IBM Applied Data Science Capstone:

Predicting the Severity of an Accident due to Environmental Aspects and Causes

Freddie Perez

October 13th, 2020

Introduction/Business Problem

The problem at hand is find a solution that can help predict the severity of an accident if an accident were to occur given various environmental information, not including human interactions or effects. This is to help potential stakeholders, which in this case includes those that have to drive to get to their final destination every day, on how to safely make the journey and/or change their driving behavior to mitigate any risk. Given various inputs of a particular day, how can we determine if there will be an accident on the way to a destination? If we can find a method to answer this problem, we can give the stakeholders that insight of choosing a different method of transportation. Not only could this insight potentially offset any risk that the person would incur if they were the ones to get an accident, but it could also allow for their ability to reach their final destination within the time frame that they were expecting to, due to not being stuck in traffic waiting for the situation to clear up.

Data:

Using the example dataset's metadata file, we can see that it is for the relatively high density population within the city of Seattle, Washington. Within this dataset, we can see that it has 37 attributes that can be of interest, including ones that detail the weather, location and level of accidents, the road conditions, among many others. Since we are trying to determine the probability of how bad or severe of an accident it will be, given that it will occur, we will use the severity attribute as the target variable within our model. In order to decide which attributes would be of most use or importance for the model building, I will apply a feature selection process by utilizing attributes that provide the most information gain on the data, given the target

variable. In addition, this dataset also contains an abundance of over 70,000 samples or observations that we can use to train a machine learning model and test against, as needed. If it were possible, a larger dataset would preferable for use. Since we'll be working with classification, some classification models that come to mind that may be of use include building a Decision Tree, using K-Nearest Neighbors, Support Vector Machines, and/or Logistic Regression. Once I've built each model, I will do a model evaluation based on a couple of scoring mechanism such as the Jaccard Score, Average F1-Score, and LogLoss metrics to see which model performed the best.

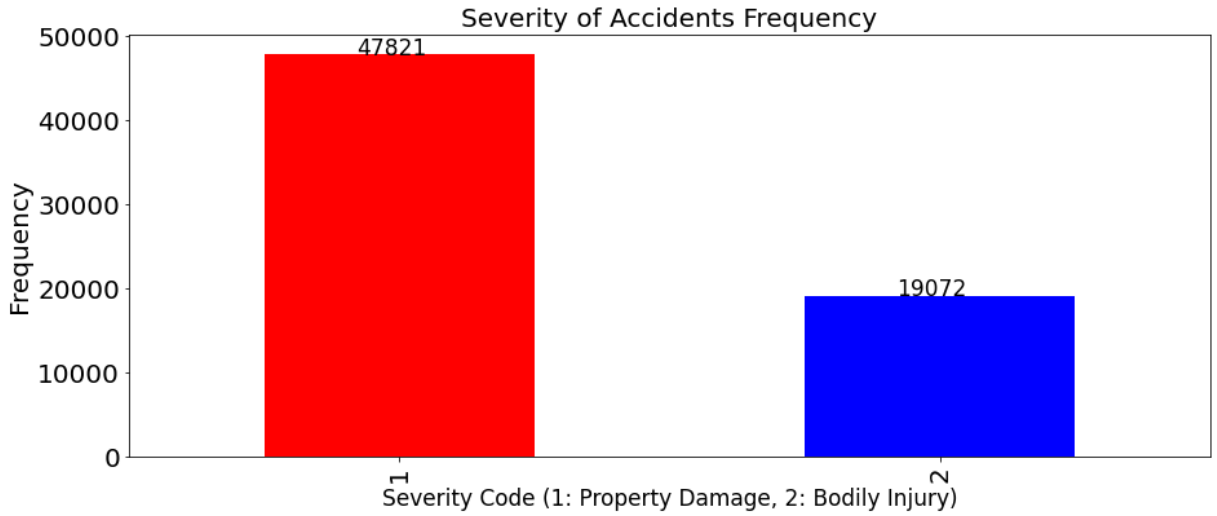
Methodology:

After looking into the Data Set summary document for the Seattle based data set, I went through the process of feature selection. Given the descriptions provided for each attribute, and the scope of the question, I've settled on a set variables that only take into account the environment of any particular instance of an accident. This includes the target attribute '***SEVERITYCODE'***, as well as '***WEATHER'***, '***ROADCOND'***, '***LIGHTCOND'***.

