CS770 Machine Learning

Final Project: Predicting Traffic Incident Duration in the Sydney Metropolitan Area

12/05/2025

Submitted by,

Joshua Kluthe – S694k397

Table of Contents

Abstract

Prediciting traffic incidents is an important public safety goal around the world. Having an accurate assessment of the expected numbers, durations, and locations of vehicular accidents allows services to plan and allocate resources to better respond to both emergency and commonplace incidents. Many machine learning techniques have been employed to try to predict these traffic incidents, with a recent paper titled “Predicting the duration of traffic incidents for Sydney greater metropolitan area using machine learning methods” [1] being of particular note. In that study, the authors used a number of machine learning algorithms to improve traffic incident duration prediction, finding their best results using XGBoost, followed by Random Forest. In this paper, I used the dataset utilized from that study and performed a number of machine learning analyses including linear regression, logistic regression, decision trees, and both linear and RBF kernel SVM, in order to both replicate the study findings and compare and contrast them to other methods. My results differed from the original study, in that XGBoost was not the best performing model. Instead, I found basic Logistic Regression and Linear SVM to be the best performing models, with RBF kernel SVM having similar metrics but taking much longer to train and describe.

1. Introduction

1.1 Background and Motivation

Traffic incidents such as crashes, breakdowns, and other hazards are a significant source of pain for anyone living in a car-centric area. These issues cause delays, economic strain, injuries and even death that propagate throughout the community. These incidents have to be attended to by police, emergency medical services, and maintainence services, making the ability to predict the frequency, severity, and duration of such incidents a high priority for traffic and city management authorities.

Traditionally forecasting such incidents has relied on the experience and best guesses of seasoned professionals in the field. Although this knowledge and experience is invaluable, with the advent of extensive traffic monitoring and data collection, there is a significant opportunity to supplement this hard-gained wisdom with machine learning data analysis methods.

1.2 Objectives

The objective of this project is to take the dataset from the “Predicting the duration of traffic incidents for Sydney greater metropolitan area using machine learning methods” [1] and try to replicate it’s findings when using Random Forest and XGBoost algorithms, and also compare these results to a larger set of machine learning models to compare and contrast their weaknesses and strengths. Specifically, I will:

- Perform EDA and data preprocessing on the initial study dataset, then clean and transform it into a dataset that is ready to be used by both linear regression and classification models to use.
- Use different methods to both attempt to predict the duration time from the continuous data, and classification methods to classify incidents as either “short” (< 30 min) or “long” (>= 30 min).
- Quantify performance using metrics appropriate to each task (MAE, RMSE, R^2 for regression; Accuracy, ROC-AUC, and confusion matrices for classification).
- Identify influential features and interpret model behavior using permutation importance and SHAP values.

1.3 Significance and Applications

This work is of high value to real world applications. Quantifying where and when traffic incidents occur, and how long they last, can support a number of logistical decisions:

- Police and emergency responders can better distribute and schedule staff.
- Traffic control authorities can better estimate traffic flow and congestion, which can be used to deploy resources and inform the public of delays.
- City planning authorities can use this information to design transit corridors to be faster and safer.

This is, however, a complex and multidimensional problem, and more work is needed. However, the results from this study and my own research shows that machine learning methods can be applied to this problem, and as more data is gathered and further studies are done, the insights gained will likely prove valuable on all of these points.

2. Data

2.1 Origin

The dataset for this experiment was sourced from the GitHub repository from the original study, “Predicting the duration of traffic incidents for Sydney greater metropolitan area using machine learning methods” [1]. According to the paper, it was compiled from the following sources:

- A dataset from described in "Data in Brief 51 (2023)". This is a public dataset covering 5.5 years of traffic incidents within the Sydney Greater Metropolitan Area.
- Data on road network metrics, bus netork details, land use and socioeconomics obtained from OpenStreetMap.
- The Australian Bureau of Statistics.

All of these data were compiled into a dataset called `merged.csv` available through the GitHub repo associated with the study. This dataset contains a total of 85611 observations across 88 features.

2.2 Description

The features described in the data cover many aspects of the time, geography, and emergency personelle responses to incidents, such as:

- Longitude, latitude, area codes, number of lanes, number of lanes affected, and distance from the central business district.
- Types of vehicles (both primary and secondary) involved in the incident, and traffic volume at the time of the incident.
- Codes describing the emergency responses instigated by the incident.
- Hour, day, month and time of the incident.
- The main feature being targeted in the original study, and for my analysis of the study, is “duration”. This is measured in minutes, and represents the length of time between the start and end of a traffic incident.

2.3 Problems Encountered

I identified a number of issues with this dataset that could cause problems for machine learning algorithms during my inspection. These include the following:

- There appears to be duplicated data regarding area codes, area names, and other features. Not all of these features are perfect duplicates, but they appear on visual inspection to be highly corelated, which would cause issues with model training.
- A large number of the features in this dataset are categorical, and many of them have very high cardinality that would be difficult for many models to make use of.
- Additionally, one column, “Actual_Number_of_Lanes “, was found to have trailing whitespace, causing problems when referencing it.

These issues motivated the decisions I made in EDA and preprocessing, as described in section 3.1.

3. Methodology

3.1 EDA and Preprocessing

I performed EDA and preprocessing on the original “merged.csv” dataset with the goal of producing a clean dataset that was ready to be imported and used by all of the models considered in this paper with minimal additional preprocessing steps. The preprocessing is performed in the notebook “clean_data.ipynb”, available at the dedicated GitHub repo for this project at https://github.com/jjmkluthe/CS770_final_project/tree/main. The resulting cleaned and augmented dataset was then saved to the file “sydney_traffic_incidents_clean.csv”.

