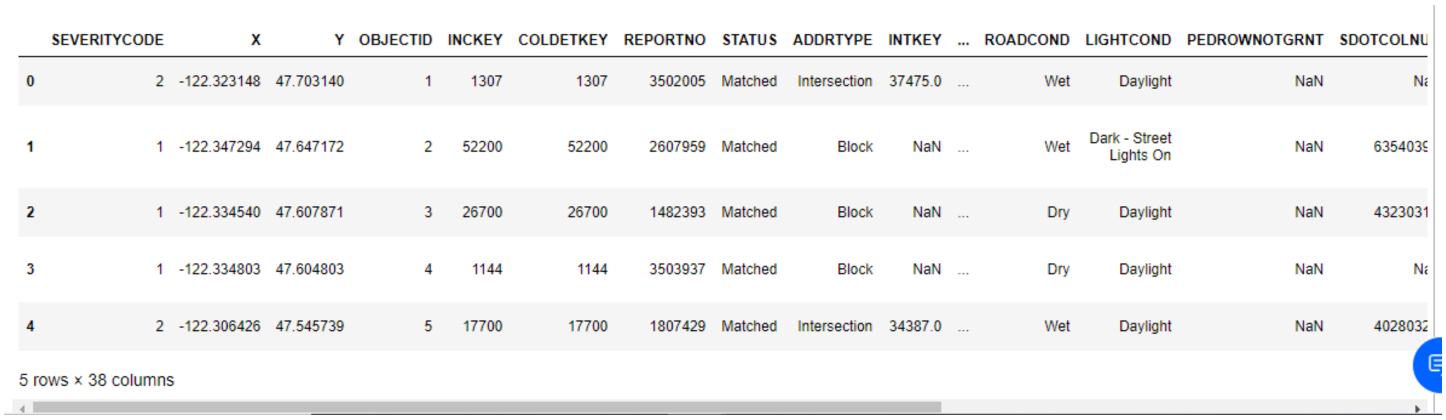
The stakeholders being the drivers helping them take precaution and the Public Development Authority of Seattle which can improve the conditions of road and street lighting to prevent more accidents.

A better understanding of the problem will be established in subsequent sections.

DATA UNDERSTANDING:

The source of this data is the resource provided in the Coursera Applied Data Science Capstone.

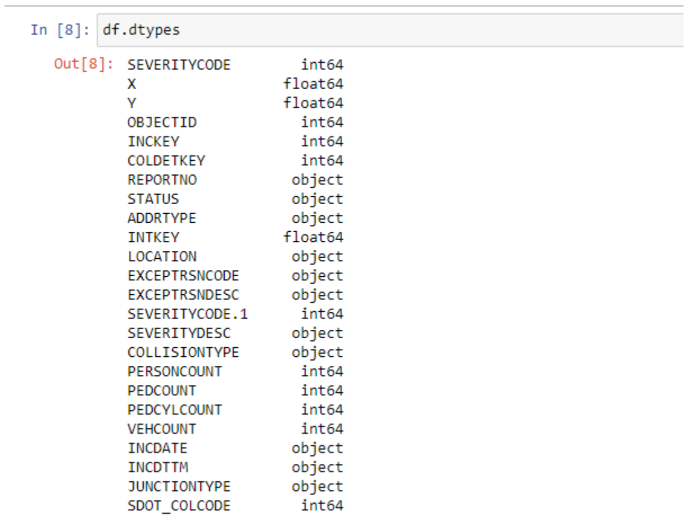
There are 37 attributes in the dataframe that we're using for the model development and not all of that information is required to build the model. So we drop the unnecessary columns before working on the model.



	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDROWNOTGRNT	SDOTCOLNU
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight	NaN	Na
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On	NaN	635403
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight	NaN	4323031
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight	NaN	Na
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight	NaN	402803

5 rows × 38 columns

To build a better understanding of data we use the dtypes method on the dataframe to know the datatypes of different columns in the table:



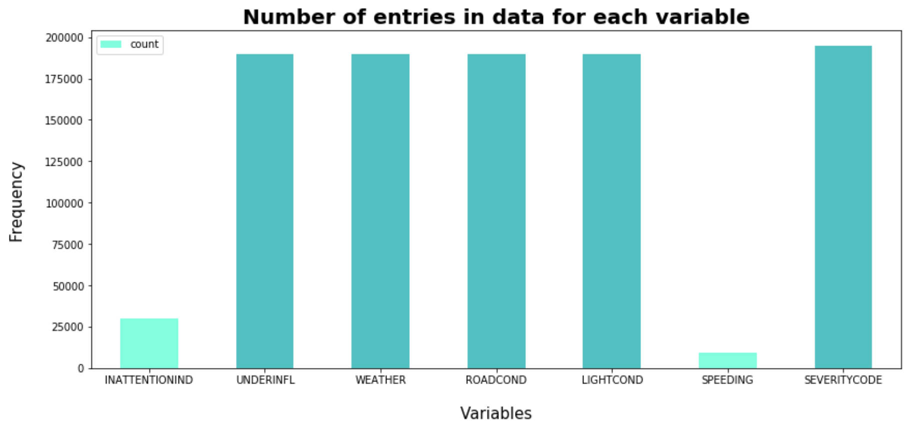
```
In [8]: df.dtypes

Out[8]: SEVERITYCODE        int64
        X                 float64
        Y                 float64
        OBJECTID            int64
        INCKEY              int64
        COLDETKEY           int64
        REPORTNO           object
        STATUS             object
        ADDRTYPE           object
        INTKEY            float64
        LOCATION           object
        EXCEPTRSNCODE      object
        EXCEPTRSNDESC      object
        SEVERITYCODE.1      int64
        SEVERITYDESC       object
        COLLISIONTYPE      object
        PERSONCOUNT         int64
        PEDCOUNT            int64
        PEDCYLCOUNT         int64
        VEHCOUNT            int64
        INCDATE            object
        INCDTTM            object
        JUNCTIONTYPE       object
        SDOT_COLCODE        int64
```

In this model our <u>target variable (X) is SEVERITYCODE</u> and the potential Independent variables can be <u>ROADCOND,WEATHER,LIGHTCOND,SPEEDING,UNDERINFL,INATTENTIONIND.</u>

But, we see that most of these variables are of type object and difficult to be deployed in the model. So, we modified the values of these variables to int type. However, even when the SEVERITYCODE is an int type data type, we see that the values it stores are 1(for Property Damage) and 2(Injury Collision). So,we would like to change these values of 1 and 2 to 0 and 1 for a better model.

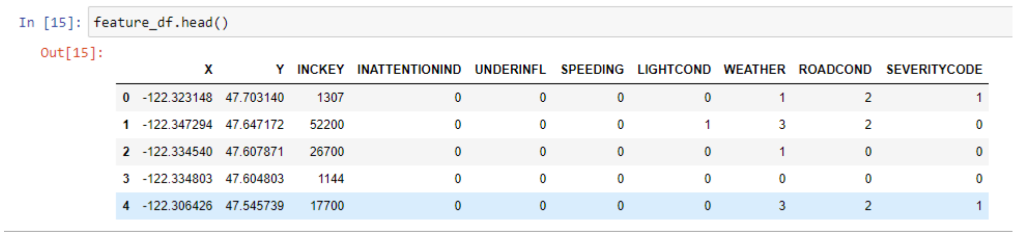
On modifying it, we use the describe() function on the modified datatype, plot a graph of the number of entries in each attribute and notice that some of our attributes have quite a less number of entries stored in them. We also can’t drop all these fields, since the data may lose it’s meaning. As a result, we not only need to change the datatype of these attributes but also fill the empty fields to make the data more reliable for building the model.



Moving on, we assign integers to each unique attribute in an attribute. So, the key that we have used to replace the values of different attributes is pretty simple. For variables storing binary information in the form of yes/no, we used 1 for Yes and 0 for No. These attributes include UNDERINFL,SPEEDING and INATTENTIONIND. For LIGHTCOND, we distributed the data in 3 types:Light,Medium and Dark. We've used 0 for Light,1 for Medium and 2 for Dark. Coming to ROADCOND the basis of indexing is 0 for Dry, 1 for Mushy and 2 for Wet. As for WEATHER, again we classified the data into 3 categories: 0 for clear or overcast,1 for Windy, 2 for Rain and 3 for Snow.

For attributes with null values, those were assigned the value 0. And for the ones storing values like other or unknown, we couldn't happen to delete the rows because it would've adversely affected our model. So we used another unique value for them in the attributes that fell in this category.

