PREDICTING EMERGENCY CALLS FOR THE SAN FRANCISCO FIRE DEPARTMENT

STATS 503 PROJECT GROUP 17

Hongru Fang fhongru@umich.edu

Thomas Fiore tmfiore@umich.edu

Ming Gao gaoming@umich.edu

Zizhao Zhang zizhaoz@umich.edu

***K*eywords** Classification · Machine Learning · ROC Curve

1 Introduction

Our project concerns binary classification of incoming calls to the San Francisco Fire Department as emergency call versus non-emergency. We chose a classification problem because it allows us to implement many machine Learning algorithms from class, for instance, logistic regression, K-nearest neighbors, random forests, support vector machines, and model evaluation/selection using Cross Validation. We found out data set from open source Kaggle : https://www.kaggle.com/datasf/san-francisco. The San Francisco Open Data Kaggle contains multiple tables about the city: "fire calls", "police calls", "trees", "bikes", and etc. We are interested in the "fire calls" as the table size is ideally large with least missing datas. Moreover, data curated and cultivated by the organization DataSF, which says: "we seek to transform the way the City works through the use of data", as fire is the most devastating one among all the emergencies recorded in the data set, we wish to put our analysis in use and improve the service of fire department of San Francisco. In that our proposed topic is **classify incoming calls to dispatch as Emergency/Non-Emergency based on features such as location, time of day, issue.**

2 Data Preprocessing & Feature Engineering

The initial data set had 4,557,045 data points with 17 columns. We decided that running machine learning models on a training set with more than one million data points would be too time-consuming and not worthwhile. Since the initial data set was huge with over 4 million points, we decided we were justified in removing all the rows/data points with any missing value. Removal was preferable to imputation since imputing missing values can bring bias into the data set and our future model. After the removal of all cases with missing values, the data set size shrunk down to around 3 million rows, which is ample.

After the removal of all cases with missing values, we took several more data cleaning steps.

1. The response variable **priority** had several other possible values besides **Emergency** and **Non-emergency**, so we removed all the rows which were neither **Emergency** nor **Non-emergency** to better indicate our emergency level.

2. After restricting attention to rows with response **Emergency** and **Non-emergency**, we skimmed through the potential features, and decided that we would need a continuously numerical feature to precisely represent the time of the call. We used string slicing and variable casting with numerical transformation to transform the feature **received_time_stamp**, which was a categorical time variable, to the new feature **called_time_passed**, which is a numerical variable between 0 and 24. This new variable indicates how many hours since the previous midnight the fire department received the call.

3. There were multiple factors representing the locations of the fire calls, we ranked all the potential features representing location in terms of the number of unique values, and chose to go with **zipcode** as it has 38 unique values and represents the call location well.

4. To avoid the potential risks of imbalanced data, we conducted a simple calculation to peek the percentage of two interest : **Emergency** with 80.67% and **Non-emergency** with 19.3%, and the percentage of spread was ideal.

5. Along with the categorical feature **zipcode**, another important categorical feature is **call_type**, which is the nature of incident (e.g. outside fire, structure fire, etc). The features **zipcode** and **call_type** are the most important records when the call is received by the fire department dispatch. Since these two categorical values are not ordinal or time related, we decided to use "one-hot encoding" to transform these two features into multiple dummy features, with binary values (0-1). This process expanded these two features into to 64 categorical feature columns (27 for call types, 37 for zipcodes).

The initial preprocessed data set which consisted of 3,000,702 cases with 67 features (including dummy variables). This was still too large: even logistic regression (which should be the fastest in terms of runtime and computation complexity) lead to problems like "dead kernel due to run time" and endless runtime errors. Thus, we decided to shrink down to make an the exploratory data set with 5,861 rows and 68 columns by taking every 512th row from the initial preprocessed data set with 3,000,702. This exploratory data set with 5,861 rows preserved the 80% / 20% split between emergency calls and non-emergency calls. We used the exploratory data set for EDA visualizations.

We did *not* use the exploratory data for training and testing. Instead, from the preprocessed data set with 3,000,702 cases we randomly selected approximately 4,500 cases with "priority-emergency" and we randomly selected approximately 4,500 cases with "priority-non-emergency". The resulting reduced data set had 9,209 rows and 61 columns. The equal representation in this reduced data set was better for training and testing.

The main variables for us were

- Priority (Emergency vs. Non-Emergency)
- Call Type (Structure Fire, Vehicle Fire, Outside Fire, Water Rescue, Hazardous Materials,.....)
- Unit type
- Incoming call time and date
- Latitude and longitude
- Zip code.

3 Exploratory Data Analysis

Note: All images for the entire project are in the Appendix.

We performed exploratory data analysis on the aforementioned exploratory sample with 5,861 rows, obtained by selecting every 512th row of the cleaned data set.

To get an idea of the geographic distribution of the data points and their emergency/non-emergency status, we plotted the latitude and longitude of each observation. See **Figure 1 in the Appendix**. A yellow dot indicates an emergency call, a purple dot indicates a non-emergency call. The visualization illustrates our finding that 80% are emergency calls, and 20% are non-emergency calls. We can clearly see the shape of San Francisco, including Treasure Island in the upper right corner of the image. The calls are not evenly distributed across the city, instead we see the northwest corner of the city is more dense.

Next we were interested in visualizing the frequency of calls at various times of day. See **Figures 2 and 3 in the Appendix**. These two frequency histograms show the incoming call times in the exploratory data set for emergency calls and non-emergency calls. The numbers on the horizontal axis indicate the number of hours since midnight. Most calls of both types are during hours in which most people are awake, namely 9 am to 11 pm.

Lastly, to get an impression of the distribution of various call types within the emergency and non-emergency calls, we made a bar plot of the counts of non-emergency and emergency calls, and color coded the portions of the bars in relation to the number of calls of each type. See **Figure 4 in the Appendix**.

