ETHANWALKER-404-FinalProject

April 13, 2020

```
/home/ethan/anaconda3/lib/python3.7/site-packages/dask/config.py:168: YAMLLoadWarning: calling y
  data = yaml.load(f.read()) or {}
/home/ethan/anaconda3/lib/python3.7/site-packages/distributed/config.py:20: YAMLLoadWarning: cal
  defaults = yaml.load(f)
```

1 Abstract

Machine Learning is becomeing a more prevelant tool in the world of criminal justice. Often it is used to predict who will commit a crime or where crimes may occur. Seldom is it used to regulate the criminal justice system, however. In this project I examine prison inmate data and determine what machine learning techniques are effective at detecting the racial bias that has been shown to exist in this data. In this report I find ———

2 Problem Statement and Motivation

Last semester I took a look a data set containing the information of more than 7.5 million individuals that have been processed by the criminal justice system. I found that racial minorities were more likely to receive extreme sentences, agreeing with existing research around bias in the criminal justice system. In this project I will be exploring the data from a machine learning perspective. My goal is to determine if this data can be classified in such a way that is predictive of race. The idea is that perhaps racial bias can be detected in various systems by seeing how effective different machine learning techniques are at classifying an inmate's by race given their data.

This is an unconventional way to approach criminal justice data with machine learning. Often we see machine learning being used to attempt to determine who might be a criminal or where crimal activity may occur using social media data and other public information, which may include data the government owns, but which is not available to te public. These approaches often ignore or discount the ways that these techniques may disproportionately affect people of color and the poor. Many organizations have made official statements reguarding the use of machine learning in this way, often called predicitive policing. The ACLU for example released a statement listing civil rights related concerns about predicitive policing which was signed by several civil rights origanizations includeing the NAACP [1].

My objective is to go against the predictive policing paradigm and use machine learning to benefit these negatively affected classes of people by using machine learning as a diagnostic tool. If it can be shown that certain machine learning techniques are effective at classifying inmates

by race given incarceration related information, then we can inform policies that will attempt to correct for these systemic racial biases.

3 Data

3.1 Source and Credibility

The data that I will be using in this analysis is from one source. It is a database hosted on Data.gov and maintained by the State of Connecticut Department of Corrections. This source is highly credible because it is a primary source for the data. This organization is an official government agency which collects, maintains, and reports on this data.

3.2 Gathering and Cleaning

All the data which I am using in this report are freely available to the public. Collection and cleaning was relatively simple as the source data was well maintained. The file that I obtained from the Connecticut Department of Corrections is a very well maintained database. The largest issue I had with this file was mild inconsistency with the way in which certain data was encoded (ex. race was encoded as both `WHITE` and `WHITE\t`). The file is

`individuals.csv`.

3.3 About the Data

This data set contains individual information for 7.77 million people that have been processed by the justice system and recorded by the Connecticut Department of Corrections. Each individual is recorded along with their age, gender, race, offense, and sentence length, among other things.

Because there is so much to consider in what is found in the data set, I chose not to feature engineer as to avoid unneeded complexity.

The sample sizes among different races that are found in the Connecticut Department of Justice data are not similar. The sample size for American Indians and Asians is much smaller than that of Whites, Hispanics, and Blacks, hence we may see some irregular outcomes in the analysis related to these racial groups.

```
Sample size for Blacks: 3287596
Sample size for Whites: 2393949
Sample size for Hispanic: 2039297
Sample size for American Indian: 21133
Sample size for Asian: 35660
```