Our data is now ready to be used. This is what it looks like:



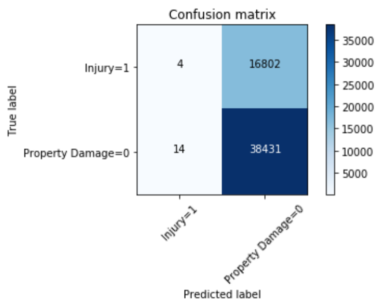
```
In [15]: feature_df.head()
Out[15]:
```

	X	Y	INCKEY	INATTENTIONIND	UNDERINFL	SPEEDING	LIGHTCOND	WEATHER	ROADCOND	SEVERITYCODE
0	-122.323148	47.703140	1307	0	0	0	0	1	2	1
1	-122.347294	47.647172	52200	0	0	0	1	3	2	0
2	-122.334540	47.607871	26700	0	0	0	0	1	0	0
3	-122.334803	47.604803	1144	0	0	0	0	0	0	0
4	-122.306426	47.545739	17700	0	0	0	0	3	2	1

METHODOLOGY:

1. Exploratory Data Analysis: It is a preliminary stage in data analysis where we summarize the main characteristics of data, gaining a better understanding of it, uncovering relationships between variables and extracting important variables. In our model, we have used some of these tools like the descriptive statistics. In descriptive statistics, we used the describe() function to describe basic features of data, summarizing the sample and measures of data. We also created a plot to visually understand the number of entries in each attribute to be used.

2. Inferential Statistical Testing: It was used after the model was built to judge the performance of each model and select the best out of them. The tools that we used for inferential statistical analysis are F1 score, Jaccard similarity coefficient and log loss (in case of Logistic Regression). Each of these evaluation metrics test the predicted values with the actual values and return the average accuracy of the model built. F1 score is built in reference to confusion matrix stating the True Positives, True Negatives, False Positives and False Negatives. It calculates the precision and recall values to arrive at F1 score. F1 score of 1 is the best and 0,the worst.



Moving on, the Jaccard Similarity Score compares members of 2 sets, i.e, Actual and Predicted values here, and determines the elements same in both and distinct. The higher the score, better the model. Lastly, Log Loss or Cross Entropy Function evaluates the probability of the correctness of the predicted value as compared to the actual value.

3. Machine Learning Model: The Machine Learning technique used to solve the problem is CLASSIFICATION. Classification is a supervised learning approach categorizing some unknown items into a discrete set of categories/classes. It was the best suited for our problem because a classification model takes target attribute as a categorical variable (Sever or Not) in our case. The algorithms that we used to construct our model are K-Nearest Neighbors, Decision Trees and Logistic Regression. We didn't use Support Vector Machines because it is well suited only for small datasets while ours was a huge one. The K-Nearest Neighbor is based on the assumption that similar cases with same class labels are near each other. So the KNN algorithm classifies classes based on their similarity to other cases that are near each other(Neighbors). Decision Tree maps out possible decision paths in the form of a tree. Each internal node corresponds to a test, each branch to a result of the test and each leaf node assigns parent to a class. Lastly, Logistic Regression is a statistical and Machine Learning technique to classify records of a dataset based on values of the input fields. It is most suitable for binary data(Yes/No,True/False,1/2 etc).

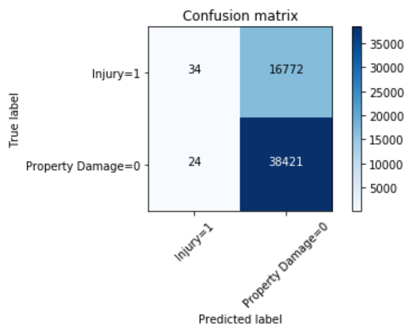
RESULT:

1. DECISION TREE CLASSIFICATION: Decision Tree maps out possible decision paths in the form of a tree. Each internal node corresponds to a test, each branch to a result of the test and each leaf node assigns parent to a class. The attributes that were used to built the model were 'entropy' for criterion and 6 for max depth.