4 Emergency Calls Prediction

4.1 Train & Test Splitting

For training we used 80% of our reduced data set, which had 9,209 rows and 61 columns. Thus our training set had 7,367 rows with 60 features. The test data was 20% of our reduced data set, thus the test data has 1,842 rows with 60 features. Both training data and test data had a 50%-50% split between emergency and non-emergency priorities.

To avoid the potential risks of data imbalance, we wish to ensure the split percentage of 50% of the response “y” within our training set. We did a sampling method of querying around 5000 random rows of data which "priority" is **Non-emergency** as well as 5000 random rows of **Non-emergency**. We then used the Python “train test split” with 1/2 of the data set as testing set, setting “stratify” API to be variable “y” to ensure the balance of emergency/non-emergency percentage of the training sets. In that the spread of “Priority” within both of the training set and test set would be 50% **Non-emergency** and 50% **Emergency**.

4.2 Standardization of Continuous Variables before Model Fitting

Since each continuous predictor had a different range, we scaled every continuous predictor to mean 0 and variance 1 in order to have equal influence.

4.3 Model Selection: Parameter Tuning via Cross Validation on the Training Set

Since the predictor is not normal, we don’t consider LDA and QDA. Instead, we use cross validaiton to select the best parameters for logistic legression, K nearest neighbor, random forest and support vector machine.

In all of the above methods, there are so many hyper parameters which we could tune. Thus, we use grid search to set a grid of possible hyper parameter choices and 5 fold cross validation to determine the best model.

4.3.1 Logistic Regression

Table 1: Best Parameters for Logistic Regression Determined by Cross Validation on the Training Set

Parameter Name	Best	Interpretation
C	5	inverse of λ value(weight) for regularization term
penalty	L1	Regularization method i.e. “LASSO”, “ridge”

As we ran through 5-fold grid search cross-validation in Python, the best penalty for regularization is “L1”, which is LASSO regularization, the penalty term shrinks and removes the coefficients can reduce variance without a substantial increase of the bias, as we have less numerical features and many one-hot encoded categorical features (57), in that “L1” penalty does great internal feature selection for the model. As our training design matrix X is pretty sparse with the dummy variables, so we expect the “L1” would reign. Concerning the parameter “C”, which is the inverse of regularization strength λ, the larger “C” is, the more the freedom is, the weaker the regularization is. The training cross-validation optimized “C” is 5, that is $\lambda = 1/5$. Cross validation did not choose a bigger “C” (or smaller λ) to ensure the model does not overfit to small perturbations or noise in the data.

We can next identify and interpret the most important predictors. Since we standardized the predictors before fitting models, we can identify the most important predictors by simply finding the largest fitted coefficients. We find that the top 3 important predictors are **call_type_Outside_Fire** with coefficient 0.562, **call_type_Alarms** with coefficient 1.074, and **call_type_Structure_Fire** with coefficient 1.99. Since these are all positive, it means the probability of the call being an emergency will increase if the type is any one of these three (all other predictors held fixed). Such result makes sense, for instance, the three binary categorical features indicates that if the incoming call is **call_type_Structure_Fire**, or **call_type_Alarms**, or **call_type_Outside_Fire**, the log-likelihood would increase.

4.3.2 K Nearest Neighbors

Table 2: Best Parameters for K-Nearest Neighbor Determined by Cross Validation on the Training Set

Parameter Name	Best	Interpretation
k	10	Number of nearest neighbors

As we ran through 5-fold grid search cross-validation in Python, the optimized parameter for “k”, which is the number of the nearest neighbors is 10, since KNN is an algorithm based mainly on distance between numerical variables, and we have used the default Minkowski distance with p = 2, which is the Euclidean distance, and such metric/parameter selection is ideal in terms of locations features on the same plane, which we have two of those in our predictors.

4.3.3 Random Forest

Table 3: Best Parameters for Random Forest Tuning Parameters Determined by Cross Validation on the Training Set

Parameter Name	Best	Interpretation
n_estimators	200	Number of trees in the forest
max_depth	10	Maximum allowed level of the each tree

As we ran through 5-fold grid search cross-validation in Python, the optimal "max_depth" is 10, as it represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data. Since our training set is pretty sparse, a tree which is too tall could caused overfitting, the depth of 10 is ideal. Moreover, the number of trees is optimized as 200 trees, as more than 200 tree would decreased the test performance.

The top 5 most important splits and their importance are:
call_type_Medical_Incident 0.195761
unit_type_MEDIC 0.125675
call_type_Alarms 0.110008
unit_type_ENGINE 0.097458
call_type_Structure Fire 0.088325.

4.3.4 Support Vector Machine

Table 4: Best Parameters for Support Vector Machine Determined by Cross Validation on the Training Set

Parameter Name	Best	Interpretation
kernel	"rbf"	kernel function of SVM, "linear", "rbf(radial)
C	1	inverse of λ value(weight) for regularization term
gamma	"auto"	kernel coefficient for the kernel function
shrinking	True	whether to shrink

As we ran through 5-fold grid search cross-validation in Python, the optimal "C" chosen for the model is 1, as choosing a difference "C" may completely change how the separating hyperplane for the classification. On the other hand, a large "C" may increase the risks of overfitting. "gamma" is set to be "auto" as same reason as for "C", large "gamma" makes the model to fit the data exactly. The optimized kernel selection is "rbf", which is the radial kernel. The 'decision_function_shape' is optimized as "ovo", as "ovr" is usually used for multi-class classification.

4.4 Final Model Assessment Using Test Set

For each of the methods we previously used cross validation to select the best parameters for model selection. Now we take each best cross-validated model and assess its performance on the test set using four different quantifications: accuracy, precision, recall, and F1 score. Table 5 shows the model assessment results on the test set. See **Figures 5, 6, 7, and 8 in the Appendix** for the color-coded confusion matrices for logistic regression, K-nearest neighbor, random forests, and support vector machines. See **Figure 9 in the Appendix** for a plot of the ROC curve for logistic regression. The ROC curve should hug the upper left corner of the square, but it does not.

