PREDICTING CRASH INTENSITY

Adeet Patel ajp4bs
Srujan Joshi sj7sf
Ruthvik Gajjala rrg5kq

November 23, 2020

1 Abstract

The primary goal of our project is to use machine learning techniques in order to predict crash severity based on a variety of factors at the time of crash. In particular, we will be considering things such as the weather conditions at the time of crash, the time of day, the exact coordinates of the crash, and the light condition to name a few. It will be important for us to select features that actually contribute to the performance of the model and prevent overfitting. The classification problem that we have at hand will be analyzed through a variety of machine learning algorithms and tuning methods in order to find the optimal model. Our secondary goal will be to potentially focus on the UVA campus and surrounding area and creating a model that identifies "high-risk" crash zones in an online-learning fashion. This would be dependent on current factors such as weather, location, and time. The motivation for this project arises from the need to prevent car crashes in high accident areas, and possibly optimizing the response to these events; this would potentially save many lives and is a problem that Virginia faces. Specifically, our goal is to enable first responders and others at scene to be adequately prepared with the necessary equipment.

2 Introduction

2.1 Motivation

Our motivation is to help first responders adequately respond to crash reports by predicting the type and severity of the crash using variables such as crash location, road condition, weather, visibility. This project is an application: specifically we would apply our model to emergency response and preparedness. Understanding the severity of crashes and where they occur could tackle the problem of creating better routes and having safer road infrastructure, including increased signage at these high risk locations to prevent crashes. Another possible application could be to create some sort of app that identifies when a driver is in a high risk zone, allowing them to take appropriate precaution while driving.

2.2 Background Information

Recent trends in road traffic data from the Virginia Department of Transport (`https://www.virginiaroads.org/datasets/virginia-crashes/data`) counter-intuitively suggest an uptick in the number of crash related fatalities in 2020, despite lighter traffic as a result of the COVID-19 pandemic. The State of Virginia has a population of 8.5 million and 7.5 million reg-

istered vehicles, which means that there is nearly one vehicle on the road for every person in the state. This clearly indicates that road safety is a pressing issue.

2.3 Related work

For our project, we have defined our topic as analyzing accident data within the state of Virginia in relation to certain conditions at the incidence, from the hour in which it occurred to the weather conditions. With this research topic in mind, there are other related works that use machine learning models to aid in car accident prevention, and these do so often through the use of road infrastructure and weather data. One particular example of similar research is the analysis of Utah's car accident data in order to map out possible high accident risk areas on regular routes. This project made use of GIS data, similar to what we are using, in order to also determine road curvature and other crucial features that could influence the likelihood of an accident at a certain time.

3 Method

For the purpose of preliminary analysis, we imported the dataset into a Pandas Dataframe Object. To verify that it was properly imported we used the `pd.head()` method to look at the first few rows of the dataset. The raw dataset had 939248 rows and 53 columns. Some of the features were duplicates and had to be dropped. Some of the features were very similar in nature (eg: District and Physical Jurisdiction), so we decided to keep only one representative feature. We also decided to drop columns which wouldn't convey useful information to our model such as such as Crash ID, Crash Date and Document Number. There wasn't much documentation that came with this dataset, and we were unable to ascertain the meaning of some of the columns such as A_People, B_People, C_People, and K_People. The majority of the columns were categorical in nature. This was a challenge since these categorical features were textual and had to be encoded numerically. We printed out the `value_counts()` of each of the categorical features and manually determined whether they should be ordinally encoded or one-hot encoded.

A problem that we faced with this dataset was the fact that there were many missing values. In fact, when we tried to remove any rows with any missing values, we were left with only 30 rows of the original 939248 rows. We decided to get rid of rows with missing values for nominal features, since that made it easier to numerically encode those nominal features. This reduced our dataset to around 90% of the original size: 837061 rows. For numerical features, we dealt with missing values by imputing the medians. Another problem with the dataset was the fact that the there were several different datatypes even amongst the numerical features: floats, ints, Objects, etc.

After numerically encoding the categorical features, the number of columns of our dataset ballooned to 170. As a result of this, we decided to perform Principal Component Analysis in order to reduce the dimensionality of the data. We chose to reduce the features to 60 principal components as this preserved 99% of the variance in the data and at the same time reduced the number of features to nearly one-third the original number.

We decided to frame the problem of predicting crash severity as a classification task. In the dataset, Crash Severity was a categorical variable with the possible values being: (1) PDO (Property Damage Only), (2) Visible Injury, (3) Nonvisible Injury, (4) Severe Injury, and (5) Fatal Injury.

