Intro to Data Mining

Final Project

Spring 2021

What Factors Influence the Occurrence of Major Injuries and Fatalities incurred during a DC Traffic Accident?

Group 3

Arianna Dunham, RyeAnne Ricker, Lydia Teinfalt

Updated: 4/30/2021

Introduction

As we start to look at the post-pandemic world and more Washingtonians are returning to commute into DC area again, our group was interested in using data mining to understand factors that contribute to car crashes severity. There are 4.1 million cars registered in DC and surrounding Virginia and Maryland areas (TPB News). With more cars coming back to the roads come more accidents. A report in 2010 by the NHTSA stated that the total economic cost of motor vehicle crashes in the United States was $242 billion (NHTSA). Additionally, the city of D.C. has implemented an initiative called Vision Zero where it plans to reach zero fatalities and serious injuries caused by traffic accidents in the year 2024 (DDOT). This is an ambitious goal, and our findings may help determine if the goal is reasonable by determining some causes fatalities in accidents.

Description of Data Set

The data set —titled *Crashes Details Table* — was downloaded from Open Data D.C. Government website on March 11, 2021. The website indicated that the data was last updated eight months ago on August 12, 2020. The dataset is derived from the crash data management system (COBALT) managed by The District Department of Transportation (DDOT) and The Metropolitan Police Department (MPD). The table provides details for individuals involved in all documented crashes including the age of those involved, the type of vehicle, the state the vehicle is registered in, if a ticket was issued, and more.

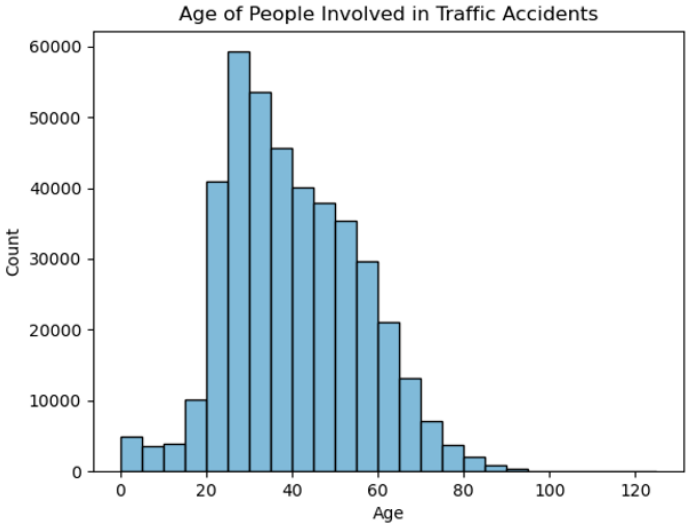
Table 1 Data Sample of Crashes Details DC from March 2020

OBJECTID	CRIMEID	CCN	PERSONID	PERSONTY	AGE	FATAL	MAJORINJ	MINO	VEHICLEID	INVEHICLETYPE	TICKETISSI	LICENSEPL	IMPAIRED	SPEEDING
430455865	27615913	18042044	86838139	Driver	49	N	N	N	3766128	Large/heavy Truc	N	MD	N	N
430455866	27615913	18042044	86838245	Driver	59	N	N	Y	3766126	Passenger Car/at	Y	VA	N	N
430455867	27615913	18042044	86836893	Driver	61	N	N	N	3766127	Bus	N	PA	N	N
430455868	26873834	16035157	84968953	Driver	28	N	N	Y	2277107	Passenger Car/at	Y	VA	N	N
430455869	26873834	16035157	84921236	Passenger	33	N	N	N	2277107	Passenger Car/at	N	VA	N	N
430455870	26873834	16035157	84748308	Driver	63	N	N	N	2277106	Passenger Car/at	Y	DC	N	N
430455871	26873836	16035159	84962811	Driver	37	N	N	N	2277098	Passenger Car/at	Y	DC	N	N
430455872	26873836	16035159	84570868	Driver	45	N	N	N	2277099	Other Vehicle	Y	None	N	N
430455873	26873838	16035120	84584071	Driver		N	N	N	2277108	Passenger Car/at	N	DC	N	N
430455874	26873838	16035120	84936111	Driver	67	N	N	N	2277108	Passenger Car/at	N	DC	N	N
430455875	26873846	16035140	84956752	Driver	35	N	N	N	2277103	Passenger Car/at	N	MD	N	N

There are 599,670 observations and 15 columns in the dataset. The CRIMEID is a unique identifier assigned to a crash. The PERSONID is a unique identifier of the person(s) in the crash. Multiple people and vehicles can be involved in a single crash. In that case, you will see the same CRIMEID repeated but with unique PERSONID information. If an individual was involved in more than one crash, the PERSONID appears multiple times. The age is associated with each person identified in the traffic accident. There can be multiple individuals involved in a crash with their four modes of transport to be driver, passenger, pedestrian, and bicyclist. FATAL, MAJORINJURY, MINORINJURY, TICKETISSUED, IMPAIRED, and SPEEDING are binary columns with possible values being 'Y' for YES and 'N' for NO. INVEHICLETYPE represents the type of vehicle involved in a crash associated with PERSONID. If a vehicle, the LICENSEPLATESTATE contains a state abbreviation where the license plate was issued. We added a column FATALMAJORINJURY which was set to the value of '1' if either of the columns FATAL or MAJORINJURY had the value of 'Y', otherwise the default value is '0'. The target column is FATALMAJORINJURY and 21,821 rows or 3.64% of the data contain a value of '1'.

