Capstone Project Report Draft

Class ALY6140.80784: Analytics Systems Technology

Prof. Daya Rudhramoorthi

Group 5 members:

Min-Chi Tsai

Fatima Nurmakhamadova

May 19, 2022

Introduction

The goal of the project is to help Chicago Police Department to predict the traffic crash type and understand the causes that lead to it. We are wondering if there are similar or common patterns that might help to predict the traffic crash. The main question we intended to answer in this analysis is “What factors affect the severity of the traffic crash type?”. For this, we built three models, logistic regression, random forest, and XGBoost because they are very popular methods widely used for classification problems, especially on large datasets as in our case. Other questions that were answered in the exploratory data analysis include "What are the common types of traffic crashes and their causes?", "Do the environmental factors play role in driving?", "What time of the day has the most traffic crashes?". Finally, based on the findings we gave recommendations which will help to reduce the number of accidents.

Exploratory Data Analysis

The first step of the project involves exploring the data and analyzing the features. The first part of the EDA includes the data extraction and cleaning where we import the dataset and handle missing values and outliers. This will help us to have a clean dataset to get accurate results. Next is the data visualization, which involves feature study by visualizing them. This will help to get to know the data and find out possible trends and patterns. The target variable here is CRASH_TYPE which will help us define whether the crash resulted in injury or not. Initially, it had two values NO INJURY / DRIVE AWAY, and INJURY AND / OR TOW DUE TO CRASH which were changed to 0 and 1 accordingly.

Data extraction

The dataset “Traffic Crashes” that was selected for the capstone project was retrieved from Chicago Data Portal (Levy, 2022). It consists of the information about daily traffic crash reports in Chicago city streets from 2015 to the present. The report was provided by Chicago Police Department. The dataset consists of 605,120 records, and 48 features. Since the dataset is huge, we have extracted the data for the year 2022, until April, together with the features that are

important to our analysis. Thus, we will focus on the traffic crashes that happened in 2022 with a dataset of 29,605 records, and 26 features.

Data cleaning

The first step in data cleaning involves checking the data type. The Figure 1 shows the feature types together with the unique values. There are 7 numerical columns, 1 date time, and the rest are strings. The string columns were converted into categorical types as all of them consist of different categories and types of values. We have removed the time in the CRASH_DATE column leaving only a date, as we will not be using it. Furthermore, the column INJURIES_TOTAL which stands for the number of people who got any injuries during the crash had a float type which we changed into an integer.

Two variables had high cardinality that was reduced by the function setting the default threshold of 85% (Figure 2). Anything below 85% of the total number of instances in the column would be assigned as Others. As we can see in the graphs above, this helped to reduce values from 36 to 11 in the PRIM_CONTRIBUTORY_CAUSE, and from 38 to 6 in the SEC_CONTRIBUTORY_CAUSE .

Next, we have checked the dataset for the missing values (Figure 3). The columns LANE_CNT, INTERSECTION_RELATED_I, NOT_RIGHT_OF_WAY_I, HIT_AND_RUN_I, and WORK_ZONE_TYPE had missing values of more than 50% thus we dropped them as they would otherwise misrepresent the data. Whereas the columns REPORT_TYPE and MOST_SEVERE_INJURY had missing values of less than 5%, thus we only dropped the rows with missing data. As the proportion of missing values is small, while the dataset is big, we did not replace them with mean, median, or mode. Further, we have checked for ‘no data’, or ‘unknown’ value counts to check if they capture the whole column, and they do not.

And lastly, we have checked the dataset for outliers. The boxplot in Figure 4 shows that POSTED_SPEED_LIMIT, NUM_UNITS, and INJURIES_TOTAL have many outliers; thus, we have rechecked them by calculating the Interquartile range (Figure 5). It shows that anything above or below 30 in POSTED_SPEED_LIMIT, 2 in NUM_UNITS, and 0 in

INJURIES_TOTAL considers an outlier. Thus, we removed only single outlier values in each column, keeping other outliers as they might be important in finding an answer to our question. So now, we are left with 28,653 rows and 21 columns.

Data Visualization

We created a series of bar charts to investigate the potential patterns of independent variables. Most of the no injury crashes happened in the conjunction where don't have any traffic control, even a stop sign (Figure 6). However, towed or injury crashes generally happened either controlled or no controlled crossing. Interestingly, as seen in Figure 7, the number of serious crashes doesn't have a great difference between under daylight and on the lighted roads in dark. Also, most of crashes happened in clear weather condition (Figure 8), followed by snow and rain. The numbers of crashes are similar during rainy and snowy. As shown in Figure 9, the speed limit of crashed location normally goes 30 - 40 mph which corresponded to the normal city or county speed limit. The police and transport apartments might add a car speed column in the report to investigate the relationship between the crash and the speeding. As reported by officers and witnesses, most of the time the no-injury crash occurred on parked motor vehicle and serious crashes happened due to turning and angle changing/blind, displayed in Figure 10. Additionally, crashing on fixed object and pedestrian always causing serious crashes.

Among the charts, Figure 11, the distribution of crashes that occurred hour could clearly answer our business question and imply the relationship between traffic high peaks and crashes. In this case, the definition of traffic high peak is 7 - 9 am and 4 - 6 pm. During traffic high peaks, the number of crashes per hour is higher than the average crash number per hour. The average number of crashes per hour is around 1193. Surprisingly, most crashes happened at 2 - 3 pm followed by 3- 4 pm, those are not included in traffic high peaks. Additionally, the serious crashes always happened between 3 – 4 pm and 10 -11 pm.

Predictive Modeling

This section represents the application of three models, logistic regression, random forest, and XGBoost. First, we have prepared the dataset for modeling by assigning the independent variables into X, and dependent, aka target variable into y. Then, the categorical features in independent variables were converted into dummy variables. Furthermore, we checked the variables for multicollinearity by looking at the correlation matrix and then at the VIF score. The correlation matrix in Figure 12 indicated the variables as having a high correlation coefficient which is more than +-.7 (Figure 13). We checked the VIF score of the highly correlated variables and found that REPORT_TYPE_ON SCENE, REPORT_TYPE_NOT ON SCENE (DESK REPORT), ROAD_DEFECT_UNKNOWN, ROAD_DEFECT_NO DEFECTS, DEVICE_CONDITION_NO CONTROLS, and TRAFFIC_CONTROL_DEVICE_NO CONTROLS had the VIF score more than 7 thus were dropped from the analysis (Figure 14).