4 Experiments

With our models, we attempted to predict the crash severity (Crash_Severity). This is a categorical variable, which takes on values in the set {"A. Severe Injury", "B. Visible Injury", "C. Nonvisible Injury", "K. Fatal Injury", "PDO. Property Damage Only"}. Because this variable has order (where "PDO. Property Damage Only" is the least severe and "K. Fatal Injury" is the most severe), it is an ordinal categorical variable. Therefore, we used classification algorithms to generate predictions. In addition, because we performed classification, we used metrics such as precision, recall, and F1 score in order to evaluate the model's performance. We also attempted to use neural networks, as we hypothesized that they will be able to learn useful patterns and make powerful predictions, given the large size and overall complexity of the dataset.

In order to make concise the code for data processing and transforming, we used pipelining on the dataset. By calling `fit_transform` and `transform` on the training and testing data, respectively, we could perform all of the encoding and transformations required by simply running only two lines of code. This pipelining can open up possibilities for implementing an online learning model, as the pipeline would be able to transform new data as it came in, and the model's weights could be updated after being trained on the new data. At first, when we tried pipelining, the sklearn code for the pipeline threw an error. We later discovered that the cause of the error was the fact that we had specified the feature `Driverage` as a numerical feature in the `variables` dictionary, but the feature was not purely numerical. This is because some of the training observations contained a list of numerical values separated by semicolons, instead of just containing one numerical value. This led to the `SimpleImputer` failing (as it is impossible to calculate the median of a non-numerical feature), which in turn prevented the pipeline from executing.

We also used the data to generate a heat map of crashes in Virginia. Our primary idea from this visualization was to create a model or use a tactic to label sites which are most prone to crashes. As a baseline we tried using sklearn's SVM classifier, since we had framed the problem of predicted crash intensity as a classification task. Although there are more powerful options available, we figured that SVM would be a good first option due to its simplicity. Even so, when we initially tried running an SVM model on our dataset, it took a very long time to run, around five to seven minutes.

To combat this, we decided to reduce the number of training examples. We decided to filter out the dataset by considering crashes only from the last two years. This reduced the number of rows of our dataset to 130544. This seemed like a logical choice to make since the most recent crashes would probably be more indicative of future crashes due to more up-to date road and vehicle information and traffic trends. We did this as opposed to random sampling of the dataset. After implementing this change, the SVM model still took around two minutes to run.

During our initial runs using SVM, we were getting "perfect" results with Precision, Recall, and Accuracy Scores of 100% for each class. In order to verify that our results weren't skewed towards this particular model, we tried using our data set on other models such as Logistic Regression, Random Forests. These also yielded similar results, with only a slight decrease in Precision and Recall per class for Random Forests. We figured that this might be due to a direct/strong correlation between one of the features and Crash Severity. In order to check this, we tried printing out the correlation of each feature with Crash Severity using the `.corrwith()` function from pandas.

Upon doing so we found out that there was no strong correlation between any single feature and Crash Severity. However, this does not rule out the fact that Crash Severity might have been a direct function of a combination of the other features. With this insight we began removing some features from the data set and rerunning the previously mentioned models on our data set. This made no difference as we continued to get similar results. We then tried using a Linear SVC model on our data set. This time we encountered the opposite of our previous problem. The Linear SVC model had an extremely high bias and predicted the same class for every training example, and gave precision and recall scores of 0 for all but one of the classes. Ultimately, we decided to abandon using SVM models since they take $O(m^3)$ to train, and even after filtering, our data set still had a lot of training examples.

We then tried running a K-nearest neighbors model (with K = 5) on our data set. This gave us more reasonable accuracy, precision and recall scores. We decided to establish this as our baseline model. This model gave more realistic metrics (not trivial values such as 0 or 1). The accuracies on the training set and testing set were about 70% and 59%, respectively. The precisions were about 45% and 25%, the recalls were about 25% and 15%, and the F1 scores were about 30% and 20%.

With these results, we then realized that we were making the model much less powerful by removing features and then using a KNN. We decided to add back the features, as further correlation analysis yielded no significant correlations or glaring reason as to why our performance metrics were initially so high. We then rebuilt our models based all of the viable features. Specifically, we made the final decision of using a Logistic Regression model, which yielded very good results, even in the areas of precision, recall, and F1 scores.