Table 5: Method by Method Comparison of Accuracy, Prediction, Recall, and F1 Score, on the test set. Random forests have the highest recall and are therefore the best.

Method	Accuracy	Precision	**Recall**	F1 Score
Logistic Regression	.729	.767	.682	.722
K-Nearest Neighbor	.691	.779	.557	.649
Random Forest	.724	.754	**.688**	.716
Support Vector Machine	.732	.770	.684	.724

Although we have four quantifications of the quality of our models' predictions, we deem *high recall* as the most important quantification for safety considerations. Namely, a true emergency should be predicted as an emergency, because an overlooked true emergency (false negative) could have lethal consequences. For comparison, a true non-emergency predicted as an emergency is not dangerous, though it is a waste of resources.

We explain for a moment the meaning of "false negative". Since our null hypothesis is "non-emergency", positive means "emergency". So "false negative" means the incoming call is "emergency" but we predicted "non-emergency" and ignore it. We prioritize the quantification *high recall* because it increases when false negatives are fewer, i.e. when the lethal outcomes are fewer.

Since we prioritize high recall over the other three quantifications, we consider random forest with 100 trees and 8 maximum tree height as our best model for prediction because it has the highest recall. If on the other hand we prioritized high F1 score, by definition the harmonic mean of precision and recall, we would decide logistic regression or support vector machine is the best model for prediction, since their F1 scores are almost the same (.722 and .724) and the highest among all F1 scores.

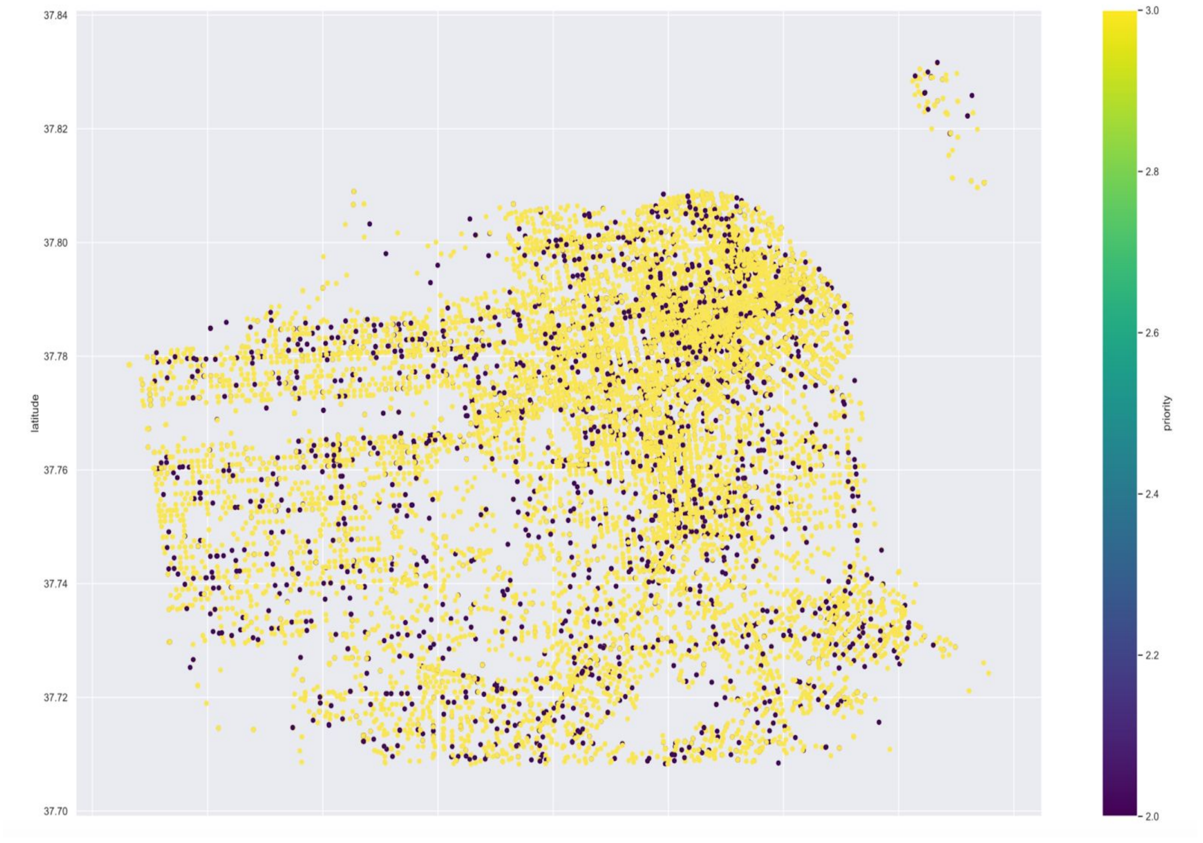
To increase recall in logistic regression, we could lower the threshold for emergency decision from default .5 to .35.