Next, the data were divided into train and test sets with a proportion of 70/30. The train set now has 20,057 rows, and 121 columns, while the test set has 8,596 rows and 121 columns.

1. **Logistic Regression**

The main reason to choose the logistic regression model is that it is a great model for predicting a binary outcome. Our target variable is categorical and has two outcomes while indicating the crash type. It either resulted in an injury or tow due to a crash or did not cause any injury and members just drove away. Moreover, the logistic regression model summary helps to distinguish the most significant variables that contribute to predicting the crash type by looking at their p values and coefficients.

The logistic regression model was fitted with L1 regularization for optimization. As our model takes 121 columns, the L1 helps to avoid overfitting by zeroing the coefficients of non-significant features. The Figure 15 shows the confusion matrix, and the results tells that the model predicted 5,904 (68.68%) of True Negative and + 1,565 (18.21%) of True Positive = 7,469 (86.89%) values correctly, and 231 (2.69%) of False Positive + 896 (10.42%) of False Negative = 1,127 (13.11%) values incorrectly. This means that the model predicted that 231 traffic crashes involved injury while they did not and that 896 crashes did not involve any injury or vehicle tow while they did. The model gave an accuracy of 86.9% which is a very good

performance. The recall is 0.96 for predicting the negative class, and 0.636 for the positive class. This means that 96% of the time the model was successful in predicting if the crash type did not involve any injury and only 63.6% in predicting if the crash involved any injury or vehicle tow. Such a big difference in recall values is probably related to the imbalance between classes. The precision is 0.87 for both classes which means that the model is good at predicting positive values.

Figure 16 shows the beginning of the logistic regression summary as there are too many variables. Here we can see the significant variables with the p-value below the alpha level of 0.1 sorted by the highest absolute value of the coefficient are TRAFFIC_CONTROL_DEVICE_SCHOOL ZONE, NUM_UNITS, DEVICE_CONDITION_FUNCTIONING IMPROPERLY, TRAFFIC_CONTROL_DEVICE_UNKNOWN, POSTED_SPEED_LIMIT, and CRASH_HOUR. The top three variables have positive coefficients which mean that the predicted probability of the crash involving any injury or vehicle tow increases as the number of units involved rises, as well as if the traffic control device is in the school zone and is not functioning properly. The last three variables have negative coefficients which mean that the predicted probability of the crash involving any injury or vehicle tow decreases as the posted speed limit and hour of the day rises, as well as if the traffic control device is unknown.

2. **Random Forest**

Based on tree-based algorithms, Random Forest modeling is an ensemble learning technique, which is widely used in modeling predictions (Zou & Schonlau, 2020). The selected random forest classifier of Skicit-learn has about 0.86 accuracy, which is acceptable. Shown on Figure 17, the precision is around 0.85 and recall is 0.63. 85% of positive predictions made are correct, and the model capture 63% of actual cases of injury and/or tow due to crash.

In order to measure the impact of all the variables in the entire random forest, we compare the relative importance of the variables returned by Skicit-learn. The larger value of importance shows that the variable affects the model more strongly. As seen from Figure 18,

INJURIES_TOTAL AND and NO INDICATION OF INJURY under the MOST SEVERE INJURY type show the greatest importance with crash type.

3. **Extreme Gradient Boosting (XGBoost)**

Gradient boosting is a type of ensemble machine learning algorithm, while XGBoost is an efficient open-source implementation of the gradient boosting algorithm. Introduced by Chen and Guestrin in 2016, XGBoost is designed with high computational efficiency making it faster than other open-source implementations. The XGBoost classifier of Skicit-learn has about 0.87 of accuracy. The precision is around 0.9 and recall is 0.6 (Figure 19). 90% of positive predictions made are correct, and the model capture 60% of actual cases of injury and/or tow due to crash. Different from the above model, Figure 20 shows that DAMAGE and INJURIES_TOTAL significantly impact the model prediction.

Interpretation

A Receiver Operator Characteristic (ROC) curve is a graphical method to evaluate a binary classifier. The curve shows the rate of the true positive rate (TPR) against the false positive rate (FPR) to present the trade-off between TPR (sensitivity) and specificity (1 – FPR). Simply, the area under the curve (AUC) larger, the higher accuracy of model prediction (Zou & Mauri, 2007). To dig out the predicted accuracy of our classifiers and regressor, three-line charts (Figure 21 -23) were generated with AUC. The AUC of the three models is close to 0.9 and slightly over that, showing the models are well-built and predictions are acceptable and accurate. Thus, it is not persuasive if we only look at the AUC and curve. Moreover, accuracy, precision, and recall also are important measures for model comparison. Accuracy is a metric to measure how the model performs across all classes, which is useful when all classes are important equally. The accuracy of the three models is over 0.86, while XGBoost's is slightly higher than the others. Precision refers to the rate of true positive against actual results, while recall refers to the rate of true positive against predicted results. XGBoost has the greatest precision value, about 0.89, having the most correct positive predictions among the three models. In other words, 89% of

positive predictions are correct in the XGBoost model. However, Logistic Regression and Random Forest have similar recall values that are greater than XGBoost's, around 0.63. 63% of predictions are correct in the two models.

Conclusion

The goal of the project has been met. Based on the analysis using three methods, we were able to define the most significant variables that were shown in all models. The most significant factors that affect the severity of the traffic crash and predict it are CRASH_HOUR, NUM_UNITS, and POSTED_SPEED_LIMIT. This means that most crashes that involves any injury or vehicle tow is caused during the high traffic peak from 7 to 9 am and 3 to 6 pm. Moreover, the probability of getting injury rises if the crash involves more than one unit whether it is a vehicle, pedestrian, or bicyclist. Another interesting factor is that the rising posted speed limit also might affect the crash severity. Thus, we would recommend the city of Chicago take the necessary precautions during the peak hours paying more attention and try to reduce the traffic.

As the next steps, we would suggest analyzing the traffic crashes by locations using the longitude and latitude. This would give an idea of where the most traffic crashes are happening so that Chicago Police Department would focus on these specific areas. Moreover, this would help the government to reduce traffics during the high traffic peaks and take other necessary actions.