Using the '***SPEEDING'*** attribute, given that these accidents were primarily caused by speeding on some stakeholders part, I've removed any instances where this attribute stated 'Y' for yes. Using the target variable '***SEVERITYCODE'***, I also removed any instances where this attribute had the value '***0'***, which means '***Unknown'*** and these instances will not helpful with training the models. After removing these instances, this left the majority of the dataset with only 2 levels of '***SEVERITYCODE'***: '***1***' which meant there was property damage, and '***2***', which meant there was bodily injury, with the majority of the accidents being classified as '***1***'. Once these instances

were removed, I then looked into the attributes '***WEATHER***', '***ROADCOND***', and '***LIGHTCOND***' to see what values they contained. Seeing that they were all categorical, I then applied 'One-Hot Encoding' to generate additional attributes based on the values within those columns.

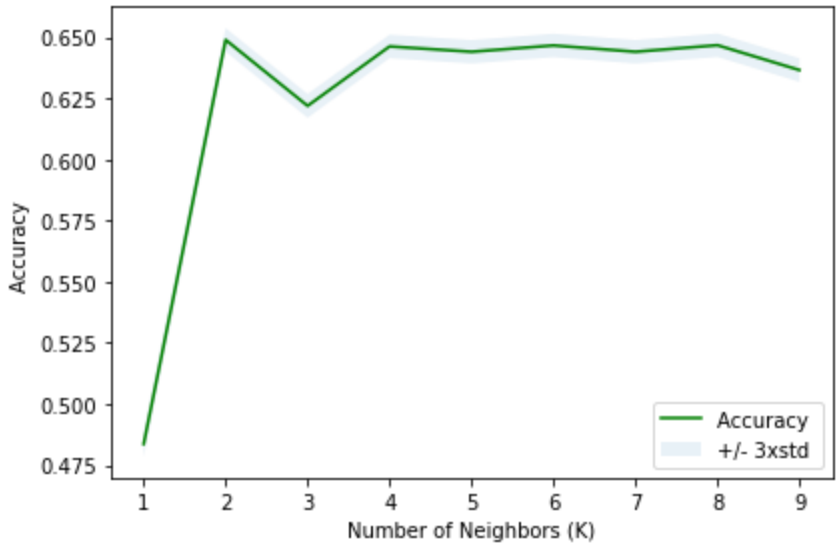
Once all of the data munging has been completed, I normalize the dataset as this will be helpful for some of Machine Learning Models that used Euclidian distance, such as the K-Nearest Neighbors algorithm. The dataset is then split into the `X` dataset, which contains all of the potential training instances/observations, and the `y` data set which has the target variable labels for each of those observations. With normalization completed, I then split the dataset into training and testing sets, called `X_train`, `X_test`, `y_train`, `y_test`. Due to the dataset being unbalanced with observations being classified as '***SEVERITYCODE***' 1 much more frequently when compared to '***SEVERITYCODE***' 2, I made sure to stratify the dataset testing and training sets to ensure there is the same proportion of each class in both types of sets. In addition, to help with how imbalanced the dataset is, I've also used the 'Oversampling' method on the Minority classes as well as the SMOTE Technique (Synthetic Minority Oversampling Technique) coupled with Undersampling the Majority class.



With these two additional datasets, I will try re-running the models to see if the scoring/evaluation metrics improve significantly. The Machine learning models that I will be using in this experiment are K-Nearest Neighbors, Decisions Trees, the ensemble method Random Forest Classifier, Support Vector Machine, and Logistic Regression.

Results:

Starting with the first of the ML algorithm K-Nearest neighbors, I started with figuring out which number of K would maximize the accuracy for each dataset. This algorithm classifies points based on how close observations are to each other with respect to Euclidian Distance. For all three dataset (the `Stratified`, `Oversampled Minority Class`, and the `SMOTE/Undersampled Majority Class` datasets), K=2 provided the highest accuracy. To compare against the other models, I will be using other evaluation metrics which may be more help in determining the best model due to how imbalanced the original dataset is.



