Github Repo Link: https://github.com/daisy-abbott/LA-Crime/blob/main/RandomForest.ipynb

CS158 Final Project

Using Random Forest For Crime-Related Prediction

By Vivien Song, Alejandra Casteñeda, Daisy Abbott

INTRODUCTION

Our project primarily focuses on using a random forest classifier for classification to predict the area of a given crime based on a variety of features. We looked at a crime dataset consisting of recent (2020-2022) crimes committed in Los Angeles that were collected and recorded through the LAPD radio communications, along with records generated post-investigation of these crimes. The features contained in this dataset include but aren't limited to: Victim Age, Victim Sex, Victim Descent, Weapon Used, AREA, MO Codes, and others. Is this sufficient information for a classifier to make a relatively accurate prediction on where a given crime will occur, or has occurred? Our classifier aims to do precisely that: look at information from relevant features and make a prediction on which area in LA the crime was committed. We will also explore how to tune our hyperparameters for the random forest classifier so that it might be more accurate. As a disclaimer, we'd like to acknowledge that applications of machine learning, such as our project, may be used negatively by law enforcement in the real world. As such, our results and any similar experiments should not be used for predictive policing to identify crime hotspots or to redirect policing resources to already marginalized communities.

EXPERIMENTAL SETUP

Feature Selection

The original dataset has twenty features, so we began our experiment by performing feature selection. First, we narrowed down repetitive, irrelevant, or unindicative features for predicting (e.g. latitude and longitude, when we already have location for area prediction; or weapon description in text, when we already have weapon description code). We also looked at the feature distributions in the Kaggle dataset and experimented with which features yielded the highest accuracy. We decided on the final categorical features of victim descent, location, victim sex, and MO codes (can have multiple MO codes for a crime, so each was encoded as a string of codes); and final numerical features of weapon code, crime code, area, time occurred, and victim age. Then, we moved on to data pre-processing.

Data Pre-Processing

We cleaned our data by dropping all null values present in the dataset, as well as eliminated examples with any features containing a "dummy" value (e.g. when age was unknown, it was marked as a zero in the dataset, which skewed our binning process. As a result, we dropped all unknown/dummy data points). We also ensured we had balanced data before experimentation. To do so, we looked at the distribution of feature labels to see that they were relatively well-represented. We did not find that the distribution was imbalanced so therefore did not decide to use weighted examples to balance the distribution. After this, we label-encoded our categorical features, giving them numerical representations. We also binned age by decade and time by hour. We ran initial accuracy tests on these to see the effects of binning—see *Appendix A.*

Establishing a Baseline

To ensure our modeling was performing well, we established baseline metrics to evaluate the accuracy of our model's predictions. We had three baseline metrics from different classifiers: first, random classifier, which would randomly predict one of 21 areas; second, majority class, which would always predict the majority class; and third, decision tree with no depth limit and looking at all features for a split. We compared these classifiers to our model, shown in *Appendix B*. We experimented with 5 iterations on the data to ensure we were getting precision in our results.

Hyperparameters

We experimented with several hyperparameters for the random forest and, specifically, the decision trees within the random forest. The primary hyperparameters were number of decision trees, number of features to find the best split, and max depth of decision tree. First, we found the best number of trees for our random forest by calculating the accuracy starting at one tree to 200 trees (shown in *Appendix C*).

Github Repo Link: https://github.com/daisy-abbott/LA-Crime/blob/main/RandomForest.ipynb

Second, we experimented using the max number of features to find the best split for each decision tree. In contrast to sklearn's normal decision tree implementation, which considers all features before finding the best feature to split on, sklearn's random forest classifier randomly considers sqrt(number of features) for each decision tree to split on. This helps prevent overfitting through randomization. Because we have 8 features, our model used only 2 max features (sqrt(8)= 2.8), so we experimented using 1 - 8 and compared our accuracies. To ensure we weren't overfitting our data, we tested on train and test, and found that while testing on train did significantly increase our data, testing on test didn't decrease—thus, we are reasonably confident we are not overfitting our data. See *Appendix D.*

Lastly, we found the best tree depth for our model by calculating accuracy from 1 node until we reached purely leaves, all listed and graphed in our results in *Appendix E.* For each of the above experiments, we ran them on 5 iterations of our data and took the average to ensure a thorough approach. All were implemented in Python using the built-in sklearn library.