References

Chen, & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Levy , J. (2022, April 20). *Traffic crashes - crashes: City of chicago: Data Portal*. Chicago Data Portal. Retrieved April 21, 2022, from https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if?category=Transportation&view_name=Traffic-Crashes-Crashes

Raschka, & Mirajalili, V. (2017). Python machine learning machine learning and deep learning with Python, scikit-learn, and TensorFlow (Second edition, fully revised and updated.). Packt.

Sangani, R. (2021, August 11). *Dealing with features that have high cardinality*. Medium. Retrieved May 15, 2022, from https://towardsdatascience.com/dealing-with-features-that-have-high-cardinality-1c9212d7ff1b

ZOU, O'MALLEY, A. J., & MAURI, L. (2007). Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models. *Circulation (New York, N.Y.), 115*(5), 654–657. https://doi.org/10.1161/CIRCULATIONAHA.105.594929

Zou, R. Y., & Schonlau, M. (2020, March 24). *The random forest algorithm for statistical learning*. The Stata Journal: Promoting communications on statistics and Stata. https://journals.sagepub.com/doi/full/10.1177/1536867X20909688 https://doi.org/10.1161/CIRCULATIONAHA.105.594929

Appendices:

Figure 1: Variables type and unique values

```
#Show the variables type
crash22.dtypes
```

```
CRASH_DATE                  datetime64[ns]
POSTED_SPEED_LIMIT                   int64
TRAFFIC_CONTROL_DEVICE              object
DEVICE_CONDITION                    object
WEATHER_CONDITION                   object
LIGHTING_CONDITION                  object
FIRST_CRASH_TYPE                    object
TRAFFICWAY_TYPE                     object
LANE_CNT                           float64
ROADWAY_SURFACE_COND                object
ROAD_DEFECT                         object
REPORT_TYPE                         object
CRASH_TYPE                          object
INTERSECTION_RELATED_I              object
NOT_RIGHT_OF_WAY_I                  object
HIT_AND_RUN_I                       object
DAMAGE                              object
PRIM_CONTRIBUTORY_CAUSE             object
SEC_CONTRIBUTORY_CAUSE              object
WORK_ZONE_TYPE                      object
NUM_UNITS                            int64
MOST_SEVERE_INJURY                  object
INJURIES_TOTAL                     float64
CRASH_HOUR                           int64
CRASH_DAY_OF_WEEK                    int64
CRASH_MONTH                          int64
dtype: object
```

```
#Show the unique values in each column
crash22.nunique()
```

```
CRASH_DATE                  20158
POSTED_SPEED_LIMIT             20
TRAFFIC_CONTROL_DEVICE         17
DEVICE_CONDITION                8
WEATHER_CONDITION              12
LIGHTING_CONDITION              6
FIRST_CRASH_TYPE               17
TRAFFICWAY_TYPE                20
LANE_CNT                        2
ROADWAY_SURFACE_COND            7
ROAD_DEFECT                     7
REPORT_TYPE                     2
CRASH_TYPE                      2
INTERSECTION_RELATED_I          2
NOT_RIGHT_OF_WAY_I              2
HIT_AND_RUN_I                   2
DAMAGE                          3
PRIM_CONTRIBUTORY_CAUSE        36
SEC_CONTRIBUTORY_CAUSE         38
WORK_ZONE_TYPE                  4
NUM_UNITS                       9
MOST_SEVERE_INJURY              5
INJURIES_TOTAL                  9
CRASH_HOUR                     24
CRASH_DAY_OF_WEEK               7
CRASH_MONTH                     4
dtype: int64
```

Figure 2: Feature engineering

```
#for PRIM_CONTRIBUTORY_CAUSE with 36 unique values
transformed_column,new_category_list=cumulatively_categorise(crash22['PRIM_CONTRIBUTORY_CAUSE'],return_categories_list=

#Check the unique values
transformed_column.value_counts()
```

```
UNABLE TO DETERMINE                     12196
Other                                    3921
FAILING TO YIELD RIGHT-OF-WAY            3111
FOLLOWING TOO CLOSELY                    2381
NOT APPLICABLE                           1415
IMPROPER OVERTAKING/PASSING              1361
FAILING TO REDUCE SPEED TO AVOID CRASH   1231
IMPROPER BACKING                         1028
DRIVING SKILLS/KNOWLEDGE/EXPERIENCE      1016
IMPROPER LANE USAGE                       992
WEATHER                                   971
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64
```

```
#for SEC_CONTRIBUTORY_CAUSE with 38 unique values
transformed_column2,new_category_list=cumulatively_categorise(crash22['SEC_CONTRIBUTORY_CAUSE'],return_categories_list=

#Check the unique values
transformed_column2.value_counts()
```

```
NOT APPLICABLE                          11817
UNABLE TO DETERMINE                     11216
Other                                    3778
FAILING TO REDUCE SPEED TO AVOID CRASH    960
DRIVING SKILLS/KNOWLEDGE/EXPERIENCE       952
FAILING TO YIELD RIGHT-OF-WAY             900
Name: SEC_CONTRIBUTORY_CAUSE, dtype: int64
```

Figure 3: Missing values of each column in percentage

```
# Check percentage of missing values
percent_missing = crash22.isnull(). sum() * 100 / len(crash22)
percent_missing
```

```
CRASH_DATE                   0.000000
POSTED_SPEED_LIMIT           0.000000
TRAFFIC_CONTROL_DEVICE       0.000000
DEVICE_CONDITION             0.000000
WEATHER_CONDITION            0.000000
LIGHTING_CONDITION           0.000000
FIRST_CRASH_TYPE             0.000000
TRAFFICWAY_TYPE              0.000000
LANE_CNT                    99.989873
ROADWAY_SURFACE_COND         0.000000
ROAD_DEFECT                  0.000000
REPORT_TYPE                  3.206968
CRASH_TYPE                   0.000000
INTERSECTION_RELATED_I      75.930865
NOT_RIGHT_OF_WAY_I          95.284070
HIT_AND_RUN_I               65.739459
DAMAGE                       0.000000
PRIM_CONTRIBUTORY_CAUSE      0.000000
SEC_CONTRIBUTORY_CAUSE       0.000000
WORK_ZONE_TYPE              99.804206
NUM_UNITS                    0.000000
MOST_SEVERE_INJURY           0.222800
INJURIES_TOTAL               0.000000
CRASH_HOUR                   0.000000
CRASH_DAY_OF_WEEK            0.000000
CRASH_MONTH                  0.000000
dtype: float64
```