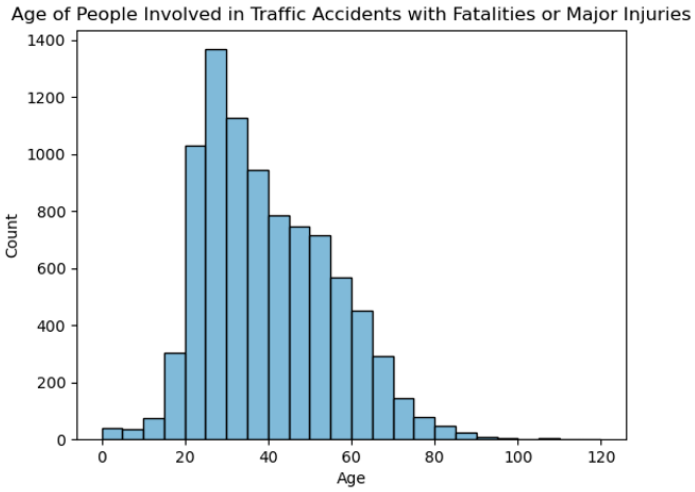
Exploratory Data Analysis

Before exploring our data, we hypothesized that age, vehicle type, and traffic violations (impairment, speeding, ticket issued) would be significant indicators of major injuries and fatalities in traffic accidents. We predicted that the presence of children or senior citizens, larger vehicles, and the presence of traffic violations would be indicators of sustained major injuries and/or fatalities.



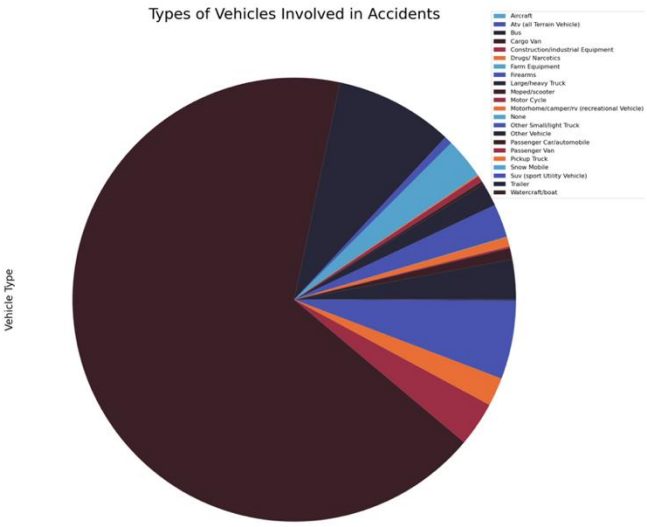
We began the exploration with the AGE column, our only quantitative variable. This column describes the age of each individual involved in the crash. After reviewing the summary statistics, it was apparent that this column needed some cleaning. The minimum value was -7990 and the maximum was 237. Additionally, we noticed a few instances where some individuals were listed as drivers and had a reported age of zero. We know that this is not possible. So, we noted that this column needed to be cleaned in pre-processing by dropping rows with unnatural ages and rows that described an outrageously young driver.

After determining what cleaning needed to be done in this column, we found that the average age of individuals involved in all accidents was 37. As seen in the figure above, a majority of people were between the age of 20 and 40. The average age of those who sustained major injuries and/or fatalities was slightly older at 39. The age distribution of these accidents (shown to the right) is similar to the age distribution of all accidents. This is different from what we hypothesized. As mentioned earlier, we predicted that children and seniors would be more likely to sustain major injuries or fatalities.



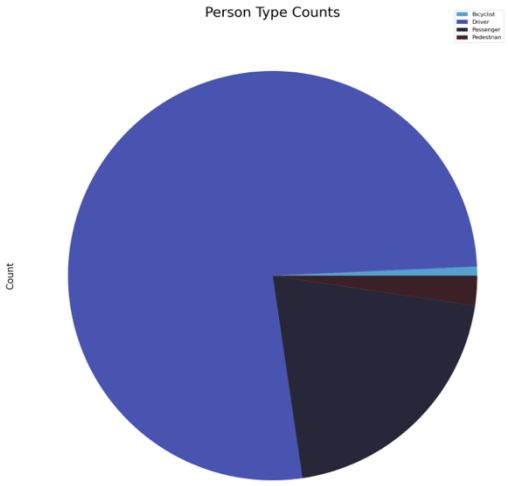
Next, we reviewed the LICENSEPLATESTATE column. This column indicates the state that each vehicle is registered in. At first glance, we noticed that there were several abbreviations that we could not identify with any U.S. state or non-U.S. country. These abbreviations included: Ot, Ou, Vi, Pu, Un, Am, and Di. Since we could not identify these, we noted that rows with these abbreviations should be dropped.

When reviewing this column, we were interested in the number of vehicles that were outside of the DMV area (District of Columbia, Maryland, and Virginia), as these may indicate drivers who are not familiar with the area. About 21 percent (125, 949 out of 594,542 total crashes) of all vehicles in the dataset were



registered outside of the DMV. Of the accidents that resulted in a fatality or major injury, about 35 percent (7,596 out of 21,745 crashes) involved a vehicle registered outside of the DMV.

After looking at license plates, we explored PERSONID. This column provides an ID for individuals within the MPD system. This means that if there are duplicates of this value, that indicates that the same individual was involved in multiple accidents. We checked for duplicates in this row and found a total of eight individuals who were involved in multiple accidents, an insignificant number. All eight of these individuals were listed as drivers and they each sustained minor injuries in their accidents. Since nearly all these values are unique, we predict that they will not have a large impact on our target.

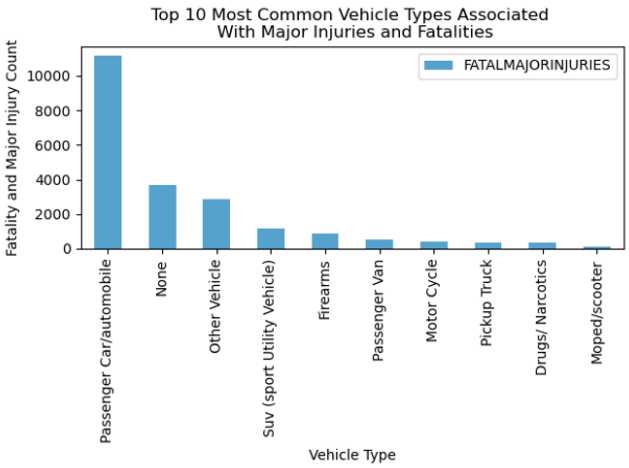


Following PERSONID, we reviewed PERSON TYPE. This column describes each individual as one of four possible road users: driver, passenger, pedestrian, or bicyclist. As shown in the figure to the left, we found that driver was the most common type, followed by passenger, pedestrian, and, finally, bicyclist.

