PREDICTING CRASH SEVERITY GIVEN ROUTE CONDITIONS

Marina Kun
mk3qa@virginia.edu

Thrishna Sonnakul
ts8dv@virginia.edu

Charles Lim
charleslim@virginia.edu

April 28, 2020

ABSTRACT

This project hopes to provide insight on improving Virginia's road conditions to reduce the severity of Virginia's car crashes by using multiclass classification models to predict the crash's severity. The initial dataset included data not related to road features, so there was a data cleaning step performed prior to any model development. After processing the data, six different models were run to achieve the highest accuracy score. The final model was a Stacking Classifier that used XGBoost as its final classifier with Logistic regression, AdaBoost, and Decision tree as its base classifiers; it had an accuracy of 66.176%. The second part of the project examined which features were the most significant in predicting crash severity. We extracted the feature importance scores from an XGBoost Classifier. The most important features were the presence of a pedestrian, the presence of a deer, and the lighting conditions of the road.

Project Category: Supervised Classification

1 Introduction

Virginia, especially Northern Virginia, experiences a high volume of traffic that both causes and is caused by crashes. There were 128,172 crashes in Virginia during 2019. The biggest reported crash of 2019 in Virginia included 69 cars colliding on an icy interstate. Because automobile accidents are so prevalent and dangerous, there have been numerous data analysis and data science projects using crash data. Many researchers have used the Virginia Crash dataset from the Virginia Roads website to visualize crashes in different locations around Virginia. Another particular study was performed by analyzing DMV crash data and they used many features to predict crash severity along with multiple machine learning methods. The purpose of their project was to present statistical methods and models to determine factors that caused fatal car crashes and high damage cost. Their methods included classification using Logistic Regression to predict whether an accident was fatal or not. Their methods also included classification using K Nearest Neighbors (KNN) and regression using Lasso regression to analyze the damage cost[7].

Our goal was to provide insight into mitigating the severity of crashes in Virginia by investigating the road conditions that are likely to result in crashes. We took this approach because we hypothesized that road conditions would be a significant factor that contributed to the severity of a crash. We aspired to develop a machine learning classification algorithm that can predict a car crash severity given the road features. The various crash severity classes present in the dataset are: Severe Injury, Visible Injury, Non-visible Injury, Fatal Injury, and Property Damage Only. In order to successfully develop the model using the Virginia Roads Crash dataset, we only used features related to the route conditions. We used multi-class classification models so that we could have the best chance at predicting the correct severity level. Our hope is that analysis of car crash severity can direct the government of Virginia on how to best improve road and route conditions to minimize the severity of car crashes.

2 Method

The purpose of the model is to predict a class of crash severity given the route features. In order to predict one of the five crash severity classes, the overall method for our project was multiclass classification evaluated with an accuracy score. In addition to constructing a multiclass classification model, we analyzed the importance of each route attribute of the given dataset so that we could make interpretations about the features to give us insight on how to better our roads.

The crash dataset was provided by Virginia's Department of Motor Vehicles; it had 66 crash features and 859,972 crash entries. We manually dropped features that did not relate with the road conditions. The features we kept were Carspeedlimit, Latitude, Deer_nodeer, Longitude, Light_Condition, Ped_Nonped, Roadway_Surface_Cond, Speed_Notspeed, Weather_Condition, and Work_Zone_Related. We used Crash_Severity as the label. Once we defined our X attributes and y label, we split the data into a training set and testing set.

There were two parts to our data cleaning and preprocessing stage. First, we filled in any missing numeric values using SimpleImputer's Median strategy which replaces any missing values using the median along each column. Then, we standardized the numeric features by removing the mean and scaling it to unit variance via StandardScaler. For the categorical features, we encoded their values as a one-hot numeric array via OneHotEncoder. Any unknown values were encoded with zeros. Finally, the label values were encoded with values between 0 and 4 via LabelEncoder.

Once the data was cleaned and preprocessed, we began trying multiple implementations of a multiclass classification model. The majority of our data were categorical values, so our heterogeneous features encouraged us to try decision trees first since they are more robust and can work with any kind of data. The first algorithm we tried was Random Forest. We selected Random Forest because it utilizes ensemble learning. This makes the model even more robust, which should allow it to work well despite noisy data. We tuned our Random Forest model using both Grid Search and Random Search to find the best hyperparameters and improve the model's accuracy on the test set predictions.

Then, we steered from ensemble learning and decided to implement a pure Decision Tree algorithm because Random Forest did not perform as well as we expected. By implementing a pure Decision Tree algorithm, we wouldn't have to deal with the stochastic element that comes with Random Forest when selecting features for the tree. Instead, the tree would be built on the entire dataset.

Next, we tried a boosting algorithm, AdaBoost. It is similar to random forest in that it reduces variance by combining the weighted sum of its classifiers. However, it is better than Random Forest in that it fits the tree to the entire training set.