Appendix for Figures

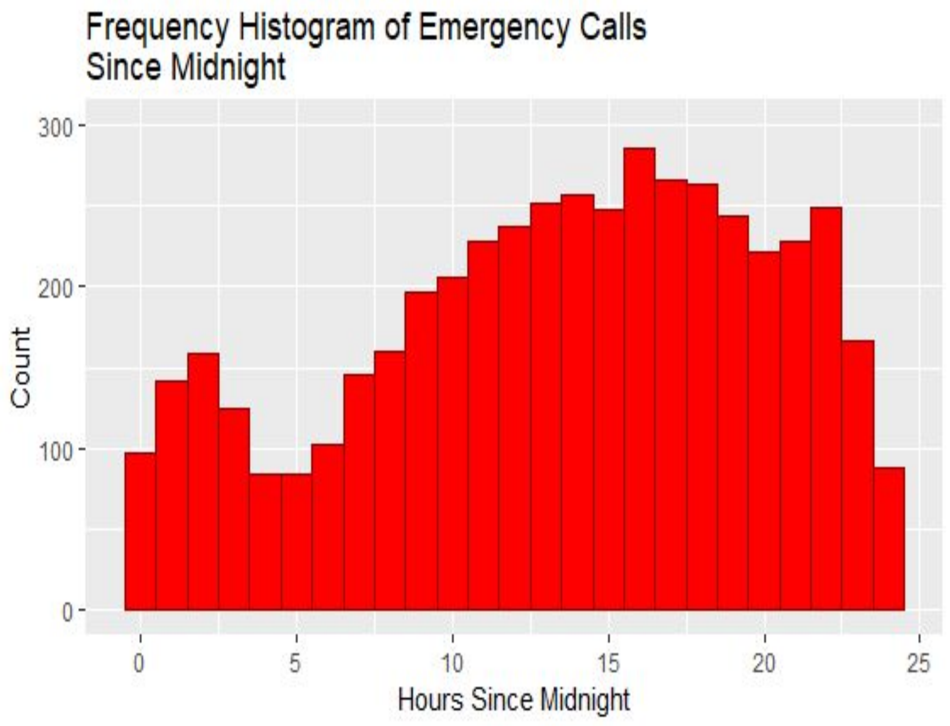
Hongru Fang, Thomas Fiore, Ming Gao, Zizhao Zhang

Visualizations for 3. Exploratory Data Analysis Visualizations

Figure 1. This is a plot of the latitude and longitude of each row in the smaller exploratory data set with 5,861 rows. A yellow dot indicates an emergency call, a purple dot indicates a non-emergency call. Clearly there are many more emergency calls. In fact 80% are emergency calls, and 20% are non-emergency calls. We can clearly see the shape of San Francisco, including Treasure island in the upper right corner.



Figures 2 and 3. These two frequency histograms show the incoming call times in the exploratory data set for emergency calls and non-emergency calls. The numbers on the horizontal axis indicate the number of hours since midnight. Most calls of both types are during hours in which most people are awake, namely 9 am to 11 pm.



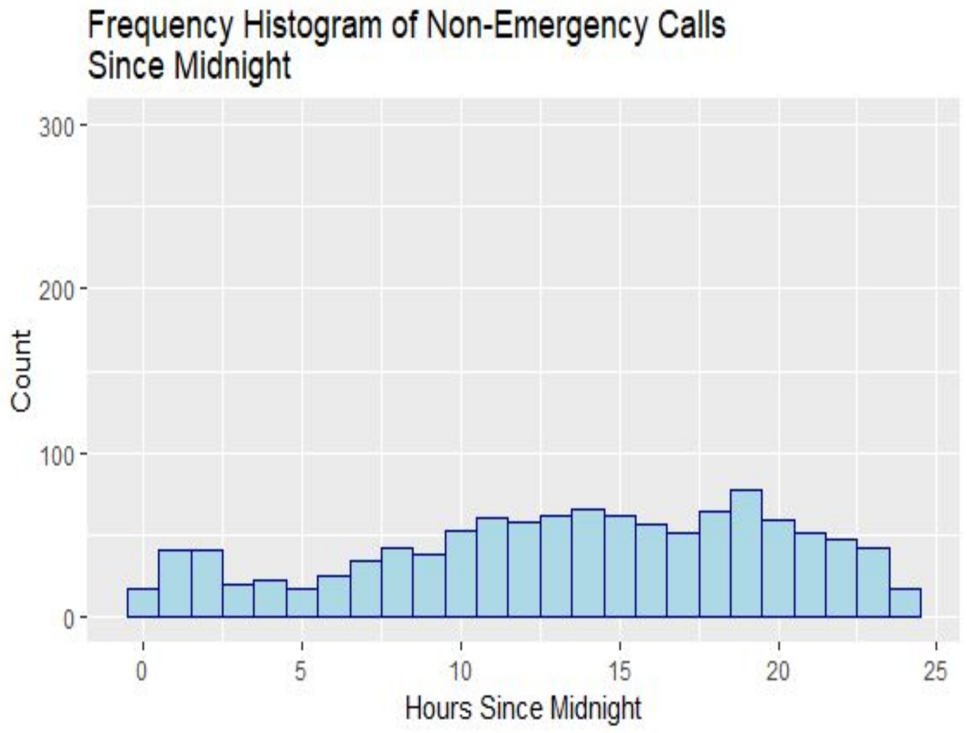
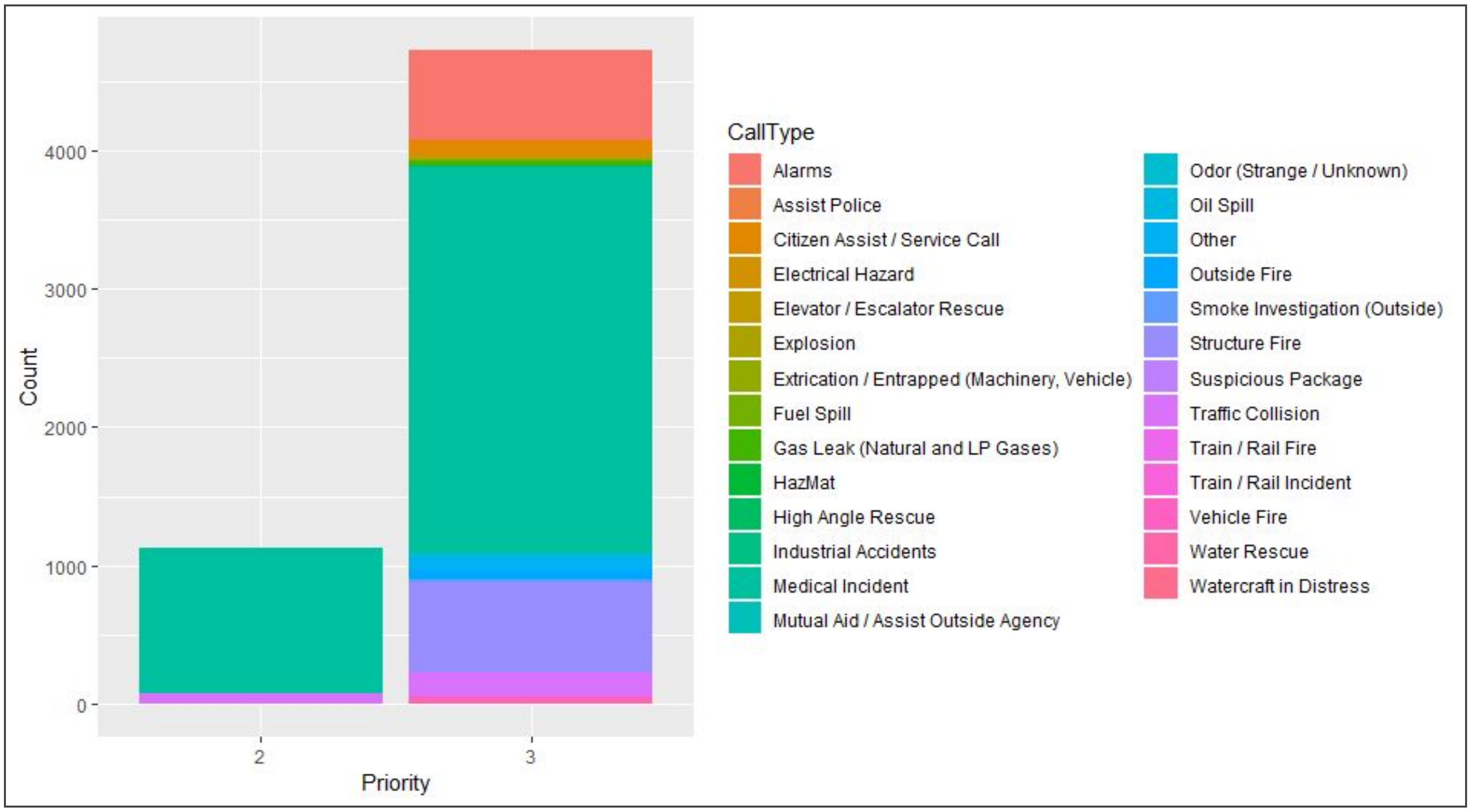


