MACHINE LEARNING

ROAD COLLISION ANALYSIS AND PREDICTION

horizontal line

# 

# 

**Table of Contents**

[**Data Exploration**](#_heading=h.tyjcwt)3

[**Dataset**](#_heading=h.3dy6vkm)3

[Season/Month](#_heading=h.u6g4dnu2vkc5) 3

[Hour](#_heading=h.qy6tfe1wwf1w) 4

[District](#_heading=h.2s8eyo1) 5

[Road Surface Condition](#_heading=h.17dp8vu) 6

[Environmental Condition (Visibility)](#_heading=h.26in1rg) 6

[Involved Party](#_heading=h.tblilaa1quph) 7

[Age of Involved Party](#_heading=h.35nkun2) 7

[Traffic Control](#_heading=h.1ksv4uv) 7

[Accident Location](#_heading=h.44sinio) 7

[Direction](#_heading=h.86abo41cmdto) 7

[Maneuver](#_heading=h.r67ovplg1lrl) 8

[Driver Action](#_heading=h.3j2qqm3) 8

[Driver Condition](#_heading=h.bz503acvpoz8) 8

[**Predictive Modeling**](#_heading=h.1y810tw)8

[Data Preparation](#_heading=h.4i7ojhp) 8

[Predicting Major Accidents](#_heading=h.2xcytpi) 9

[Model Design](#_heading=h.1ci93xb) 9

[Random Forest Classifier](#_heading=h.3whwml4) 9

[Decision Trees Classifier](#_heading=h.2bn6wsx) 9

[AdaBoost Classifier](#_heading=h.qsh70q) 10

[Extra Tree Classifier](#_heading=h.naieq3rgithw) 11

[Gradient Boosting Classifier](#_heading=h.49x2ik5) 11

[eXtreme Gradient Boosting (XGBoost) Classifier](#_heading=h.71683bb596u6) 12

[Model Evaluation](#_heading=h.23ckvvd) 12

[Learning Curves](#_heading=h.ihv636) 12

[Model Performance Metrics](#_heading=h.32hioqz) 12

[ROC Curve](#_heading=h.xficgovwalap) 14

[Precision-Recall Curves](#_heading=h.41mghml) 15

[Confusion Matrix](#_heading=h.2grqrue) 16

[**Conclusion**](#_heading=h.3fwokq0)18

[**Appendix A: Exploratory Data Analysis**](#_heading=h.1v1yuxt)21

[**Tree Visualization**](#_heading=h.19c6y18)35

[**Appendix B: Predicting Fatalities**](#_heading=h.3tbugp1)36

[**Appendix C: Clustering**](#_heading=h.4k668n3)38

[**Appendix D: Attribute List**](#_heading=h.2zbgiuw)39

**Introduction**

For Canadians, car accidents are one of the leading causes of both death and hospitalization. In 2020, for example, there were approximately 100,000 car accidents in 2020, of which 7,868 resulted in serious injuries and 1,591 resulted in deaths. (Canadian Transportation Safety Board, 2020). According to Transport Canada (2020), the risks of 16 to 24-year-old drivers suffering fatal car accident injuries are significantly higher than in other age groups. As well as older adults with heart disease or cognitive disorders also face higher risks.

Understanding traffic collisions' primary characteristics is an important step for developing effective countermeasures in traffic safety. Although there is a downtrend in the number of deaths due to road collisions in developing countries, the cost of traffic-related injuries and property loss remains a significant concern. To minimize the financial and physical cost of traffic collisions, predictive models can be developed to determine the collision types or severity levels. These severity levels can create an infrastructure/facility as a countermeasure for the collision situation.

**Objective**

The objective of this study is to develop a classification machine learning model for the Motor Vehicle Collision dataset to determine the likelihood of major/fatal accidents in the City of Toronto using over 15 years of data, which is taken from the open data (2006-2020). Exploratory data analysis (EDA) will be carried out to visualize and analyze trends in the data set. It will also be used to help us determine what predictive model(s) to use. The data will then be prepped (nulls will be filled, unnecessary variables will be removed, etc.) for modelling. We will then compare the results against several other models and discuss the strengths and weaknesses of each. We will also use various evaluation techniques, such as roc curve, precision-recall curve, learning curves, etc.

The practical application of the model could be to dedicate better resources in the areas where the model predicts a higher probability of a major/fatal accident; or to serve as a guide to traffic participants that certain areas have a high incidence of major/fatal accidents depending on time of day, season, road conditions, etc.

# Data Exploration

# Dataset

The dataset came from Toronto’s open data catalog: (https://open.toronto.ca/dataset/motor-vehicle-collisions-involving-killed-or-seriously-injured-persons/). This dataset includes 16,860 observations of traffic collisions events where a person was either Killed or Seriously Injured (KSI) from 2006 to 2020. One of limitations that was noted was that the location of occurrences has been deliberately offset to the nearest road intersection node to protect the privacy of parties involved in the occurrence. Due to the offset of occurrence location, the numbers by Division and Neighborhood may not reflect the exact count of occurrences reported within these geographies. Therefore, the accuracy, completeness, and timeliness of the data cannot be guaranteed.

The dataset had an extensive amount of null variables, which we used *fillna()* to replace with a value, which was most relevant to that column. For example, if it was a binary column yes/null, the nulls were filled with “no”. If it was a categorical column that already had the value of “unknown”, it was filled with “unknown” or “other”. We also defined the dependent variable to be a binary indicator, based on the *injury* column, which equals 1 when the injury to the individual involved in a collision was a fatality or had a major injury, and 0 for any other injuries including minor injuries, no injuries, or property damage. We also found that the date variable was not useful on its own, in the string format it originally came in. Hence, we decided to extract the day of the week, month of the year, and season to create more meaningful analysis. These steps were done prior to EDA, but the rest will be discussed during the pre-processing section.

**Exploratory Data Analysis (EDA)**

EDA is an approach to data analysis that uses a variety of techniques, largely to maximize insight, reveal underlying structure, check for outliers, test assumptions, and determine optimal factors.

We have performed EDA on the collision data set using Seaborn and Matplotlib. These features were found to have the highest relation to accident seriousness. While they may all have an impact, not every feature will be discussed in the following sections; only a subset of the 53 features that were originally present in the dataset will be discussed. The rest of the features as well as some detailed charts are included in the appendix.

### Season/Month

As mentioned previously, one of the features we were able to extract from the *Date* column was *Season*. A *barplot()* and *hisplot()* was used to measure the probability of a fatal/major incident and the total counts by season, respectively. Based on the findings, it was determined that the probability of a fatal/major incident seems to be higher during the winter/fall months. In terms of counts, however, there seems to be more major/fatal accidents in the Summer and Fall, which may be contrary to one’s beliefs (Figure 1). Our analysis of month (Appendix Figure A1) confirmed similar findings, which was that the probability of a fatal/major incident seems to be higher during September to December (fall/winter). August had the lowest probability. In terms of counts, there seems to be more fatal/major incidents that occurred through Summer/fall months.

*Fig 1: Volume of Accident by Target (Major Accident Outcome) by Season*

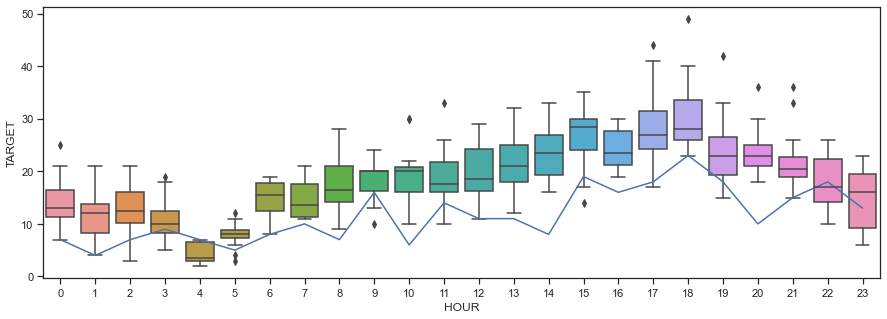
### Chart, bar chart Description automatically generated

### Hour

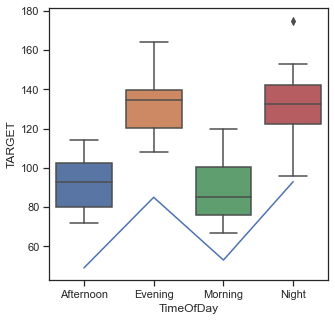
Hour was a useful feature, because it provided a distinction between rush hour times and early morning hours, where traffic is generally slow. We used a *boxplot()* to illustrate our findings, which is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). The findings confirmed that there seems to be an increase in fatal/major incidents in the evening hours, which is evident by the increasing trend line and the higher box plots for evenings and night. It starts to level off at the end, which may be because people are sleeping. There also seems to be more significant outliers in the evening hours. *Please refer to Figure 2.*

From *hour,* we were able to extract a *time-of-day* feature. To be more specific, “Night” was defined as 9pm and 6am, “Morning” was defined as 6am - 11:59 am, “Afternoon” was 12pm - 3pm, and “Evening” was 3 pm and after. There are more incidents in the evening and at night. There also seems to be an outlier at night. It should be noted that in 2020 there was a pandemic with shutdowns, hence, this chart may not reflect normal traffic conditions. *Please refer to Figure 3.*

*Fig 2: Box Plot of Target Variable (Major Accident Outcome) by Hour of Accident 2020*



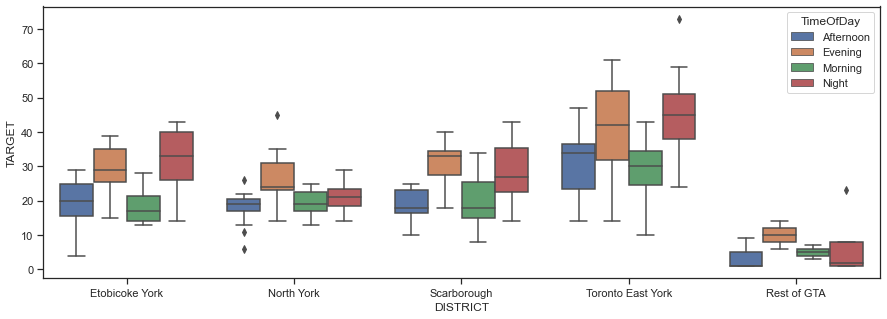
*Fig 3: Box Plot of Target Variable (Major Accident Outcome) by Time of Day of Accident*