The next model I tried using to help with this classification problem is the Decision Tree model. This algorithm splits the data based on binary splits that provides the highest information gain. For this model building process, I used the criterion entropy due to it having the greatest information gain. Currently, the `Stratified` training/testing dataset and the `Oversampled` training/testing datasets had the highest accuracy value compared to the `SMOTE/under sampled` training/testing datasets. In conjunction with the Decision Tree, I also tried running the ensemble method Random Forest Classifier to see if this could produce better results. The Random Forest model generates multiple decision tree structures based on a subset of attributes, and each of the decision trees vote on the result/classification of each of the observations within the testing set.

The next model that I tried on the accident data set is Support Vector Machine. SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data is not considered linearly separable. Decision boundaries between the categories is found, then the data is transformed in such a way that the separator could be drawn as a hyperplane.

The last model that I tried using is the Logistic Regression. This model is a variation of Linear Regression, where the observed dependent variable or target attribute, `***y***` or in our case '***SEVERITYCODE***', is categorical. It also creates a formula that can predict the probability of the class or target variable as a function of the independent attributes.

Discussion:

After running each of the classification models, it seemed like each of the model performed similarly in terms of the evaluation metrics that I used, with some slight variations. The evaluation metrics that I chose to use, outside of accuracy are the Average F1 score, Jaccard Similarity Score, and the LogLoss for the Logistic Regression. The F1 score is a measure of accuracy for a model such that it calculates the mean of a models precision and recall. The highest value that the F1 Score can take is 1, which dictates that there was perfect precision and perfect recall. For all of the models evaluated, the F1 score is the average of the individual F1 scores of the two target attribute values, 1 for property damage and 2 for injury. Precision refers to the number of True positive results over the Total number of observations within the testing set predicted to be positive. Recall refers to the number of True Positive results over the number of instances that were predicted to be positive and the number the predicted to be Negative despite being the class actually being positive. Out of all the models that was used, the Decision Tree model with the `Stratified` training and testing data set did the best with an average F1 score of 0.6254. The worst performing model in terms of average F1 Score was Logistic Regression that was trained on the dataset that had both `SMOTE and Under sampling` applied on it.

The second evaluation metric that I used on these machine learning algorithms is the Jaccard Similarity Score. The Jaccard Similarity is a statistic used to understand the similarities between sample sets, and in this case our target attribute testing set against the predicted values that each model provided using the testing dataset. The model that produced the highest value for this metric was the Logistic Regression model that was trained on the `Stratified` dataset, which has a Jaccard similarity of 0.7149. The Random Forest classifier that was also trained on the

`Stratified` testing dataset came in a close second with a Jaccard Similarity score of 0.7148. The model that had the lowest Jaccard Similarity score ended up being the Logistic Regression model, which was trained on the dataset that had `SMOTE and under sampling` techniques applied to it. One model that did well throughout all the three different testing datasets was the K-Nearest Neighbors model, which ended up obtaining Jaccard Similarity scores 0.6386 for the `Stratified` dataset, 0.7126 for the `Oversampled` dataset, and 0.7146 for the `SMOTE/Under sampled` dataset.

KNN	Jaccard	F1-score	LogLoss
Stratified	0.6386	0.6020	NA
Oversampled	0.7126	0.5965	NA
SMOTE/Undersampled	0.7146	0.5965	NA

Decision Tree	Jaccard	F1-score	LogLoss
Stratified	0.6508	0.6262	NA
Oversampled	0.6522	0.6254	NA
SMOTE/Undersampled	0.3047	0.4632	NA