Figure 4: Boxplot of the numerical variables

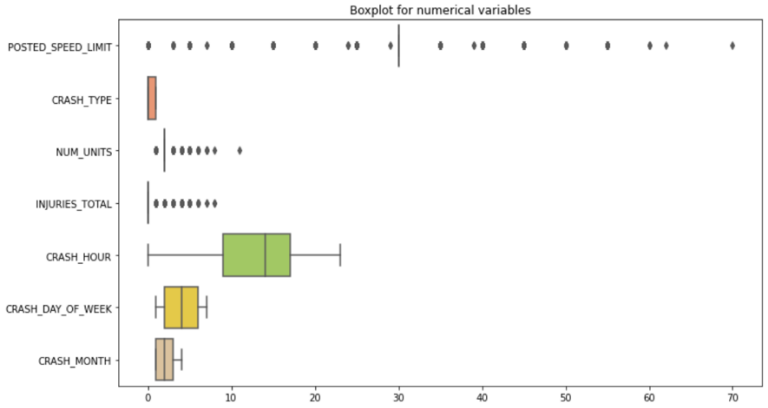

Figure 5: The Inter quartile range score of numeric variables

```
The IQR score
Lower limit :
POSTED_SPEED_LIMIT    30.0
CRASH_TYPE            -1.5
NUM_UNITS              2.0
INJURIES_TOTAL         0.0
CRASH_HOUR            -3.0
CRASH_DAY_OF_WEEK     -4.0
CRASH_MONTH           -2.0
dtype: float64
Upper limit :
POSTED_SPEED_LIMIT    30.0
CRASH_TYPE             2.5
NUM_UNITS              2.0
INJURIES_TOTAL         0.0
CRASH_HOUR            29.0
CRASH_DAY_OF_WEEK     12.0
CRASH_MONTH            6.0
dtype: float64
```

Figure 6: Traffic Control Device Condition

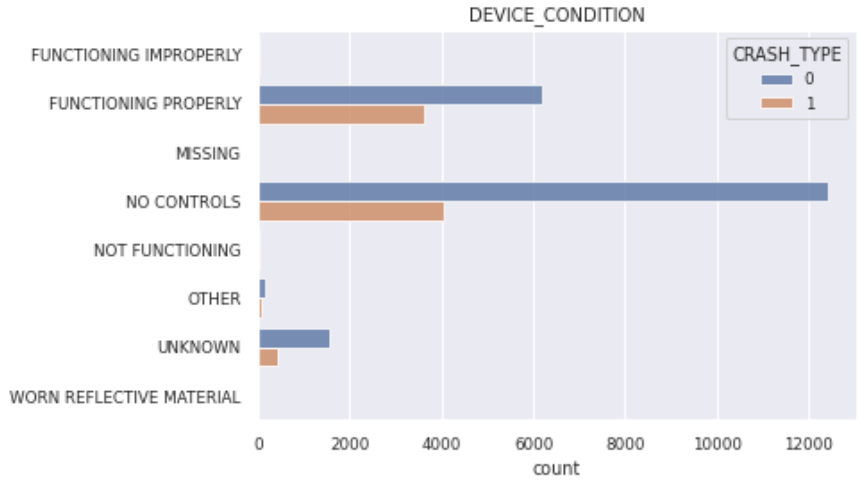

Figure 7: The Lighting Condition

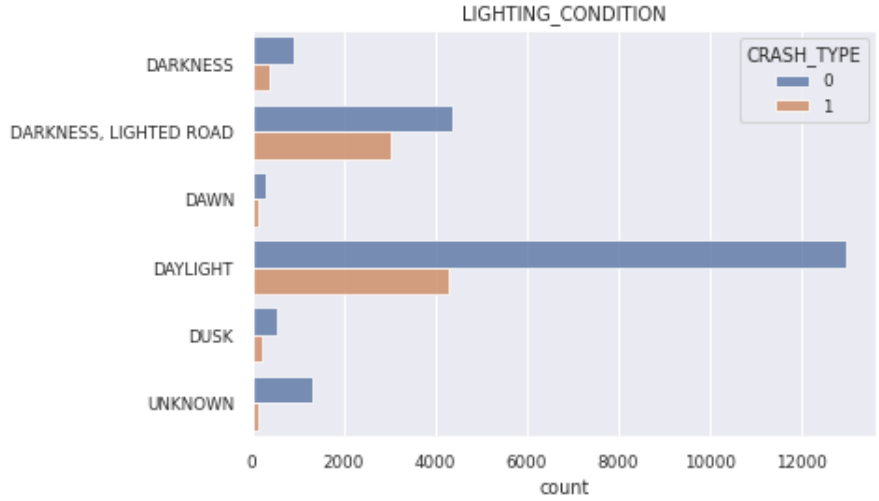

Figure 8: The Weather Condition When Crashes Happened

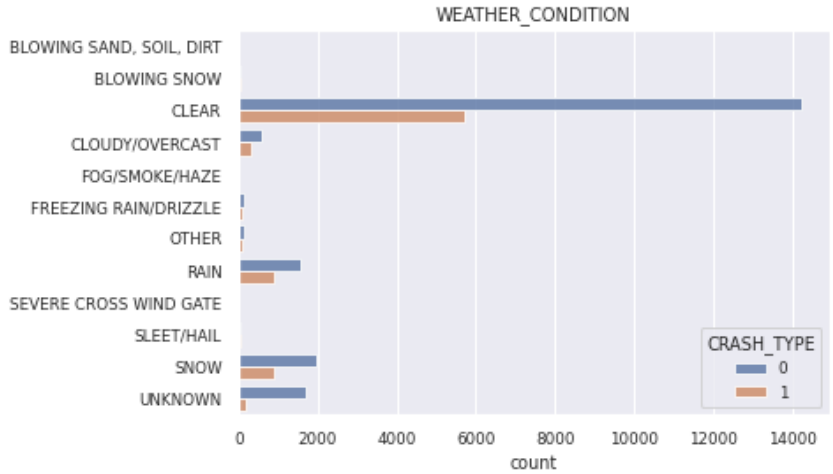

Figure 9 ： The Distribution of Posted Speed Limitation on Where the Crash Happened