### District

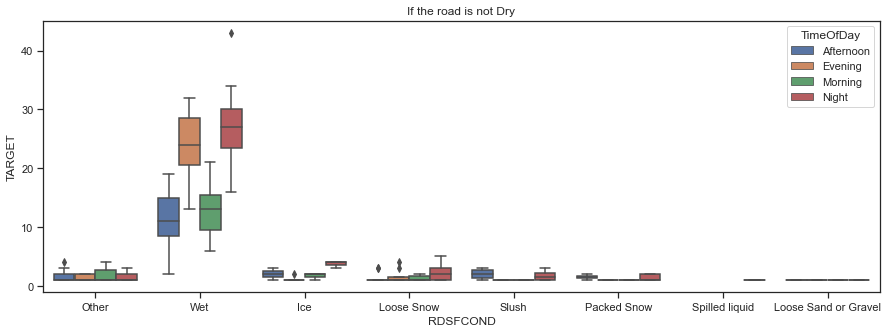
If we carry this analysis (*Time of Day*) to visualize the *District*, Toronto East York shows the highest number of serious accidents/fatalities regardless of *Hour of Day.* Like the previous chart, the evening and night labels seem to have a higher boxplot and more significant outliers. *Please refer to Figure 4.*

*Fig 4: Box Plot of Target Variable (Major Accident Outcome) by Time of Day and District*



### Road Surface Condition

A significant number of accidents have occurred during Dry Road surface condition; hence, we did not include it in this chart. After removing dry, we found that wet conditions had the second highest number of accidents, relative to ice, Loose Snow, Slushing or packed snow condition. *Please refer to Figure 5.*

*Fig 5: Box Plot of Target Variable (Major Accident Outcome) by Road Condition, excluding Dry*

### Environmental Condition (Visibility)

*Visibility* is another important feature in fatal/major road accidents. In terms of counts, the highest number of major accidents/fatal incidents occurred in clear environments than any other environmental conditions such as Snow, Fog, Freezing Rain. These could be due to drivers not being cautious while driving in a clear environment. However, the probability of a major/fatal accident is 10% higher during drifting snow and strong winds than in a clear environment. *Please refer to Figure 6.*

*Fig 6: Volume of Accident by Target (Major Accident Outcome) by Visibility*

Icon

Description automatically generated with low confidence

### 

### Involved Party

The likelihood for major/fatal accidents seems to be higher when cyclists, cyclist passengers, and wheelchairs are involved. Followed by moped drivers, motorcycle drivers, and pedestrians. In terms of counts, pedestrians are the most affected group in major accidents as they are not using any vehicle safety features such as brakes or airbags to avoid accidents and protect themselves. Please refer to *Appendix A, Figure A2.*

### Age of Involved Party

Although there is a large percentage of unknowns about Age band of the involved party or *invage*, the likelihood seems higher for fatalities/major incidents when ages 65+ are involved. In terms of counts, the bars seem to be spread out between 25 to 60 years old, and then they decrease after age 65, like the introduction suggested. Please refer to *Appendix A, Figure A3.*

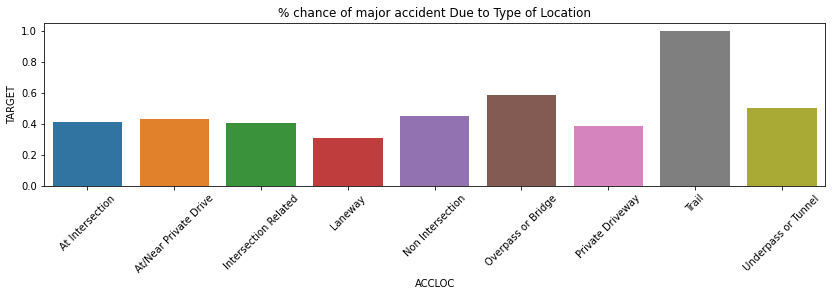
### Traffic Control

The main objective of *Traffic Control* is to avoid accidents. In terms of counts, most areas with accidents were traffic signals and uncontrolled. The probability of having a major incident/fatality seems to be higher when there is police control and school guard, which is surprising because one would think it would be the opposite. The yield sign also appears to have a high probability of major incident/fatality. Please refer to *Appendix A, Figure A4.*

### Accident Location

Not surprisingly, in terms of counts, most of the major accidents occurred at intersections, where there is incoming and outgoing traffic flow in all directions. However, there is a 50% higher chance of major accidents on trails where there might be less visibility, light, and traffic control.

*Fig 8: Rate of Target Variable (Major Accident Outcome) by Type of Location*



Chart, waterfall chart

Description automatically generated

### Direction

The probability of having a major incident/fatality seems to be higher when individuals are traveling North. Perhaps the weather conditions get more severe as they travel more North. There also seems to be a lot of unknowns. Please refer to *Appendix A, Figure A5*.

### Maneuver

In terms of counts, most fatal/serious accidents occur when the driver is going ahead and there seems to be a lot of unknowns in this feature. In terms of probability, having a major incident/fatality seems to be higher when individuals are pulling onto shoulder, overtaking, and merging. Please refer to *Appendix A, Figure A6.*

### Driver Action

The probability of having a major incident/fatality seems to be higher when the driver has lost control or exceeded speed limit. In terms of counts, it looks like most drivers in this dataset drive properly or other. Please refer to *Appendix A, Figure A7.*

### Driver Condition

The probability of having a major incident/fatality seems to be higher when the driver has medical or physical disability. Fatigue and alcohol impairment appear to be significant. In terms of counts, there seems to be a lot of unknowns, and normal driving. Please refer to *Appendix A, Figure A8.*

The rest of the features are in Appendix A with an explanation for each chart. They were less useful to our analysis, due to the extensive number of nulls and unknowns. In general, our overall expectations were confirmed in that we proved there would be certain features that had a high impact on the severity of the accident. For example, road surface condition, visibility but there were also other features with very low correlations between them and accident severity.

# Predictive Modeling

## Data Preparation

Based on our data exploration, we had the intuition that our initial hypothesis of predicting accident outcomes using the available details about individuals and collisions may be feasible using some machine learning methods from this course. As mentioned previously, we defined the dependent variable to be a binary indicator that equals to 1 when the injury to the individual involved in a collision was a fatality or had a major injury, and 0 for any other injuries including minor injuries, no injuries, or property damage. From there, additional work was required to prepare the dataset for machine learning modeling:

* We identified categorical variables that included the geographical location of the accident (roads, neighborhood, police division, etc.), road and lighting conditions as identified in the police report, and indicator variables for the presence of pedestrians, drivers, motorcycles, passengers, aggressive driving, red lights, alcohol, or disability in the accident.
* We found that the feature *Date* was not useful on its own, in the string format it originally came in. However, when we broke it down into the day of the week, and month of the year, and season it became more useful.
* As mentioned previously, we noticed that the dataset had an extensive amount of null variables, which we used .*fillna()* to replace with a value, which was most relevant to that column. For example, if it was a binary column or yes/null, the nulls were filled with “no”. If it was a categorical column, and it already had “unknown”, the blanks were grouped with the other “unknowns”.
* Another challenge was the potential for data leakage. Data leakage refers to a problem where information about the holdout dataset, such as a test or validation dataset, is made available to the model in the training dataset. In order to avoid this problem, *injury*, *acclass*, and *fatal\_no* were dropped. *Injury* was used to make the target variable, so this needed to be dropped; *acclass* would have revealed that there was a fatality to the target variable, and *fatal\_no* implies that someone has died.
* We removed *geometry,accnum, object\_id* because *accnum* and *object\_id* are unique identifiers, and they add no value to the dataset. For ‘*geometry*’, we have other variables that provide location, and this is typically used in geo-mapping applications.
* We built a pipeline to clean up the dataset, which included
  + OneHotEncoder for all categorical variables
  + MinMaxScaler to normalize any numerical variables
  + Pipeline and ColumnTransformer to put together all the preparation steps.

**Predicting Major Accidents**

Approximately 40% of our dataset represented the major accident outcome, which was sufficiently balanced, although technically anything below 50% is unbalanced. In some initial algorithms, we saw an overall accuracy of over 80%. This provided us with confidence in the predictive power present in this dataset and opened the opportunity for trying different models and fine-tuning the models to improve that accuracy, which continues in the section below.

## Model Design

## Random Forest Classifier

Random forest is a commonly used machine learning algorithm, which combines the output of multiple decision trees to reach a single result. Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction (IBM, 2022). The low correlation between models is the key. (“What is Random Forest?”)

The hyper-parameters were selected by using Randomized Search CV. Randomized Search implements a “fit” method and a “predict” method like any classifier except that the parameters of the classifier used to predict is optimized by cross-validation. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n\_iter (sckit-learn, 2022). We decided to use the default n\_iter = 10, which is the number of parameter settings that are sampled. n\_iter trades off runtime vs quality of the solution. We also used n\_jobs = -1. N\_jobs which refers to the number of jobs to run in parallel, and -1 means using all processors. Finally, verbose controls the verbosity, the higher, the more messages. We set our verbose at 1. We used these metrics for the rest of the Randomized Search models.

The hyper-parameters selected for this model were *criterion, max\_depth, max\_features, min\_samples\_leaf, min\_samples\_split, and n\_estimators*. *Max\_depth* is the max number of levels in each decision tree, *min\_samples\_split* is the minimum number of data points placed in a node before the node is split, *min\_samples\_leaf*  is the min number of data points allowed in a leaf node. The hyper-parameter tuning with Randomized Search improved the scores from the default model, in terms of accuracy score (86.15% vs. 84.52%), F1 score (81.62% vs. 77.88%), and roc auc score (94.48% vs.92.34%).

Random forest was selected because it works well with unbalanced and missing data, which is the case in this dataset (Blackwell, 2022). Random forest also offers a superior method for working with missing data. Missing values are substituted by the variable appearing the most in a particular node. In the case of this dataset, we already filled the missing values with values that made sense for that column. For example, missing values in yes/null columns were filled with “no”, so this aspect of the model was not necessary.