Figure 4. Breakdown of non-emergency calls and emergency calls by call type. We see the largest category in both non-emergency and emergency calls is medical incident.



Visualizations for
4. Final Model Assessment Using Test Set

Figure 5. We trained a *logistic regression* model for the binary response non-emergency/emergency with all explanatory variables described in Section 2. For training we used 80% of our reduced data set, which had 9209 rows and 61 columns. Thus our training set had 7367 rows with 60 features. The test data was 20% of our reduced data set, thus the test data has 1842 rows with 60 features. Both training data and test data had a 50%-50% split between emergency and non-emergency priorities. The color coded confusion matrix for the test data is below. The two dark squares indicate logistic regression performs well in recognizing both true non-emergency calls and true emergency calls.

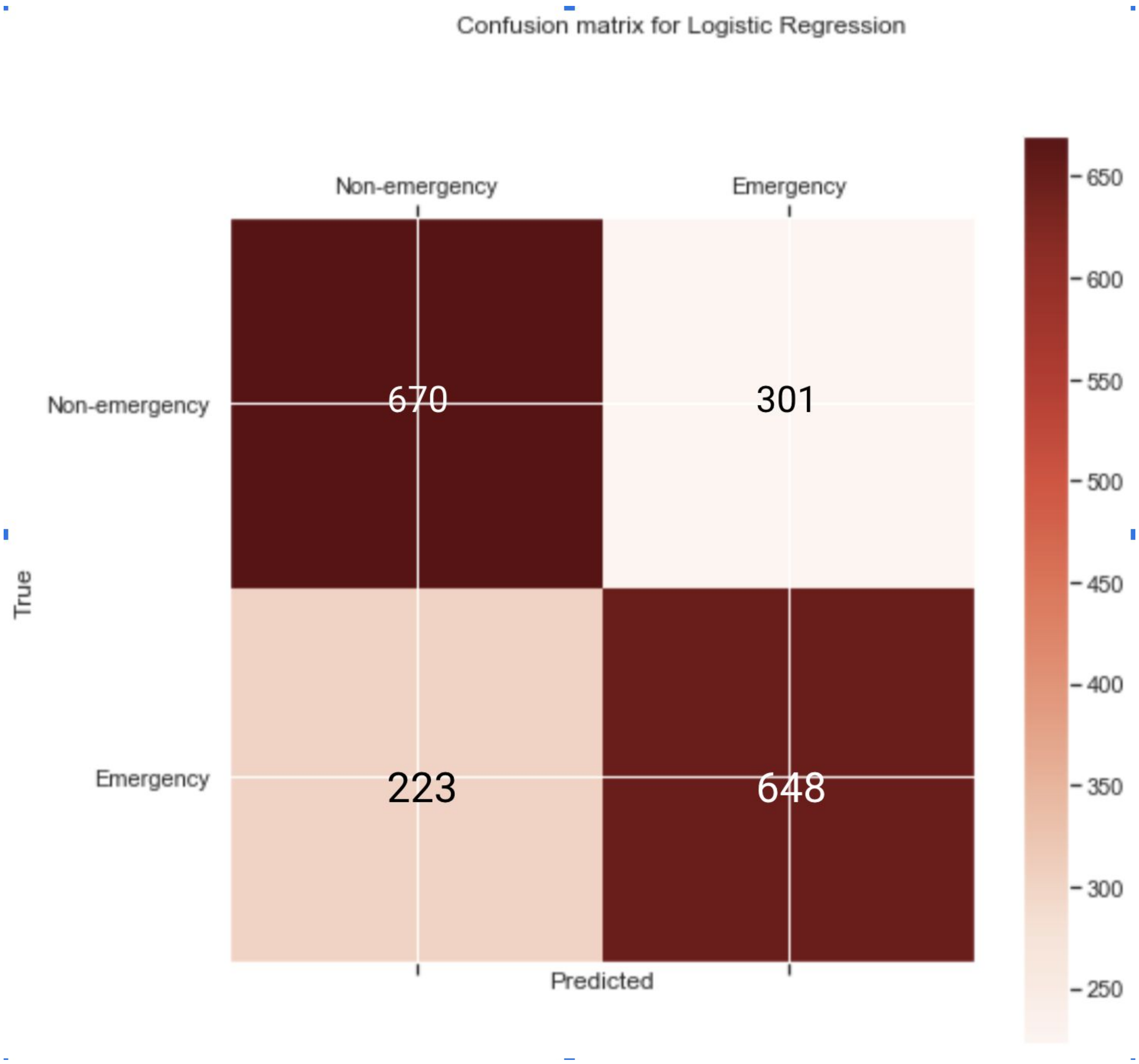