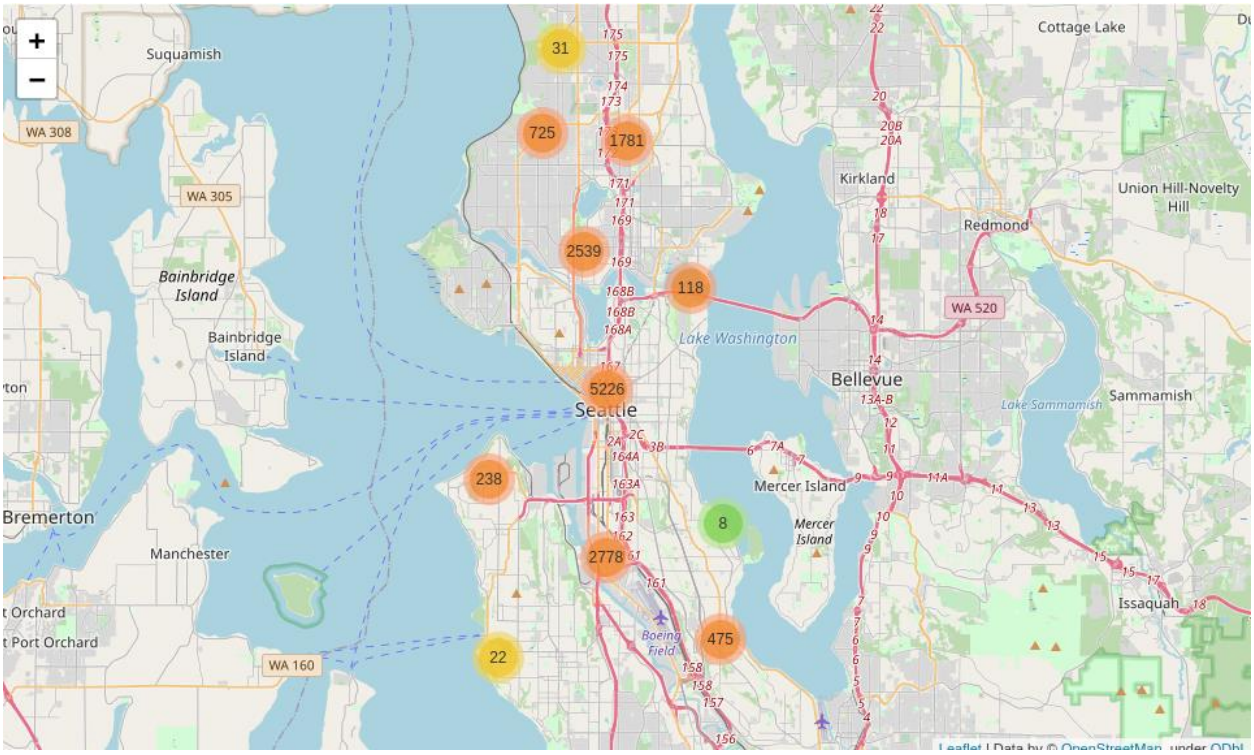
Random Forest	Jaccard	F1-score	LogLoss
Stratified	0.7148	0.5967	NA
Oversampled	0.3232	0.4773	NA
SMOTE/Undersampled	0.3034	0.4632	NA

SVM	Jaccard	F1-score	LogLoss
Stratified	0.7148	0.5964	NA
Oversampled	0.3220	0.4764	NA
SMOTE/Undersampled	0.3036	0.4627	NA

