**Predicting Car Collision’s**

Anirudh Ramesh

08 September, 2020

1. Introduction

The problem that we will be aiming to figure out, through creating a model, if there is a way to advise drivers of potential future accidents by analyzing a combination of variables and predicting the collision severity. From the Data Collisions file provided, several potential indicators of an accident include Weather, Road conditions, and Light conditions. This model will be used to assist drivers and hopefully assist in deterring accidents when driving conditions are less than ideal. I believe there is a need for solving this type of problem as often times it is unclear whether or not conditions for driving in certain areas lead to higher accident probability, and therefore collision severity. I have personal experience with this as I have been involved in several accidents, that were not my fault but occurred as a result of driving conditions that made it less than ideal for me to succeed. Hopefully with the creation of this model, we are able to accurately ascertain with at least some certainty the factors that influence collision severity.

1. Data

I will be using the data provided to us in week 2. The data comes in a csv file with lots of columns that will not be used in our model. There are few variables(columns) that I believe will influence the model which include the Light Conditions (LIGHTCOND), the Road conditions (ROADCOND), and the Weather (WEATHER). The target column is Severity Code (SEVERITYCODE). Therefore, the first step is going to be cleaning the data and selecting for the aforementioned columns. The next step will be to transform the values in some of the columns to respective integer values. For example, in respect to the column Road conditions, the values are 'Wet', 'Dry', and 'unknown' each of which can be mapped to a respective integer value of 1, 2, and 0. The same sort of methodology can be followed to map the values from Light Conditions, and Weather to numerical values.

A screenshot of a cell phone

Description automatically generated

Figure 2.1 : Chart of initial pandas Data Frame

As we can tell from the first couple rows of the table, it is going to be 38 columns long and clearly all of these columns are not necessary, so we can begin with clearing some of these messy and unnecessary columns.

This can be done by creating an array with the names of the columns that you want to delete. If your table is in the correct data frame format, you can call inbuilt functions to handle this.

Once this step is done your data should look a lot cleaner.

A screenshot of a social media post

Description automatically generated

Figure 2.2 – Columns to be dropped

This looks a lot better. There is a column called UNDERINF, which refers to drivers under the influence. This has values that could take on either a character value of 'Y' and 'N', as well as an integer value that can be 0 or greater. We want to remove rows where this column value is either 'Y', or has a value greater than 0.

A screenshot of a cell phone

Description automatically generated

Figure 2.3 – Drivers under influence are removed from table

Now that we are sure that the entries for drivers in our dataframe were not under the influence, we can go ahead and get rid of the column entirely. Additionally I am going to get rid of another column that has all NaN values.

A screenshot of a cell phone

Description automatically generated

Figure 2.4 – Final version of table

1. Methodology

We need to convert finally the Weather, Road condition, and Light condition to category variables. Once this is done, they are then in a suitable format for training the model. In order to use the scikit library's LabelEncoder, the respective rows need to be rid of empty strings. This can be fixed by turning the empty string into NaN and subsequently the LabelEncoder can be ran upon it.

A screenshot of a social media post

Description automatically generated

Figure 3.1 – Begin Label Encoding

A screenshot of a cell phone

Description automatically generated

Figure 3.2 – Label Encoding complete

Now that we have the severity code in a int64 format, as well as roadcond-cat, lightcond-cat, and weather-cat, we can go ahead and start with the machine learning process.

There are 3 different types of Machine Learning models we will be working with, KNN, Decision Tree, and Logistic Regression.

We will be using KNN as our first algorithm. This will calculate the distance between the k closest values in order to predict similar data points. This comes from the basis that all points in a given region are very similar.

We will use the Decision Tree as another supervised learning algorithm. By starting at the root and looking at predictors such as weather, the road condition, and light condition, one can traverse down the decision tree and calculate a relative probability after taking aforementioned conditions into consideration. This will be a categorical variable decision tree.

Finally, we will use Logistic Regression in order to calculate a discrete probability, and check the accuracy of our model.

A screenshot of a cell phone

Description automatically generated

Figure 3.3 Feature and train and test setup

Here we can see that the train and test classes are being set up. We ae importing the metrics from the sklearn package, as well as train\_test split. After this the data is fed into the model given a training size of 30% and a model testing size of 70%.

We then go ahead and display a couple of values from the resultant array. This concludes most of the preprocessing that needed to occur on the data before it could be in a format that the various algorithms could work with.

--

Now it’s time to work on the data with the KNN machine learning model.

In order to continue working on it we will use the sklearn package and import a K Neighbors Classifier package. Upon doing this, we must peruse possible values of “k” that are going to be fed into the model. The 5 values I ended up trying to work with were at k=9, k=16, ,k=19, k=25, and k=29.

A screenshot of a social media post

Description automatically generated

Figure 3.4 - Displays model with some values of k

A screenshot of a social media post

Description automatically generated

Figure 3.5 – Displays graph of k Neighbors

As we can tell from the graph over here the data was relatively linear across the entirety.

The next step was to attempt to work with a Decision Tree Model. I felt that this was a good model to use because it complemented the type of data that we are working with well, which is collision data. When deciding about making a decision, one can look at ‘Is it light or is it dark’, and then continue down one of either of those paths when looking at weather, road conditions and other necessary features that would greatly influence this model. Because of this nested, and ‘tree-like’ structure of the data and potential to traverse in either direction, I believe that a decision tree model works perfectly in this situation

A screenshot of a social media post

Description automatically generated

Figure 3.6 – Displays model of Decision Tree Classifier

Now that we we have looked at both the K Nearest Neighbors Model as well as the Decision Tree Classifier Model, the next model we are going to be considering is that of Logistic Regression.