### Decision Trees Classifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation (scikit-learn, 2022). The key difference between decision trees and random forests is that while decision trees consider all the possible feature splits, random forests only select a subset of those features (IBM, 2022). Please refer to *Appendix A, Figure A25* for the decision tree visualization.

The hyper parameters were selected by using *Grid Search* and *Randomized Search CV*. In this case, Grid Search outperformed Randomized Search CV. As mentioned previously, Grid Search is an exhaustive search over specified parameter values for an estimator. The Important members are fit, and predicted. GridSearchCV implements a “fit” and a “score” method. It also implements *score\_samples*, *predict*, *predict\_proba*, *decision\_function*, *transform* and *inverse\_transform* if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid. As mentioned previously, the key difference from Grid Search is in random search, not all the values are tested and values tested are selected at random. For example, if there are 500 values in the distribution and if we input n\_iter=50 then random search will randomly sample 50 values to test (Ismiguzel, 2022). This is perhaps why *Grid Search* is performing better. The hyper-parameters that were tuned were *criterion, max\_depth*, *min\_samples\_leaf*, and *splitter*. The hyper-parameter tuning with Grid Search significantly improved the scores from the default model, in terms of accuracy score (86.60% vs. 83.54%), F1 score (81.00% vs. 78.50%), and roc auc score (93.53% vs.82.73%).

Decision trees were selected as a model because they are simple to understand and to interpret and can be visualized. They also usually require little data preparation. The downside is that they can be prone to problems, such as bias and overfitting (scikit-learn, 2022). We believe therefore the decision trees performed worse with hyper parameter tuning.

### AdaBoost Classifier

AdaBoost is an ensemble learning method (also known as “meta-learning”) which was initially created to increase the efficiency of binary classifiers. Adaboost helps you combine multiple “weak classifiers'' into a single “strong classifier” (towardsdatascience, 2022). Ada works by putting more weight on difficult to classify instances and less on those already handled well. The weak learners in AdaBoost are decision trees with a single split, called decision stumps.

The hyper-parameters for AdaBoost were selected by Randomized Search CV. The following hyper-parameters were selected *learning\_rate*, and *n\_estimators.* The learning\_rate is the parameter which is provided to shrink the contribution of each classifier. By default, it is provided with a value of 1. The number of base estimators or weak learners we want to use in our dataset. AdaBoost with hyper-parameter tuning performed slightly better in terms of F1 score (81.83% vs. 81.72%) , it performed slightly worse in terms of roc auc score and accuracy score.

AdaBoost was selected because it is less prone to overfitting as the input parameters are not jointly optimized. It also improves the accuracy of weak classifiers. The main disadvantage of AdaBoost is that it needs a quality dataset (Thailappan). Noisy data and outliers have to be avoided before adopting an Adaboost algorithm (Thailappan, 2022).

### 

### 

### Extra Tree Classifier

Extra Tree Classifier is a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting (scikit-learn, 2022). Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees (Gupta, 2022).

The optimal hyper-parameters were selected based on Randomized Search CV. An important hyperparameter for Extra Trees algorithm is the number of decision trees used in the ensemble. Typically, the number of trees is increased until the model performance stabilizes. Intuition might suggest that more trees will lead to overfitting, although this is not the case. Bagging, Random Forest, and Extra Trees algorithms appear to be somewhat immune to overfitting the training dataset given the stochastic nature of the learning algorithm. The number of trees can be set via the “*n\_estimators*” argument and defaults to 100, which is sufficient for this model. *Max\_depth* is the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples. *Min\_samples\_leaf*  is the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. Max\_features is the number of features to consider when looking for the best split.If “auto”, then max\_features=sqrt(n\_features). If “sqrt”, then max\_features=sqrt(n\_features). The following hyper parameters were selected criterion, max\_depth, max\_features, and min\_samples\_leaf. It performed worse with the hyper-parameter tuning, in terms of accuracy (79.72% vs. 84.61%), roc auc (89.69% vs. 90.72%), and F1 score (65.56% vs. 78.45% ). This may be due to insufficient range of parameters.

Extra trees were selected based on two key differences from Random Forest. One of the differences between Extra Trees and Random Forest is the selection of cut points in order to split nodes. Random Forest chooses the optimum split while Extra Trees chooses it randomly. However, once the split points are selected, the two algorithms choose the best one between all the subset of features. Therefore, Extra Trees adds randomization but still has optimization (Aznar, 2022). Since splits are chosen at random for each feature in the Extra Trees Classifier, it’s less computationally expensive than a Random Forest.

### Gradient Boosting Classifier

Gradient Boosting for classification is a supervised ensemble learning method that builds an additive model of many weaker models (typically decision trees) by iteratively fitting on the negative gradient of a loss function (“sklearn.ensemble.GradientBoostingClassifier — scikit-learn 1.1.2 documentation”). This approach is known to provide high accuracy and flexibility in handling imperfect data, which was relevant to our dataset, and tends to have higher overall accuracy compared to other models, which was also confirmed in our experience. While gradient boosting can overfit on the training set due to its attempts to minimize all errors, and can be computationally expensive (Kurama), we did not face these challenges, and the biggest drawback of this algorithm was its lack of explainability compared to "simpler" models such as a single decision tree. In terms of performance, Gradient Boosting was a very powerful technique in approaching this machine learning problem.

The hyper-parameters for Gradient Boosting Classifier were selected by Randomized Search CV. Max\_depth is the maximum depth of a tree. It’s used to control over-fitting as higher depth will allow the model to learn relations very specific to a particular sample. N\_estimator is the number of sequential trees to be modeled. The random state was added to achieve consistent results (Jain, 2016). Gradient booster performed better after hyper-parameter tuning, in terms of accuracy (87.93% vs. 86.59%), F1 score (82.91% vs.80.24%), and roc curve (94.93% vs 94.65%).

Gradient boosting trees were selected because they can be more accurate than random forests. Because we train them to correct each other’s errors, they’re capable of capturing complex patterns in the data. However, if the data are noisy, the boosted trees may overfit and start modeling the noise (Simic, 2022).

### 

### 

### eXtreme Gradient Boosting (XGBoost) Classifier

The eXtreme Gradient Boosting classifier attempts to improve on the Gradient Boosting approach by regularizing the processing to improve performance, so that trees are built in parallel and can take advantage of multicore computers (Khandelwal).

Similar to the other models, hyper-parameter tuning was selected based on Randomized Search CV. Similar to gradient boosting, *max\_depth* is the maximum depth of a tree, and is used to control over-fitting as higher depth will allow the model to learn relations very specific to a particular sample. The *min\_child\_weight* defines the minimum sum of weights of all observations required in a child. This is similar to *min\_child\_leaf* in GBM but not exactly. This refers to min “sum of weights” of observations while GBM has min “number of observations”. It is used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. Too high values can lead to under-fitting (Banerjee, 2020). XGBoost performed better with hyper-parameters tuning, in terms of accuracy (86.89% vs. 86.60%), F1 score (81.58% vs. 81.95%), and ROC curve(94.74% vs. 94.48%) .

This model was designed to be computationally efficient without sacrificing accuracy. In our experience, we did observe that XGBoost took less time than the gradient boost with only a marginal drop in accuracy on the test set. The disadvantages of this algorithm are similar to that of gradient boosting: a potential for overfitting and a lack of explainability.

## 

## Model Evaluation

In order to evaluate the performance of all the models we tried, we looked at several different approaches as outlined below. We were interested in understanding how well individual vs ensemble models performed on the 20% of the data we had set aside as a test set across the various evaluation metrics.

### Learning Curves

Learning curve in machine learning is used to assess how models will perform with varying numbers of training samples. This is achieved by monitoring the training and validation scores (model accuracy) with an increasing number of training samples. Models that have underfitting / high-bias, both the training and validation scores are very low and also lesser than the desired accuracy. Models that have overfitting / high-variance, there is a large gap between training and validation accuracy. Also, training accuracy may come to be more than desired accuracy.

We found that for both AdaBoost and Gradient Boost Classifiers, when the training sample size was less than 3800, the difference between training and validation accuracy is much larger. This is the case of overfitting. For training sizes greater than 3800, the model is better. It is a sign of good bias-variance trade-off. Past 3800 the training accuracy and validation accuracy are wider apart for Gradient Booster than for AdaBoost. You can also see that around 8200 it starts to diverge again for Adaboost.

|  |  |
| --- | --- |
| *Fig 9: Learning Curves for Adaboost and Gradient Boost Classifiers* | |
|  |  |

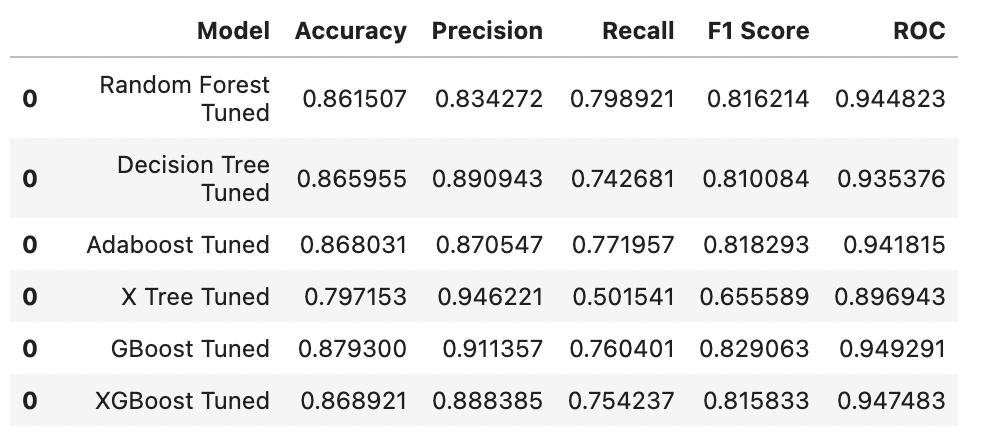
### Model Performance Metrics

The following metrics were used to evaluate model performance:

* **Accuracy** is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.
* **Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.
* **Recall (Sensitivity)** is the ratio of correctly predicted positive observations to all observations in actual class.
* **F1 Score** is the weighted average of Precision and Recall, which is better than accuracy because we have a relatively unbalanced data set.
* **AUC - ROC** **curve** is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.