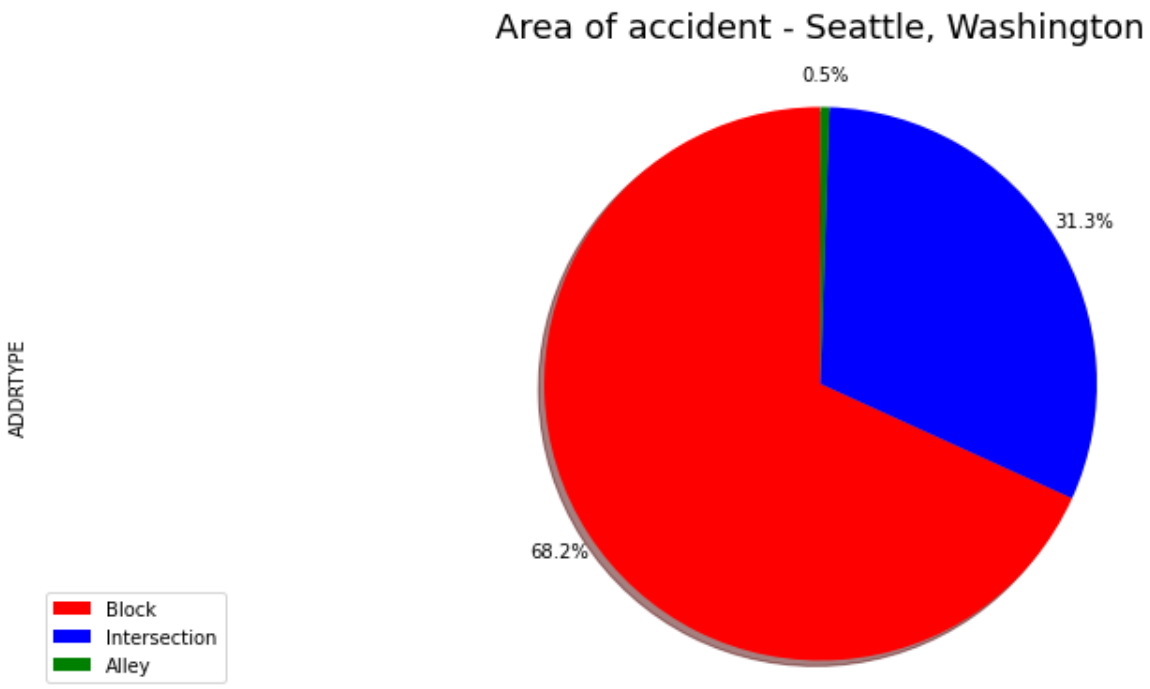
Logistic Regression	Jaccard	F1-score	LogLoss
Stratified	0.7149	0.5960	0.58
Oversampled	0.3567	0.5011	0.66
SMOTE/Undersampled	0.2793	0.4424	0.68

Recommendations:

Through the exploration process of the data and the machine learning model building process, various insights were brought up that could be of use for the stakeholders at hand. One insight is with respect to the locations that had the highest frequency of accidents, which were primarily centered on interstate-5 as it passes through highly dense, populated areas.



We can also see that the 68.2% of accidents occur around the city blocks, with intersection being a close second at 31.3%.



In addition, the majority of accidents were caused by either severe weather, poor lighting conditions, poor driving conditions, or a combination of all 3 major attributes. In general, methods to mitigate the potential to incur the risk of having some type of accident occur on your personal property would be to note the flow of traffic, increase your following distance to allow for better breaking opportunity, as well driving more defensively and safely during adverse conditions.

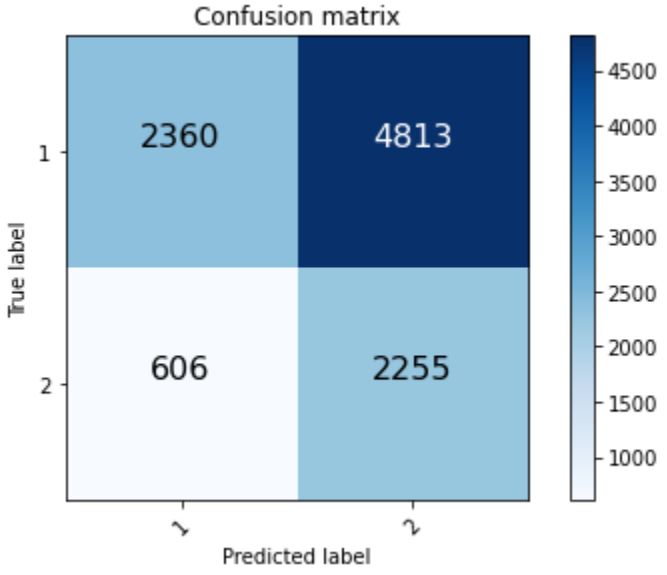
Conclusion:

Overall, it seemed like the best models in each case did average in terms of performance. None of them performed excellently or extraordinarily, and for a lot of the models, the accuracy was

always capped at 0.71 or 71%. Despite having the average F1 scores to work with as an additional metric, if you take a look at the Precision and Recall as additional metrics, you can see that most of the models were always skewed toward one class or the other (Severity Code 1 or 2) depending on how the algorithm was trained on the training set. This could primarily be due to having a very skewed/imbalanced dataset to begin work with. The classification report results from the Random Forest model, SVM model, and the Logistic Regression, all with the dataset that had `SMOTE/Under sampling` applied, had a very high precision score of 0.8 for class 1 and recall value of 0.79-0.82 for class 2. Meanwhile, the Recall was very low for class 1, and the Precision was also low for class two, despite any hyper-tuning applied to the models.