Figure 6. We also trained a *K-nearest neighbors* model for the binary response non-emergency/emergency with all explanatory variables described in Section 2. The training data and test data are the same as above for logistic regression. The color coded confusion matrix for the test data is below. The upper left dark square indicates K-nearest neighbors performs well in recognizing true non-emergency calls, but the light red square on the bottom right indicates its performance in recognizing true emergencies is not so good, a very dangerous situation indeed.

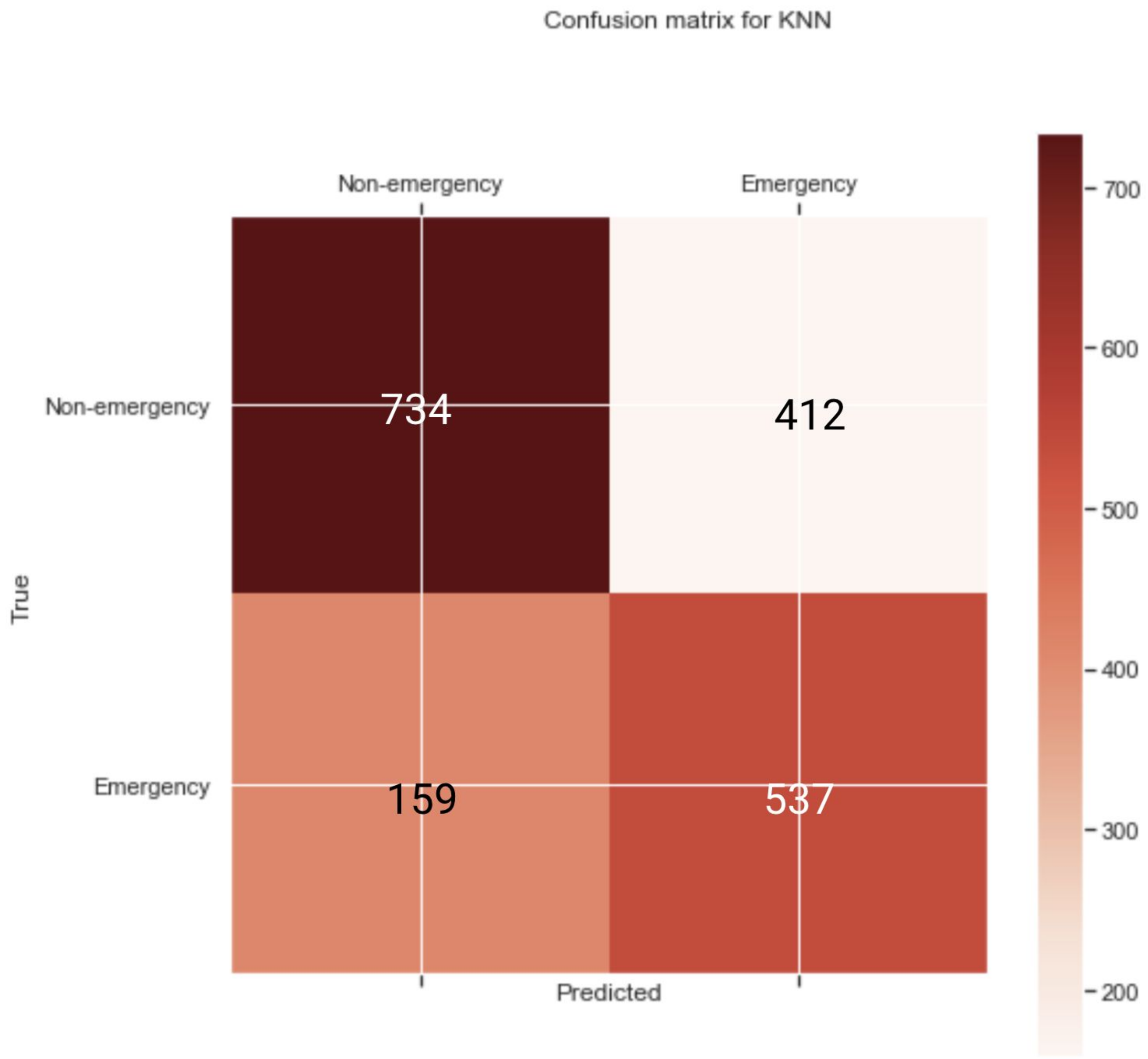


Figure 7. We also trained a *random forest* model for the binary response non-emergency/emergency with all explanatory variables described in Section 2. The training data and test data are the same as above for logistic regression and K-nearest neighbors. The color coded confusion matrix for the test data is below. The two dark squares indicate random forests perform well in recognizing both true non-emergency calls and true emergency calls.