It looks like Gradient Boosting Classifier and XGBoost seem to be outperforming the other models, in terms of ROC curve, accuracy score, and F1 score. Gradient Boosting classifier outperformed XGBoost in terms of roc auc curve This is not surprising because Gradient Boosting Classifier often provides predictive scores that are far better than other algorithms, and it works well with missing values. There are some disadvantages, however, which is that this method is sensitive to outliers. Outliers will have much larger residuals than non-outliers, so gradient boosting will focus a disproportionate amount of its attention on those points. Using Mean Absolute Error (MAE) to calculate the error instead of Mean Square Error (MSE) can help reduce the effect of these outliers since the latter gives more weight to larger differences. The parameter *criterion* helps you choose this function. It can also be prone to overfitting if the number of trees is too large. The parameter *n\_estimators* can help determine a good point to stop before our model starts overfitting. Finally, computation can take a long time. In the chart below, we did find that it took a bit longer than some of the other models, however, it was still shorter than Random Forest and decision trees. The hyper-parameter tuned Extra Tree Classifier performed the worst in terms of accuracy score, F1 score, and ROC curve. This may be because it does not perform bootstrap aggregation like in the random forest. In simple words, it takes a random subset of data without replacement. Thus, nodes are split on random splits and not on best split.

*Fig 10: Comparing Model Metrics across different Classifiers*

**

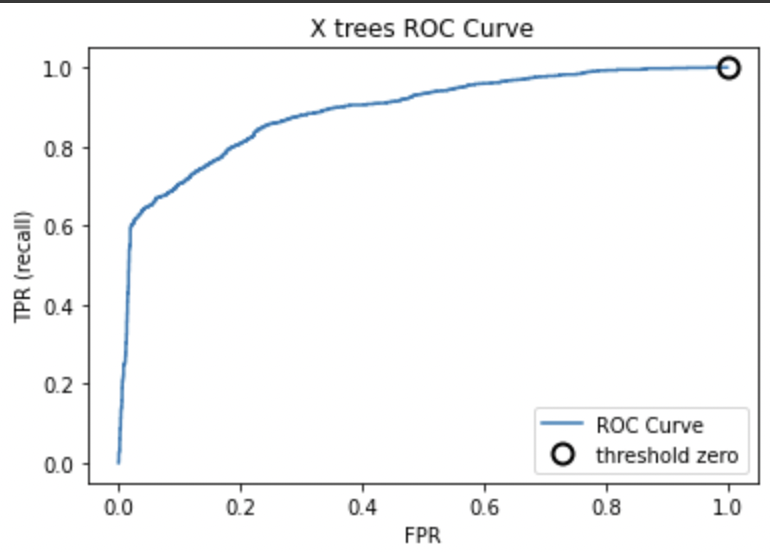
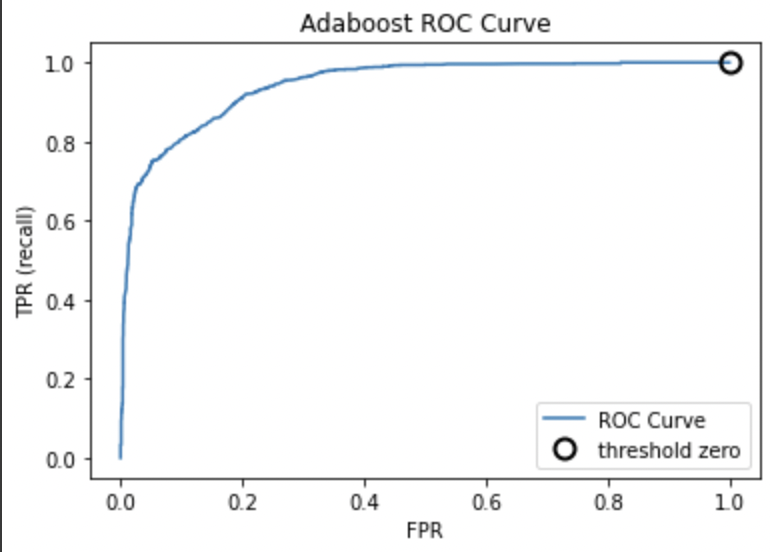
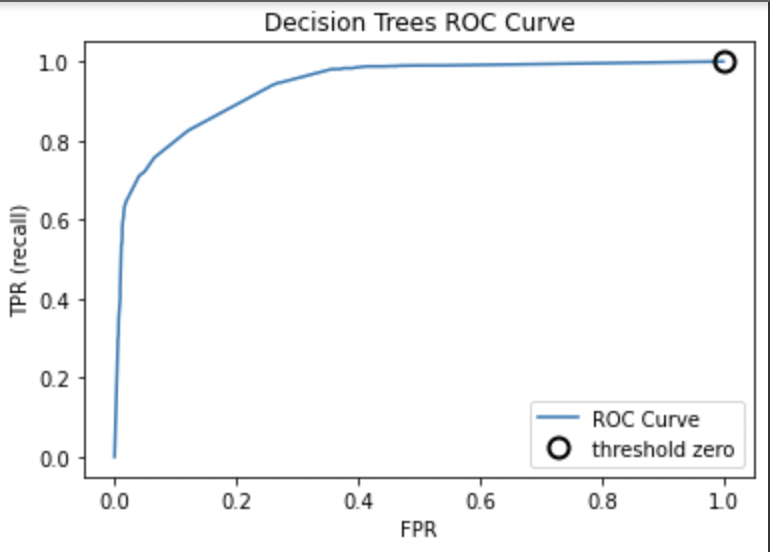
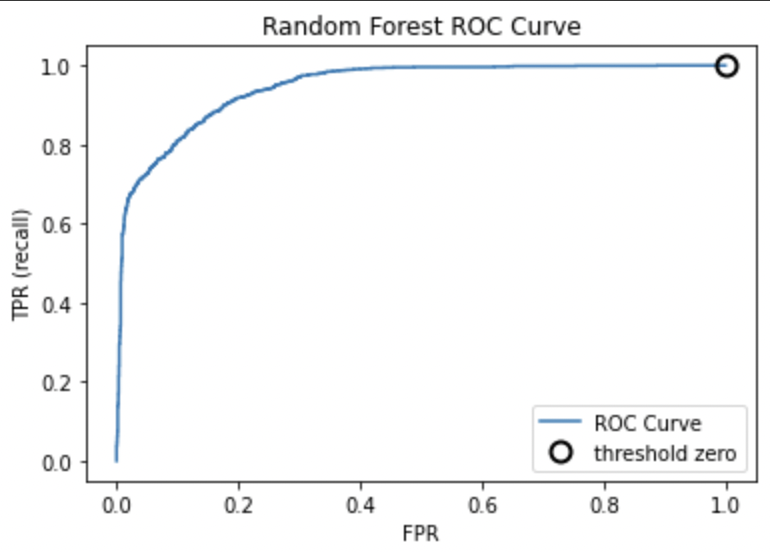
### 

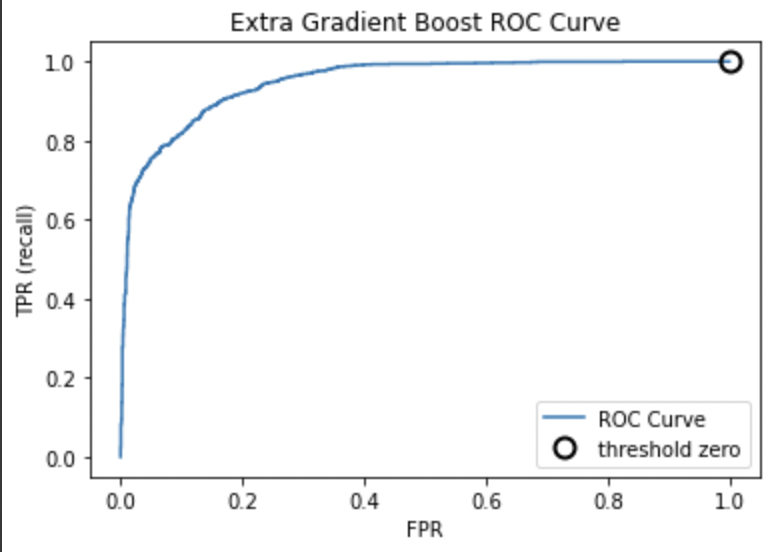
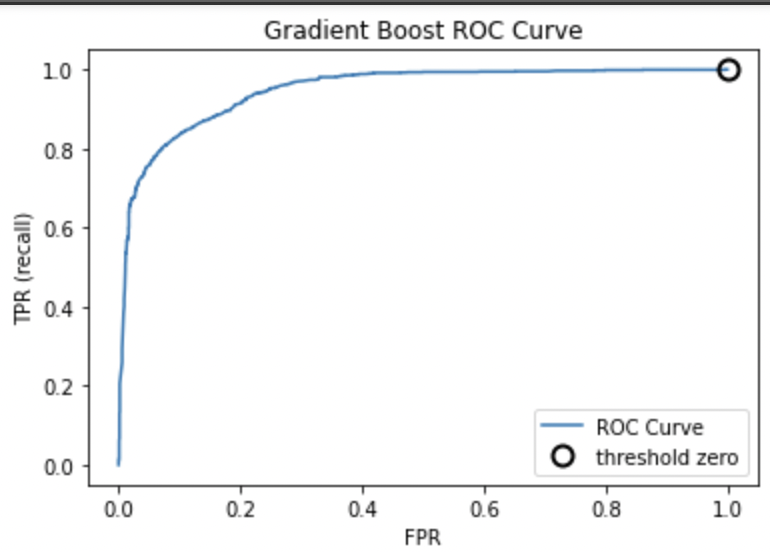
### ROC Curve

The ROC Curve is used to show the positive rate (FPR) against the true positive rate (TPR). For an ROC curve, the ideal curve for a classifier is to be close to the top left: you want a classifier that produces a high recall (true positive rate) while keeping a low false positive rate. We need to be careful of overfitting though. An overfit model will have essentially rote-learned the dataset and will show a great-looking ROC curve for training data but not for test data. The point closest to the top left might be a better operating point than the one chosen by default. By allowing some True Positives to be missed, we will reduce the number of False Positives a lot.

By examining the graphs below, we can see that most of the classifiers are doing well, as long as we're cautious not to overfit the model. Extra Trees Classifier did not do as well as the other models, which is evident by the slightly steeper curve (its ROC score was around 0.88). XGBoost had the highest ROC curve (around 0.949). Followed by Gradient Boost (0.947), and Random Forest (0.946). The ROC curve is useful in our case, because we have a relatively unbalanced dataset.