Drivers also suffered the most minor injuries as well as the most major injuries and fatalities. They were followed by passengers, pedestrians, and, finally, bicyclists in both minor injuries and major injuries/fatalities.

Then, we assessed traffic violations, which include the columns SPEEDING, IMPAIRMENT, and TICKETISSUED. These columns indicate if an individual was speeding, impaired, or received a ticket at the time of the crash. We found that most accidents did not involve any of these. We also found this to be true for accidents that resulted in minor injuries and accidents involving major injuries and/or fatalities. Additionally, there were incredibly weak correlations between these incidents and accidents resulting in major injuries and fatalities. As mentioned above, before looking into the data, we had hypothesized that speeding, impairment, and ticket issued would have a larger impact on the injuries sustained in an accident.

Finally, we looked at the VEHICLEINTYPE column. This column describes the type of vehicle that each individual was in during the crash. As shown in the chart to the left, there are a total of 22 vehicle types, including “None.” This vehicle type is used to describe pedestrians and bicyclists. The three most common vehicle types are: Passenger Car, Other Vehicle, and SUV.



As shown in the chart "Top 10 Most Common Vehicle Types Associated with Major Injuries and Fatalities", we found that the top 10 vehicle types most frequently involved in accidents resulting in major injuries and/or fatalities were Passenger Car, None, Other Vehicle, SUV, Firearms, Passenger Van, Motorcycle, Pickup Truck, Drugs/Narcotics, and Moped/Scooter. It is expected that the vehicle type None would be included in the top ten most dangerous, as that description depicts pedestrians and bicyclists, who are highly vulnerable to major injuries and fatalities when in an accident with a motor vehicle. The other two vehicle types in the top ten that stand out are Firearms and Drugs/Narcotics.

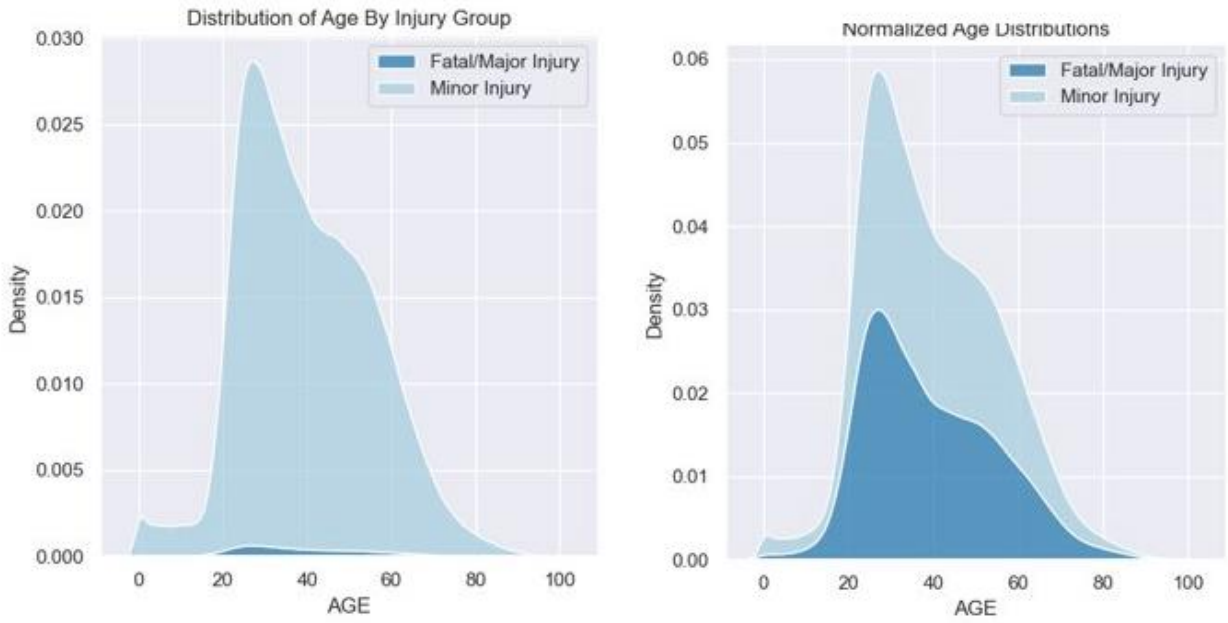
After completing our exploratory data analysis, we began to hypothesize that vehicle type would be one of the most important features in our models. Age and traffic violations appeared to have a smaller effect on the injuries sustained than we originally hypothesized. We also discovered that the license plate state might have more influence than originally hypothesized.

Statistics

Six out of seven of the features are categorical features, as well as the target. Therefore, to investigate the relationship between each of these features and the target, a Chi-Square Test of Independence was conducted. As Rosner discusses, a Cramer's V score of greater than 0.10 is considered a moderate relationship, greater than 0.15 is considered strong, and greater than 0.25 is very strong. The results of the chi-squared tests are shown in the table below and indicate that vehicle type and person type are both strongly related to whether some incurs a major injury or fatality and that the state someone is from (license plate state) is moderately related. Potentially, the association between vehicle type and injury type could indicate that those in smaller vehicles are more likely to experience major injuries or fatalities because their vehicles are not able to provide as much protection. Or, that those from out of state are less familiar with D.C. driving and therefore may experience major injuries or fatalities at a higher rate. These questions could be interesting to explore further in another investigation.

