Introduction:

The goal of this project was to determine whether the features age, whether the driver was speeding, whether the person was impaired, the type of vehicle the person was in, whether the individual was a passenger, driver, etc., and what state the driver was from could be used to predict whether the person would experience a major injury or fatality. This was a binary classification problem, with a positive event being that the individual did experience a major injury or fatality and the negative event being that the individual experienced a minor injury.

Description of Individual Work:

Originally, each of us worked on each component of the project; EDA, preprocessing, modeling. However, once it was realized that there was a problem with using merged data (replicated rows) and decided to use only one data set, we had to restart. Upon restarting, Arianna worked on the EDA and did the models Random Forest and Logistic Regression. Lydia worked on making the script to read in the data and eliminate the identifiers as well as the GUI. I did the preprocessing, statistics, and the models Naïve Bayes, XGBoost, and Voting Classifier. Due to some of our earlier work, however, there are a few lines in the readdata.py, eda.py, and preprocessing.py scripts that have codes from one another.

For preprocessing, I loaded the data that Lydia had already modified a bit – the data where the target was already transformed from the three columns of 'MAJORINJURY', 'MINORINJURY', and 'FATALITY' to 'MAJORFATALINJURY'. Additionally, Lydia had removed all columns that were identifiers such as 'VEHICLEID', 'CNN', and 'CRIMEID'. Lydia also wrote the code that removed non-sensible license plates (non-state or territories). Additionally, Arianna wrote the code that cleaned out the ages that were negative or over 100. I was responsible for removing the drivers under the age of 10 (though Arianna did a similar code to work on the EDA), creating the matrix of only features and target, filling in missing data, normalizing the quantitative variables, and doing a label encoder on the data.

I wrote the entire statistical script myself. This included a Chi-Square test for independence on each of the categorical features and the target. I used researchpy as it returned the Cramer's V value, unlike scipy which allowed for easier comparison of the strength of the relationship between each categorical feature and the target. For the quantitative variable, 'AGE', I performed a *t*-test to compare the mean age of those experiencing a major injury or fatality to the mean age of those who only experienced minor injuries. This test arose from the question of whether it tended to be younger children or potentially older individuals who experienced more dramatic injuries during a car crash. Prior to the t-test, the PDFs of the two groups were also plotted, along with calculation of the means and standard deviations of the distributions.

Arianna and I split up doing the models. I took XGBoost and Naïve Bayes, she took Random Forest and Logistic Regression. I originally wrote the structure of how we would write the models (written as a class) and then she followed suit using a similar design. Upon deciding to see if using a Voting Classifier would increase the accuracy, I then wrote that script. Each of us worked on the powerpoint presentation. I made the preprocessing, statistics, XGBoost, Naïve Bayes, Voting Classifier, and conclusion slides. In regard to the paper, we again split it up to the portion that each of us worked us. Therefore, I wrote the preprocessing, statistics, description of the 3 models

I wrote, results of the 3 models I wrote, and then some of the conclusion. We all edited each other's portions.

Note: We all worked pretty well together. There was a lot of discussion and collaboration. We edited each others components, looked over each others codes, and discussed all issues with one another.

Results:

Upon optimization, the models each scored an AUC between 0.7 and 0.8. Because of the similarity, additional measures were evaluated. The specificity and sensitivity of the models were calculated and showed that despite Naïve Bayes having an AUC of 0.7, the sensitivity of the model was less than 1%, indicating that it is unable to discern a positive instance. Therefore, it is a very poot model. Logistic Regression, Random Forest, and XGBoost all scored very similar AUCs, as well as similar values for sensitivity and specificity, with XGBoost being the highest. The table below shows the evaluation results of the testing data using each of the models.

Though this level of accuracy, sensitivity, and specificity indicate that the models Random Forest, XGBoost, and Logistic Regression perform better than randomly guessing the outcome, it was still the hope that there could be an improvement to the models, even after optimization. Therefore, a voting classifier using the three of the four models (the three best; logistic regression, random forest, and XGBoost) was designed. However, there was not much of improvement, if any, in the accuracy using either the soft or hard voting methods. Voting Classifier is unable to calculate the AUC and therefore the overall classification accuracy had to be used instead, along with the sensitivity and specificity. Since classification accuracy can be misleading in sets where there is a class imbalance, we had to be wary relying on overall accuracy. The Voting Classifier did have a higher classification accuracy than XGBoost but much lower than Naïve Bayes, which we already determined was a bad model due to its inability to identify positive cases (sensitivity = 17%). Additionally, Voting Classifier had a slightly higher specificity than XGBoost, but lower sensitivity. Ultimately, Voting Classifier did not substantially increase the accuracy of the model. Ultimately, there was not one best classifier, as XGBoost has the highest AUC, Random Forest had the highest sensitivity, and Voting Classifier had the highest Specificity.