*Fig 11: ROC Curves across Classifiers*





### Precision-Recall Curves

The precision-recall curve shows the tradeoff between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

The closer a curve stays to the upper-right corner, the better the classifier. A point at the upper right means high precision and high recall for the same threshold. The curve starts at the top-left corner, corresponding to a very low threshold, classifying everything as the positive class. Raising the threshold moves the curve toward higher precision, but also lower recall. Raising the threshold more and more, we get to a situation where most of the points classified as being positive are true positives, leading to a very high precision but lower recall. The more the model keeps recall high as precision goes up, the better.

In general, the curves appear to be relatively flat and become steeper towards the bottom. It is evident by the graph that the Extra Trees classifier performed worse than the other classifiers in terms of F1 score (0.65). The rest of the classifiers hovered in the range of 0.80 to 0.83. Gradient Boost had the highest F1 score (0.825), followed by XGBoost (0.822).

|  |  |
| --- | --- |
| *Fig 12: Precision-Recall Curve Comparisons across Classifiers* | |
|  |  |
|  |  |
|  |  |

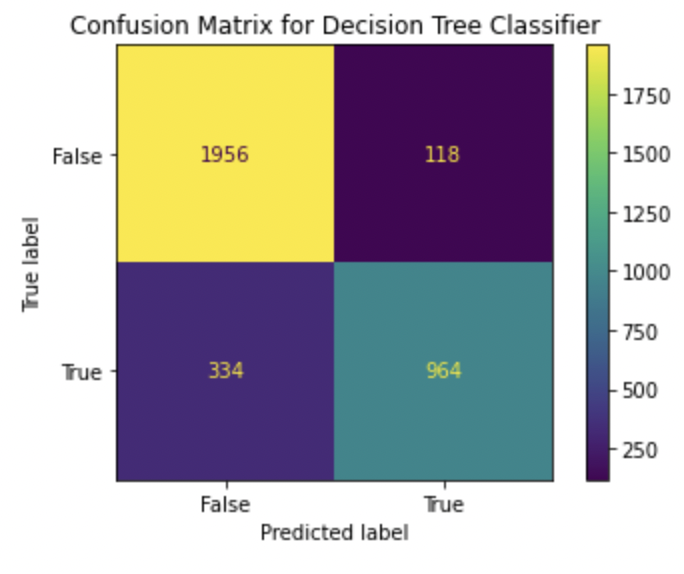
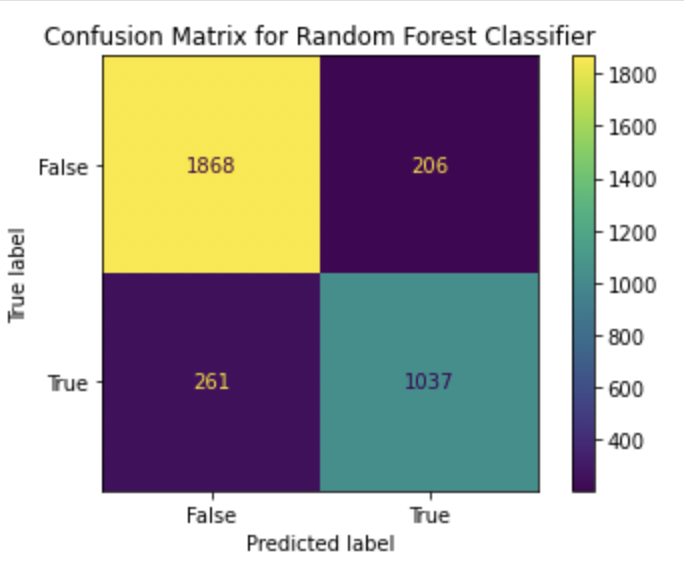
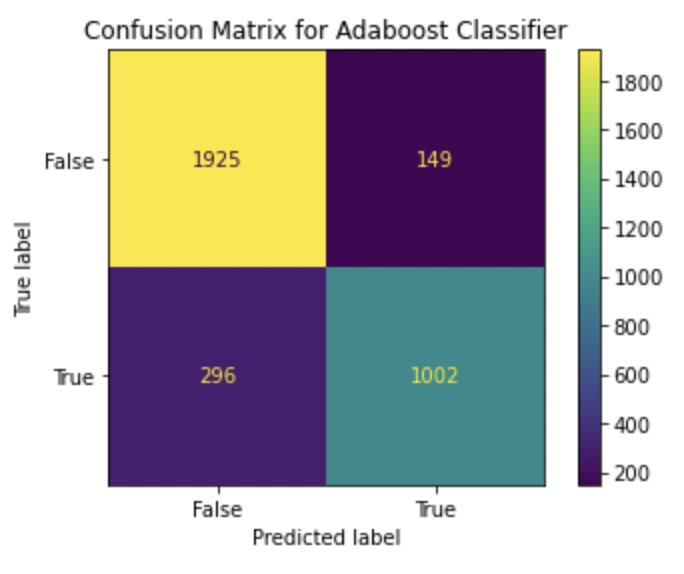
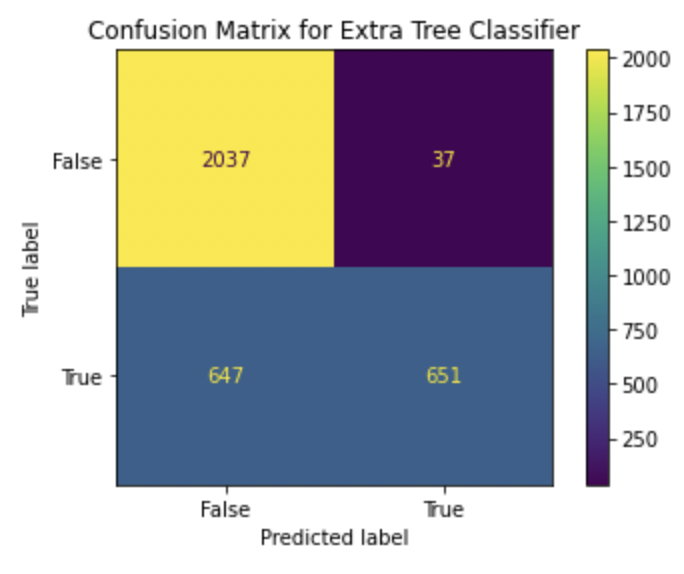
### Confusion Matrix

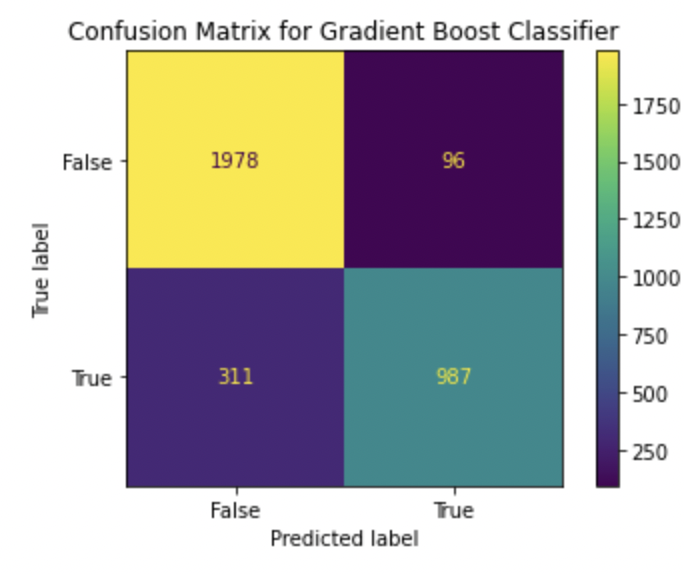
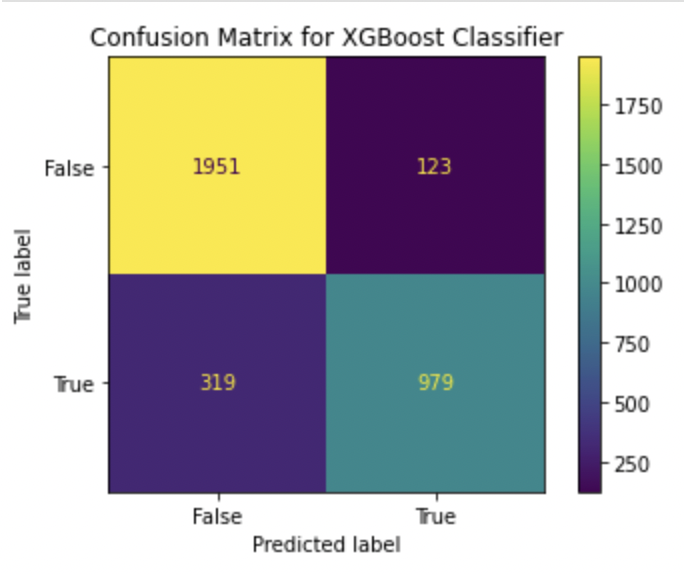
A confusion matrix is a technique for summarizing the performance of a classification algorithm. A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It gives your insight not only into the errors being made by your classifier but more importantly the types of errors that are being made. It is this breakdown that overcomes the limitation of using classification accuracy alone.

These numbers are then organized into a table or a matrix where the X axis represents predicted classes, the Y axis represents true classes. The counts of correct and incorrect classification are then filled into the table. The total number of correct predictions show up along the diagonal. The off-diagonal entries show how often that misclassification occurs.

The Extra Tree Classifier seems to have significantly more false positives (684) than the rest, which means it has more cases that it incorrectly predicted would not be fatality/major injury. For the purposes of our report, this would be the worst outcome. Gradient Booster seems to have less wrong predictions in general (407). Followed by XGBoost (442), which is not surprising because these ones also have the highest scores out of the group.