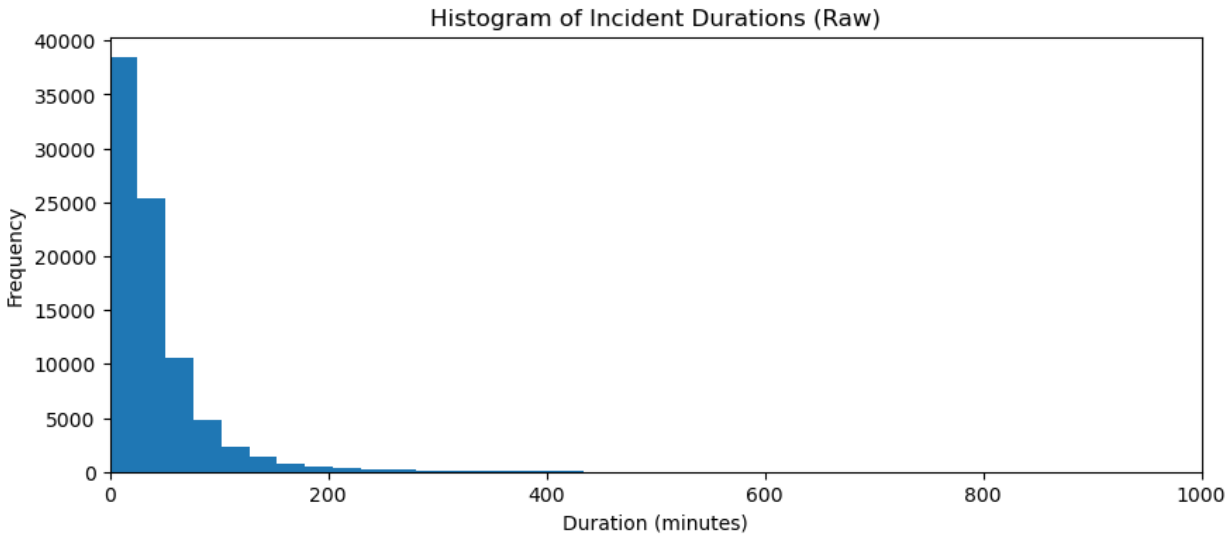
The first step was to remove any trailing whitespace in column headers, as it was discovered that this existed in at least one column. After exploring the column headers in code, it was found

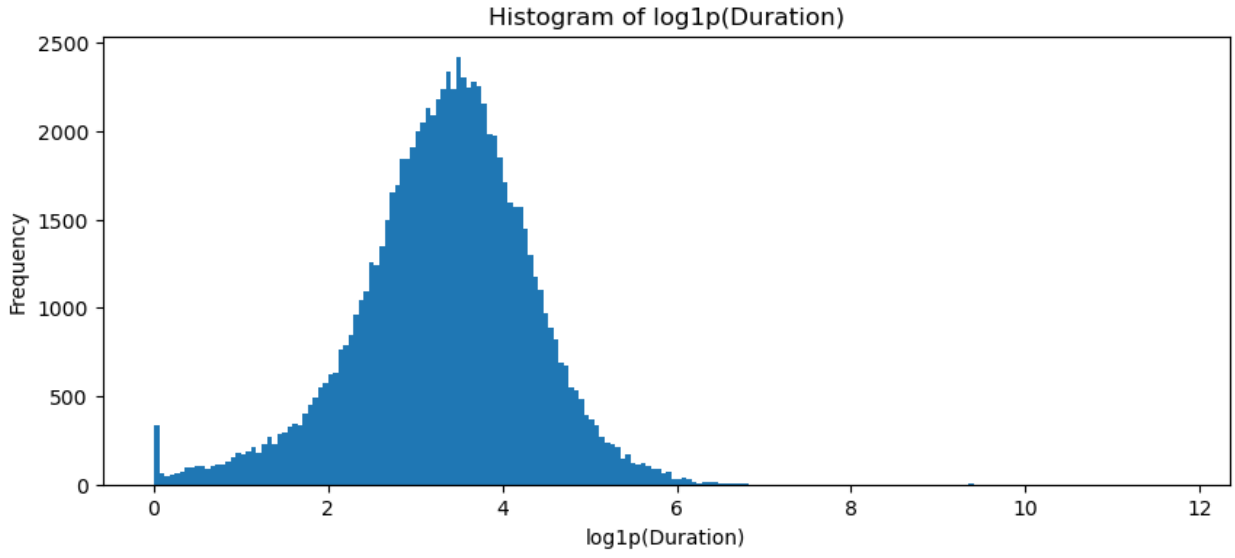
that this issue only affected a single column, and the whitespace was stripped and the column header was renamed.

In the next step, I examined the “duration” column to see it’s general shape and if there were any significant outliers skewing the data. A quick inspection of the column shows that there are likely outliers in the target:

```
count     85611.000000
mean         44.863947
std         454.097287
min           0.000000
25%          14.943592
50%          28.719767
75%          51.747558
max      127729.214500
Name: duration, dtype: float64
```

The extreme value of the maximum is clearly out of step with the rest of the data. For a closer look, I generated a histogram of the raw “duration” data, and another of the log transformation:





This clarifies that most of the “duration” data is well distributed, but we have both extremely large outliers and some durations of 0 minutes. Traffic incidents by nature must take longer than 0 minutes, so it is likely these datapoints represent mistakes in the records. As an arbitrary cutoff, I removed all records with 0 duration and all records > 500 minutes:

```
# Count clearly invalid and extreme duration values
n_nonpositive = (df_raw["duration"] <= 0).sum()
n_long = (df_raw["duration"] > 500).sum()

print("Durations <= 0:", n_nonpositive)
print("Long outliers > 500 minutes:", n_long)
```

```
Durations <= 0: 92
Long outliers > 500 minutes: 88
```

These outliers are a tiny fraction of the total records available, and can be safely removed. I also inspected the full dataset for NaN values, and found there were 2807 rows containing NaNs. Again, this is a very small percentage of the total data available, and these were dropped as well.

The next problem to address was that of duplicate or highly correlated features. On visual inspection of the dataset, I found that several columns, particularly those like “SA2_CODE21” and “SA3_CODE21” appeared to contain much of the same information. I inspected the data in code to find these columns, as this duplicate data would cause problems in modeling, with the following results:

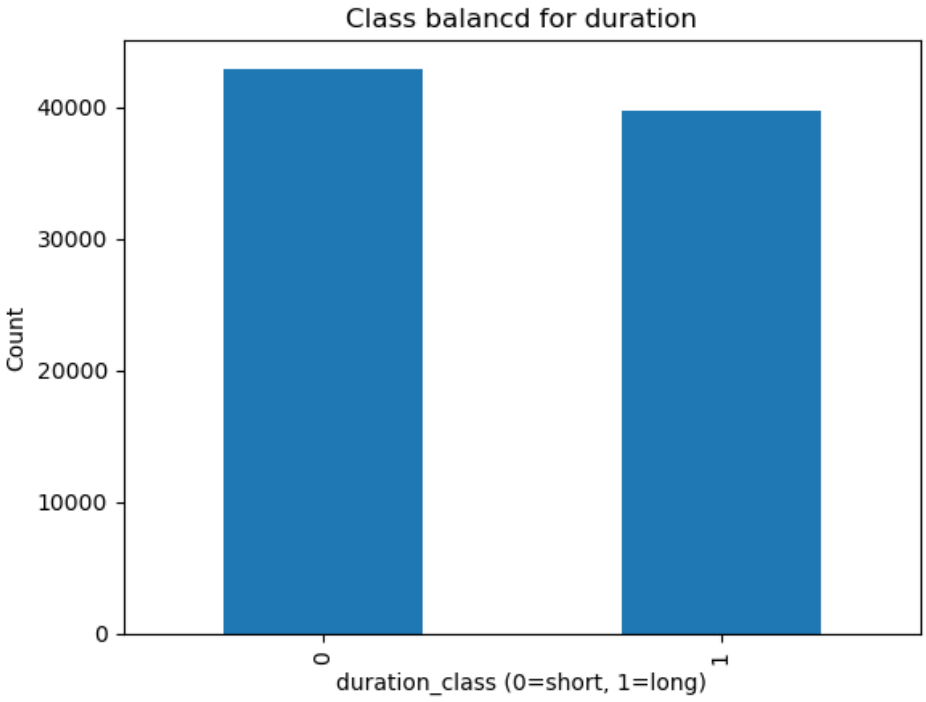
```
Highly correlated columns > 0.99:
  SA2_CODE21      SA3_CODE21      corr=0.9999999994
  SA2_CODE21      SA4_CODE21      corr=0.9999956414
  SA2_CODE21      SA4_NAME21      corr=0.9999956414
  SA2_CODE21      0_ZID           corr=1.0000000000
  SA3_CODE21      SA4_CODE21      corr=0.9999956448
  SA3_CODE21      SA4_NAME21      corr=0.9999956448
  SA3_CODE21      0_ZID           corr=0.9999999994
  SA4_CODE21      SA4_NAME21      corr=1.0000000000
  SA4_CODE21      0_ZID           corr=0.9999956414
  SA4_NAME21      0_ZID           corr=0.9999956414
  AREASQKM21      1_Area          corr=1.0000000000
```

I kept the columns “SA2_CODE21” and “AREASQKM21”, and dropped the following columns as being essentially duplicates:

- SA3_CODE21
- SA4_CODE21
- SA4_NAME21
- 0_ZID
- 1_Area

After performing these cleaning steps, the dataset now contains 82624 records across 83 columns, remaining a robust sample for my experiments. One further step remains, however. The dataset contains the target “duration”, which is made up of continuous values measured in minutes. This is appropriate for tasks such as linear regression, but for classification experiments I want the data to be in a categorical format, with durations <= 30 minutes being classified as short and > 30 minutes as long. To do this, I created a new column called “duration_class”, transforming the data from “duration” into 0 for short durations, and 1 for long durations.

With this step complete, I then checked the balance of the target “duration_class” variable:



This is an excellent result! The target classes are almost ideally balanced, and further preprocessing steps are needed at this stage. The result was saved to “sydney_traffic_incidents_clean.csv” to be imported into the modeling notebooks.

3.2 Model Descriptions

This project evaluates a variety of machine learning models in predicting either traffic incident duration for linear regression, or incident class (short vs long) for classification algorithms. Due to the focus on using multiple algorithms to assess the data rather than tuning the best possible model, default hyperparameters were used in most cases. Some hyperparameter tuning was done informally, but a detailed experimental hyperparameter tuning process is not included in these experiments. A brief description of the models used is as follows:

- Linear Regression: Linear regressions was used as a baseline model for predicting the “duration” target. It assumes a linear relationship between features and the target variable, with a continuous target output. I performed both simple regression and an XGBoosted version of regression.
- Logistic Regression: Logistic regression is our baseline binary classifier, and was used to target the “duration_class” target.
- Decision Trees: A decision tree splits the features into hierarchical trees to predict the target, and can be used to easily infer the most important features affecting model performance.
- SVM (Linear and RBF): Linear SVM uses a linear decision boundary, while the RBF kernel extends this to nonlinear feature mappings. It requires feature scaling and category encoding in order to function properly.
- Random Forest: Random forest uses a “forest” of simple decision trees and takes a “majority vote” from them to predict the target. The original study found this to be one of the strongest models.
- XGBoost: XGBoost is another tree model that uses gradient boosting to increase computational efficiency and model output. The original study found this to be the most effective mode.

3.3 Model Architecture and Parameters

Each model was trained on the cleaned dataset using an 80/20 train/test split with random_state=42 for reproducibility. Models were configured with commonly used and recommended parameter, with some informal tuning for model efficiency. In this paper I do not systematically explore the effects of hyperparameter tuning, focusing instead on comparing and contrasting baseline models. The core configuration for each model family is summarized below.

In each case, whether “duration” or “duration_class” was used as the target, both variables were dropped from the feature set to avoid overfitting since the two columns are so closely correlated. Additionally, many of the features exhibit categorical values with very high cardinality. This posed a problem for several models, and in those cases high cardinality (cardinality > 50) were dropped.

For linear models, the typical StandardScaler was used for normalizing features. SVM and Logistic Regression models also required one-hot encoding to avoid treating higher values of

categorical features as being “greater” than lower valued categories. Random Forest and XGBoost are able to interpret these categories, however, so normalization and one-hot encoding was not required for these models.

3.4 Interpretability

Model interpretability was assessed using a combination of:

- Feature Importance from Random Forest and XGBoost
- Permutation Importance
- SHAP values for XGBoost via the model’s built in interpretation module

These tools allowed a deeper look into which features had the most impact on the models ability to predict traffic incident duration.

4. Results and Discussion

4.1 Regression (Simple and XGBoosted)

For this experiment, I compared Simple Linear Regression and an XGBoosted version, XGBRegressor, to see how well these models could predict the actual continuous duration of traffic incidents. In both cases, the high cardinality categorical features were dropped from the training set, and the remaining variables were one-hot encoded. Additionally, the StandardScaler was applied to the numeric column values. The results are below.