We tried improving our boosting approach by implementing a model that is better fit to real world data, so we implemented XGBoost. XGBoost works better than AdaBoost with complex and high-dimensional problems, which is similar to our problem. We also tuned our XGBoost model in order to try to improve its accuracy.

A Logistic Regression model was next. We included this because we thought it would help in the next model, which uses multiple models together to predict the severity class. It performed as well as the AdaBoost and Decision Tree models.

Because our models had not performed as well as we had hoped, we aimed to combine the well-performing classifiers that were previously implemented to further improve the accuracy. A Stacking model usually performs better than its individual classifiers, so we thought this approach would help us in slightly improving the accuracy score. The base classifiers of the model included Logistic Regression, Adaboost, and Decision tree. The final classifier of the Stacking model was selected to be XGBoost because it performed the best individually out of the multiple models we tried.

3 Experiments

The first experiment was a RandomForestClassifier model tuned with RandomSearchCV; it was used to classify the five types of crash severity based on the selected route condition features. Using RandomSearchCV we determined the value for the number of trees (n_estimators), the number of features to consider when looking for the best split (max_features), and the function to measure the quality of a split (criterion). RandomSearchCV was fit using only 3000 data points of the training data because of the large dimension size and the duration of the training process. RandomSearchCV selected 'auto' as the best value for max_features, meaning that the number of features per split should be the square root of the total number of features. It chose 200 n_estimators and Gini impurity for criterion. We instantiated a RandomForestClassifier model based on the parameters selected by RandomSearchCV and trained the model on the entire training data set. Once the training process concluded, we ran the model on our testing set and determined the accuracy score of its predictions.

The second experiment was another RandomForestClassifier model. However, this model was tuned with GridSearchCV. The parameters for GridSearchCV were n_estimators, max_features, and criterion. GridSearchCV was run on the same 3000 values as RandomSearchCV was run on to save time. The best parameters were entropy for criterion, auto for max_features, and 100 for n_estimators. Using these parameters, a RandomForestClassifier was created and fitted on the training set. Lastly, we ran the classifier on our test set and calculated its accuracy score.

The third experiment was a DecisionTreeClassifier model. It was used to fit our training set with max_depth parameters of 10 and 5. Once the model was fitted, we predicted the labels on the testing set using the new model. Utilizing the predicted labels and actual labels, we observed the resulting confusion matrix and accuracy score between the different depth models to determine the most accurate model.

For the fourth experiment, we began exploring boosting algorithms by implementing an AdaBoostClassifier. We initialized the classifier with 50 as the maximum number of estimators at which boosting is terminated (n_estimators) and 1 as the learning rate (learning_rate). Then, we trained the classifier on the entire training set and predicted on the testing set. Finally, we calculated the accuracy score of its predictions.

The fifth experiment was another boosting algorithm, XGBoost. Initially, we instantiated the XGBClassifier from the package xgboost. We specified the learning objective as 'multi:softmax' to specify the problem as multi-classification. Then, we trained the model on the training set and predicted on the test set. We computed the accuracy of its predictions. We tried to improve its accuracy by tuning its hyperparameters using RandomSearchCV. We tuned the number of trees (n_estimators), the maximum depth of a tree (max_depth), the subsample ratio of columns when constructing each tree (colsample_bytree), the ratio of the training instances to prevent overfitting (subsample), and the learning rate (learning_rate). The score used for RandomSearchCV was the f1_score. Once the model was trained on the entire training data, RandomSearchCV determined the best hyperparameters to be 50 for n_estimators, 9 for max_depth, .2 for colsample_bytree, .6 for subsample, and 1 for learning_rate. Then, the model was used to predict on the testing data and its accuracy was computed. Finally, each route feature's significance was analyzed via the XGBClassifier's feature_importances_ attribute which returned the importance score of each attribute. We aligned these scores with the attribute name and determined which attributes significantly contributed to the prediction of the crash severity.

The sixth experiment was a Logistic Regression model. The LogisticRegression model was instantiated with the default parameters, in which C was 1 and the L2 norm was used. Once the model was training on the entire training set, we used its score function to return the mean accuracy on the test set.

The seventh and final experiment was a stacking ensemble, StackingClassifier from the sklearn package. Its base estimators were LogisticRegression with default parameters, DecisionTreeClassifier with a max_depth of 10, and AdaboostClassifier with 50 n_estimators and a learning_rate of 1. The final estimator was XGBClassifier with multi:softmax as the objective. The StackingClassifier model was trained on the entire training set. Then the model predicted on the test set. We evaluated the predictions by computing an accuracy score.