Classification Report:

	precision	recall	f1-score
0	1.00	0.70	0.82
1	0.00	0.59	0.00
micro avg	0.70	0.70	0.70
macro avg	0.50	0.64	0.41
weighted avg	1.00	0.70	0.82

Confusion Matrix:

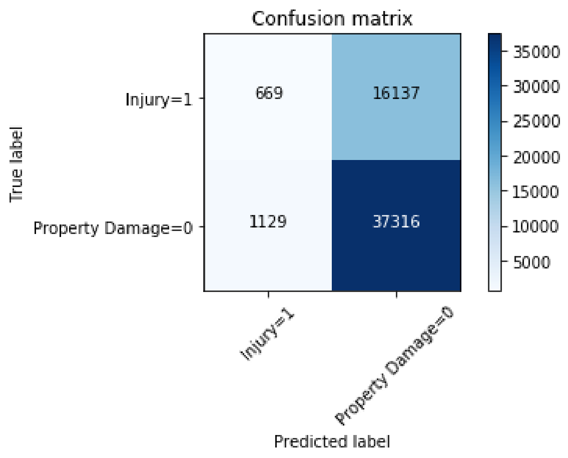


2. K-NEAREST NEIGHBOR: It is based on the assumption that similar cases with same class labels are near each other. So the KNN algorithm classifies classes based on their similarity to other cases that are near each other(Neighbors). The value of k used in our model is 8 as it is the best suited value.

   Classification Report:

	precision	recall	f1-score
0	0.70	0.97	0.81
1	0.37	0.04	0.07
micro avg	0.69	0.69	0.69
macro avg	0.54	0.51	0.44
weighted avg	0.60	0.69	0.59

Confusion Matrix:

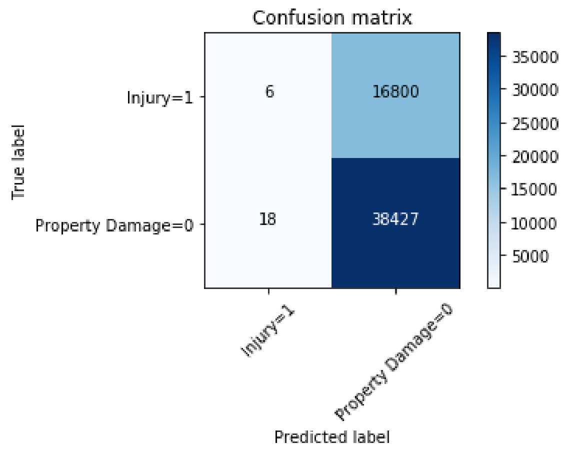


3. LOGISTIC REGRESSION: A statistical and Machine Learning technique to classify records of a dataset based on values of the input fields. It is most suitable for binary data(Yes/No,True/False,1/2 etc). The solver we used is 'liblinear' for this particular model.

Classification Report:

	precision	recall	f1-score
0	0.70	1.00	0.82
1	0.25	0.00	0.00
micro avg	0.70	0.70	0.70
macro avg	0.47	0.50	0.41
weighted avg	0.56	0.70	0.57

Confusion Matrix:



DISCUSSION:

Algorithm	F1 Score	Jaccard Similarity	Log Loss
Decision Tree	0.8198	0.6960	NA
K-Nearest Neighbor	0.7880	0.6875	NA
Logistic Regression	0.8201	0.6956	0.6108

The above table shows the accuracy of the different models used in our model development. A brief description of each of these evaluation metrics is as follows:

1. F1 Score: It is built in reference to confusion matrix stating the True Positives, True Negatives, False Positives and False Negatives. It calculates the precision and recall values to arrive at F1 score. F1 score of 1 is the best and 0,the worst.

$$F1 = 2 \text{ x } \frac{Precision * Recall}{Precision + Recall}$$

2. Jaccard Similarity Coefficient: Compares members of 2 sets, i.e, Actual and Predicted values here, and determines the elements same in both and distinct. The higher the score, better the model.

$$\mathbf{J(X,Y) = |X \cap Y| \ / \ |X \cup Y|}$$

3. Log Loss: Log Loss or Cross Entropy Function evaluates the probability of the correctness of the predicted value as compared to the actual value.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

CONCLUSION:

On comparing the accuracy of different models through different measures, Logistic Regression is the best suited algorithm to solve our problem. It shows the highest accuracy among all the 3 algorithms.

However, the model has not performed as well as it should have performed because of imbalance in data as we it contained a lot of null values which had to be filled with 0 by default at some places. This led to inefficiency in the model.
No matter what, this model can still be used to:

- Predict the severity of a road accident.
- Help drivers make choices as to avoid taking the route involving these accidents.
- Help the government determine the conditions of the road and street lighting that led to major accidents, and hence improve the conditions.