Simple Linear Regression:

```
Training time: 3.25 seconds
MAE: 25.98973634487247
RMSE: 41.65279368704854
R2 Score: 0.185416093757792
```

XGBRegressor:

```
Training time: 1.10 seconds
MAE: 27.816036021811282
RMSE: 44.648345221062314
R2 Score: 0.06403784211544539
```

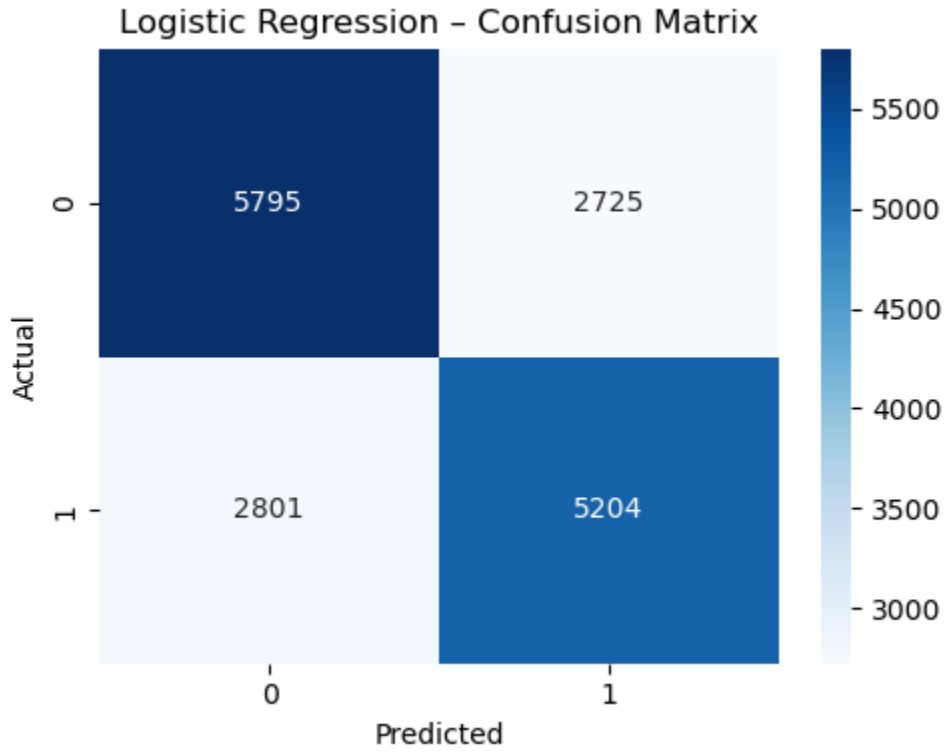
Surprisingly, Simple Linear Regression performed better than XGBRegressor. However, neither method demonstrated a strong ability to predict actual duration times.

4.2 Logistic Regression

As with Linear Regression, high cardinality columns were dropped, with the remaining categorical columns being one-hot encoded and the StandardScaler being applied to numeric column values. The model was then evaluated on precision, recall, and F1-Score:

```
Accuracy: 0.6655975794251134

Classification Report:

              precision    recall  f1-score

           0       0.67      0.68      0.68
           1       0.66      0.65      0.65

    accuracy                           0.67
   macro avg       0.67      0.67      0.67
weighted avg       0.67      0.67      0.67
```

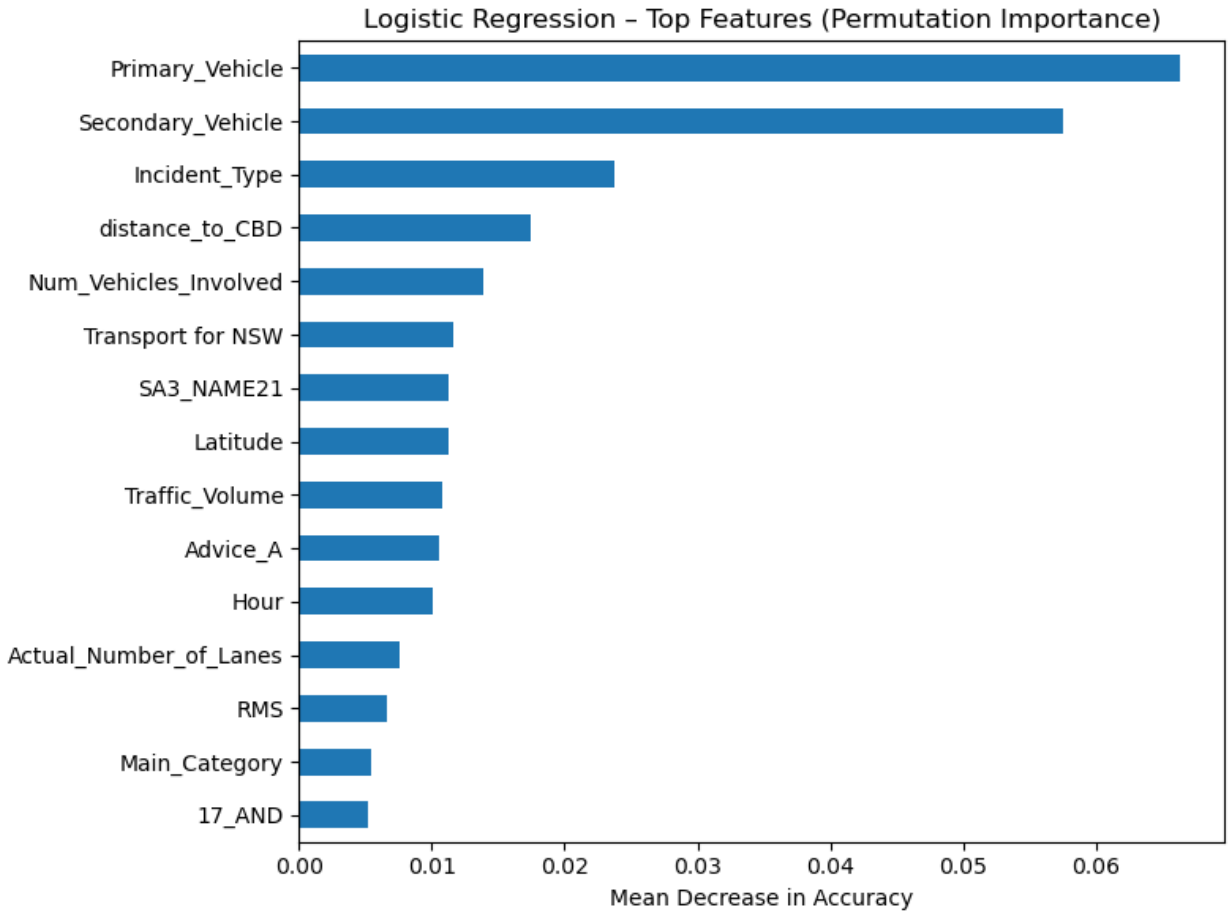
This simple baseline model actually performed quite well compared against the best results achieved in the source paper for XGBoost. Now a look at the confusion matrix:



These results are much better than I would have expected, and the model is showing roughly

equal error on each side of the long/short duration class boundary. Additionally, the training time for this model was very short, at 2.29 seconds.