RESULTS

Through our initial experimentation with binning time by hour and ages by decade (shown in *Appendix A),* we found that binning ages gave us a lower accuracy (35%) than not binning (38%). Similarly, binning time by hour yielded a lower accuracy (39%), than not binning it (40%) . We also looked at dropping ages listed at 0 because those are dummy values for unknown ages. Without dropping, the accuracy was at 38% and with dropping the accuracy was at 39%.

For decision tree experimentation (see *Appendix C)*, we found that there was a sharp increase in initial accuracy from 1 to 50 decision trees (35% to 39%), and then a gradual increase in accuracy until we reached 180 decision trees (42% accuracy). Then, accuracy plateaus. From this, we see the best number of trees is 180, as it yielded the highest accuracy.

For max features experimentation (see *Appendix D*), we found that accuracy begins at 30% for 1 feature and steadily increases to 75% with all 8 features. This is a large improvement from the default parameter of 2 (with an accuracy of 39%), where sklearn's random forest used the square root of the number of features to prevent overfitting. Our model was only using 2 features because we had 8 features total. For decision tree depth experimentation (see *Appendix E*), we found that accuracy began at 11% for decision tree depth of 1 and steadily increased until 23 (accuracy of 40%). After 23, accuracy begins to plateau or decrease which could indicate overfitting. Thus, we determined that the best decision tree stopping depth was 23.

After testing and hyperparameter tuning, we came to our best model: a random forest classifier with 180 trees, 23 as the max depth, 8 features for max split, and no binning. We evaluated this model using a classification report (see *Appendix F*) and a confusion matrix (see *Appendix G*) as well as running the model through 5 iterations of the data which had an average accuracy of 75.06%.

CONCLUSION

Our final model achieved an accuracy of 75% due to data pre-processing, feature selection, and our hyperparameter modification. The hyperparameter with the most significant impact was max features for each decision tree to split on. We did not bin. Initially, we were surprised that binning lowered model accuracy. After further research, we discovered that binning may introduce arbitrary cutpoints and can result in a loss of information and granularity, skewing model accuracy.

To verify that our conclusions were significant, we produced a classification report through sklearn as well as a confusion matrix. The classification report (*Appendix F)* calculated average precision, recall, f-1, and accuracy for the 21 classes/labels, and all four of these average numbers were 74-75%. The confusion matrix (*Appendix G)* represents the correct and incorrect predictions per class — the diagonal represents the frequency at which the predicted labels match the true labels. Darker squares correlate to greater frequency of occurrences; lighter squares to less. The diagonal is significantly darker than the rest of the matrix (frequencies of 2000-5000), indicating reliable model performance.

APPENDIX A. Initial experimentation with binning and dropping ages.

Github Repo Link: https://github.com/daisy-abbott/LA-Crime/blob/main/RandomForest.ipynb