	Fatal/Major Injury Occurrence
Speeding	0.02
Ticket Issued	0.07
Vehicle Type	0.18
License Plate State	0.10
Impaired	0.01
Person Type	0.18

Age is the only quantitative variable. Given that the over-arching question of this project is to determine whether we could predict with *individuals* in a crash would experience a major injury or fatality, it seemed pertinent to look at whether there was a difference in ages between the group of those with major injuries and fatalities and those that only acquired minor injuries. To do this, the PDFs of these groups were first examined. Due to the high class imbalance, the PDFs were also normalized. The non-normalized and normalized PDFs can be seen below. Once normalized, the PDFs appeared quite similar. The mean ages of those who experienced a major injury or fatality was 39.3 years of age and 39.7 years for those obtaining only a minor injury. Then, a Welch's *t*-test was performed to determine whether there was a significant difference between the ages of those who experienced major injury or fatality and the ages of those who acquired a minor injury. A Welch's was specifically used as we did not want to assume an equal variance. Surprisingly, the result of the *t*-test was a p-value of 0.014, resulting in a significant difference in the mean ages of the groups with an alpha value of 0.05. However, if the confidence level was instead increased to 99%--or an alpha of 0.01--the results would no longer be considered significant.



Experimental Setup

Preprocessing

The first step of the preprocessing was to clean up the data. The exploratory data analysis revealed that the AGE column listed values that did not make sense. For instance, there were negative valued ages and multiple people in the 120's. All rows with negative ages were removed, as a negative age is an impossibility. There were 70 rows with ages less than 70. Though people can live over the age of 100, it is uncommon and ultimately it was decided to remove rows where people exceeded age 100 due to the concern that the date was erroneous. 276 rows had ages listed above 100. It was also decided to remove any rows in which the Driver was listed below the age of 10. This cutoff was decided upon as, what we felt, was the youngest age likely to be driving a car. This dropped 11,230 rows. The last step in cleaning out the data was to remove any row in which the license plate state listed was not one of the 50 states or US territories. This resulted in dropping 12,488 rows. In total, 24,064 rows were removed during cleaning.

The next step in the preprocessing was to eliminate the columns in which would not be used as features or targets. The identifier rows were eliminated. This included OBJECTID, CRASHID and PERSONID, CCN,

and VEHICLEID. Additionally, as the data mining question was what features could predict whether an individual in a crash would experience a major injury or fatality, the columns FATAL, MINORINJURY, and MAJORINJURY were combined into one column with the value '1' placed into it if either FATAL or MAJORINJURY occurred and a '0' if not.

Missing values in the data were evaluated next. 333 rows were found to be completely empty, and therefore removed. After these rows were removed, the only columns found to have missing values were in the AGE column. After removing the 333 rows, there were 160,637 empty rows in the AGE column, or roughly 28% of the AGE was missing. These values were imputed using the mean age value.

As the earlier transformation of the columns FATAL, MINORINJURY, and MAJORINJURY into FATALMAJORINJURY resulted in an encoded target, label encoding was unnecessary here. However, label encoding of the features was still needed. OrdinalEncoder() was used on the features PERSONTYPE, TICKETISSUED, SPEEDING, VEHICLETYPE, LICENSEPLATESTATE, and IMPAIRED.

The last preprocessing step was to normalize the quantitative variable, 'AGE'. This was done using StandardScaler(). Even though the Decision Trees do not require normalization, we used a normalized feature matrix for the DTs as the other models, Naïve Bayes and Logistic Regression, required normalization of quantitative variables, as using normalized data in the DT's would not hinder the models. This way, the same feature matrix could be used for each of the models.

Machine Learning

Four classification algorithms were originally chosen for this data; Naïve Bayes, Logistic Regression, Random Forest, and Extreme Gradient Boosted Decision Tree. Each of the algorithms was split into a training and testing set, with 70% used for training and 30% for testing. Each algorithm was cross-validated for accuracy. These cross-validation lines can be commented out in the scripts in which this takes hours (XGBoost and Random Forest). The models were scored in multiple ways. First, the overall accuracy was assessed. It was found that prior to optimizing parameters, the accuracy was around 99% for each of the models. However, due to the class imbalance, this was a poor reflection of the model. Therefore, AUC (area under the ROC curve) was determined, as well as the sensitivity (ability to correctly identify positives) and the specificity (ability to correctly identify non-positives). Cross-validation was performed using 5 folds to validate the accuracy of the models.

Naïve Bayes

Naïve Bayes is a simple algorithm used for classification based on Bayes Theorem (Navlani, 2018). This algorithm uses the concept of prior probabilities to predict a target and assumes that each feature is independent of one another, regardless of whether they are, hence the 'naive' (Navlani, 2018). This algorithm is quick, efficient, and does not accept hyperparameter optimization. The probability of falling within a class *given* the data is calculated. Below, Bayes equation can be seen. It shows that the probability of falling into a class given the data (P(h|D), or the posterior probability) is equal to the probability of the data *given* the class (P(D|h), the posterior probability of the data) times the probability of falling into a class (P(h), the prior probability of the class) divided by the probability of the data (P(D), also known as the prior probability of the data). For each instance, the probability of falling into each class is calculated and the data is classified into the category with the highest probability (Navlani, 2018).

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Extreme Gradient Boosted Decision Tree

Extreme Gradient Boosting, or XGBoost, is a decision tree model. It is an ensemble method in which gradient boosting is performed. The forest is

constructed by learning from the previously built, weak learners (Pathak, 2019). This leads to an increase in accuracy as the forest continues construction. XGBoost has a multitude of parameters, some of which are used for regularization. Xgboost is an extension of gradient boosted decision trees and is fast, relative to other types of gradient boosting (Pathak, 2019).

Random Forest

We also implemented a Random Forest model. The Random Forest model merges N number of decision trees, which provides a stable prediction and typically produces accurate results with limited hyperparameter tweaking (Donges, 2019). This model is versatile and relatively quick to run. However, running the model with cross-validation—as we did in our code—can add several minutes to the run time.

Logistic Regression

Since our target is binary, it made sense to implement a Logistic Regression model, which finds an equation to predict the target (Hoffman, 2019). This model is straightforward and quick to run. However, it runs the risk of overfitting due to its simplicity. Logistic Regression also requires more hyperparameter tuning than some of our other models. Because of this, we wrote a script to optimize the hyperparameters for this model.