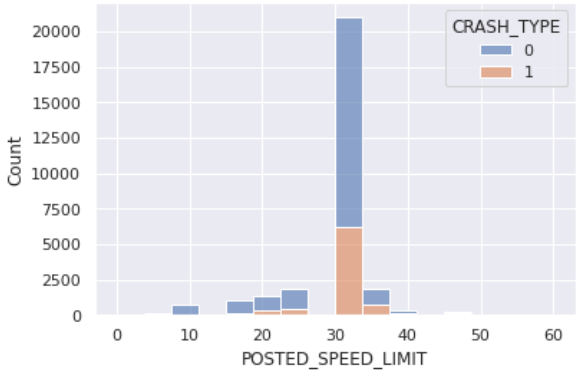

Figure 10: The Type of First Crash identified by the officers

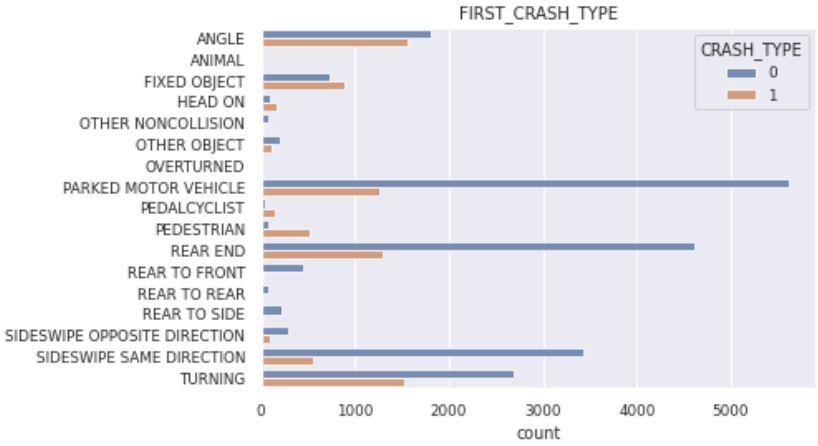

Figure 11: The Distribution of Crash Time in Hour

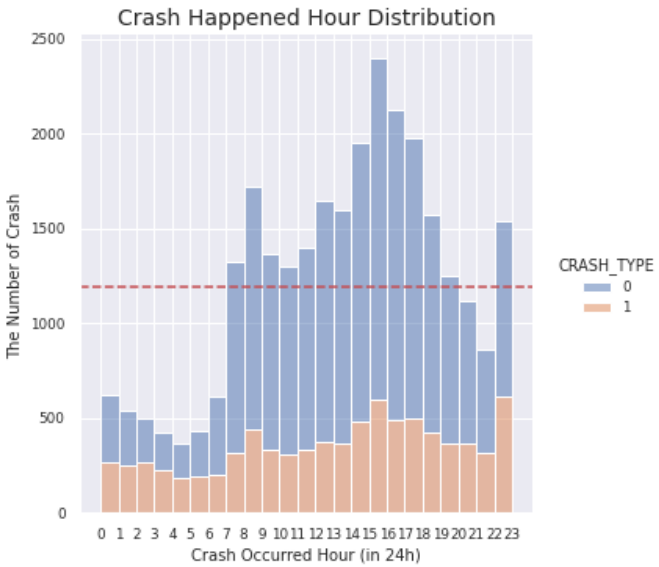

Figure 12: Correlation Matrix of all independent variables

```
#Plot the correlation matrix for all independent variables
xCorr = X.corr()
filteredDf = xCorr[((xCorr >= .7) | (xCorr <= -.7)) & (xCorr !=1.000)]
plt.figure(figsize=(30,10))
sns.heatmap(filteredDf, annot=True, cmap="Reds")
plt.show()
```

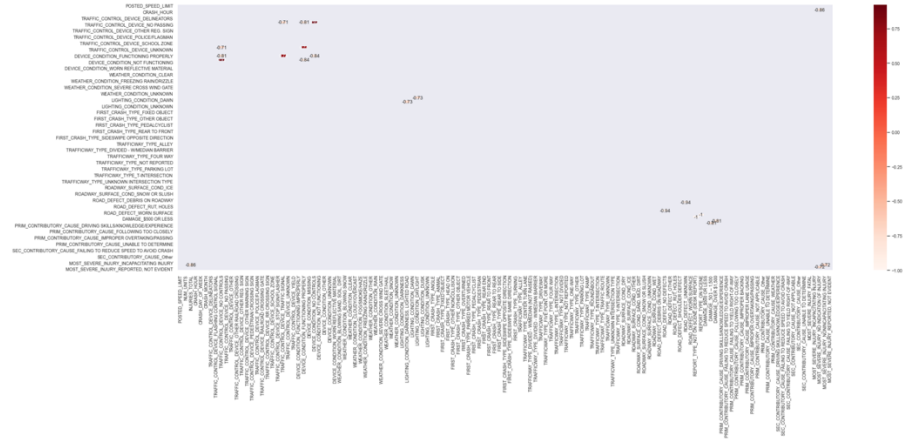

Figure 13: The list of most highly correlated variables

```
#Show most highly correlated variables
def corrFilter(x: pd.DataFrame, bound: float):
    xCorr = x.corr()
    xFiltered = xCorr[((xCorr >= bound) | (xCorr <= -bound)) & (xCorr !=1.000)]
    xFlattened = xFiltered.unstack().sort_values().drop_duplicates()
    return xFlattened

corrFilter(X, .7)
```

```
REPORT_TYPE_ON SCENE                          REPORT_TYPE_NOT ON SCENE (DESK REPORT)        -1.000000
ROAD_DEFECT_UNKNOWN                           ROAD_DEFECT_NO DEFECTS                        -0.944185
INJURIES_TOTAL                                MOST_SEVERE_INJURY_NO INDICATION OF INJURY    -0.857666
DEVICE_CONDITION_FUNCTIONING PROPERLY         DEVICE_CONDITION_NO CONTROLS                  -0.839448
DAMAGE_$501 - $1,500                          DAMAGE_OVER $1,500                            -0.807635
DEVICE_CONDITION_FUNCTIONING PROPERLY         TRAFFIC_CONTROL_DEVICE_NO CONTROLS            -0.806895
LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD     LIGHTING_CONDITION_DAYLIGHT                   -0.728432
MOST_SEVERE_INJURY_NO INDICATION OF INJURY    MOST_SEVERE_INJURY_NONINCAPACITATING INJURY   -0.723675
TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL         TRAFFIC_CONTROL_DEVICE_NO CONTROLS            -0.710867
                                              DEVICE_CONDITION_FUNCTIONING PROPERLY          0.736137
DEVICE_CONDITION_NO CONTROLS                  TRAFFIC_CONTROL_DEVICE_NO CONTROLS             0.923540
POSTED_SPEED_LIMIT                            POSTED_SPEED_LIMIT                                  NaN
dtype: float64
```