Next, I will use the permutation_importance module to assess what the most important features in the model were found to be:



The Primary and Secondary vehicles were found to be far and away the most important, followed by the incident type, distance to the central business district, and the number of vehicles involved before features level off in importance. This is intuitively understandable, and will be interesting to compare to feature importance in other models.

4.3 SVM (Linear and RBF)

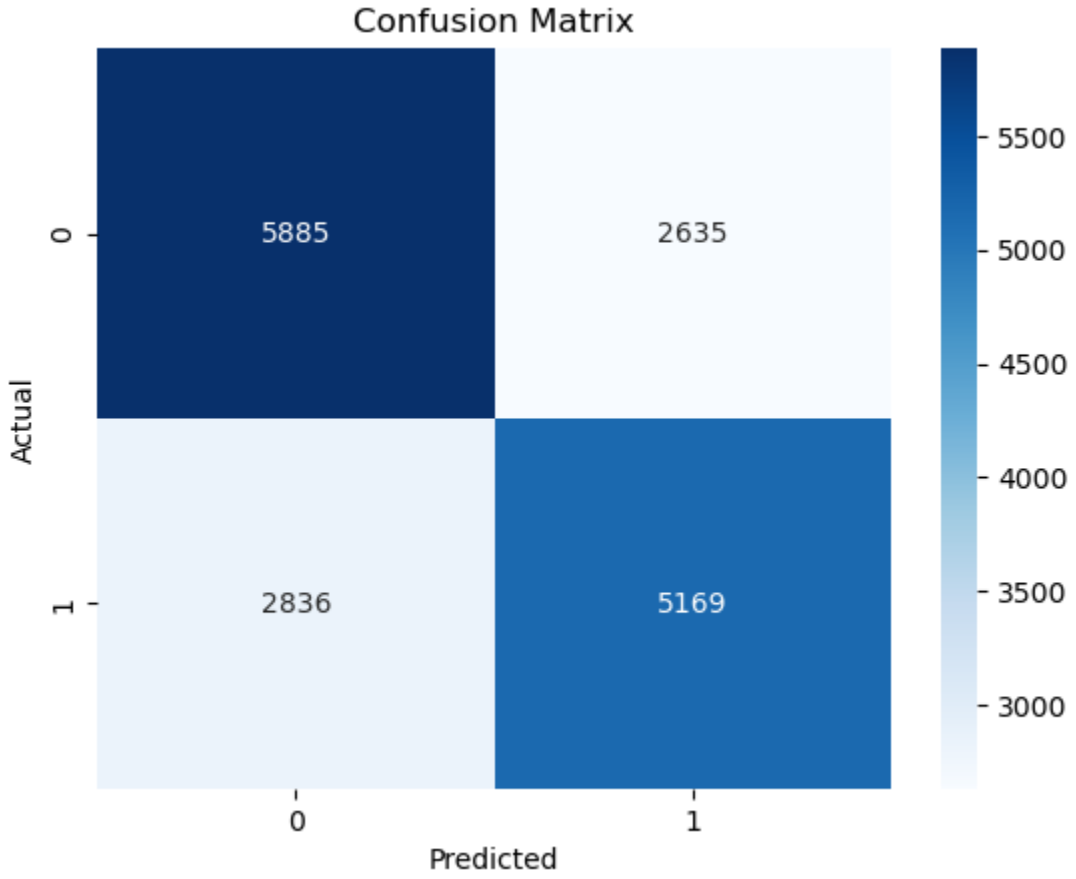
For these models I again dropped high cardinality categorical columns, one-hot encoded the remaining categories, and used StandardScaler. Additionally, I evaluated the models using the same metrics as before.

The models ended up performing very similarly, except that the Linear model was *much* faster than the RBF model, by two orders of magnitude. RBF took 900 seconds to train, compared to Linear at 9.76 seconds.

I then attempted to extract the top features from both models using permutation_importance, but this proved extremely time consuming for the RBF kernel version. After running the permutation for over two hours I abandoned this attempt, and used permutation_importance on the Linear model. This took significantly longer than simply training the model had, but was completed in a reasonable amount of time nonetheless. The results are below.

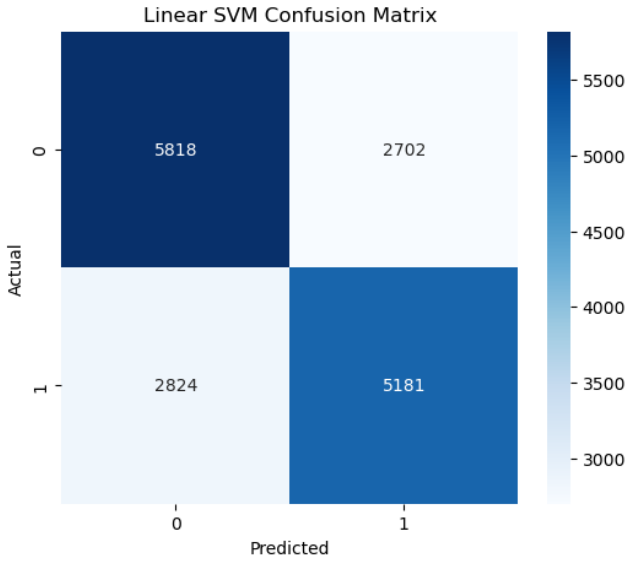
RBF Kernel:

```
Accuracy: 0.6689258698940999

Classification Report:
              precision    recall  f1-score

           0       0.67      0.69      0.68
           1       0.66      0.65      0.65

    accuracy                           0.67
   macro avg       0.67      0.67      0.67
weighted avg       0.67      0.67      0.67
```



Linear SVM:

```
Accuracy: 0.6655975794251134

Classification Report:

              precision    recall  f1-score

           0       0.67      0.68      0.68
           1       0.66      0.65      0.65

    accuracy                           0.67
   macro avg       0.67      0.67      0.67
weighted avg       0.67      0.67      0.67
```



And Feature Importance from Linear SVM:

```
Primary_Vehicle            0.066638
Secondary_Vehicle          0.057682
distance_to_CBD            0.015806
Num_Vehicles_Involved      0.013047
Transport for NSW          0.011244
SA3_NAME21                 0.011147
Traffic_Volume             0.010675
Hour                       0.010045
Latitude                   0.010033
Advice_A                   0.009489
Actual_Number_of_Lanes     0.007455
RMS                        0.006209
17_AND                     0.005241
Motorway Crew              0.005047
36_PD2MV                   0.004381
Heavy vehicle tow truck    0.004091
Incident_Type              0.003897
42_NPTtWbyTx               0.003328
Is_Major_Incident          0.003256
6_TrRL                     0.003062
dtype: float64
```