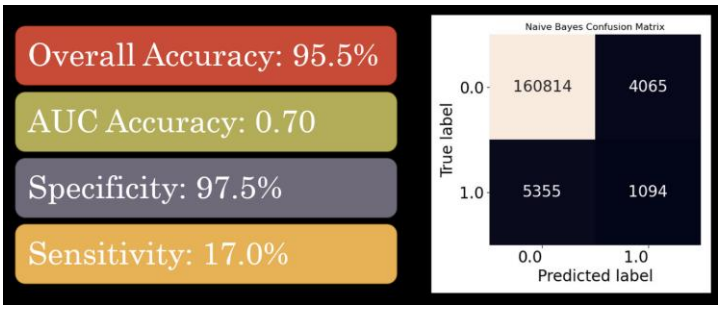
Final Voting Classifier

In an attempt to increase the accuracy of the models, a model that combines the classification of the three best models was added, VotingClassifier(). The idea behind this model is to aggregate the results of three models and classify each instance based upon the collective vote of the three models. This model allows for either "hard voting", in which each model gets an equal vote, or "soft voting", in which the vote is based upon the average probability of each model's classification (ML Voting Classifier Using Sklearn, 2019).

Results

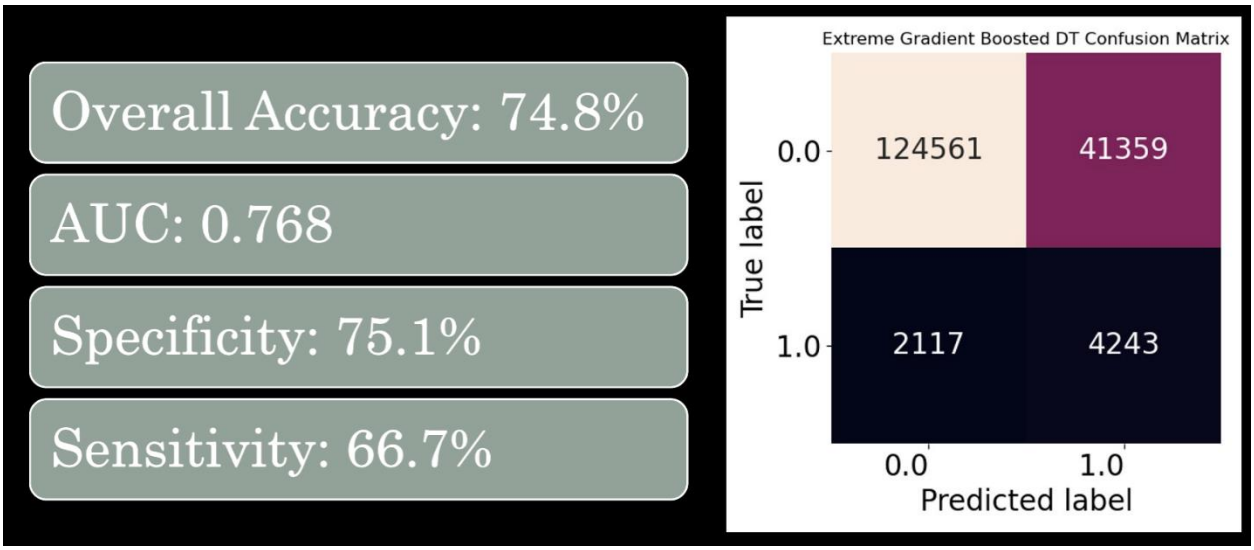
Naïve Bayes

Naïve Bayes does not allow for parameter inputs, thus there was no optimization of the model. Despite the natural ability of this model to deal with class imbalance using the information on prior probabilities, this model was unable to handle the class imbalance. The overall accuracy was 95.5%, but the AUC was only 0.7 and the sensitivity was 17.0%. This indicates that the model was classifying almost all of the data points as the negative class or predicted that almost everyone would experience only a minor injury. Or, in other words, the model was not sensitive enough to detect a positive (major injury/fatality) occurrence. The confusion table below depicts the inability of the model to predict a positive instance correctly. As can be seen, there are a total of 6,449 instances of a positive ('1'), but only 1094 are classified correctly as a positive instance.



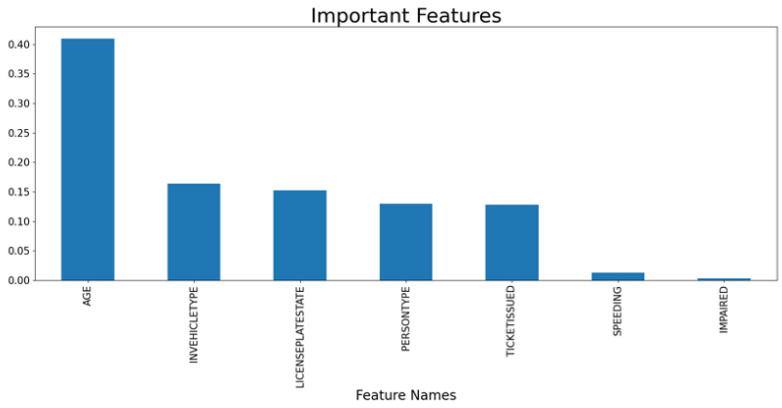
Extreme Gradient Boosted Decision Tree

Prior to optimizing the parameters, the model performed worse than Naïve Bayes, with a sensitivity of less than 0.01. The most important parameter in increasing the accuracy was that of 'scale_pos_weight'. As the data contained 25 times more negative class instances than positive, the 'scale_pos_weight' was reset to 25. This decreased the overall accuracy (dropping from around 95% to 74.8%) but increased the sensitivity to 66.7%. The most important feature was whether a ticket was issued, followed by the person type (driver, passenger, etc.) and then age. The confusion matrix below shows how the negative class is classified incorrectly at a much higher rate than Naïve Bayes, however, the positive class is classified correctly much more often than in Naïve Bayes.