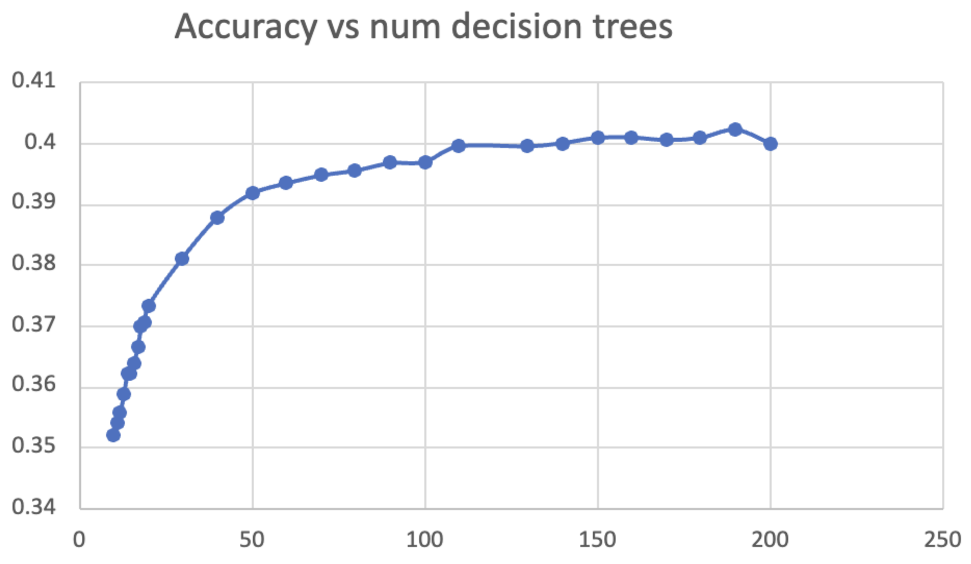
	Age Binned	**Age Not Binned**
Time Binned	0.3877	0.3574
Time Not Binned	0.3969	0.4034

	Dropping 0 value for ages for binning	**Not dropping 0 values for binning**
Accuracy	0.3891	0.3969

APPENDIX B. Baseline classifiers.

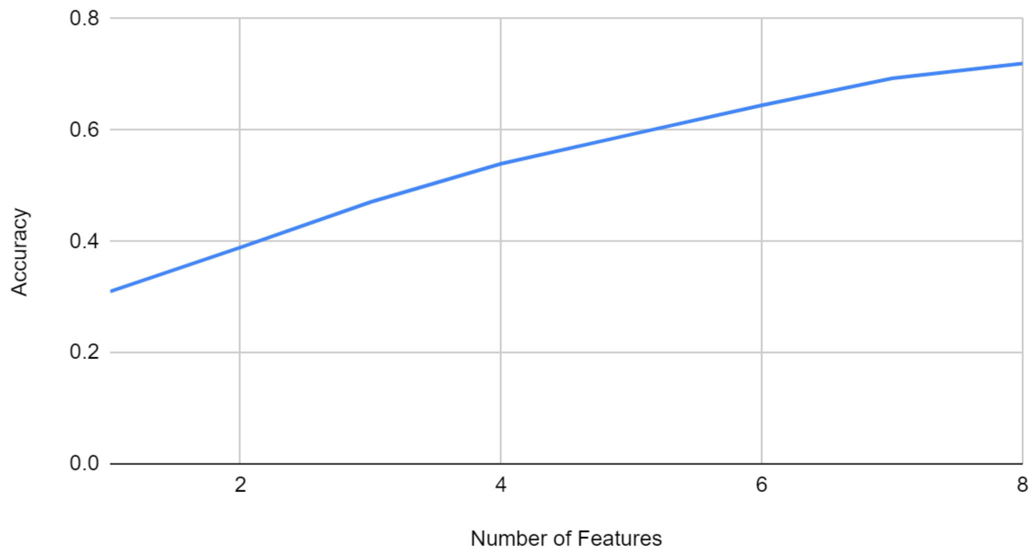
	Random Classifier	**Majority Class Classifier**	**Decision Tree Classifier**
Iteration 1	0.045669	0.067898	0.310142
Iteration 2	0.046831	0.067898	0.314974
Iteration 3	0.047989	0.067898	0.330955
Iteration 4	0.047252	0.067898	0.331238
Iteration 5	0.045410	0.067898	0.329769
Average	0.046630	0.067898	0.3234156

