ANALYSIS OF CHARLOTTESVILLE PARKING TICKETS

A PREPRINT

Jacob S. Rodal
jr6ff@virginia.edu

Harun Feraidon
hf6mw@virginia.edu

Kaan Katircioglu
kk2dh@virginia.edu

April 26, 2019

ABSTRACT

This project aimed to find a strategy for where citizens of Charlottesville should and shouldn't park illegally based on local parking ticket data as well as predict if a ticket appeal will be successful. We first cleaned our data which involved geocoding and date/time format standardization and then we plotted a map that shows the distribution parking tickets in Charlottesville. Classification models were used to predict whether or not a parking ticket is appealed based on various factors leading to the initial ticketing. Through this project, we gained valuable insights about parking ticket patterns in Charlottesville.

1 Introduction:

The problem we are tackling involves being strategic with parking around Charlottesville. Because Charlottesville is a college-city, many tourists and visiting families will attend the city for a short duration. During their stay, because they are not completely fluent with parking regulations around the city, people will often struggle in finding parking spots. Some areas will be more heavily regulated, resulting in a greater amount of parking tickets. On the other hand, other areas might be less regulated, resulting in a lower amount of parking tickets. By exploring this dataset [1], we will find a strategy for where one should park and should not park. We did so by exposing how certain features are related to the chances of getting a ticket, such as the area, the time of day, and what violation may be related to the ticket. Using this information, we also determined what factors contribute to successfully appealing. Citizens of Charlottesville could use this information to decide whether or not they should bother appealing a ticket.

The topic challenges the idea of parking with a relative confidence of avoiding a ticket in such an urban and chaotic city. Similar research projects have been done in the past, such as an experiment involving data analytics and decision trees to obtain a similar goal but in the city of San Francisco [2].

2 Method

Before creating our models, we first needed to clean our data. One thing we did was geocode roughly 12,000 street addresses into GPS coordinates. To do this, we utilized the google maps geocoding API and stored the results back into our pandas dataframe [3]. After obtaining GPS coordinates, we then modified a feature that contains date information. Essentially, the date data wasn't in a format that pandas works well with and some dates made no sense (e.g., some dates were from the future), so the date feature was transformed. Another feature that needed to be cleaned was a time feature. A consequence of the dataset being updated for 20 years is that the format in which time data is reported has changed multiple times throughout the years. The feature was put into a single format that pandas can understand. It was important to clean the date and time features because it will be helpful later on when we perform regression and classification that depends on these features. Once all of the data was cleaned, we plotted the GPS data over top of a basemap of Charlottesville so we could get a better understanding of where all of the tickets occured in Charlottesville.

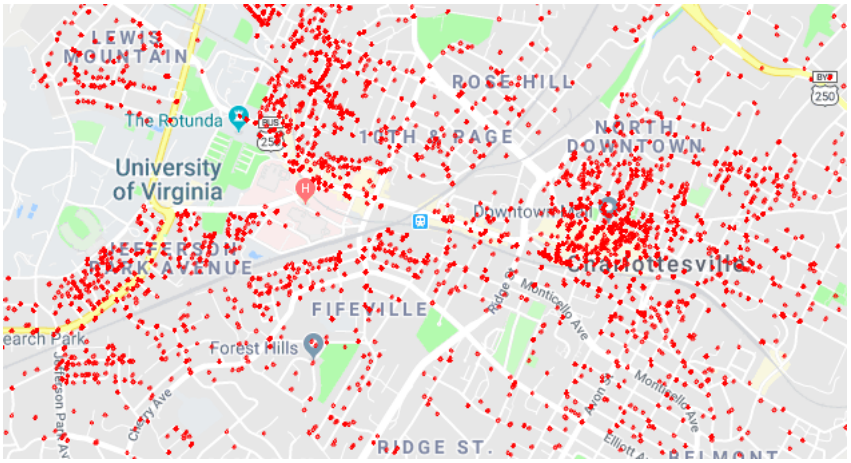

Figure 1: A portion of the map of Charlottesville. See the full map here.
[4]

When our program was finally ready to be trained, we experimented with several different models to see how they would perform against our data. Firstly, because our problem involved classification techniques, we fit our data among a variety of algorithms: Logistic Regression, Linear SVM, and Decision Tree. We also fit a Random Forest model on our data to get a perspective on how ensemble learning would perform. Finally, we also tried the following algorithms for classification: Neural Network, Naive Bayes, Gradient Boosting Classifier, and Nearest Neighbors to compare their performance against our other classification algorithms.