4 Possible questions

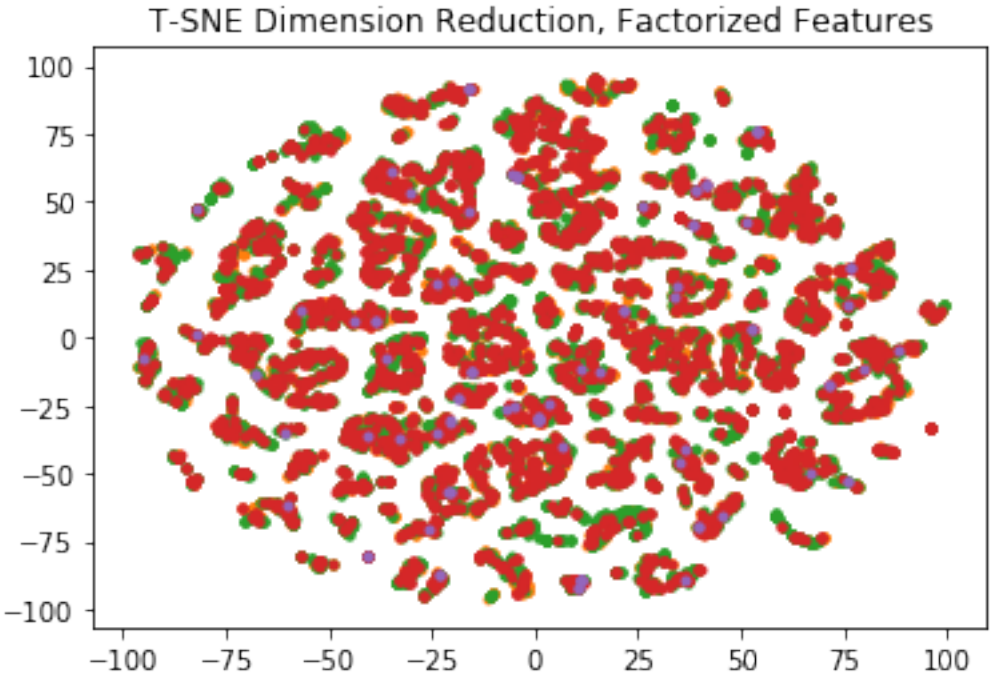
can we use ml techniques to correctly classify this data? which ones fail and why? can we create a predictive model for sentence lengths? should sentencing be offloaded to a ml algorithm? what does it mean to have an effective classfifier for this data set.

5 Methods

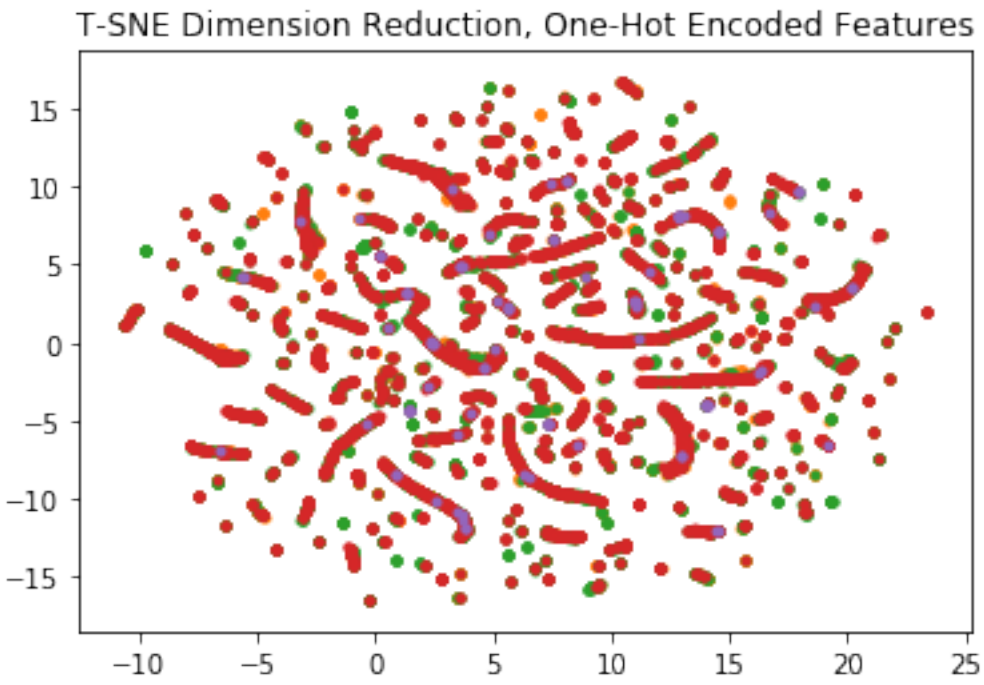
Before I begin discussing the methods that I did use, I will talk about some of the methods that I did not use. There are manay techniques that are not applicable to this data. For example, since this data set is not a time series models like ARMA and HMM are not applicable here.