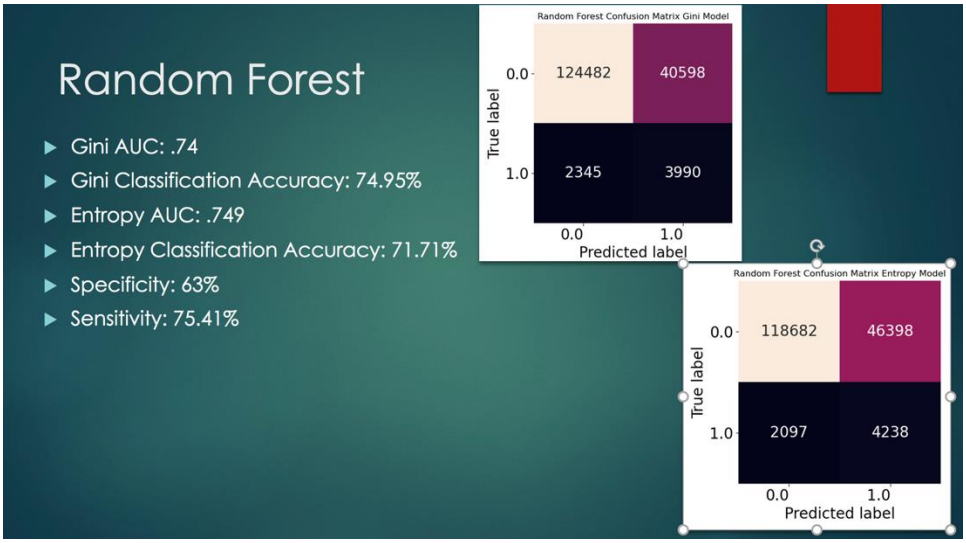
Random Forest

Next, we ran a random forest model with 100 trees, a balanced subsample, and a random state to ensure consistent outputs. After checking for important features, we determined that speeding and impairment did not have a significant influence on our target—as shown in the graph above. This confirmed what we



noticed in EDA and proved our initial hypothesis to be wrong, as we expected these features to have a greater impact on our target. Age was the most important feature.

We ran a Gini model with all the features and an Entropy model with the important features: Age, Vehicle Type, License Plate State, Person Type, Ticket Issued. As seen in the image below, the Gini model had a slightly higher classification accuracy at 74.95% accuracy; compared to 71.71% accuracy in our Entropy model. Both models had similar AUCs, with the Entropy model being only slightly higher at .749 (compared to .74 for the Gini model, an insignificant difference). As shown in the confusion matrices below, both models predicted false positives at a much higher rate than false negatives.

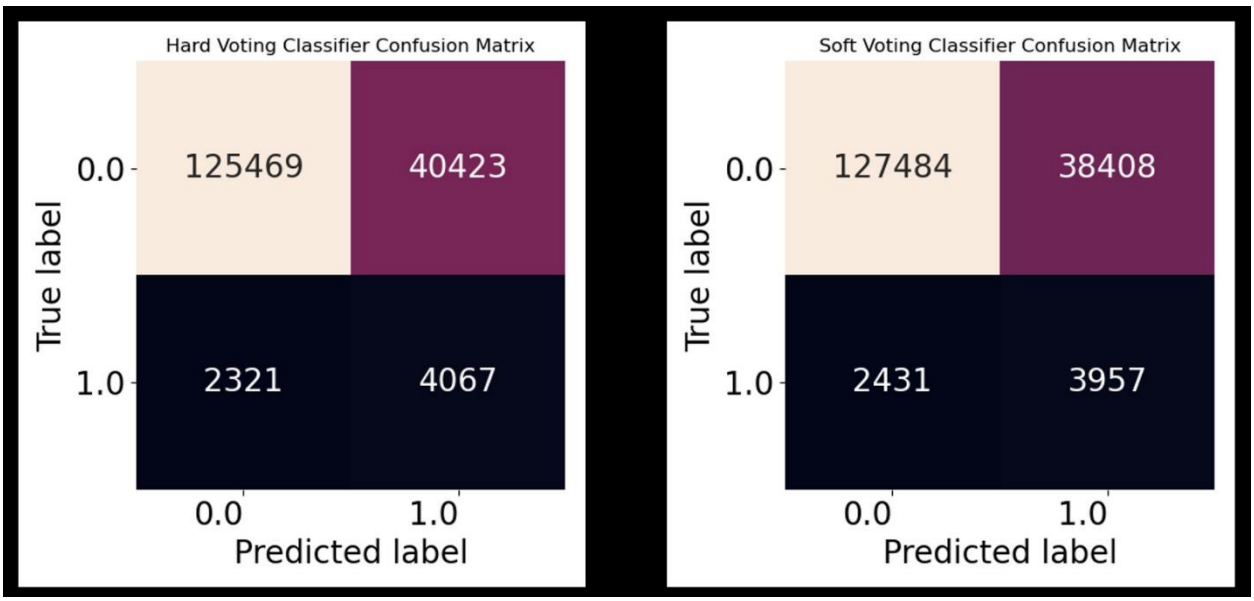


Logistic Regression

We initially ran a logistic regression with default parameters and got poor results. We ran code to find optimal hyperparameters and determined to set penalty to L2 and C to .0001. After running the model again with these hyperparameters our results did not change dramatically. Our model predicted that none of the accidents resulted in major injuries and/or fatalities. This resulted in a 96.3% classification accuracy and .667 AUC. The high classification accuracy makes sense, since a majority of accidents do not result in major injuries and/or fatalities, but we know that classifying 100% of accidents as 0 signifies a weak and inaccurate model. In other words, these results were an instance of probability rather than genuinely accurate classification. So, we decided to adjust our model further and set class weight to balanced. After setting the class weight to balanced, our AUC decreased slightly to .664 and our classification accuracy decreased to 73.32, as depicted in the image below. However, the model was no longer classifying all accidents as 0.