Figure 14: The VIF Score of highly correlated variables

	feature	VIF
0	REPORT_TYPE_ON SCENE	71.154025
1	REPORT_TYPE_NOT ON SCENE (DESK REPORT)	76.264787
2	ROAD_DEFECT_UNKNOWN	9.336687
3	ROAD_DEFECT_NO DEFECTS	9.259547
4	INJURIES_TOTAL	3.801230
5	MOST_SEVERE_INJURY_NO INDICATION OF INJURY	4.930353
6	DEVICE_CONDITION_FUNCTIONING PROPERLY	4.515433
7	DEVICE_CONDITION_NO CONTROLS	8.660927
8	DAMAGE_OVER $1,500	2.909204
9	DAMAGE_$501 - $1,500	2.930201
10	TRAFFIC_CONTROL_DEVICE_NO CONTROLS	7.260381
11	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	2.255270
12	LIGHTING_CONDITION_DAYLIGHT	2.247147
13	MOST_SEVERE_INJURY_NONINCAPACITATING INJURY	2.102278
14	TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL	2.404926
15	POSTED_SPEED_LIMIT	1.051912

Figure 15: Logistic Regression Model Confusion Matrix

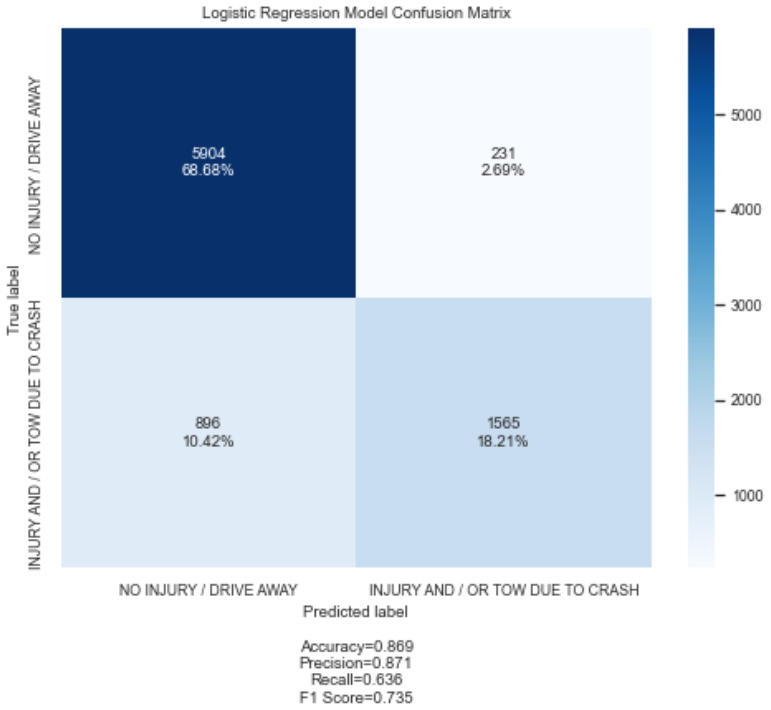

Figure 16: Logistic Regression model summary

Logit Regression Results

Dep. Variable:	CRASH_TYPE	No. Observations:	20057
Model:	Logit	Df Residuals:	19944
Method:	MLE	Df Model:	112
Date:	Sat, 14 May 2022	Pseudo R-squ.:	0.4848
Time:	03:00:04	Log-Likelihood:	-6188.0
converged:	False	LL-Null:	-12010.
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
POSTED_SPEED_LIMIT	0.0203	0.005	3.868	0.000	0.010	0.031
NUM_UNITS	1.1876	0.062	19.165	0.000	1.066	1.309
INJURIES_TOTAL	-2.7669	514.586	-0.005	0.996	-1011.337	1005.803
CRASH_HOUR	-0.0177	0.004	-4.574	0.000	-0.025	-0.010
CRASH_DAY_OF_WEEK	-0.0165	0.011	-1.443	0.149	-0.039	0.006
CRASH_MONTH	0.0194	0.024	0.799	0.424	-0.028	0.067
TRAFFIC_CONTROL_DEVICE_DELINEATORS	1.1095	0.715	1.551	0.121	-0.293	2.512
TRAFFIC_CONTROL_DEVICE_FLASHING CONTROL SIGNAL	0.1933	1.297	0.149	0.882	-2.350	2.736
TRAFFIC_CONTROL_DEVICE_NO PASSING	-16.3397	2.61e+04	-0.001	1.000	-5.12e+04	5.12e+04
TRAFFIC_CONTROL_DEVICE_OTHER	-0.2894	0.314	-0.922	0.357	-0.905	0.326
TRAFFIC_CONTROL_DEVICE_OTHER RAILROAD CROSSING	-9.4059	152.989	-0.061	0.951	-309.259	290.447
TRAFFIC_CONTROL_DEVICE_OTHER REG. SIGN	0.4529	0.715	0.634	0.526	-0.948	1.854
TRAFFIC_CONTROL_DEVICE_OTHER WARNING SIGN	0.9776	0.767	1.275	0.202	-0.525	2.481
TRAFFIC_CONTROL_DEVICE_PEDESTRIAN CROSSING SIGN	0.0769	0.856	0.090	0.928	-1.600	1.754
TRAFFIC_CONTROL_DEVICE_POLICE/FLAGMAN	0.9672	1.070	0.904	0.366	-1.130	3.064
TRAFFIC_CONTROL_DEVICE_RAILROAD CROSSING GATE	-0.7786	0.900	-0.865	0.387	-2.543	0.986
TRAFFIC_CONTROL_DEVICE_RR CROSSING SIGN	0.9473	1.217	0.779	0.436	-1.437	3.332
TRAFFIC_CONTROL_DEVICE_SCHOOL ZONE	1.4141	0.833	1.697	0.090	-0.219	3.047
TRAFFIC_CONTROL_DEVICE_STOP SIGN/FLASHER	-0.1866	0.137	-1.365	0.172	-0.455	0.081
TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL	-0.2193	0.139	-1.582	0.114	-0.491	0.052
TRAFFIC_CONTROL_DEVICE_UNKNOWN	-0.4716	0.217	-2.173	0.030	-0.897	-0.046
TRAFFIC_CONTROL_DEVICE_YIELD	0.7608	0.557	1.367	0.172	-0.330	1.852
DEVICE_CONDITION_FUNCTIONING IMPROPERLY	0.7408	0.364	2.037	0.042	0.028	1.453
DEVICE_CONDITION_FUNCTIONING PROPERLY	0.1135	0.132	0.862	0.389	-0.145	0.372
DEVICE_CONDITION_MISSING	1.5257	1.419	1.075	0.282	-1.256	4.307
DEVICE_CONDITION_NOT FUNCTIONING	-0.7534	0.526	-1.431	0.152	-1.785	0.278
DEVICE_CONDITION_OTHER	0.1149	0.298	0.386	0.700	-0.468	0.698
DEVICE_CONDITION_UNKNOWN	0.1158	0.165	0.701	0.483	-0.208	0.439
DEVICE_CONDITION_WORN REFLECTIVE MATERIAL	-14.1811	3253.918	-0.004	0.997	-6391.744	6363.381
WEATHER_CONDITION_BLOWING SAND, SOIL, DIRT	-5.5737	nan	nan	nan	nan	nan