A method that I attempted to use, but found little success with were dimension reduction tools like PCA, T-SNE, and UMAP. This dimension reduction would have been helpful, especially because once one-hot encoded this data becomes extremely high dimensional. However PCA, T-SNE, and UMAP all failed to create meaningful clusters, so I abandoned the the pursuit of clustering early on. Perhaps some kernel methods would have been helpful in this endeavor, however I could not find a kernel that could create a metric on crimes and I dp not feel qualified to write one myself.

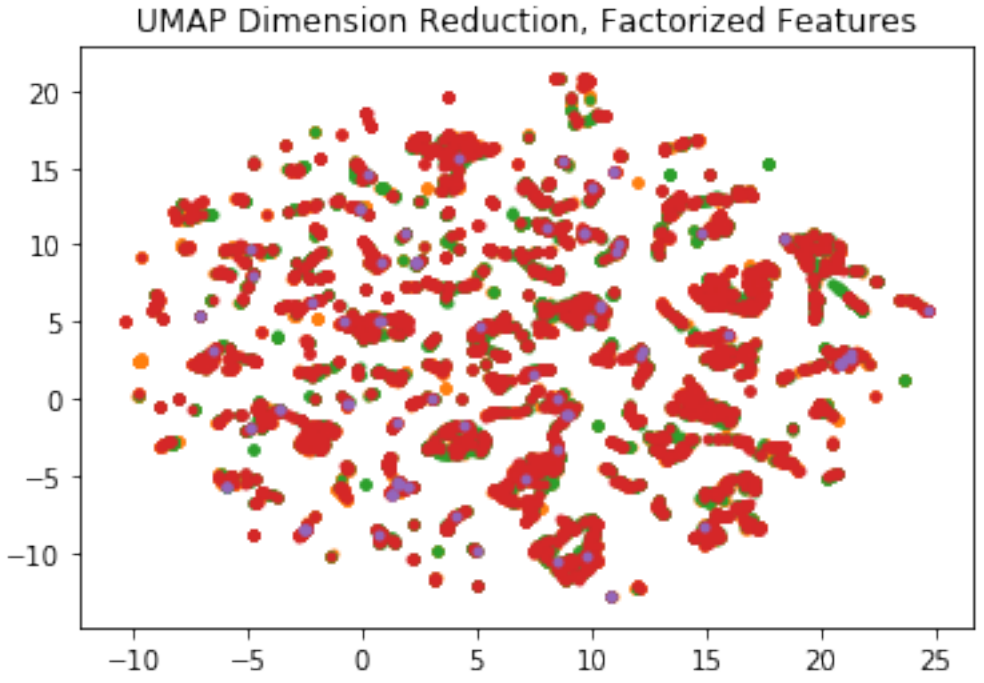
Below I have images of my attempt to use PCA, T-SNE, and UMAP to cluster the data. It is apperent that it simply is not effective.