Voting Classifier

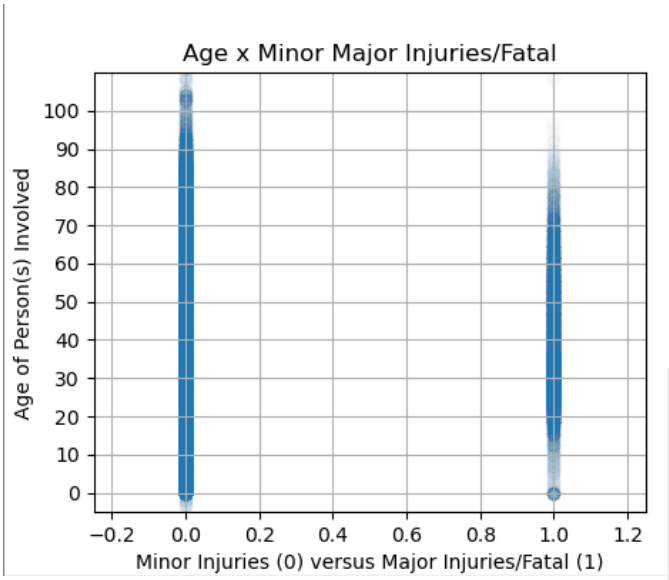
Both the soft voting and hard voting methods of the voting classifier were constructed. This model does not have the ability for parameters to be optimized, nor for the AUC to be calculated. Neither method resulted in higher accuracy, specificity, or sensitivity than the XGBoost model. The soft voting method was slightly better at distinguishing true negatives while the hard voting method was slightly better at distinguishing true positives. The confusion matrices below show the classifications using both the soft and hard voting. The accuracy, sensitivity, and specificity for the hard voting method were 75.2%, 63.7%, and 75.6%, respectively. The accuracy, sensitivity, and specificity for the soft voting method were 76.3%, 61.9%, and 76.8%, respectively.



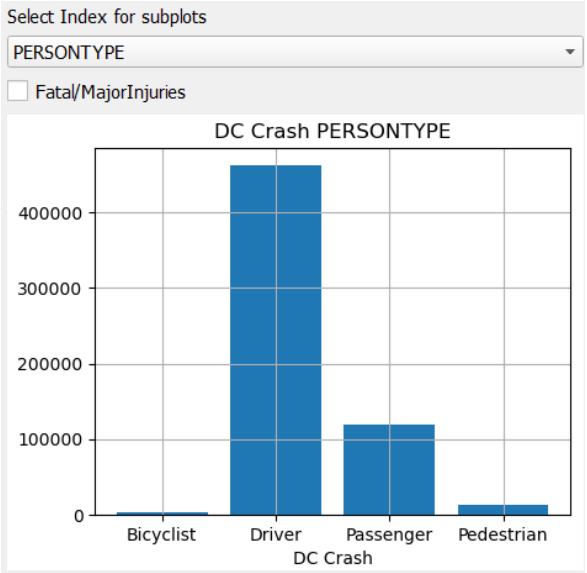
GUI

The project's GUI desktop application is accessed by executing Main.py. The menu items are File – Data - EDA – ML Models. From the File menu – Quit (CTRL + Q) will close the GUI application. Data menu

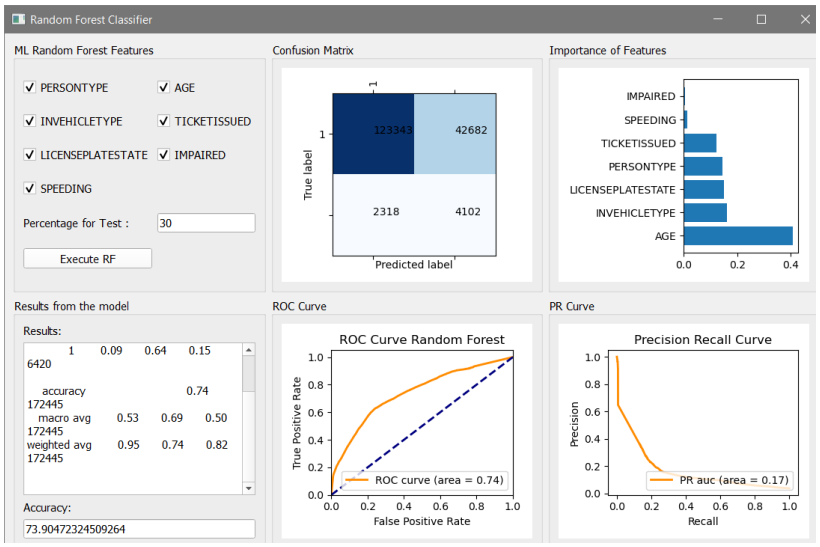
– Sample displays a few rows of data from DC Crash Details before preprocessing. Under the EDA menu are three exploratory data plots. The first is the Age Histogram which displays "Age of People Involved in Traffic Accidents with Fatalities or Major Injuries" where Age is on the x-axis and the number of crashes on the y-axis. The second EDA is the "Age Scatter," which plots the ages of persons who sustained Minor Injuries versus those who sustained Major Injuries or Fataluty. The range of ages in the data show that we had negative ages and ages above 100 years old.



The third EDA menu option Crash Graphs allows user interaction where an index can be selected by one of the following options to see a resulting graph: impaired, speeding, ticket issued or person type. Mark the checkbox to see crashes that resulted in major injuries or fatality.



The ML Models menu has three models developed by the group: Logit, Random Forest Classifier, and XGBoost DT. For the models, the feature columns are person type, age, in vehicle type, ticket issued, license plate state, impaired and person type. For the Logit model displays findings implemented in the Logit.py class and discussed in previous sections. For both the Random Forest Classifier and XGBoost DT displays findings implemented in the



randomforest.py and xgboost.py classes but without cross-validation.

