MACHINE LEARNING

ROAD COLLISION ANALYSIS AND PREDICTION SUMMARY REPORT

horizontal line

# 

# 

**Objective:** Crash injury severity prediction is an exciting area of study in traffic safety. Traffic safety has always been an important issue in sustainable transportation development, and the prediction of traffic accident severity remains a crucial challenging issue in the domain of traffic safety. Traffic accidents are a leading cause of death despite the development of traffic safety measures. The goal 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).

**Data Preparation:**https://open.toronto.ca/dataset/motor-vehicle-collisions-involving-killed-or-seriously-injured-persons/

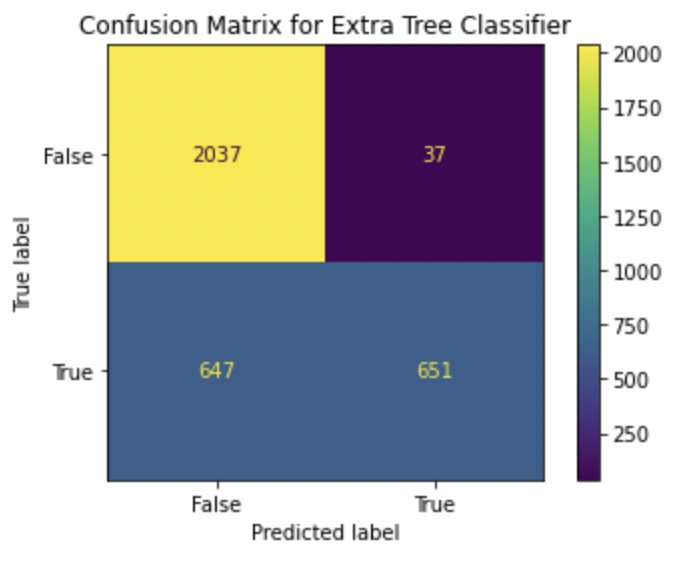
Our dataset comes from Toronto’s open data catalog. One challenge we faced is that we did not have a clear target variable already embedded in the dataset. We had 3 different columns that signified that there was fatality or major incident, which was *acclass*, *fatal\_no,* and *injury.* We decided to use the injury to create our target variable, because the distinction between death and major injury was clearer. 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. Another challenge was the potential for Data leakage. 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. There were also certain features in the dataset that 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/no, the nulls were filled with “no”. If it was a categorical column, it was filled with “unknown” or “other”. We noticed that the feature *Date* was not useful on its own, given the string format. Hence, we broke it down into the year, the day of the week, month of the year, and Season, which allowed us to do more meaningful EDA. We removed *ObjectId*, *ACCNUM* since they were unique identifiers and they did not add much value to the modeling or EDA . *Geometry* did not add much value, since we had other location features. We built a pipeline to clean up the dataset, which included OneHotEncoder for all categorical variables, MinMaxScaler to normalize any numerical variables, and Pipeline and ColumnTransformer to put together all the preparation steps.

**Exploratory Data Analysis (EDA):** We began our EDA, by examining some of the time variables against the target variable: *hour* , *time of day*, *month*, and *season*. We found that the likelihood of fatal/major accidents increases during the evening hours. The probability of a fatal/major incident seems to be higher during the winter/fall months. We then carried our analysis of time of day to the district feature, and determined that Toronto East York shows the most serious accidents regardless of Hour of Day. Next, we examined *road surface conditions*, and found a significant number of accidents have occurred during dry road surface conditions instead of ice, loose snow, slushing or packed snow conditions. Wet conditions had the second highest number of accidents. Environmental conditions also played a role, since the likelihood of fatal/major incidents was higher during drifting snow and strong winds. Another, interesting finding was that the likelihood of a fatal/major incident was higher when it involved groups age 65+. We also found that the likelihood of fatal/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. Through EDA, we also determined that our target variable is slightly unbalanced, which means accuracy may not be the most useful scoring metric to use.