APPENDIX C. Accuracy vs. Number of Decision Trees

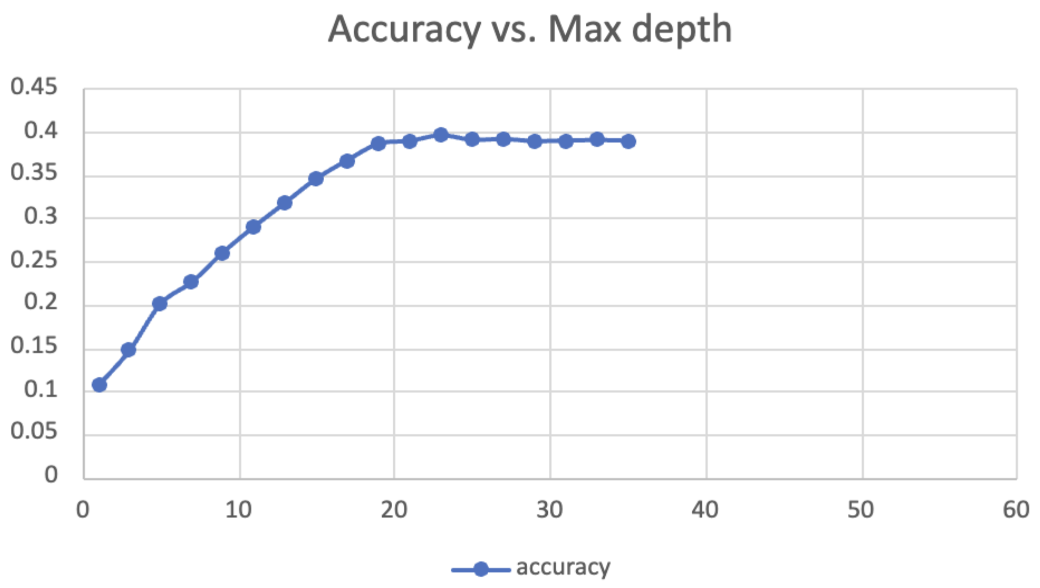


Github Repo Link: https://github.com/daisy-abbott/LA-Crime/blob/main/RandomForest.ipynb

APPENDIX D. Accuracy as the number of features for decision tree splitting increases.

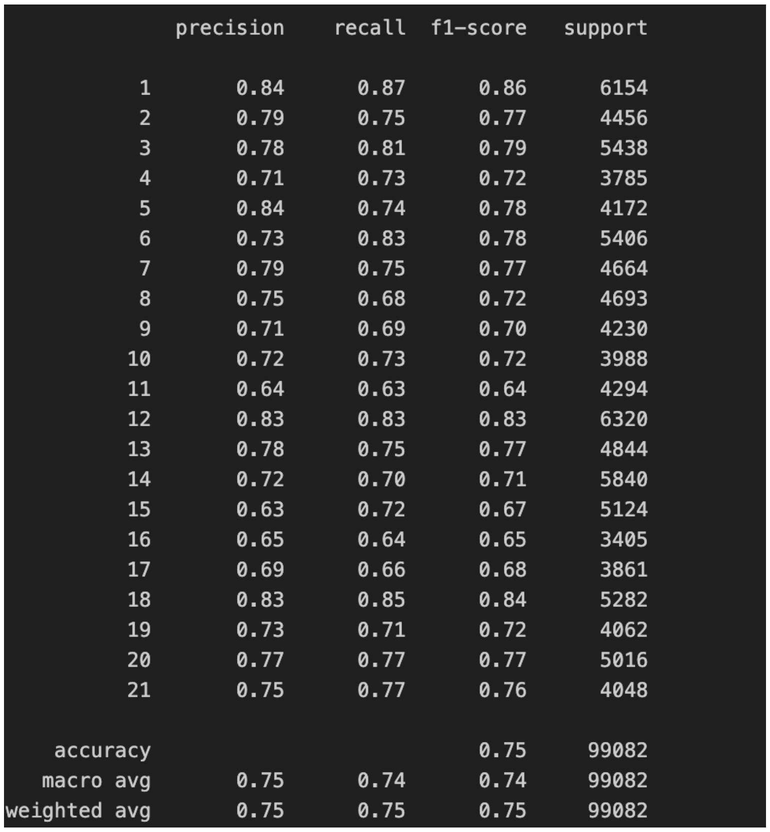


APPENDIX E. Accuracy over Max Depth of Decision Trees.