There are a number of reasons as to why Logistic Regression works well with the data. The first reason would be that Logistic Regression works with discrete data and provides a tangible result in form of a percentile. The data we are working with has been converted into a categorical format, meaning that it works very well with a Logistic Regression model.

A screenshot of a social media post

Description automatically generated

Figure 3.6 – Displays Model of Logistic Regression

Now that we have the model trained in 3 separate ways, K Nearest Neighbors, Decision Tree, and Logistic Regression, we are ready to move on to the results and gather some valuable metrics from this data.

1. Results

We are now ready to gather some metrics from the data. In order to analyze this data we need to once again import from sklearn, in order to find the jaccard similarity score, the F1 score, as well as the log loss. First lets look at the results of the K nearest neighbors test

A screenshot of a cell phone

Description automatically generated

Figure 4.1 – KNN Results

So we can see here in the 2 sets that there was 73% precision in the training set for X and around 34 percent accuracy in the 2nd set. We also see a f2 score of 0.67 in the first set and a score of 0.39 in the 2nd set. Overall with the K nearest neighbors test we achieved a prediction accuracy of around 57%. I believe that this stems from the large amount of entries in both sets. If there is some balancing enacted I believe that both the precision as well as the overall accuracy would increase. We would also work on selecting a greater number of neighbors. As there are a large amount of entries in the set there are a large number of neighbors so potentially increasing the range of neighbors that we look at to examine distance and similarity could potentially lead to an increase in these values as well.

Next we can go ahead and look at the results of doing the decision Tree.

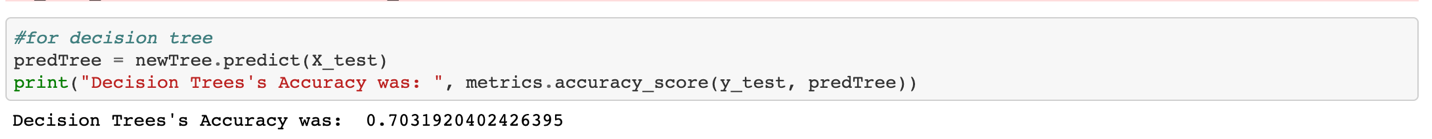


Figure 4.2 – Decision Tree Results

From this we can see that the Decision Tree has an accuracy of about 0.703 which rounds up to about 71 percent. I feel that this is a pretty good accuracy and this definitely benefitted from the large amount of data available. Ways to improve the decision tree accuracy is to add more data that has more information available on road conditions, weather conditions, and light conditions in other regions of the world. By doing this and comparing accidents we can see if the same weather, road, and light conditions in another area potentially has less accidents and use the data to continue to learn and improve the model.

Finally let’s go ahead and look at the results of the data for the Logistic Regression.

A screenshot of a cell phone

Description automatically generated

Figure 4.3 – Logistic Regression Results

Looking at this data, we can see that there was a precision of 70% , and an f1 score of 0.83, which is on the higher side. Because we have a harmonic mean of 83 percent, we can tell that the in this LR model the accuracy was definitely on the higher side, and the numbers definitely support it, with the reported accuracy being 70 percent.

Now with the 3 different models complete, each different form of testing resulted in a different accuracy and precision, and all have been calculated and their data has been displayed above.

1. Discussion

As mentioned earlier in regards to the K nearest neighbors model test we achieved a prediction accuracy of around 57%. I believe that this stems from the large amount of entries in both sets. If there is some balancing enacted I believe that both the precision as well as the overall accuracy would increase. We would also work on selecting a greater number of neighbors. As there are a large amount of entries in the set there are a large number of neighbors so potentially increasing the range of neighbors that we look at to examine distance and similarity could potentially lead to an increase in these values as well. The Decision Tree has an accuracy of about 0.703 which rounds up to about 71 percent. I feel that this is a pretty good accuracy and this definitely benefitted from the large amount of data available. Ways to improve the decision tree accuracy is to add more data that has more information available on road conditions, weather conditions, and light conditions in other regions of the world. By doing this and comparing accidents we can see if the same weather, road, and light conditions in another area potentially has less accidents and use the data to continue to learn and improve the model. Additionally reporting on similar data from different states and counties during different times of the year would help a lot as to gauge the effect with the same weather and light conditions have on the road over years. The LR test was probably the best and most indicative test that was ran on the data. We achieved a harmonic mean of 83 percent showing that accuracy was definitely on the higher side. Because of the very discrete nature of the data as numbers, I felt that LR was the best test method. In order to improve results, one might potentially widen the test set and work on balancing between the training and testing sets.

1. Conclusion

Over the past two weeks I had the opportunity to work with car collision data. I started off with a simple question, whether or not one could grab certain predictor columns in order to check and see if accidents can be prevented by checking a multitude of conditions. After going through various time consuming steps of cleaning the data, the data was transformed via working with numpy and pandas into a format that was suitable for data analysis and training our machine models. After running multiple experiments ranging from KNN, Decision Trees, and LR testing, I was able to conclude that for working with the collision data, LR testing was the most sound method. There is definitely a correlation between certain adverse weather, road, and light conditions, and the combination of these three can prove fatal to unwary drivers. In order to combat this, we trained our models and did discover that there are combinations of weather and road conditions that can prove unsafe paired with light conditions, and with more data in the future over various seasons, and more trained models, we can hopefully predict with more certainty before car accidents occur.