Figure 17: Confusion Matrix of random forest model

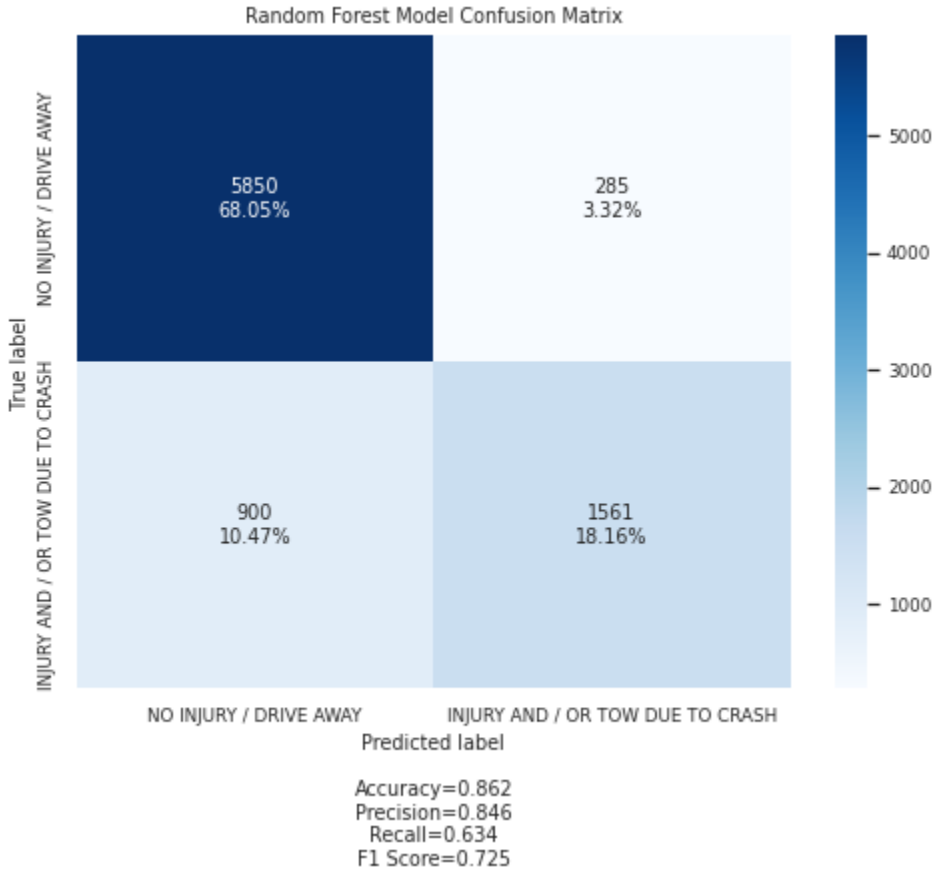

Figure 18: Ranking of feature importance in the random forest model.

```
# create feature importance barchart
feat_importances = pd.Series(rfc.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

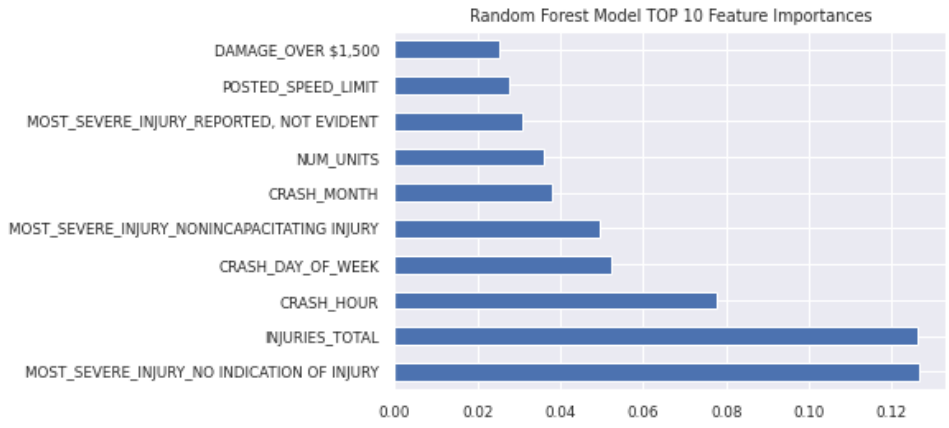

Figure 19: Confusion Matrix of XGBoost model.

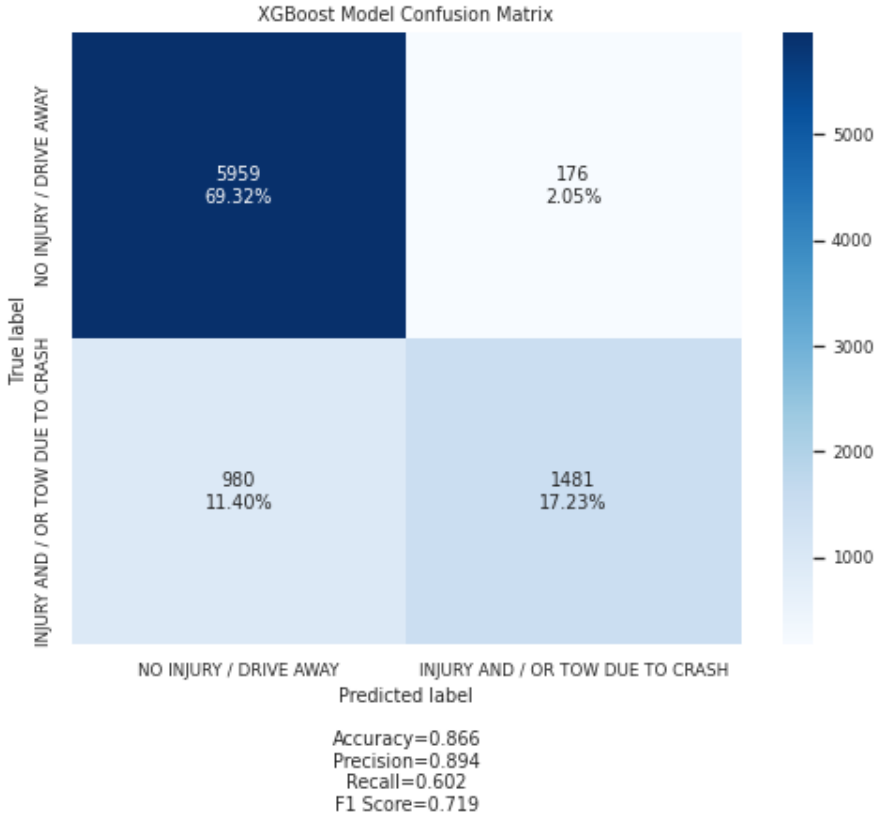

Figure 20: Ranking of feature importance in the XGBoost Model

