911 CALL PRIORITY: MACHINE LEARNING IN VIRGINIA

A PREPRINT

Muhammad H. Sareini
University of Virginia
Charlottesville, VA 22903
mhs3vh@virginia.edu

Sammy R. Hecht
University of Virginia
Charlottesville, VA 22903
srh2kq@virginia.edu

Javier Rosas Ruiz
University of Virginia
Charlottesville, VA 22903
jr2dj@virginia.edu

May 3, 2019

ABSTRACT

Our project aims to predict the severity of a 911 call in order to help law enforcement prioritize their 911 calls, to help with public safety. Our data labels the severity of a 911 call with a number from 1-5 and our goal is to predict this number through categorical machine learning methods such as Logistic Regression, Support Vector Machine and a Decision Tree. We are going to implement each of these ML techniques individually, and then use an ensemble method to merge them. We are going to be analyzing the error at each step of the way, with the goal of reducing it as much as possible. The errors that we will be looking at are the accuracy, recall, precision and F-1 score.

1 Introduction

Our project used various machine learning classification algorithms to classify the priority of a 911 call (1-5, with 5 being the highest priority). We know that time is of the essence in a major emergency. For example, a small public disturbance where no harm has occurred should not be addressed before a severe car crash, where people's lives are on the line. The main goal of this is to provide a classifier that can flag calls automatically during high-traffic times, allowing emergency resources to be allocated as effectively and as quickly as possible hoping that this small incrase in efficiently would be enough to save someones life. In our Proposal section, we go over how we created three initial classifiers: Logistic Regression, SVM, and Decision Tree Classifier, and how they performed compared to each other. In our Final Report section, we explain how we iterated on some of the algorithms from the Proposal, and go over our new experiments with the introduction of three new classifiers: Random Forest Classifier, Ensemble Classifier, and Boosting. Each algorithm was implemented in hopes of performing better than the previous, allowing better classification of 911 calls. Our report finishes up with a Results section, where we highlight how our algorithms performed and which we believe is the best classifier for the task at hand.

2 Proposal

2.1 Motivation

Crime is detrimental to the well-being and prosperity of any society. We have obtained a data set that includes multiple metrics about crimes in the state of Virginia. Our intention is to use various metrics/features of the data set in order to predict the severity of a crime. The severity of a crime can be measured by a categorical number that goes from 1 to 5 (5 being the most severe crime, like murder, for example). This prediction could potentially reveal useful information that law enforcement could use in order to prevent severe crimes from occurring in the future.

2.2 Dataset

https://data.vbgov.com/Public-Safety/Police-Calls-for-Service/7gep-fpmg

2.3 Related Work

With some research, it was found that there are some similar studies have been completed regarding ML and 911 call data. A project completed at RIT analyzed emergency events to capture deviations in 911 call patterns to support emergency responders [1]. The goal of this study was to analyze and prioritize emergency cases to aid in making efficient decisions for high priority calls.

Another study was done to predict 911 calls using spatial analysis [2]. Analysis, like hot spot analysis, were used to see which areas are most active to help policy makers better distribute resources. The better distribution of resources, the study hoped, would lead to better response time, and a reduction in calls due to proactive measures [2].

2.4 Intended Experiments

We intend to conduct experiments in order to predict the severity of 911 calls. The data set specified includes features such as location, time of call, end time of call, date and others. Initial experiments will involve plotting the correlation between these features with respect to the priority of the call. These experiments will expose important attributes of the data set and hint at the types of prediction techniques that will be helpful in the future. If possible, regression techniques will be used to predict the priority of the call, but more complex models may be developed if the error is inadequately large. The priority of each call is categorical in nature, but is represented numerically as a number between one and five. Thus, the data will be trained on eighty percent of the data set and tested against the further twenty percent. Given a vector of features, the model will predict a floating point number between one and five. The performance of the model will be measured as the RMSE between the floating point predictions and the true priorities of the testing data. This metric will be used iteratively with different techniques in order to discover the model which minimizes the error to an acceptable level.

3 Checkpoint

3.1 Methods

To classify our calls, we decided to try three basic machine learning algorithms. We chose Logistic Regression, SVM, and Decision Trees. We chose these because they are three different implementations to solve classification problems. By using three different types of classifications, we hope to find that one model is better suited for our kind of data than the other two. The success of some models over others may additionally hint at underlying tendencies in the data. Any insight gained through this experiments will prove useful when we consider using more complex and powerful classification models.

3.2 Preliminary Experiments

3.2.1 Logistic Regression

The first and most obvious model to try on this data was logistic regression. Logistic regression takes an instance of some sample, and predicts its classification based upon a predicted probability of it belonging to one class or another. This model can be used for a multi-class problem by developing a binary model for each class and then comparing the predictions for each class against each other. Logistic regression has one hyperparameter, C, which controls the regularization of the parameters. With some broad-stroke tuning of C, a value of 1 produced the best results. The measurements achieved with this value are as follows:

Accuracy: 0.9693387452776377
Precision: 0.6657533149104221
Recall: 0.9687119406349376
F1 Score: 0.6970868468870449

This model produced the best accuracy and recall scores from the preliminary experiments with both approaching .97. This model suffered slightly with precision, which in turn brought down the F1 score. The logistic regression model provided a good baseline to work from, and shows promise for further improvement through more fine-grained hyperparameter tuning. This model maintains great accuracy and recall, but perhaps a more complex model could improve the precision and F1 scores.