5 Results

Overall:

Metric	Training Set	Testing Set
Accuracy	0.999992	1.000000

Property Damage Only:

Metric	Training Set	Testing Set
Precision	1.000000	1.000000
Recall	1.000000	1.000000
F1 Score	1.000000	1.000000

Nonvisible Injury:

Metric	Training Set	Testing Set
Precision	0.999897	1.000000
Recall	1.000000	1.000000
F1 Score	0.999949	1.000000

Visible Injury:

Metric	Training Set	Testing Set
Precision	1.000000	1.000000
Recall	0.999962	1.000000
F1 Score	0.999981	1.000000

Severe Injury:

Metric	Training Set	Testing Set
Precision	1.000000	1.000000
Recall	1.000000	1.000000
F1 Score	1.000000	1.000000

Fatal Injury:

Metric	Training Set	Testing Set
Precision	1.000000	1.000000
Recall	1.000000	1.000000
F1 Score	1.000000	1.000000

These metrics are for a Logistic Regression model. The model achieved a 100% recall for each of the five crash severities, which means that for each crash severity, the model was able to correctly predict all crashes of that type. We determined that for this problem, we need to prioritize recall over precision, because we want to ensure that for each crash type, we maximize the percentage of correctly predicted crashes of that type. Because we were able to achieve nearly 100% metrics on both the training set and the test set, it means that our model is able to generalize to new crashes, with various crash severities.

GitHub Repository: `www.github.com/adeet1/ml4va`

6 Conclusion

We believe that the implementation of our idea to an advanced degree definitely has the potential to save many Virginian lives. In order to further develop our model we would need to establish a strong relationship with the Virginia Department of Transportation so that we can obtain the latest, unadulterated traffic data and enhance data collection mechanisms. Such a data-driven approach would give us the ability to better analyze and predict the latest trends in road traffic so that we can enable better emergency preparedness and response and also keep Virginians informed about road safety.

Another area of improvement for us is to perform further analysis on the metadata associated with our dataset. Some features are difficult to interpret, and we are not sure how they would apply to our larger goal of building a functioning model. This leads to the natural question of how to encode such categorical variables, and it is worth researching how different encodings can make the performance of our models ultimately differ. Given that the modeling process itself is relatively "black box" in Python, we feel that we should emphasize the data cleaning and understanding step of our process as much as possible to maximize performance in controllable areas of our project. We also found towards the end of our analysis that there was a severe imbalance of classes in the target variable of our dataset, and this most likely had a heavy influence on the training process of the models we used. Before doing anything else, we need to determine how to deal with this imbalance to produce models that can generalize better to the real world. Thus, further data discovery is also something to look into.

Perhaps our biggest next step is the hyperparameter tuning of the models we are focused on implementing. So far we have implemented various different models just to gauge the possibility that we can find an effective way to predict crash severity from the given features in the dataset, and we have shown that the results have been good. However, we deemed that these models were mediocre at completing our intended task in a real world environment, and we discussed various ways that we wanted to make models that perform relatively better than the preliminary ones we have chosen. The main path that we could take is to be able to implement a fully regularized and tuned ensemble of neural networks so that we can assess whether neural networks in general would work out for our project. To do this, examining different loss functions, activation functions, regularization techniques, number of nodes, number of layers, and various other specifics would be necessary. This is an intriguing area of research that we could focus on in the future.

Another aspect of the modeling process we could examine is tuning the lower complexity models we have already implemented, such as SVM, Random Forest, and Logistic Regression. By comparing these models to the deep learning approach, we can determine the appropriate path to take. We could use grid search and other techniques to find alternate hyperparameters for these models with respect to our training data. Following from our discussion of the necessity of dimensionality reduction, feature selection will also be very important for our specific dataset. This is because we know that in realistic circumstances, first responders may not always have access to features such as whether the crash victim was intoxicated, or the crash victims gender. For this purpose, we could also assess which features will best fit the motivations behind the project.

7 References

- Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly.
- Wilson, Daniel. (2018). Using Machine Learning to Predict Car Accident Risk. Medium.

8 Contributions

- Srujan Joshi
  - Researched the metadata associated with the data set
  - Initial data discovery including plotting the distribution and correlation of the features
  - Feature selection including dropping of irrelevant/redundant features
  - Model selection based on precision, recall, and accuracy metrics
- Adeet Patel
  - Cleaned data, removing rows with missing nominal features
  - Created a pipeline for data transformation
    - Transformed categorical features via one-hot encoding and ordinal encoding
    - Transformed numerical features via imputing and standard scaling
    - Ordinally encoded labels (crash severity)
  - Performed principal component analysis (PCA) as a dimensionality reduction technique
  - Trained, tested, and evaluated logistic regression and KNN models
- Ruthvik Gajjala
  - Worked on exploring the available data and created visualizations for the distributions, including scatter matrices and histograms
  - Created geographical map representing where crashes occur in the state of Virginia, and of what severity
  - Attempted to build, compile, and train a neural network on the data
  - Tried various other models, including random forest and SVM