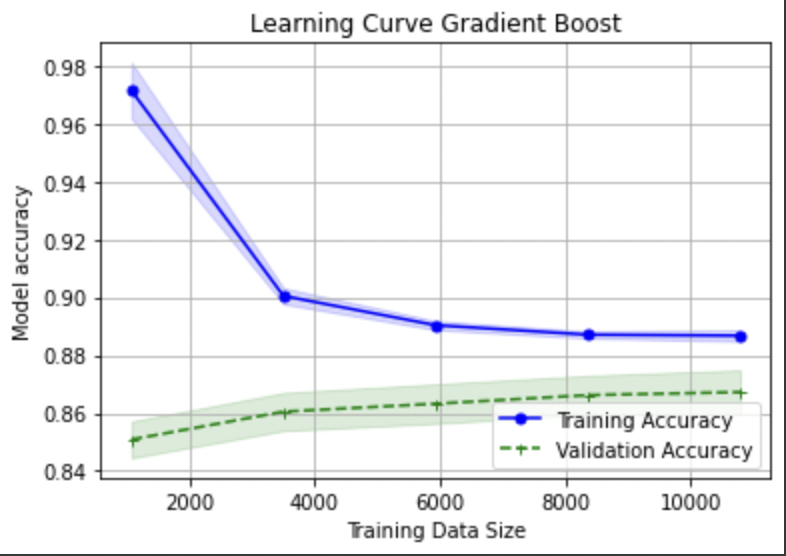
**RandomForestClassifier:** Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction (IBM, 2022). Random 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. The hyper-parameters selected were *criterion*, *max\_depth*, *max\_features*, *min\_samples\_leaf*, *min\_samples\_split*, and *n\_estimators*. They were optimized by using Randomized Search CV, which 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%).

***DecisionTreeClassifier:*** Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. A tree can be seen as a piecewise constant approximation (scikit-learn, 2022). 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). The hyper parameters were selected by doing a comparison of Grid Search and Randomized Search CV, and selecting the best performer. The hyper-parameters that were tuned were *criterion*, *max\_depth*, *min\_samples\_leaf*, and *splitter*. Grid Search outperformed Randomized Search CV. Grid Search also 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%).

***AdaboostClassifier:*** AdaBoost is an ensemble learning method, which helps combine multiple “weak classifiers'' into a single “strong classifier”. Ada works by putting more weight on difficult to classify instances and less on those already handled well. 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). AdaBoost was tuned with Randomized Search CV, with the following hyper-parameters were selected *learning\_rate*, and *n\_estimators*. 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. From the learning curve we found that for training sample size less than 3800, the difference between training and validation accuracy was 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. However, you can also see that when the training dataset passes 8200 it starts to diverge again.



***ExtraTreeClassifier:*** 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). 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. Extra trees were selected because it’s less computationally expensive than a Random Forest. Extra trees were tuned with Randomized Search CV, and the following hyper parameters were selected *criterion*, *max\_depth*, *max\_features*, and *min\_samples\_leaf*. It performed worse with the hyper-parameter tuning, this may be due to insufficient range of parameters. It also performed worse, relative to the other models (accuracy score: 79.72%, F1 score: 65.56%, and ROC curve: 89.69%). It also had the largest number of incorrect predictions, relative to the other models.

***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. Gradient Boosting was selected because it is known to provide high accuracy and flexibility in handling imperfect data, which was relevant to our dataset. The drawback is that gradient boosting can overfit on the training set due to its attempts to minimize all errors, and can be computationally expensive (Kurama). Gradient Boost was tuned with Randomized Search CV, and the following hyper parameters were selected: *max\_depth*, *n\_estimators*, and *random\_state*. 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%). It also outperformed the other models. The learning curve was similar to adaboost. For training sizes greater than 3800, the model is better. However, past 3800 the training accuracy and validation accuracy are also wider apart than for AdaBoost.

***Extreme Gradient Boosting 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). 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. XGBoost was tuned with Randomized Search CV, and the hyper-parameters selected were *max\_depth*, *n\_estimators*, and *random\_state*. XGBoost performed better with hyper-parameters tuning, in terms of accuracy (86.89% vs. 86.60%), worse in terms of F1 score(81.58% vs. 81.95%), and better in terms of ROC curve(94.74% vs. 94.48%) .

**Conclusion:** Obtained results demonstrate that the Gradient Boosting Classifier enhanced the decision-making process and outperformed other models with 0.87 accuracy, 0.91 precision, 0.76 recall, 0.829 F1-score, and 0.947 roc using the most significant features in predicting the severity of accidents. One insight we took away is that 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. We also learned that fatal/serious vehicle collisions are more likely to occur if the driver is 65+ years old. 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).