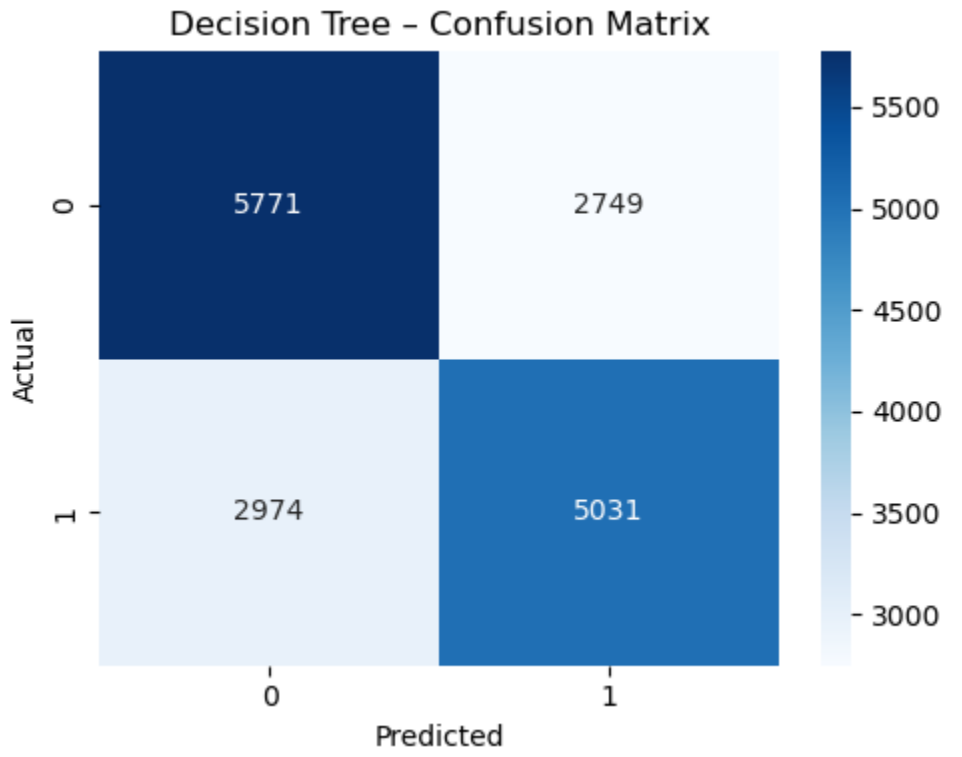
Interestingly, both the performance and the top features of these models are very much the same as normal Logistical Regression.

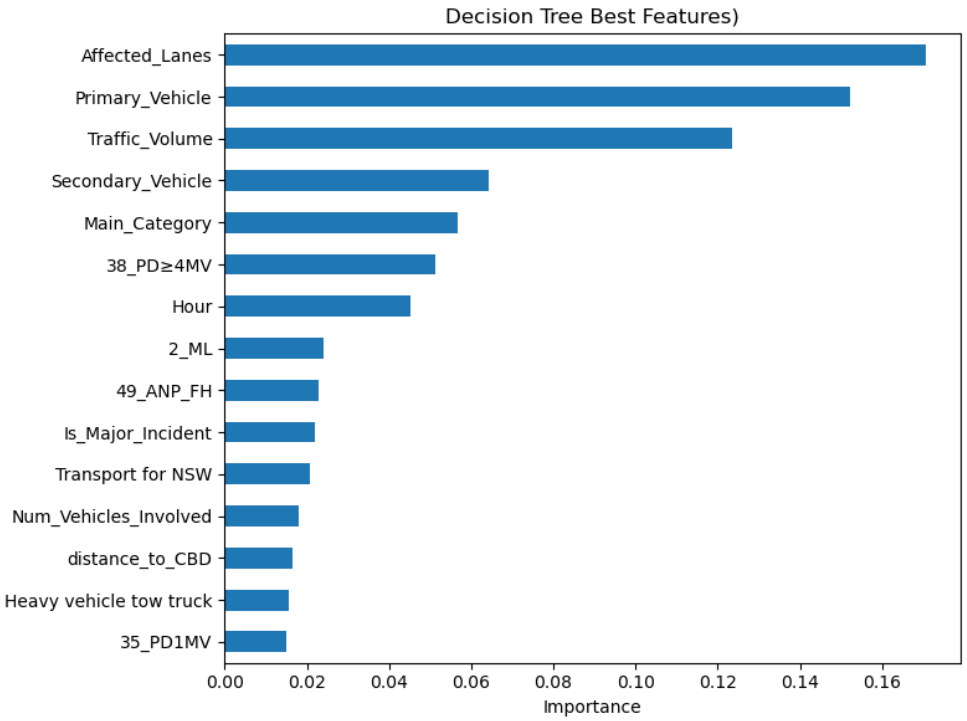
4.4 Decision Tree

Although Decision Tree algorithms are better able to handle categorical features than the previously considered methods, there is a danger of overfitting, so once again the high cardinality categorical features were dropped. However, for this algorithm one-hot encoding and scaling are unnecessary, and were skipped. The model trained extremely quickly at 0.58 seconds, but comparing it's metrics to logistical regression and SVM it performed slightly worse:

```
Accuracy: 0.6536762481089259

Classification report:

              precision    recall  f1-score

           0       0.66      0.68      0.67
           1       0.65      0.63      0.64

    accuracy                           0.65
   macro avg       0.65      0.65      0.65
weighted avg       0.65      0.65      0.65
```

The most important features were easier to extract from this model, and are listed below along with the confusion matrix:





We again have a similar list of important features, but it is interesting to note the distance_to_CBD is much lower in importance to this model's results.