We used specificity and sensitivity as a metric to evaluate our models up to this point. But for the sake of visualization for the GUI, a Precision-Recall (PR) curve was implemented in addition to the ROC curve for Random Forest and XGBoost models. Precision (P) is defined as the number of true positives (Tp) divided by the sum of true positives (Tp) and false positives (Fp). Recall is defined as the number of true positives divided by the sum of true positives and false negatives. (Pedregos, 2011)

$$P = \frac{T_p}{T_p+F_p} \quad R = \frac{T_p}{T_p+F_n}$$

The XGBoost Classifier is a slightly better algorithm than the Random Forest Classifier with both the ROC and PR curves.

Conclusion

While our Naïve Bayes model produced the highest classification accuracy, the model was not highly sensitive, indicating that the predictions made were a result of probability and rather than accurate prediction. This was also the case for our initial Logistic Regression model. Once we set the class weight to balanced, the classification accuracy decreased to about 73%, which is closer to the accuracy of our other models, which were more sensitive to the features in our data set. There was no one best classifier, with XGBoost having the highest AUC, Random Forest having the highest sensitivity, and Voting Classifier having the highest specificity (excluding Naïve Bayes), as can be seen in the table below. Due to the class imbalance, the models worked very poorly prior to optimizing the class balance parameters, resulting in sensitivities of less than 1%.

	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%

These results indicate that there is some relationship between our features and the injuries sustained in a D.C. traffic accident, however these features may be too broad to provide highly accurate classification predictions. For example, the vehicle type feature describes vehicles in broad categories, rather than providing makes and models of vehicles. These categories provide us some insights, but they do not tell the entire story. As mentioned earlier in the report, most of the vehicles are described as “Passenger Car.” This category includes a wide range of vehicles that offer different safety features. In other words, a Subaru Outback—a vehicle with a 5-star safety rating—and a Fiat 500L—a vehicle with one of the lowest safety ratings—would both be placed in the “Passenger Car” category (Sackman, Subaru). Another

example of broad features is the Ticket Issued column. There is no indication of what the ticket was issued for. Tickets can be issued for texting while driving, not wearing a seatbelt, and other violations that could potentially impact the severity of injuries sustained in an accident.

References

TPB News, *Vehicle Census Shows What's on our Region's Roads*, Metropolitan Washington Council of Governments (District of Columbia), 2018. Accessed on: March 11, 2021 [online]. Available: https://tinyurl.com/j87hfhe9

U.S. Department of Transportation National Highway Traffic Safety Administration, *The Economic and Societal Impact of Motor Vehicle Crashes*, 2010. Revised: May 2015. [PDF]. Accessed March 11, 2021. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812013

District Department of Transportation, The Vision Zero Initiative. Accessed March 11, 2021 [online]. Available: https://ddot.dc.gov/page/vision-zero-initiative

District Department of Transportation, Metropolitan Police Department, *Crashes Details Table*, Open Data DC, (District of Columbia): Vision Zero Data Planning Work Group, 2020. Accessed on: March. 11, 2021. [online]. Available: **https://opendata.dc.gov/datasets/crash-details-table**

James, G., Witten, D., Hastie, T., Tibshirani, R., *An Introduction to Statistical Learning - with Applications in R.* New York: Springer, 2013.

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.

Barnet-Woods, Bryan, "How Roadway Capacity and Safety Intersect with Vision Zero Goals", Greater Washington, Oct. 12, 2020. Accessed Mar. 13, 2021 [online]. Available: https://ggwash.org/view/79050/how-roadway-capacity-and-safety-intersect-with-vision-zero

"VotingClassifier". *Sklearn.* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

"ML Voting Classifier Using Sklearn". *GeeksforGeeks.* Nov 25, 2019. [Online]. https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/

Navlani, Avalash. "Naive Bayes Classification Using Scikit-Learn". *DataCamp.* Dec. 4, 2018. [Online]. https://tinyurl.com/wkfhc6mp

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.[Online] https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html. Acessed Apr. 25, 2021.

Pathak, Manish. “Using XGBoost in Python”. *DataCamp*. Nov. 8, 2018. [Online]. https://www.datacamp.com/community/tutorials/xgboost-in-python

Appendix: List of Packages Utilized in Project

- Pandas
- Numpy
- Matplotlib.pyplot
- matplotlib.figure
- Seaborn
- From sklearn.preprocessing import OrdinalEncoder
- from sklearn.preprocessing import StandardS
- from sklearn.preprocessing import StandardScaler
- from sklearn.preprocessing import LabelEncoder
- from sklearn.model_selection import train_test_split
- from sklearn.linear_model import LogisticRegression
- from sklearn.metrics import accuracy_score
- from sklearn.metrics import roc_auc_score
- from sklearn.metrics import confusion_matrix
- from sklearn.metrics import roc_curve, auc
- from sklearn.metrics import precision_reca
- from sklearn.model_selection import cross_val_score
- from sklearn.model_selection import RepeatedStratifiedKFold
- from sklearn import linear_model, decomposition, datasets
- from sklearn.pipeline import Pipeline
- from sklearn.model_selection import GridSearchCV
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.metrics import classification_report
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.naive_bayes import GaussianNB
- from sklearn.ensemble import RandomForestClassifier
- from sklearn import tree
- from sklearn.ensemble import VotingClassifier
- Xgboost
- collections
- os
- Warnings
- sys
- from PyQt5.QtWidgets import (QMainWindow, QApplication, QWidget, QPushButton, QAction, QComboBox, QLabel, QGridLayout, QCheckBox, QGroupBox, QVBoxLayout, QHBoxLayout, QLineEdit, QPlainTextEdit, QTableWidget, QTableWidgetItem)
- from PyQt5.QtGui import QIcon
- from PyQt5.QtCore import pyqtSlot
- from PyQt5.QtCore import pyqtSignal
- from PyQt5.QtCore import Qt
- Scipy
- from scipy import stats

- from itertools import cycle
- from PyQt5.QtWidgets import QDialog, QVBoxLayout, QSizePolicy, QMessageBox
- from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
- from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as NavigationToolbar
- import xgboost as xgb
- researchpy
- Statsmodels