3.2.2 SVM

The sklearn LinearSVC library supports multi-class classification, so this was an obvious method to try out. There are several hyper-parameters that we had to control in order to get the best possible results. For example, the C hyper-parameter controls the trade off between the slack variable and the "street width". After playing around with various C hyper-parameters, we concluded that the best C value was 1. The number of iterations is a hyper-parameter had to stay relatively low because the data set was almost 600,000 entries, so it took an extremely long amount of time to fit the model with a large amount of iterations. Another hyper-parameter we used was hinge loss, which is a technique used for SVMs that can be utilized for multi class classification. After tuning our hyper-parameters, we got the following measurements:

Accuracy: 0.9691505519240343
Precision: 0.7707131568089028
Recall: 0.6610431841598198
F1 Score: 0.6901913267759773

With a really good accuracy, and decent precision, recall and F1 scores, we can determine that this model is fairly good at classifying the priorities levels of 911 calls. This model is probably going to be very useful later on for the ensemble method, where we will combine various models to get better accuracy, precision, recall and F1 score.

3.2.3 Decision Tree

We used the Decision Tree Classifier as one of our initial classification models on our data. To make sure we did not over-fit the data, we decided to set the tree maximum depth as 2. After training the model, we had the following performance scores:

Accuracy: 0.7848583333333333
Precision: 0.5412268928390945
Recall: 0.30713560243964166
F1 Score: 0.3405258278733306

With only a decent accuracy, and terrible ratings for precision, recall, and F1, we can see that this model was not good for classifying our data. By increasing the maximum depth, we increase all performance scores but we run the risk of over-fitting.

3.3 Next Steps

The next step for our ML4VA project is to use ensemble learning in our models to see if we can get a better classifier compared to how our model currently performs. We hope to use random forest and boosting, as well as other ensemble methods time permitting. Within our current models, there is still room for improvement as well. Further experimentation with one-versus-one and one-versus-all strategies could lead to performance boosts. We would also like to do a more comprehensive tuning of the hyperparameters on each model. If the logistic regression model or the SVM model can be tuned to increase their precision and f1 scores, then different methods may not be necessary. By experimenting with and tuning these different models, we hope to maintain high accuracy and recall, while progressively improving the other important metrics. In addition to that, we are planning to implement a Gaussian Kernel for the SVM to see if we can classify the data more accurately by transforming our data into higher dimensions.

4 Final Report

4.1 Methods

After experimenting with simple models such as logistic regression, we looked to try more complex classification models to see if we could improve upon the precision, recall, and f1 scores which our simple models were lacking in. These more complex models included random forest, an ensemble of the previous methods, and gradient boosting. We hoped that these more powerful models might be able to generalize better to the testing set, and yield more well rounded performance metrics. We also looked to improve upon parameter tuning with some of our previous models to see if we might be able to get acceptable results from the models we were already trying. Our decision tree and SVM classifiers were the main candidates that we saw room for improvement on, and thus we continued to pursue better models.

4.2 Experiments

For our final report, we focused upon five experiments. We started by trying to further tune and improve our SVM model, which consisted of trying out different C hyper-parameters. In addition to that, we kernelized the SVM with a Gaussian RBF. Then we investigated the effect of the max depth hyper-parameter on our decision tree. With a max depth of 2, we were not getting acceptable results, which encouraged us to experiment more with its configuration. After improving our decision tree model, we looked to further pursue this idea and construct a random forest classifier. We hoped that this approach would build off of our improvements with the decision tree in order to create a more powerful model. With our results beginning to improve, we then experimented with an ensemble learning model which utilized our decision tree, logistic regression, and SVM models. This finally brought us to our most complex model in gradient boosting. Our ensemble model performed well, but we hoped to make gains in recall and f1 score by employing a gradient boosting algorithm. After performing these experiments, we looked to evaluate the results, and choose the model which best aided in our problem of classifying 911 call priority.

4.2.1 SVM

After trying several different C parameters, we found that C=1 was the optimal value which yielded the highest accuracies. After that we kernelized the SVM with a Gaussian RBF. This was an extremely time consuming task to do because our data set was so huge. To train the kernelized SVM and get the accuracy score, it took us over one hour and a half. Unfortunately, the kernelized SVM did not improve our results at all. We used an alpha of 0.1 for our kernel, and C=1.