4 Results

Once we ran all the models stated above in the experimentation section, we analyzed and compared the accuracy score of each model on the test set. First, the RandomForestClassifier model tuned with RandomSearchCV resulted in an accuracy of 0.568871. Next the same type of model was tuned to retrieve the ideal parameters using GridSearchCV had an accuracy of 0.5694. The insignificant difference between tuning models with RandomSearchCV and GridSearchCV may be due to the fact that we trained both models on a small portion of the training set. In addition to limited training, the range of the parameter options was relatively small. A DecisionTreeClassifier model with a depth of 10 had an accuracy of 0.6611, and it was observed that adjusting the depth to 5 and to 15 only changed the accuracy with a 0.005 difference. Next, we shifted to boosting models; the AdaBoostClassifier provided an accuracy of 0.66118. The following XGBoost method was implemented using XGBClassifier that gave an accuracy of 0.66123. However, when we attempted to tune the parameters for the model using RandomSearchCV the accuracy decreased slightly to 0.65901. The mean accuracy of the Logistic Regression model was 0.66102. The last model we ran utilizing the stacking method of a logistic regression model, decision tree model, and XGBoostClassifier model, and this last model provided the highest accuracy of 0.66176. The final model selected was the StackingClassifier as it yielded the highest accuracy on the test set.

Another way we performed data analysis of the dataset was by observing the best features. This was done using the tuned XGBoostClassifier model and observing its feature importance scores. Using these scores, we grouped the attributes based on the category they belonged to prior to the encoding step. Then, we summed the scores of the attributes for each category. The pie chart below represents the result of this process. The total feature importance score for Ped_NonPed is 52.0% making it the most significant feature that contributes to predicting the crash severity.

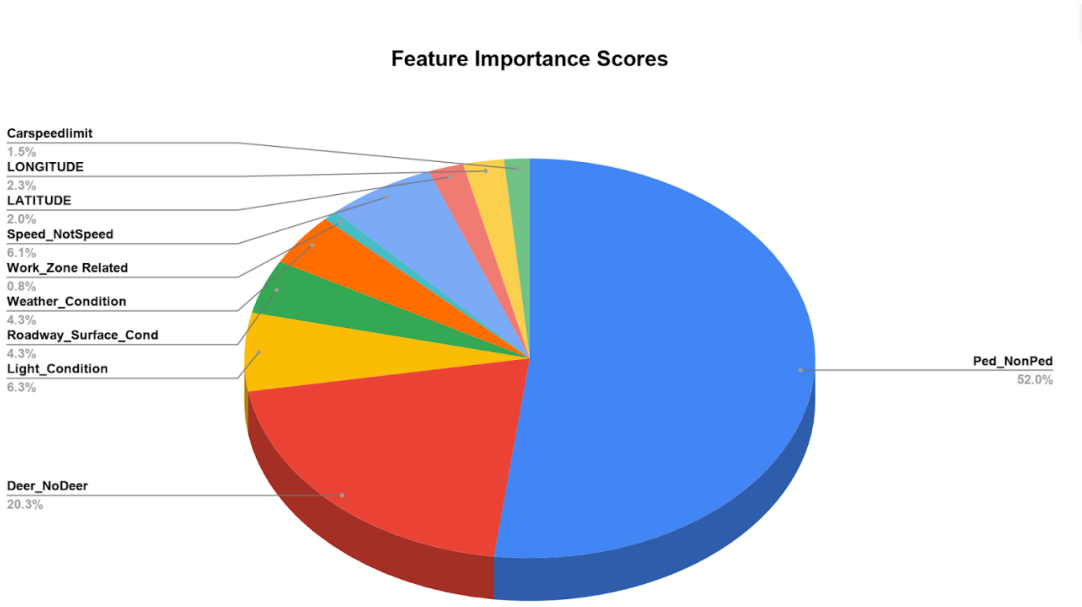

Figure 1: Feature importance distribution chart

The second most significant feature is Deer_NoDeer with 20.3%. The other feature scores are 6.3%, 6.1%, 4.3%, 4.3%, 2.3%, 2.0%, 1.5% and .8% for Light_Condition, Speed_notSpeed, Roadway_Surface_Cond, Weather_Condition, LONGITUDE, LATITUDE, Carspeedlimit, and Work_Zone_Related respectively. This observation explains that the top three most significant features are the presence and absence of pedestrians, the presence and absence of animals, and the light condition of the roads.