```
# create feature importance barchart
feat_importances = pd.Series(xgb.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
```

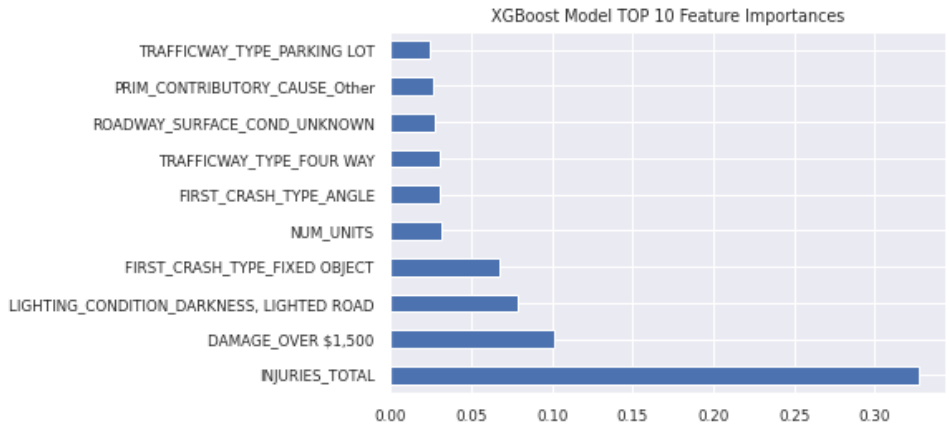

Figure 21: Random Forest ROC Curve

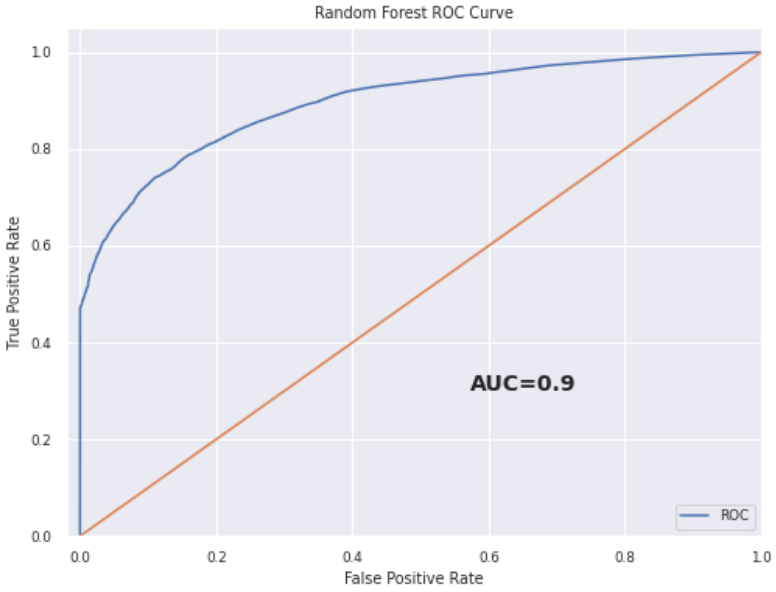

Figure 22: XGBosst ROC Curve

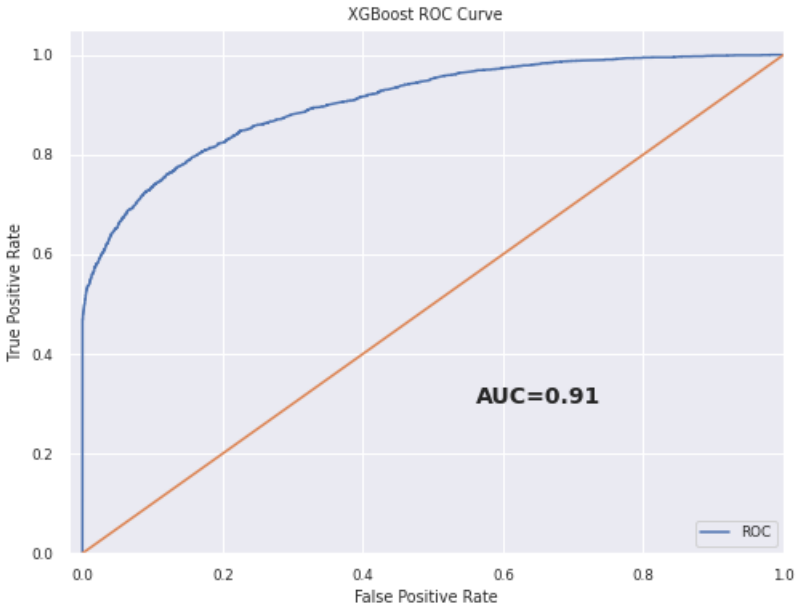

Figure 23: Logistic Regression ROC Curve