3 Experiments

Both the labels and an important feature, violation description, were converted to categorical info with one hot encoder. During the preliminary experiments, the entries with appeal status, labels, "pending" were ignored. Many of the potentially useful features were also ignored, due to additional cleaning and tuning required. Then, seven different classifiers (random forest, decision tree, linear SVM, nearest neighbors, neural net, logistic regression, naive bayes) were trained using the training data with the help of scikit-learn. Each classifier was given a train and test score based on the accuracy metric, and the top three models were chosen (random forest,linear SVM, and non-linear SVM).

After the preliminary experiments, we focused on cleaning and extracting more features that were omitted previously. First, from the date feature, we extracted the day of the week, as it is possible that the tickets given out on different days may be possibly easier or harder to appeal. Next, an hour of day feature was extracted from the time of day feature. As there were 30 violation descriptions, the team spent considerable effort on reducing this to a few meaningful features. First, thorough categorizing the different descriptions objectively, we were able to reduce the total description count to 21. This reduction included instances that were practically the same violation in real life but labeled using different words. We tried different combinations of combining the features subjectively. Additionally, categorized less frequent violation types into "other". As a final attempt to find a significant relationship between violation descriptions and appeal status we mapped each violation to its ticket prices and use the monetary values instead.

The following hyperparameters were analyzed for random forest classifier by using a random search with cross validation (100 combinations, 3 folds):

1. Number of estimators
2. The minimum number of samples required to split an internal node
3. The minimum number of samples required to be at a leaf node
4. Number of features to consider when looking for the best split
5. Maximum depth of the tree

We ran different permutations in parallel to speed up the hypertuning process for the random forest classifier. The Linear SVM model was hypertuned as well in an effort to better the test score. Linear SVMs with C values ranging from .001 to 1000 were trained, cross validated (3 folds), and tested for potential improvements in accuracy. Additionally,

Non-linear SVMs were also trained and hypertuned with C values ranging from .001 to 100 and γ values ranging from 0.0001 to 0.1 using a grid search with cross validation (3 folds).

The codebase for our experiments is hosted on github [5]. Some code for hypertuning the random forest classifier [6] and comparing the accuracy of various classifiers [7] was borrowed from the sources linked in the references.

4 Results

The seven tested classifiers in the preliminary stage and the results are shown below. The score metric is accuracy.

Classifier	Train Score	Test Score
Random Forest	0.723428	0.593286
Decision Tree	0.723428	0.589206
Linear SVM	0.577119	0.561387
Nearest Neighbors	0.646680	0.560645
Neural Net	0.561352	0.547663
Logistic Regression	0.549898	0.539132
Naive Bayes	0.548414	0.538205

The best parameters of Random Forest Classifier calculated in the experiment where different configurations ran in parallel are presented below.

```
{'n_estimators': 800,
 'min_samples_split': 10,
 'min_samples_leaf': 4,
 'max_features': 'sqrt',
 'max_depth': 50,
 'bootstrap': True}
```

Figure 2: Random Forest Best Hyperparameters

In Figure 3, different test scores over changing hyperparameter C are presented.

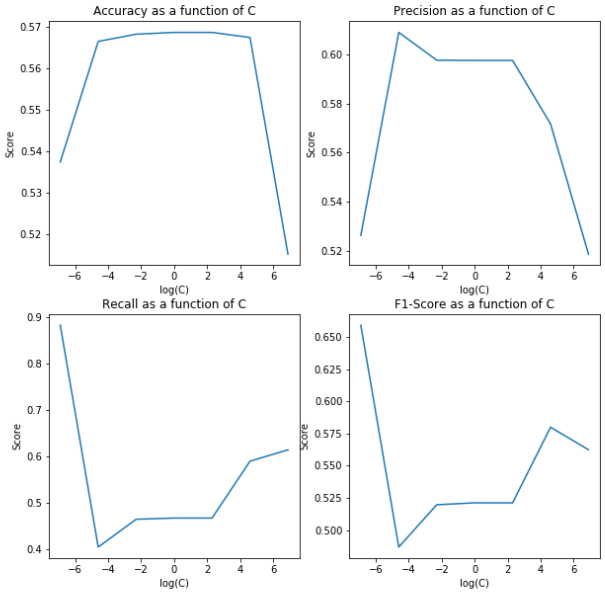

Figure 3: Linear SVM Scores

$C = 1$ was the optimal hyperparameter, meaning that our accuracy for the linear SVM likely cannot be improved much beyond the initial accuracy score achieved in the preliminary experiments.