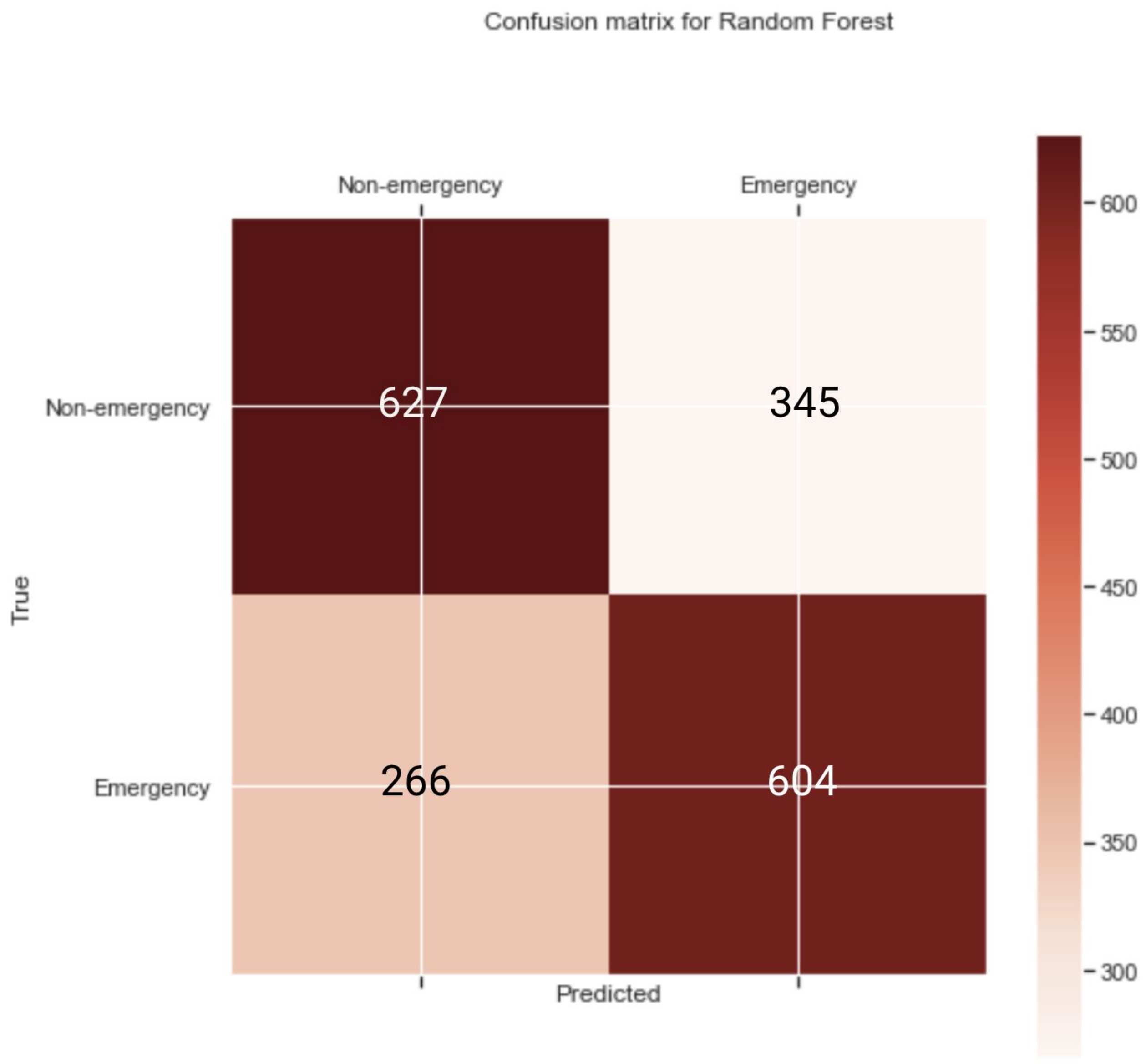


Figure 8. We also trained a *support vector machine* model for the binary response non-emergency/emergency with all explanatory variables described in Section 2. The training data and test data are the same as above for the other three models. The color coded confusion matrix for the test data is below. The two dark squares indicate support vector machines perform well in recognizing both true non-emergency calls and true emergency calls.