Regular SVM metrics:

Accuracy Score: 0.9689936222111335
Precision Score: 0.920261552357195
Recall Score: 0.6628074524914764
F-1 Score: 0.6941962931810439

SVM with Kernelized Gaussian RBF metrics:

Accuracy Score: 0.928133522111335
Precision Score: 0.8974524924976
Recall Score: 0.71291381899338
F-1 Score: 0.6010431962931813

4.2.2 Decision Tree

In the project checkpoint, we decided to choose a maximum depth of 2 for our DecisionTreeClassifier to "prevent over-fitting". However, after some more parameter tuning, we realized that this model was severely under-fitting our data. For the amount of features the data has, this was a much too low minimum depth. Tuning allowed us to converge to a maximum depth of 70. This gave us the following performance scores:

Accuracy: 0.9610583333333333
Precision: 0.7716911449224426
Recall: 0.7358102746749348
F1 Score: 0.7516427398957578

All of the measurements increased significantly, making this model now comparable to the others unlike the poor perfoming model from the checkpoint.

4.2.3 Random Forest

We figured that since the Decision Tree Classifier is performing well now, we should try and see if a Random Forest Classifier can do even better. We used a maximum depth of 70 (since that is what performed best for the Decision Tree). After fine-tuning, we used 10 estimators in our Random Forest. This gave us the following performance scores:

Accuracy: 0.9685416666666666
Precision: 0.8939483268758561
Recall: 0.6959151692412575

F1 Score: 0.7421529437200082

As you can see, this gave us a huge increase in precision (almost 12%) at only a small cost to recall (1%). The small hurt to recall, in turn, slightly reduced the F1 score as well.

4.2.4 Ensemble Learning

At this point, we had trained many different classification models which all were performing at around the same levels. We wanted to see if we could combine these already created classifiers into a better performing Ensemble Classifier. We ended up choosing the Logistic Regression, SVM, and Decision Tree classifiers in our Ensemble Classifier. We did not include the Random Forest because when both Random Forest and Decision Tree were utilized, performance was hurt significantly. The Ensemble Classifier gave us the following performance scores:

Accuracy: 0.969925
Precision: 0.9703112433006822
Recall: 0.6685719583226668
F1 Score: 0.7009708502651231

This gave us the highest precision by far, and kept a very high accuracy as well. The performance did take a small hit to F1 and recall measures. Overall, this was a well performing model.

4.2.5 Boosting

After trying an ensemble method based upon our previous methods, we tried a more complex ensemble method in gradient boosting. Gradient boosting works by combining many weak models, usually trees, into a stronger model. It iteratively adds models, which work to improve upon the error of that particular models predecessor. The learning rate and number of classifiers are two important hyperparameters which can be tuned to yield a more accurate model and try to prevent overfitting. We found the best results with a learning rate of .2, and 50 classifiers. Our final Gradient Boosting model yielded the following performance scores:

Accuracy: 0.9648581844635313
Precision: 0.8866753181067324
Recall: 0.7310775582409732
F1 Score: 0.7759839794883954

This method outperformed our other ensemble method in recall and F1 score, but suffered in precision. The sequential nature of gradient boosting makes it very computationally expensive, which in turn makes it difficult and time consuming to tune the hyperparameters. Even with this limitation, however, gradient boosting still yielded arguably our best model as it was reasonably good in every metric, and boasted the highest F1 score. With more time to tune each hyperparameter, perhaps we could have increased the precision of this model to make it definitively more powerful than our other models.

5 Results

Many models performed well on this dataset especially in the metric of accuracy. However, the more powerful algorithms were able to target recall and F1 score more effectively. This is important to keep in mind with the dataset we're discussing, since most crimes are classified as relatively low priority. This can lead models to over predict the common values, and miss others. With an issue such as 911 calls, it is important that our model is sensitive enough to catch less less common calls such as the ones with the highest priority. For this reason, it appears that our Gradient Boosting model provided the best fit for the problem. This model had a very high accuracy score of .96, but it also had the highest F1 score out of any model at .78. Recall and F1 score are particularly important in the area of crime response, since the cost of false negatives is so high. Falsely predicting a high priority crime could lead to a loss of life. False positives, while they lead to a waste in resources, generally have much more benign consequences. This model is by no means a replacement for those who monitor 911 calls, but it could serve as a resource to quickly estimate the criticality of different situations. While our model is by no means perfect, it was impressive to see how well these algorithms were able to predict priorities from a set of features that seemed only minorly relevant.

6 Contributions

Muhammad H. Sareini - I was responsible for the data cleaning and pre-processing, as well as training the Decision Tree classifier and analyzing its performance. I was responsible for the random forest classifier and the ensemble method experiments as well. I also helped with writing various parts of the report.

Sammy R. Hecht - I was responsible for training and tuning the logistic regression classifier and reporting on it. I also created and reported upon the gradient boosting classifier. Finally, I reported our methods and experiments for the final report.

Javier Rosas Ruiz - I was responsible for training the SVM and analyzing its performance using accuracy, recall, precision and F1 score. I had to tune the hyper parameters to get the most optimial results.

References

[1] https://www.cs.rit.edu/usr/local/pub/GraduateProjects/2165/ss8248/Poster.pdf

[2] https://www.researchgate.net/publication/221541935_Predicting_911_Calls_Using_Spatial_Analysis