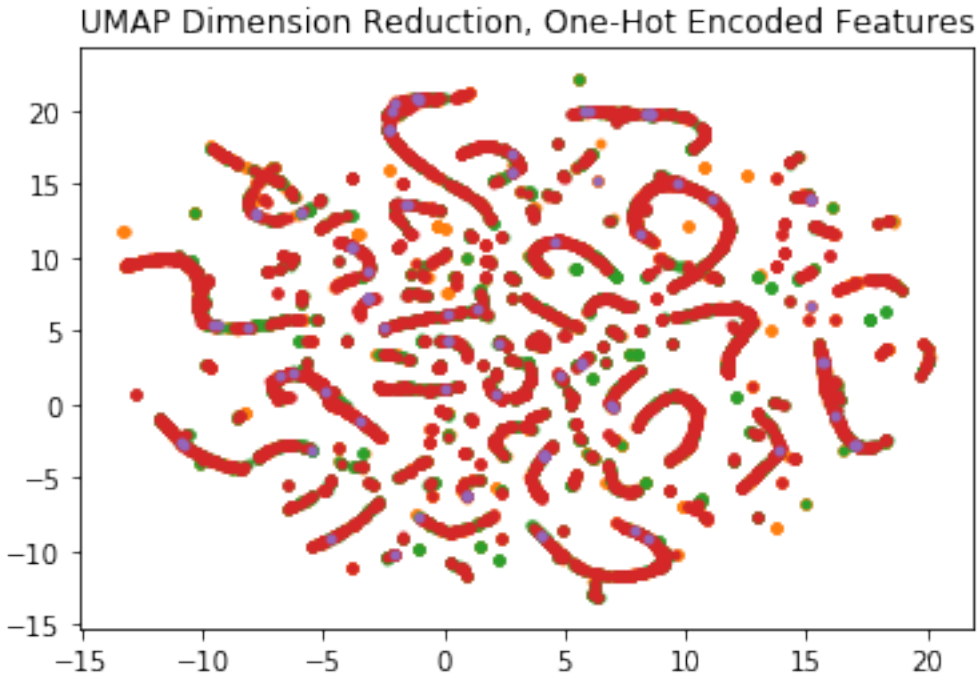

```
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/manifold/spectral_embedding_.py:237: U
  warnings.warn("Graph is not fully connected, spectral embedding"
```

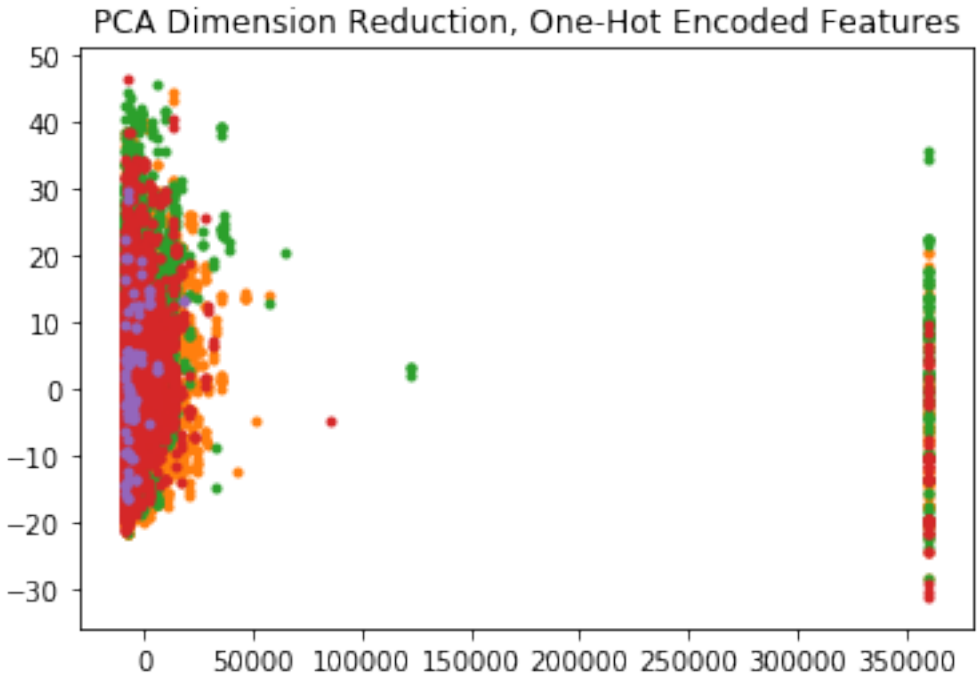



```
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/manifold/spectral_embedding_.py:237: U
  warnings.warn("Graph is not fully connected, spectral embedding"
```



```
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/manifold/spectral_embedding_.py:237: U
  warnings.warn("Graph is not fully connected, spectral embedding"
```





This is only a small sample of things that I tried to do to get useful clusterings given the dimension reduction, though it is representitive of the results I found.