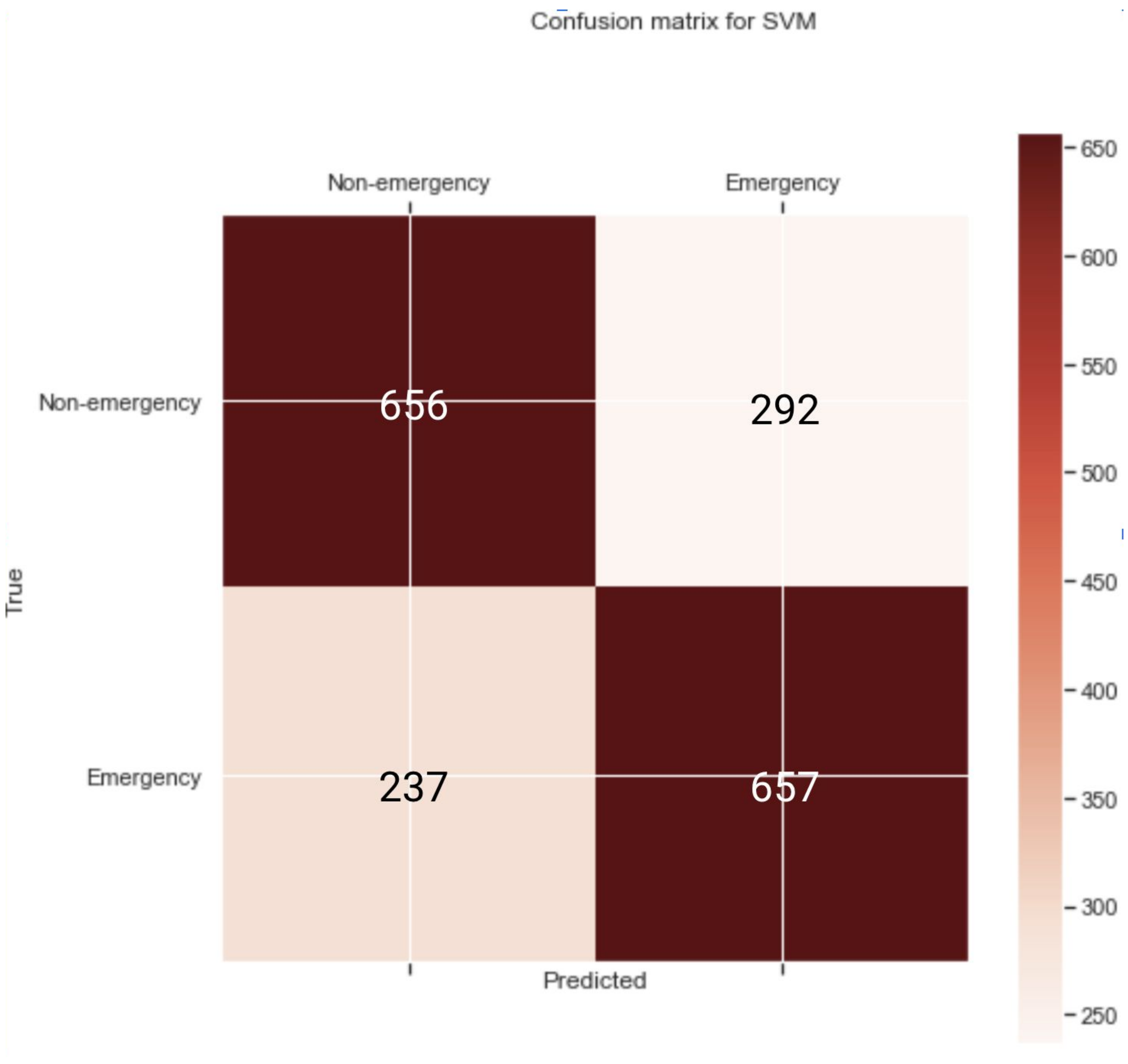
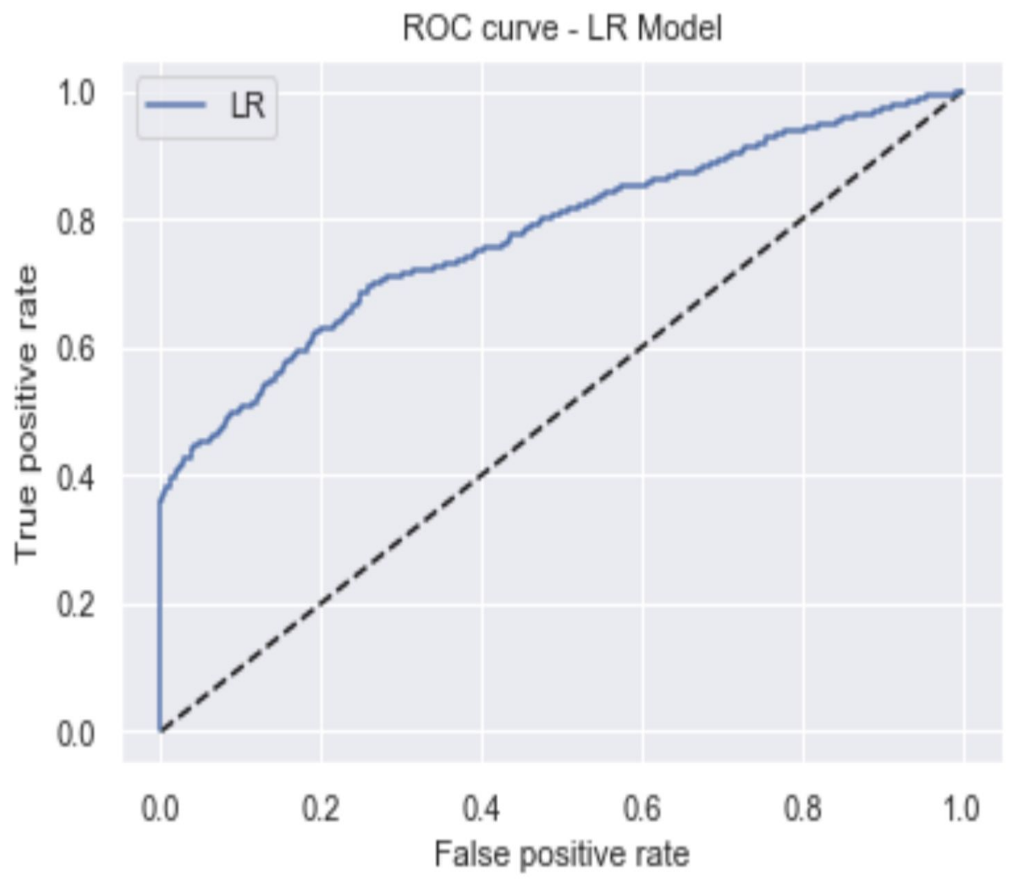


Figure 9. Below is the ROC curve for *logistic regression*. In the plot, the *true positive rate* (sensitivity) is plotted as a function of the false positive rate (1 minus specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.


ROC curve - LR Model
LR
True positive rate
False positive rate