*Fig 13: Confusion Matrices across Classifiers*

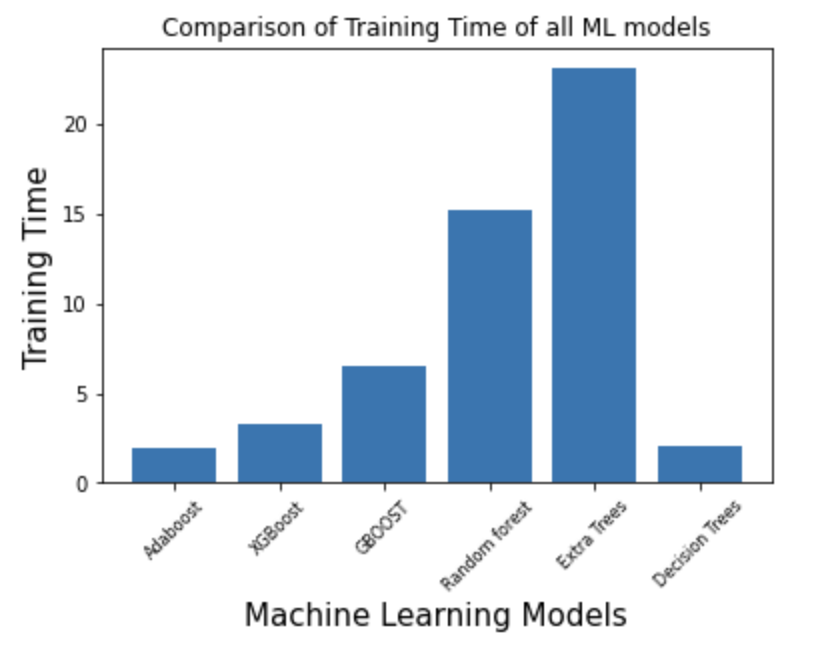
*  *

* *

**Training Time of Models**

Out of all the Models Random Forest and Extra tree machine learning model took the longest time over the Adaboost and decision trees, but their performance was not the strongest out of the group.

*Fig 14: Training Time comparison across Classifiers*



# Conclusion

In this project, road accident severity level is predicted by trying different machine learning models such as Decision Tree, Random Forest, XGBoost on the dataset provided by the City of Toronto Police. The results showed that the Gradient Boosting Classifier enhanced the decision-making process and outperformed other models with 0.87 accuracy, 0.91 precision, 0.746 recall, and 0.946 F-score using the most significant features in predicting the severity of accidents.

We learned from our dataset that most fatal/serious vehicle collisions occur at night or when it's relatively dark outside. We also learned that fatal/serious vehicle collisions are more likely to occur if the involved party is 65+ years old. The chance of getting into a major accident is twice as high if under the influence or fatigued, while the chance is 4-fold if the driver is suffering from a medical or physical disability.

A classification model such as the one demonstrated in our report could potentially be a useful tool for City of Toronto Police as first responders to triage responses to accidents. If it’s possible to predict the outcome of an accident (major accident or fatality vs minor accident), then it may be possible to dispatch the right response team from the police force, thus improving care for thousands of severely injured parties every year, while reducing unnecessary use of trauma center resources for non-severely injured parties.

To improve the model further, we would continue to work with the hyper-parameter tuning, utilizing Grid Search CV, not just Randomized Search to see if higher scores could be obtained. We would also perhaps try to include additional parameters or different ranges. We could also see if there are other datasets for Toronto vehicle collisions to see if we can reconcile some of the null values. Another option may include feature engineering via gathering additional data to supplement our dataset (e.g. including data around the overall weather conditions on the day, or the average traffic conditions at those locations within the city, or additional information around the safety of the vehicles involved).

Accident predictive models can help support decisions for the transportation division in traffic control activities or city planning, especially if additional work is undertaken to understand the key driving variables of major accidents. That is an opportunity for potential extension of the work undertaken in this report.

**References**

Aznar, Pablo. “What is the difference between Extra Trees and Random Forest?” *Quantdare*, 17 June 2020, https://quantdare.com/what-is-the-difference-between-extra-trees-and-random-forest/. Accessed 13 August 2022.

Banerjee, Prashant. “A Guide on XGBoost hyperparameters tuning.” *Kaggle*, https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning/notebook. Accessed 13 August 2022.

Benyamin, Dan. “A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System.” *CitizenNet Blog*, 9 November 2012, https://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics. Accessed 12 August 2022.

Brownlee, Jason. “How to Develop an Extra Trees Ensemble with Python.” *Machine Learning Mastery*, 22 April 2020, https://machinelearningmastery.com/extra-trees-ensemble-with-python/. Accessed 13 August 2022.

CFI Team. “Random Forest - Overview, Modeling Predictions, Advantages.” *Corporate Finance Institute*, 1 September 2021, https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/. Accessed 13 August 2022.

Gupta, Alind. “ML | Extra Tree Classifier for Feature Selection.” *GeeksforGeeks*, 1 July 2020, https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/. Accessed 12 August 2022.

ISMIGUZEL, IDIL. “Hyperparameter Tuning with Grid Search and Random Search | by Idil Ismiguzel.” *Towards Data Science*, 29 September 2021, https://towardsdatascience.com/hyperparameter-tuning-with-grid-search-and-random-search-6e1b5e175144. Accessed 13 August 2022.

Jain, Aarshay. “Gradient Boosting | Hyperparameter Tuning Python.” *Analytics Vidhya*, 21 February 2016, https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/#h2\_6. Accessed 13 August 2022.

Khandelwal, Neetika. “A Brief Introduction to XGBoost. Extreme Gradient Boosting with XGBoost! | by Neetika Khandelwal.” *Towards Data Science*, 7 July 2020, https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6. Accessed 12 August 2022.

Kurama, Vihar. “Gradient Boosting for Classification.” *Paperspace Blog*, https://blog.paperspace.com/gradient-boosting-for-classification/. Accessed 12 August 2022.

Simic, Milos. “Gradient Boosting Trees vs. Random Forests.” *Baeldung*, 25 February 2022, https://www.baeldung.com/cs/gradient-boosting-trees-vs-random-forests. Accessed 13 August 2022.

“sklearn.ensemble.GradientBoostingClassifier — scikit-learn 1.1.2 documentation.” *Scikit-learn*, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html. Accessed 12 August 2022.

“sklearn.model\_selection.RandomizedSearchCV — scikit-learn 1.1.2 documentation.” *Scikit-learn*, https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.RandomizedSearchCV.html#sklearn.model\_selection.RandomizedSearchCV. Accessed 13 August 2022.

Tests.ca. “2021 Driving Statistics: The Ultimate List of Canadian Driving Stats.” Free Driving Test Practice: Canadian Driver's Test Prep 2022. Accessed August 14, 2022. https://tests.ca/driving-statistics/.

Thailappan, Dhanya. “AdaBoost | AdaBoost Explained | Introduction to Ensemble learning.” *Analytics Vidhya*, 1 June 2021, https://www.analyticsvidhya.com/blog/2021/06/adaboost-a-brief-introduction-to-ensemble-learning/. Accessed 12 August 2022.

Transport Canada. 2022. *Canadian Motor Vehicle Traffic Collision Statistics: 2020*. [online] Available at: <https://tc.canada.ca/en/road-transportation/statistics-data/canadian-motor-vehicle-traffic-collision-statistics-2020> [Accessed 16 August 2022].

“What is Random Forest?” *IBM*, 7 December 2020, https://www.ibm.com/cloud/learn/random-forest. Accessed 12 August 2022.

# Appendix A: Exploratory Data Analysis

**Month vs. Target**

The probability of a fatal/major incident seems to be higher during the fall/winter months (September to December). August had the lowest probability. In terms of counts, there seems to be many fatal/major incidents that occur through Summer/fall.

*Fig A1: Average and Counts of Accidents by Month*

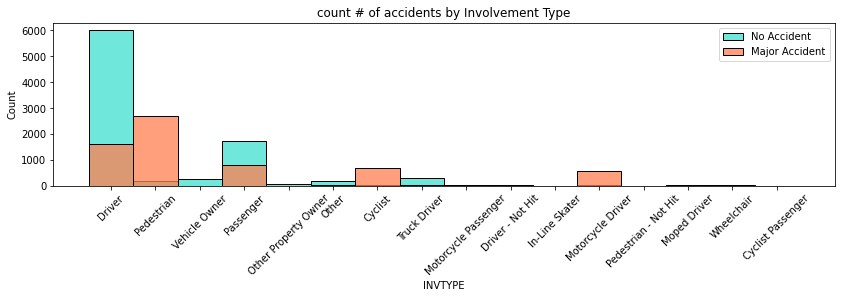
Chart, bar chart

Description automatically generated

**Involved Party vs. Target**

The likelihood for major/fatal accidents seems to be higher when cyclists, cyclist passengers, wheelchairs. Followed by moped drivers, motorcycle drivers, and pedestrians.

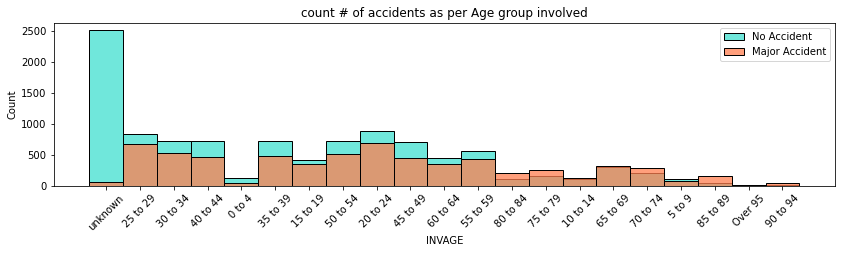
*Fig A2: Counts of Accidents by Involved Party*



**Age of Involved Party vs. Target**

There seems to be a large percentage of fatalities/major incidents involving groups 65+. Even though ages 25 to 65 seem to have higher counts.