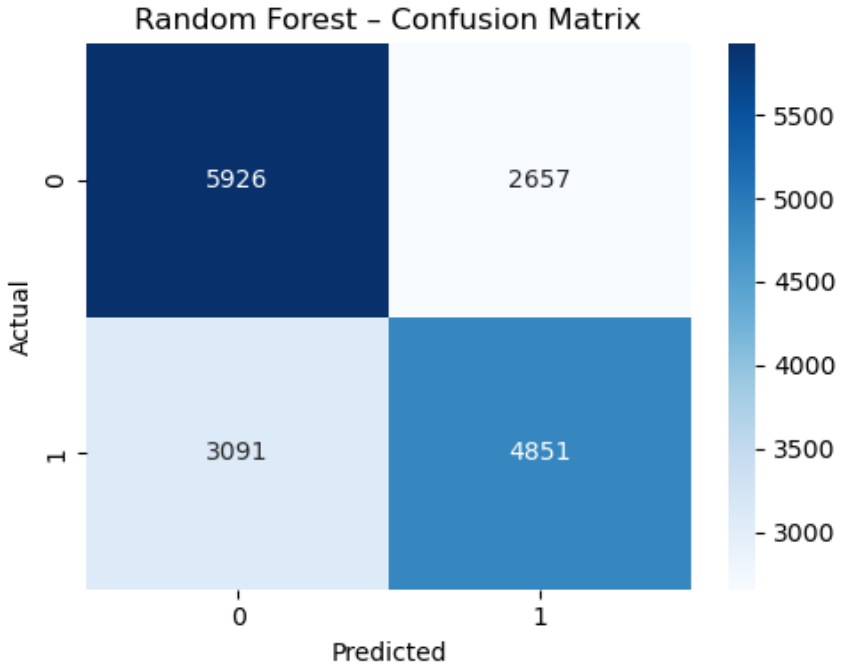
4.5 Random Forest

For the random forest algorithm, there is no need to drop high cardinality categorical columns,

perform one-hot encoding, or scaling, the cleaned dataset can simply be used as-is. This model also trained very quickly, at 1.19 seconds. However, the results were not impressive. Evaluating on the same metrics as other classifiers, the results are as follows:

```
Accuracy: 0.6521633888048411
Precision: 0.6461108151305275
Recall: 0.610803324099723
F1 Score: 0.6279611650485437
ROC-AUC: 0.7192072327473331
```

These scores are actually lower than any previous classifier, in contrast to the original study which claimed Random Forest as one of its best performing models. These results may be improved with additional data preprocessing and hyperparameter tuning, but that is outside of the scope of this paper. The confusion matrix and top features are as follows:




```
Primary_Vehicle      0.028829
Traffic_Volume       0.008169
Hour                 0.006245
Secondary_Vehicle    0.005604
Affected_Lanes       0.004841
                        ...
45_NPTtWbyO         -0.001452
37_PD3MV            -0.001719
Latitude            -0.001755
2_ML                -0.002082
distance_to_CBD     -0.002917
Length: 82, dtype: float64
```

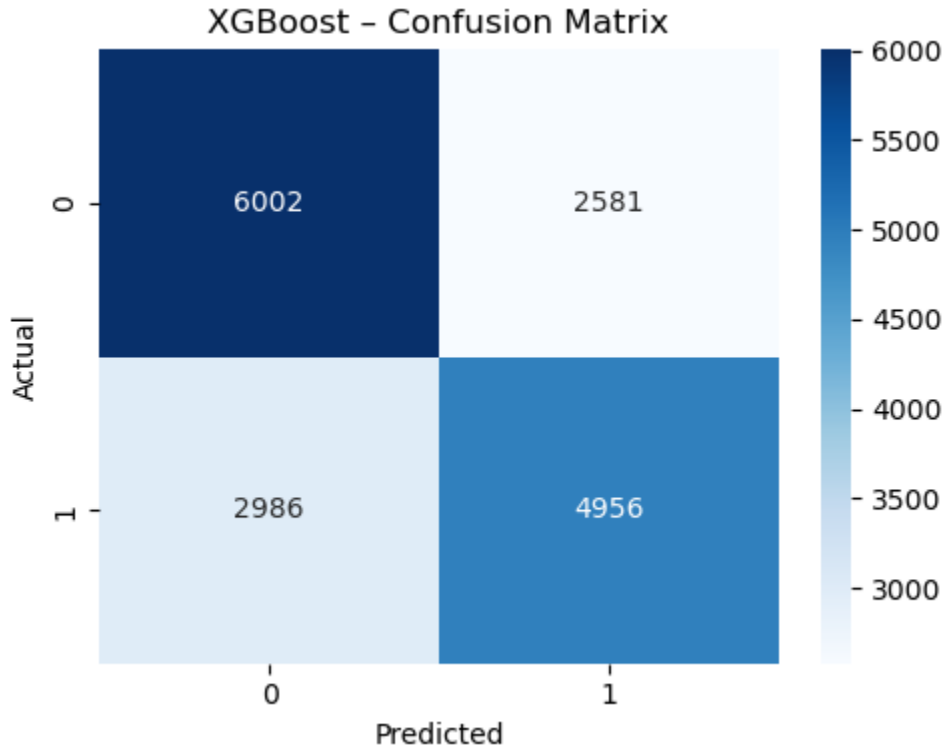
We again see Primary_Vehicle and Secondary_Vehicle as important features, although interestingly distance_to_CBD has dropped considerably in importance to this model compared those previously considered.

4.6 XGBoost

XGBoost, like Random Forest, is a tree algorithm that does not require scaling, encoding, or dropping of high cardinality categorical columns. It also trained very quickly at just 2.01 seconds, and was evaluated on the same metrics as before:

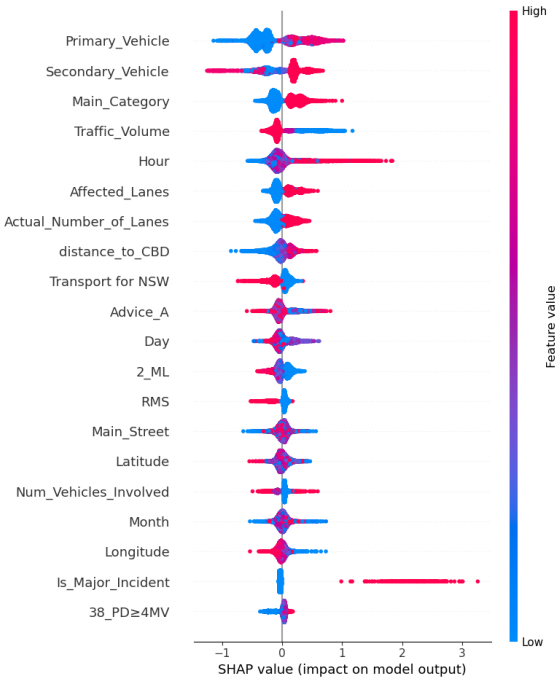
```
Accuracy: 0.6631164901664145
Precision: 0.6575560567865198
Recall: 0.6240241752707126
F1 Score: 0.6403514438917243
ROC-AUC: 0.7273619210556976
```