The following links contain the code for data cleaning and preprocessing a, model implementations, and further data analysis of route features:

https://colab.research.google.com/drive/1MkOSXd2lN7VCpGDTCW2W3AG7aTCj0JYT
https://colab.research.google.com/drive/1wbH8xZoBz5vhc2ZupJVxsFLHw7yFNIgA
https://colab.research.google.com/drive/11O2W4UxNhxUyZTFmaMCmNLoh3O8IuCgL

5 Conclusion

Overall, our models were somewhat successful in accomplishing the goal we set in the beginning. We had multiple models that hovered around 66% accuracy, which implies that our results support the hypothesis that road conditions significantly contribute to how severe a crash turns out to be. However, it is important to note that because our models were not stellar, road conditions are not the only contributing factors towards crash severity. Because we ended up having multiple models perform well but not amazingly, we hoped that by stacking the models together the stacked model would be able to predict better for the entire dataset as a whole. However, the stacked model did not perform significantly better than its individual base classifiers. This shows that the base classifiers implemented seem to predict accurately for similar subsets of the data. This implies that while the Virginia government should be concerned with its road conditions, other aspects to safe driving should not be ignored.

Our feature analysis showed that the presence of a pedestrian was the biggest contributing factor for crash severity, which implies that further work can be done to make the presence of vehicles safer around pedestrians. In hindsight, this makes sense because if there was a pedestrian involved in the crash, then risk of severe injury is high. There is no specific recommendation that we can give because we did not perform analysis on that, but some possibilities would include lowering the speed limit or putting additional barriers between pedestrians and vehicles. The next most important was the presence of a deer. This is also understandable because when a vehicle collides with a deer,

the vehicle can be severely damaged. Perhaps lawmakers should attempt to tackle the problem of deer in our roads more seriously. The third most important feature was the lighting conditions of the road. Because limited visibility is dangerous, the government should consider lighting our streets with lamps and such more.

One fallback during our procedure was that we did not reduce the size of our data. Our rationale was that by including all the data, we could ensure that we do not miss any aspect of the data in our models. However, this became a significant problem when we realized that we were not able to run a neural network on such a large dataset because we do not have world class computing power. Another limitation of our project was the extent of our hyperparameter tuning. We could have tried more extensive ranges and increased the number of folds for cross validation. However, we would be sacrificing the runtime for a better hyperparameter tuning. We also should have tuned every model we implemented. By improving on our limitations, we may have seen slightly better accuracy scores. Other future work may include separating the data between rural and urban locations to determine how these location features may result in different findings for improving crash severity.

References

[1] Caponetto, Guilherme. "Random Search vs Grid Search for Hyperparameter Optimization." Medium, Towards Data Science, 14 Nov. 2019, https://towardsdatascience.com/random-search-vs-grid-search-for-hyperparameter-optimization-345e1422899d.

[2] Ceballos, Frank. "Stacking Classifiers for Higher Predictive Performance." Medium, Towards Data Science, 14 Sept. 2019, https://towardsdatascience.com/stacking-classifiers-for-higher-predictive-performance-566f963e4840.

[3] Liu, Cheng-Hsiang. "Classification with Mixed Numeric and Categorical Data Using Improved Extension Theory ." Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, 13 Mar. 20136, http://www.iaeng.org/publication/IMECS2013/IMECS2013$_p p337 - 340.pdf$.

[4] Morde, Vishal. "XGBoost Algorithm: Long May She Reign!" Medium, Towards Data Science, 8 Apr. 2019, https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.

[5] Nguyen LH, Holmes S (2019) Ten quick tips for effective dimensionality reduction. PLoS Comput Biol 15(6): e1006907. https://doi.org/10.1371/journal.pcbi.1006907.

[6] United States, Congress, "2019 Virginia Traffic Crash Facts." 2019 Virginia Traffic Crash Facts, Department of Motor Vehicles. https://www.dmv.virginia.gov/safety/crash$_d ata/crash_f acts/crash_f acts_1 9.pdf$

[7] W. Tong, P. Cherian, J. Liu, H. Li and Q. Gu, "Statistical analysis of DMV crash data," 2016 IEEE Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, 2016, pp. 113-117.

6 Contributions

Marina - Marina worked on the data cleaning and preprocessing stage. She implemented several experiments including RandomForest tuned with RandomSearchCV, KNN (although it is not included in the report), XGBoost, Logistic Regression, Adaboost, and stacking. She also analyzed the feature importance scores. In addition, she attended group meetings to pair-program with her teammates and discuss further implementations. Lastly, she worked on the final report.

Charles - Charles set up a lot of things in the Google Colab like finding the best way to get the data accessible, and he also ran some of the tests like random forest on the entire dataset. He attended meetings to work together and discuss how we should approach the project in general. He tried to include some features back into the model after realizing that they might be helpful, but they ended up not being helpful so it was not included. He also read over and edited the report before transferring to LaTeX and submitting.

Thrishna - Thrishna attended multiple group meetings, in which she helped decide extra features to be dropped, planning next model methods, and writing the final report. She helped by recommending the method to set crash severity as the label and encode it. Also, peer programmed with a group member to make the data compatible with the classifier models. She developed the RandomForestClassifier tuned with GridSearchCV, DecisionTreeClassifier, and KMeans for visualization. Lastly, she helped with the checkpoint report and the ending analysis of the models to write the final report.