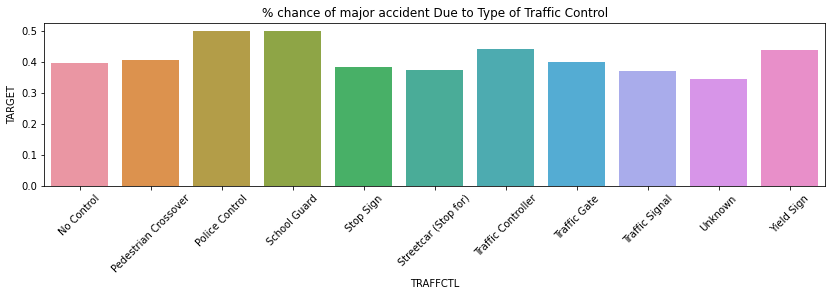
*Fig A3: Counts of Accidents by Age of Involved Party*

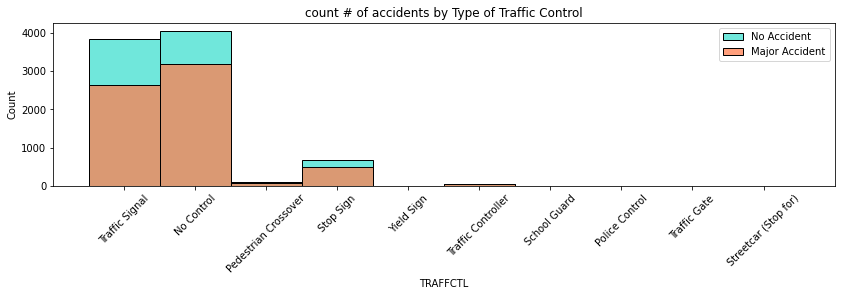


**Traffic Control vs. Target**

The probability of having a major incident/fatality seems to be higher when there is police control and a school guard is present, which is surprising cause one would think it would be the opposite. Perhaps, the areas where a police officer and school guard are already inherently more dangerous. The yield sign also appears to have a high probability of major incident/fatality.

*Fig A4: Average and Counts of Accidents by Traffic Control at Accident Site*





**Direction of Traffic vs. Target**

The probability of having a major incident/fatality seems to be higher when individuals are traveling North. Perhaps the weather conditions get more severe as they travel more North. There also seems to be a lot of unknowns.

*Fig A5: Average and Counts of Accidents by Direction*

Chart, bar chart

Description automatically generated

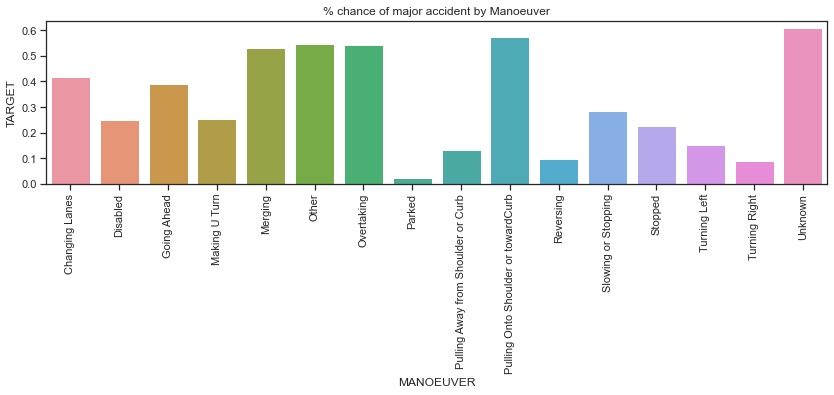
Chart, histogram

Description automatically generated

**Maneuver vs. Target**

The probability of having a major incident/fatality seems to be higher when individuals are pulling onto shoulder, overtaking, and merging.

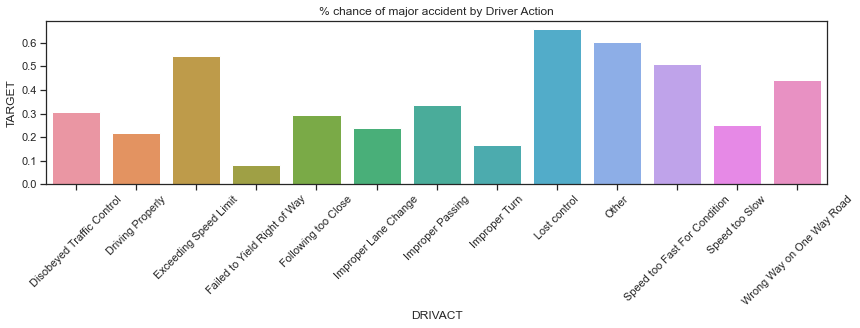
*Fig A6: Average Major Accident Rate by Manouever*



**Driver Action vs. Target**

The probability of having a major incident/fatality seems to be higher when the driver has lost control or exceeded speed limit. In terms of counts, it looks like most drivers in this dataset drive properly or other.

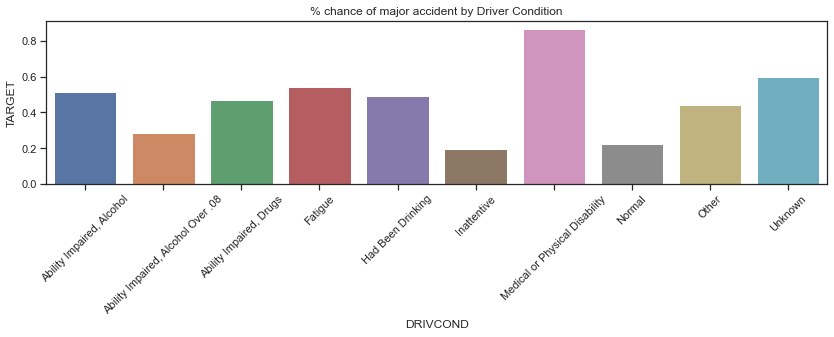
*Fig A7: Average Major Accident Rate by Driver Action*



**Driver Condition vs. Target**

The probability of having a major incident/fatality seems to be higher when the driver has medical or physical disability. Fatigue and alcohol impairment appear to also be significant. In terms of counts, there seems to be a lot of unknowns, and normal driving.

*Fig A8: Average Major Accident Rate by Driver Condition*



**Vehicle Type vs. Target**

The probability of having a major incident/fatality seems to be higher for TTC, emergency vehicles, motor cycles, moped (which is a type of scooter), and bicycles.

*Fig A9: Average Major Accident Rate by Vehicle Type*

Chart, bar chart, waterfall chart

Description automatically generated

**Light vs. Target**

The probability of having a major incident/fatality seems to be higher when its dark outside, also when its artificial dawn.

*Fig A10: Average and Volume of Major Accident Rate by Type of light*

Chart, waterfall chart

Description automatically generated

**Pedestrian Condition vs. Target**

The probability of having a major incident/fatality seems to be consistent among the pedestrian conditions. Alcohol, drugs, and fatigue appear to be slightly higher than the rest.

*Fig A11: Average and Volume of Major Accident Rate by Pedestrian Condition*

Chart, waterfall chart

Description automatically generated

\*\*\* Variables not in Report \*\*\*

**Year**

From 2006 to 2020, there were the highest numbers of major accidents reported in the initial years. However, there was a reduction in the number of Major Accidents observed over the recent years probably due to the traffic control system in the City of Toronto.

*Fig A12: Volume of Accidents by Year*

Chart, bar chart, histogram

Description automatically generated

**Cyclist Action vs. Target**

The probability of having a major incident/fatality seems to be higher when a cyclist is following too close or went the wrong way or onto a one way road. In terms of counts, there seems to be a significant number of cyclists driving properly

*Fig A13: Volume of Accidents by Cyclist Action*

# Chart, bar chart, waterfall chart Description automatically generated

**Pedestrian vs. Target**

The probability of having a major incident/fatality seems to be higher when there is a pedestrian involved.

*Fig A14: Volume of Pedestrians Involved in Collisions*

Chart, bar chart

Description automatically generated

**Cyclist vs. Target**

The probability of having a major incident/fatality seems to be higher when there is a cyclist involved.

*Fig A15: Volume of Cyclists Involved in Collisions*

Chart, bar chart

Description automatically generated

**Automobile vs. Target**

The probability of having a major incident/fatality seems to be higher when there is no automobile involved. Perhaps this is because other vehicles are more common.

*Fig A16: Number of Times a Driver was Involved in Collisions*

Chart, bar chart

Description automatically generated

**Motorcycle vs. Target**

The probability of having a major incident/fatality seems to be higher when there is a motorcycle involved.

*Fig A17: Number of Times a Motorcyclist is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Truck vs. Target**

The probability seems higher when there is no truck involved.

*Fig A17: Number of Times a Truck t is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Transit City Vehicle vs. Target**

The probability seems higher when there is no transit city truck involved

*Fig A18: Number of Times a Transit City Truck is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Emergency Vehicle vs. Target**

The probability seems higher when there is no emergency vehicle involved.

*Fig A18: Number of Times an Emergency Vehicle is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Passenger vs. Target**

The probability seems higher when there is no passenger involved.

*Fig A19: Number of Times an Passenger is Involved in Collisions*

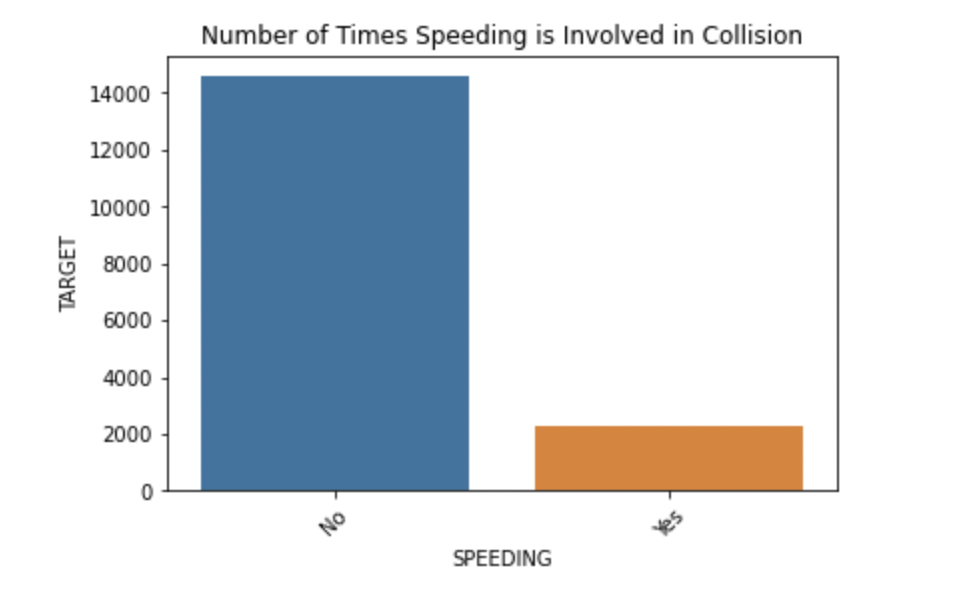
Chart, bar chart

Description automatically generated

**Speeding vs. Target**

The probability seems higher when there is no speeding involved. The discrepancy with previous EDA is most likely due to the low counts of "yes".