Github Repo Link: https://github.com/daisy-abbott/LA-Crime/blob/main/RandomForest.ipynb

APPENDIX F. Classification Report for our best model

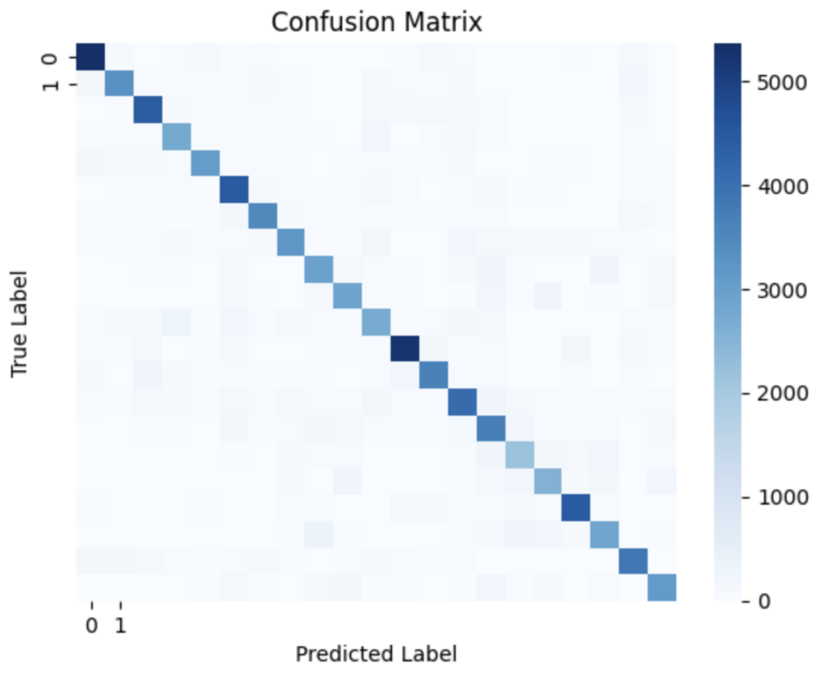


```
              precision    recall  f1-score   support

           1       0.84      0.87      0.86      6154
           2       0.79      0.75      0.77      4456
           3       0.78      0.81      0.79      5438
           4       0.71      0.73      0.72      3785
           5       0.84      0.74      0.78      4172
           6       0.73      0.83      0.78      5406
           7       0.79      0.75      0.77      4664
           8       0.75      0.68      0.72      4693
           9       0.71      0.69      0.70      4230
          10       0.72      0.73      0.72      3988
          11       0.64      0.63      0.64      4294
          12       0.83      0.83      0.83      6320
          13       0.78      0.75      0.77      4844
          14       0.72      0.70      0.71      5840
          15       0.63      0.72      0.67      5124
          16       0.65      0.64      0.65      3405
          17       0.69      0.66      0.68      3861
          18       0.83      0.85      0.84      5282
          19       0.73      0.71      0.72      4062
          20       0.77      0.77      0.77      5016
          21       0.75      0.77      0.76      4048

    accuracy                           0.75     99082
   macro avg       0.75      0.74      0.74     99082
weighted avg       0.75      0.75      0.75     99082
```

APPENDIX G. Confusion Matrix for our best model


Confusion Matrix
True Label
Predicted Label