	Naïve Bayes	XGBoost	Random Forest	Logistic Regression	Voting Classifier
Overall Accuracy	95.5%	74.8%	74.9%	77.3%	76.3%
AUC	0.70	0.768	0.74	0.664	---
Sensitivity	17.0%	66.7%	74.7%	74.3%	61.9%
Specificity	97.5%	75.1%	63.5%	48.7%	76.8%

Both Random Forest and XGboost evaluated feature importance when constructing the models. Surprisingly, the results were vastly different. The most important features in Random Forest was found to be age followed by vehicle type while the most important features in XGBoost were found to be whether the person was given a ticket then followed by person type (driver, passenger, etc.).

The final finding of this project was an analysis of parameter optimization. Though a thorough parameter optimization was not conducted for this project, parameters were changed and evaluated to find better fitting models. For example, a low accuracy was calculated when the forests in Random Forest or XGBoost were used (forests smaller than 50) or if the learning rates were too large. However, by far the most important parameter was setting the class weights, due to the high class imbalance. In XGboost, this was optimized by calculating to what degree the class imbalance occurred and then changing the 'scale_pos_weight' parameter from the default of 1 to 25. In Random Forest and Logistic Regression, the 'class_weight' parameter was changed to 'balanced'. This enabled the algorithms to go from sensitivities of less than 1% to up in the 70's.

Summary and Conclusions:

The best model was XGBoost, with an AUC of 0.768 and a sensitivity of 66.7%. However, it should be noted that although XGBoost has the highest AUC and sensitivity, it is the slowest to run with a cross-validation. Adding in the Voting Classifier did not significantly increase the accuracy of the models. If was found that because of the class imbalance, the models only performed well when the parameters addressing class imbalance were optimized. Additionally, because of the class imbalance, using the classification accuracy as the evaluation metric was quite misleading. Therefore, metrics such as AUC, sensitivity, and specificity were used. Ultimately, the models, other than Naïve Bayes, classify whether an individual will experience a major injury or fatality better than a random guess does using the defined features.

Code Calculations:

Total Lines of Code: 40 + 45 + 56 + 39 + 36 = 216
Lines of Code Written Myself: 24 + 30 + 9 + 12 + 10 = 85
Lines of Code Copied: 16 + 15 + 47 + 27 + 26 = 131
Lines of Copied Code that was Modified: 16 + 15 + 10 + 9 + 9 = 59
Calculation: (131-59)/(131+85) = 72/216= 0.333 * 100% = 33.3% copied

References I Used Writing My Portion of the Paper:

1. James, G., Witten, D., Hastie, T., Tibshirani, R., *An Introduction to Statistical Learning - with Applications in R.* New York: Springer, 2013.
2. "ML Voting Classifier Using Sklearn". *GeeksforGeeks.* Nov 25, 2019. [Online]. https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/
3. Navlani, Avalash. "Naive Bayes Classificaion Using Scikit-Learn". *DataCamp.* Dec. 4, 2018. [Online]. https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn?utm_source=adwords_ppc&utm_campaignid=1565261270&utm_adgroupid=67750485268&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_ad

postion=&utm_creative=332661264374&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9007810&gclid=Cj0KCQjwvYSEBhDjARIsAJMn0lj1DfpdDWQ5NbCTjk8GlsSJ21kKd8WcdrU5FLhU1Yy7NYkOM3vHUikaAuUREALw_wcB

4. Pathak, Manish. "Using XGBoost in Python". *DataCamp.* Nov. 8, 2018. [Online]. https://www.datacamp.com/community/tutorials/xgboost-in-python
5. Rosner, B. (2015). *Fundamentals of Biostatistics* (8th ed.). Boston, MA: Cengage Learning.
   Ott, R. L., and Longnecker, M. (2010). *An introduction to statistical methods and data analysis.* Belmon, CA: Brooks/Cole.
6. "VotingClassifier". *Sklearn.* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html