*Fig A20: Number of Times an Speeding is Involved in Collisions*



**Aggressive and Distracted Driver vs. Target**

The probability seems higher when there is no aggressive and distracted driver involved.

*Fig A21: Number of Times an Aggressive and Distracted Driver is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Redlight vs. Target**

The probability seems higher when there is no red light involved.

*Fig A22: Number of Times a Redlight is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Alcohol vs. Target**

The probability seems higher when there is alcohol involved.

*Fig A23: Number of Times a Alcohol t is Involved in Collisions*

Chart, bar chart

Description automatically generated

**Medical or Physical Disability vs. Target**

The probability seems higher when there is no disability involved. The discrepancy with previous EDA is most likely due to the low counts of "yes" values.

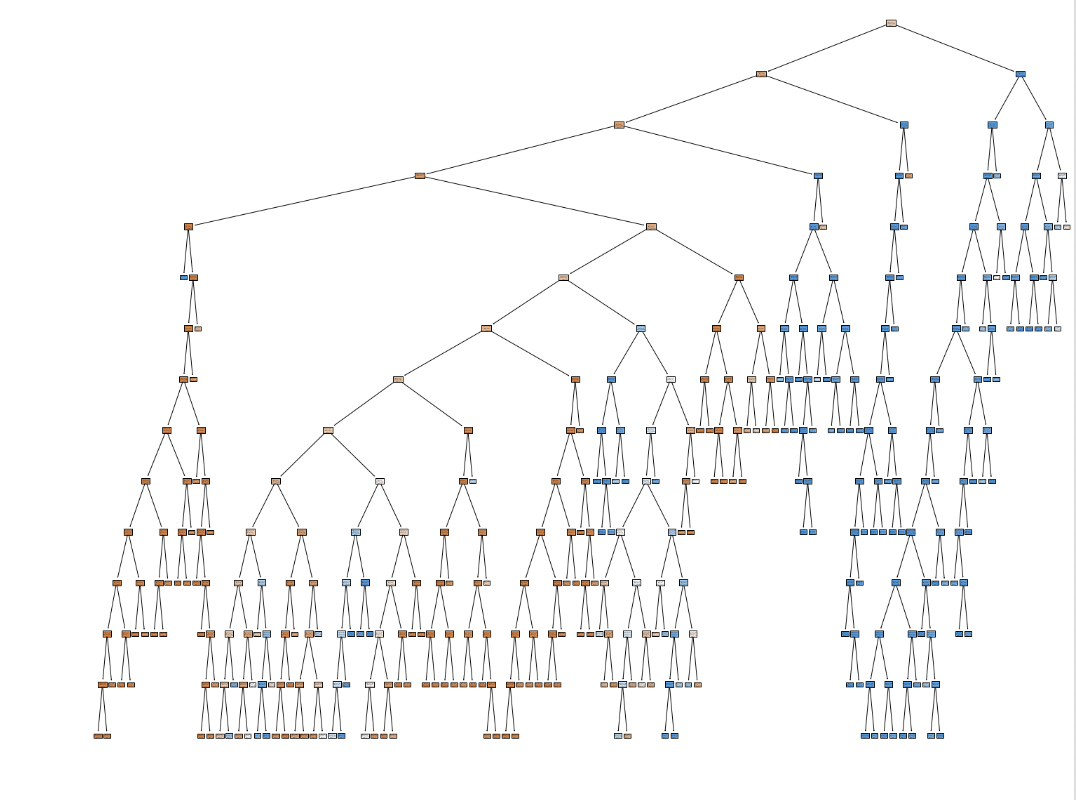
*Fig A24: Number of Times a Disability is Involved in Collisions*

Chart, bar chart

Description automatically generated

# Tree Visualization

*Fig A25: Decision Tree Visualization*



# 

# Appendix B: Predicting Fatalities

Our initial interest was in predicting not major accidents but only fatalities, which may have been a more challenging but more interesting data science question.

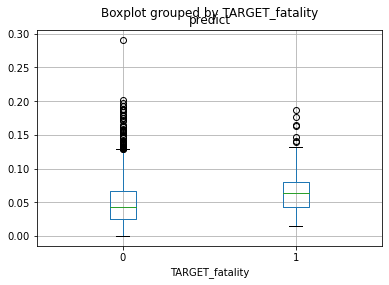
We understood that predicting fatalities would present a challenge, since injuries that involved fatalities represented only 5% of the dataset, nevertheless we proceeded and tried several algorithms. Our work with random forest and decision trees models, as well as parameter search for both, did not meet our expectations. When evaluating our models we saw that while the accuracy was very high at 95%, we quickly realized by looking at the confusion matrix that the models were doing no better than the null hypothesis and were essentially predicting that “no one dies”. We attempted different modeling techniques as well as using a subset of the data representing only pedestrians involved in accidents, where fatalities represented 16% of the dataset. Nevertheless, no amount of fine-tuning or approaches yielded any better results, essentially predicting the null hypothesis as illustrated in the confusion matrix in Figure B1 below.

|  |  |  |
| --- | --- | --- |
| *Fig B1: Confusion Matrix of Random Forest Model predicting fatalities in Pedestrians involved in accidents* | | |
|  | Predicted non-Fatality | Predicted Fatality |
| Actual Fatality | 479 | 3 |
| Actual non-Fatality | 90 | 3 |

In order to understand the problem better, we were curious to go back to basics and attempt a logistic regression in order to see how it would perform. Although this was just a curiosity and we did not go through the full extent of statistical rigour to verify the fundamental assumptions of this model such as checking for collinearity of the variables, we wanted to see whether the findings above that the null hypothesis cannot be rejected would be corroborated by a fundamental model.

Here we reached a similar conclusion: a pseudo r-squared value of less than 0.05 (meaning less than 5% of the dependent variable can be explained by the model), and within the box plots of predictions vs actuals we saw limited predictive power potential. This confirmed our idea that predicting fatalities was not a feasible data science approach due to the sparseness of this indicator within the data set.

*Fig B2: Box Plot of Predicted vs Actual indicators for fatalities using Logistic Regression*



We concluded that this interpretation of our initial hypothesis - that we will be able to predict a fatality outcome of an accident - was wrong and that the null hypothesis cannot be disproven and given the data and modeling we’ve attempted. Nevertheless, we were not dismayed, and our overall intuition that we can predict the outcome of an accident persisted, so we proceeded with some other approaches for tackling this problem.

This first exploration is now moved to the appendix as it was a stepping stone to our understanding of this dataset and the outcomes of this report.

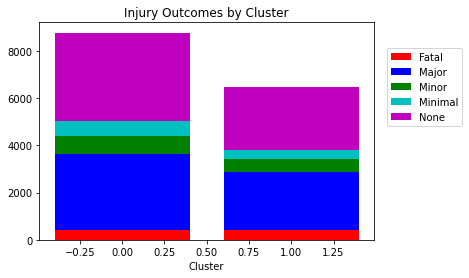
# 

# Appendix C: Clustering

Another one of our curiosities within this dataset was around whether it was possible to do some unsupervised learning to see whether any underlying structure of the dataset would become evident, and whether that underlying structure may be relevant to the injury outcome of the accident. Using the cleaned up data from previous modeling exercises, we ran a k-means model and determined the optimal number of clusters to be 2 clusters using the elbow method. We visualized the clusters compared to their injury outcomes to see whether a natural pattern would emerge, in the stacked bar chart in *Fig X*.

We saw that the two clusters outcome of the k-means model had few differences between the share of collision outcomes, and that they both contained a similar share of major accidents (38% and 39% for the two clusters). We thus concluded that the natural pattern approach was also not feasible given our dataset.

*Fig C1: Optimal number of clusters using k-means algorithm  
compared to the accident outcomes within those clusters*



# 

# Appendix D: Attribute List

\_id: Unique row identifier for Open Data database

ACCNUM: Accident Number

YEAR: Year Collision Occurred

DATE: Date Collision Occurred

TIME: Time Collision Occurred

HOUR: Hour Collision Occurred

STREET1: Street Collision Occurred

STREET2: Street Collision Occurred

OFFSET: Distance and direction of the Collision

ROAD\_CLASS: Road Classification

DISTRICT: City District

WARDNUM: City Ward Identifier, will show multiple if collision occurred along a border

DIVISION: Police Division(s), will show multiple if collision occurred along a border

LOCCOORD: Location Coordinate

ACCLOC: Collision Location

TRAFFCTL: Traffic Control Type

VISIBILITY: Environment Condition

LIGHT: Light Condition

RDSFCOND: Road Surface Condition

ACCLASS: Classification of Accident

IMPACTYPE: Initial Impact Type

INVTYPE: Involvement Type

INVAGE: Age of Involved Party

INJURY: Severity of Injury

FATAL\_NO: Sequential Number

INITDIR: Initial Direction of Travel

VEHTYPE: Type of Vehicle

MANOEUVER: Vehicle Manouever

DRIVACT: Apparent Driver Action

DRIVCOND: Driver Condition

PEDTYPE: Pedestrian Crash Type - detail

PEDACT: Pedestrian Action

PEDCOND: Condition of Pedestrian

CYCLISTYPE: Cyclist Crash Type - detail

CYCACT: Cyclist Action

CYCCOND: Cyclist Condition

PEDESTRIAN: Pedestrian Involved In Collision

CYCLIST: Cyclists Involved in Collision

AUTOMOBILE: Driver Involved in Collision

MOTORCYCLE: Motorcyclist Involved in Collision

TRUCK: Truck Driver Involved in Collision

TRSN\_CITY\_VEH: Transit or City Vehicle Involved in Collision

EMERG\_VEH: Emergency Vehicle Involved in Collision

PASSENGER: Passenger Involved in Collision

SPEEDING: Speeding Related Collision

AG\_DRIV: Aggressive and Distracted Driving Collision

REDLIGHT: Red Light Related Collision

ALCOHOL: Alcohol Related Collision

DISABILITY: Medical or Physical Disability Related Collision

POLICE\_DIVISION: Toronto Police Service Division

HOOD\_ID: City of Toronto Neighbourhood Identifier

NEIGHBOURHOOD: City of Toronto Neighbourhood Name

ObjectId: Object ID (Unique Identifier)