Different test scores over changing hyperparameters C and gamma are shown below

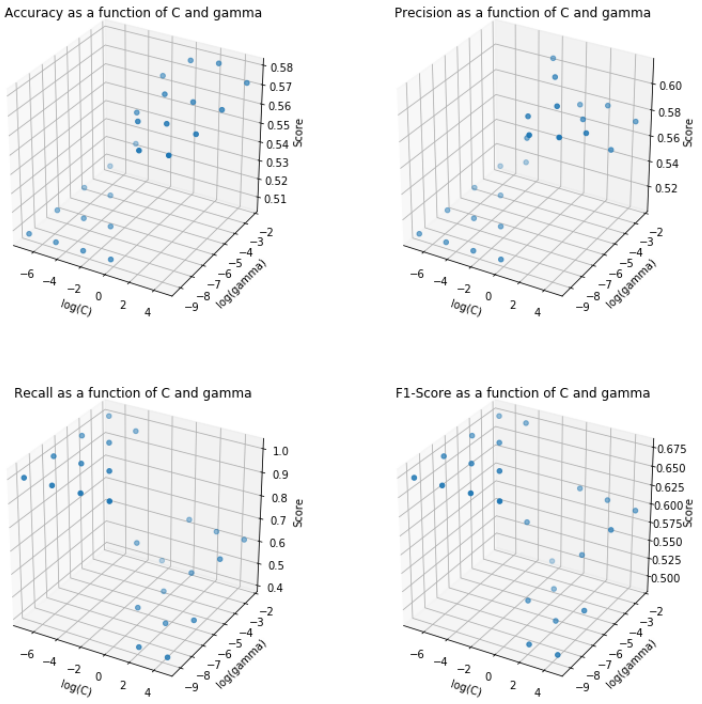

Figure 4: Non Linear SVM Hypertuning

Our top three models, the random forest, the linear SVM, and the non-linear SVM, were fit using the training data and tested against the testing data. The results are shown below:

Classifier	Train Score	Test Score
Random Forest	0.766326	0.600303
Linear SVM	0.580352	0.579674
Non-Linear SVM	0.556739	0.549962

5 Conclusion

The results have been inconclusive on finding a reliable way for Charlottesville residents to decide if their tickets would be appealed.

Based on the final results, we suggest choosing a random forest with the parameters specified in Figure 2. The models didn't perform significantly better than their baselines prior to hypertuning - it is possible that the increase in accuracy can be attributed to random error.

The accuracy is around 60%. In real life, this may mean that besides from some rare ticketing instances that consistently yield to a certain outcome, the model is not much better than a coin flip on deciding this problem. Applying to see whether a ticket will be granted an appeal is, unfortunately, still the best option that Charlottesville residents have.

Before experimenting, we suspected a high correlation between the appeal status and violation description, as some offenses may be more strict than others. We also expected to see the tickets issued at clustered locations in certain dates would have more agency on the appeal status. However, neither of these predictions were supported by the test's results.

Since the accuracy was not promising, we also had to consider the recall scores. With a high recall score, one could at least reduce the domain space of instances that would get an appeal. This way, residents would at least have a good idea of when it's definitely a waste of time to apply for appeal.

We achieved models with reasonably high recalls without losing much accuracy. However, this is still not a success. Consider the recurring cases where the recall is exactly or very close to 1 and accuracy is almost 50%. The ratio of positive instances to negative instances are $1:1$. The model is just calling "appeal granted" for every instance. Although this may be the hope that the Virginia Residents need, it's certainly not a successful result.

Another surprising shortcoming was how we were unable to find any orientation where the violation description was useful. It turns out, the cost of the ticket was not related to its appeal application at all. We can assume that our dataset was insufficient to be able to build a model that helps predict appeal outcome. However, it may be the case that no existing dataset would be able to help build a model for predicting parking ticket appeal outcomes. This is because in real life scenarios, there can be far too many random reasons why a ticket appeal may be a success or fail. To name a few, perhaps the judge was having a bad day, there might have been racial bias involved, there may have been too many tickets appealed recently thus the committee concludes that they need to fail an x amount of appeals before they can start passing them again. The list can go on.

References

[1] Charlottesville Open Data http://opendata.charlottesville.org/datasets/parking-tickets/data

[2] Sinclairs, Colin San Francisco Parking Ticket Analysis https://github.com/Csinclair0/SF_Parking

[3] Google, Google Geocoding API developers.google.com/maps/documentation/geocoding/start

[4] Rodal, Jacob; Feraidon, Harun; Katircioglu, Kaan; Parking Ticket Map of Charlottesville https://www.cs.virginia.edu/~jr6ff/pages/parking_ticket_map.html

[5] Rodal, Jacob; Feraidon, Harun; Katircioglu, Kaan; ML4VA Github Codebase https://github.com/jrodal98/ML4VA

[6] "Hyperparameter Tuning the Random Forest in Python." Koehrsen, Will, Towards Data Science, 10 Jan. 2018, https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

[7] "Classification with Scikit-Learn." Ahmet Taspinar, 1 Mar. 2018, ataspinar.com/2017/05/26/classification-with-scikit-learn/