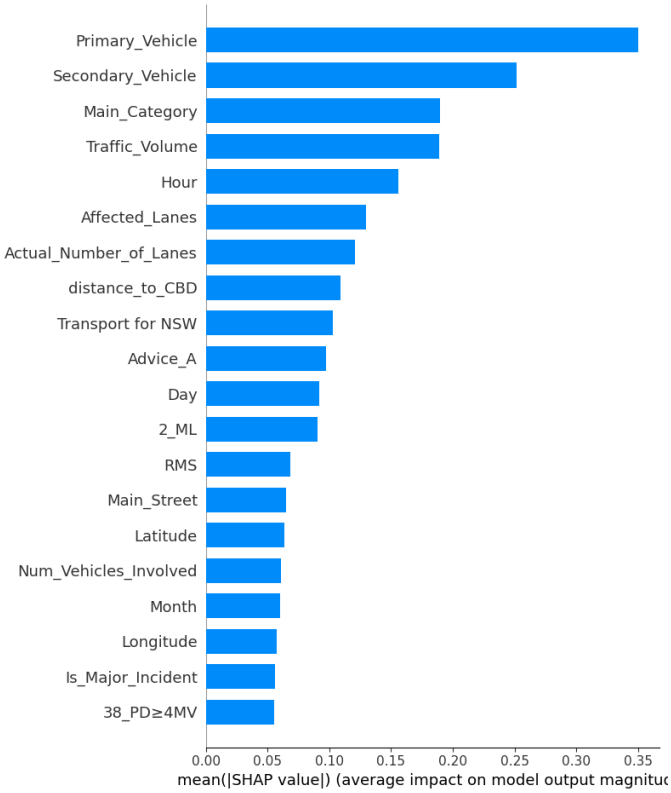
Although this model performed better than Random Forest, it is surprisingly still worse than basic Logistic Regression. The original paper found this to be their best performing model, achieving an F1-Score over 70%. It is likely that the algorithm could be improved through hyperparameter tuning, but again that is outside of the scope of this project. The confusion matrix is as follows:



I find it interesting that this model is better at predicting short duration incidents than other models, but worse at predicting long duration incidents. This phenomenon would be interesting

to explore further, but I am unable to do so at this time. SHAP was then used to explore the importance of different features to the performance of this model:





Again we see similar features being the most important, some of which are intuitive and some of which are less so. The distance_to_CBD feature has gained in importance in XGBoost as opposed to Random Forest, but not as important as it was in Logistic Regression or SVM. It is interesting to note how these two models differ in the importance assigned to distance_to_CBD in comparison to the tree-based models.

5. Conclusions

This project set out to replicate and expand on the findings of the study “Predicting the duration of traffic incidents for Sydney greater metropolitan area using machine learning methods” [1] and to compare the results obtained from that study to a broader set of machine learning algorithms. After cleaning and preprocessing the original dataset provided from the study, I evaluated Linear Regression, XGBRegressor, Logistic Regression, SVM (Linear and RBF), Decision Trees, Random Forest, and XGBoost.

My results did not fully support the findings presented in this study. Random Forest and XGBoost were not the top performing algorithms, although in my experiments they did achieve results somewhat consistent with those presented in the paper. The paper claimes that XGBoost achieved their highst classification F1-Score of 0.62, with 70.84% short term duration classificaion and 62.72% long incident classification accuracy. XGBoost actually performed better in my experiment, achieving an F1-Score of 0.64, with a similar difference between its accuracy on long and short duration classification.

However, basic Logistic Regression and both RBF and Linear SVM experiments showed even better performance. RBF kernel SVM, however, took a great deal of time to train and an excessive amount of time to extract feature importance from, making my top performing models Logistic Regression and Linear SVM, which both achieved F1-Scores around 0.66.

Both Linear Regression and XGBRegressor performed poorly in predicting continuous incident duration lengths. This is in line with the findings from the original study, which noted that this remains a difficult task.

Feature importance analysis showed similar features being the most important across the models, the most common factors being “Primary_Vehicle”, “Secondary_Vehicle”, “Main_Category”, “Incident_Type, “Num_Vehicles_Involved”, “Affected_Lanes”, and “distance_to_CBD”. Although there was some variation, in most analyses some combination of these features was considered the most important, and, satisfyingly, these features make intuitive sense.

Overall, my findings suggest that with proper data cleaning and preprocessing, simple linear methods can perform as well or better than more complex tree-based methods. With further preprocessing, hyperparameter tuning, and more careful handling of high cardinality categorical features, these results could likely be improved further.

6. Contributors

This paper was authored solely by Joshua Kluthe. All of the code contributing to the research for this paper was written by Joshua Kluthe, and is available from his GitHub repository at https://github.com/jjmkluthe/CS770_final_project/tree/main.

The author would like to also acknowledge his previous contributors, Sashank Enumula and Pavan Maddu. These contributors assisted in selecting the paper to review, discussing it's merits and issues, and presenting the research for further study. Although we were unable to complete our collaboration, their contributions to this process are appreciated.

7. References

[1] A. Grigorev, S. Shafiei, H. Grzybowska, and A.-S. Mihăiţă, "Predicting the duration of traffic incidents for Sydney greater metropolitan area using machine learning methods," *arXiv preprint* arXiv:2406.18861, 2024.

[2] "How to Use XGBoost XGBRegressor," XGBoosting, accessed Dec. 7, 2025. Available: https://xgboosting.com/how-to-use-xgboost-xgbregressor/

[3] "RBF SVM parameter selection — scikit-learn example," scikit-learn, accessed Dec. 7, 2025. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[4] "sklearn.svm.LinearSVC — scikit-learn documentation," scikit-learn, accessed Dec. 7, 2025. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[5] "sklearn.tree — Decision Trees, Random Forests, and Gradient Boosted Trees," scikit-learn, accessed Dec. 7, 2025. Available: https://scikit-learn.org/stable/modules/tree.html

[6] "sklearn.ensemble.RandomForestClassifier — scikit-learn documentation," scikit-learn, accessed Dec. 7, 2025. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[7] "XGBoost Multiclass Classification — GeeksforGeeks," GeeksforGeeks, accessed Dec. 7, 2025. Available: https://www.geeksforgeeks.org/machine-learning/xgboost-multiclass-classification/