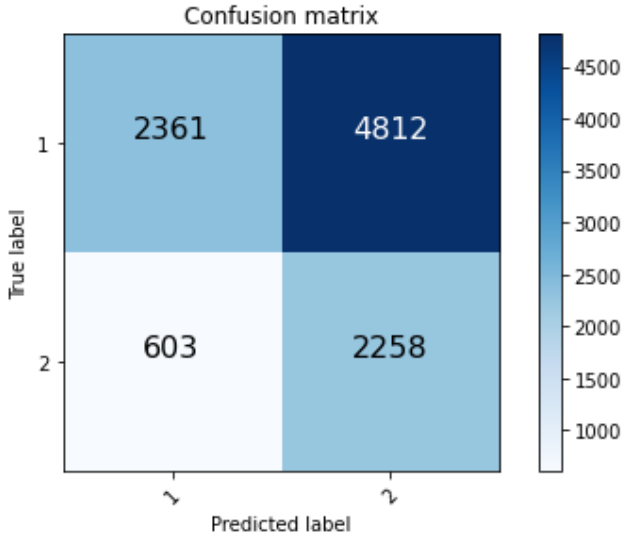
Random Forest (SMOTE/Under Sampling)

	Precision	**Recall**	**F1-Score**
1	0.80	0.33	0.47
2	0.32	0.79	0.45
Macro Avg.	0.56	0.56	0.46
Weighted Avg.	0.66	0.46	0.46



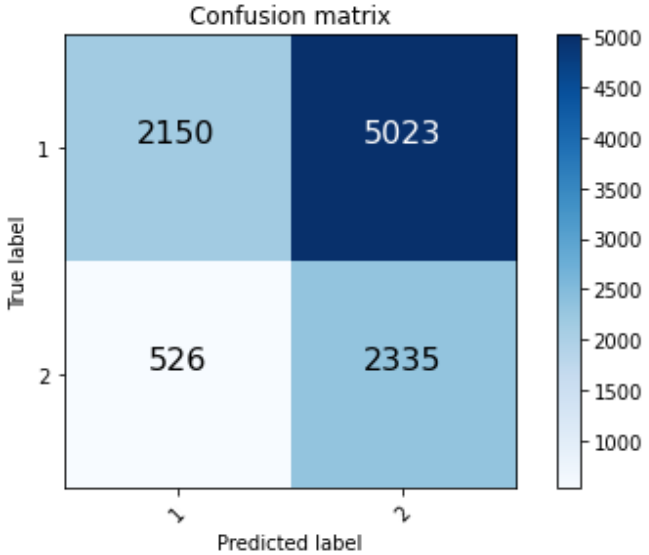
Support Vector Machine (SMOTE/Under Sampling)

	Precision	**Recall**	**F1-Score**
1	0.80	0.33	0.47
2	0.32	0.79	0.45
Macro Avg.	0.56	0.56	0.46
Weighted Avg.	0.66	0.46	0.46



Logistic Regression (SMOTE/Under Sampling)

	Precision	**Recall**	**F1-Score**
1	0.80	0.30	0.47
2	0.32	0.82	0.45
Macro Avg.	0.56	0.56	0.46
Weighted Avg.	0.66	0.46	0.46



I initially expected that applying the various over/under sampling techniques would have allowed the models to perform better, which is somewhat seen by taking a micro level view on that statistics generated. In general however, the stratified dataset, where the testing and training dataset each had the same proportions of each class with the original data, ended up performing much better when looking at the average F1 score and the Jaccard Similarity Score overall. To further improve these models, additional aspects such as a more balanced dataset and more instances where missing data was not prevalent, could have been useful for answering the business problem of how the environmental forces can play into and effect the severity of a car/mobile accident. If I had more time, I definitely would have tried to utilize some Time-Series predictive modeling to see if the time during the day, or even the month itself, had an impact on the probability of how severe an accident could be. This would add a temporal overlay that can add temporal based attributes such as time, date, and month as part of the learning process, which may improve the predictive power of the classification models if it comes to fruition.

References:

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Metadata.pdf

https://wsdot.wa.gov/