5.1 Ensamble Methods

Ensamble methods seemed imediately like the methods that I should be using in this project. I did not intially expect to successfully classify this data, however as I learned about how ensamble classifiers worked I became more confident that I would get decent results.

The primary reason that I though I would not get good results has to do with how I identified racial bias in the data. In my previous project I determined that there was racial bias in the criminal justice system based on the kurtosis of the distribution of sentence lenghts, when you block inmates by race. Interpreted this means that minorities are much more likely to receive an extreme sentence length than a white person is. This condition is very subtle and I did not think that it would be easily detectable by machine learning methods. However, ensamble methods only need each member to do slightly better than random, so there was hope that I could get good results with these methods.

The methods that I attempted were the following: random forrest classifier, gradient descent boosted classifier, XGBoost, and LightGBM.

```
Max Depth = 85
Number of Estimators = 2100.


Out[6]: 0.296365

[0.02090027 0.24716962 0.25775903 0.10433119 0.03748648 0.33235341]


/home/ethan/.local/lib/python3.7/site-packages/sklearn/base.py:306: UserWarning: Trying to unpic
  UserWarning)
/home/ethan/.local/lib/python3.7/site-packages/sklearn/base.py:306: UserWarning: Trying to unpic
  UserWarning)
```

Here we see the same feature importances as we did with the random forest

```
Out[39]: {'criterion': 'friedman_mse',
          'init': None,
          'learning_rate': 0.34,
          'loss': 'deviance',
          'max_depth': 4,
          'max_features': None,
          'max_leaf_nodes': None,
          'min_impurity_decrease': 0.0,
          'min_impurity_split': None,
          'min_samples_leaf': 1,
          'min_samples_split': 2,
```

```
'min_weight_fraction_leaf': 0.0,
'n_estimators': 600,
'n_iter_no_change': None,
'presort': 'auto',
'random_state': None,
'subsample': 1.0,
'tol': 0.0001,
'validation_fraction': 0.1,
'verbose': 0,
'warm_start': False}
```

5.1.1 XGBoost

```
0.355695
[0.2596316  0.132617   0.1624777  0.12035192 0.18768501 0.13723671]
{'objective': 'multi:softprob', 'base_score': 0.5, 'booster': None, 'colsample_bylevel': 1, 'col
```

This score is very promiseing since the score is about double chance. There actually is a lot of correct classification going on here.

Interestingly however, it seems that the feature importances are very different for this model. Gender is the most important feature nad every other feature is about equally important.

5.1.2 LightGBM

LightGBM is an attempt to improve upon the efficiency, both temporally and spatially, of different grandient boosted decision tree (GBDT) algorithms such as XGBoost. This algoritm was developed by a team at Microsoft and it uses two novel techniques to imporove GBDT algorithms. The baseline comparison used was against XGBoost, since the team found this method to be the best performer of the commonly used GBDT algorithms.

By analyzing GBDT algorithms the team found that the most expensive parts of the process is learning the decision trees and the most expensive part of learning the decision trees is finding the best split points. They decided to use a histogram based approach for efficiency. This process is dominated by the histogram building which has a complexity of $O(\#data \times \#feature)$. Now the goal is to reduce the feature number or the number of data points.

Gradient-based One-Side Sampling This is the first novel technique proposed by the Microsoft team. It is a sampling method that is meant to reduce the number of data instances while maintaing accuracy. The main idea here is that data points with small gradient are usually ignored, since the model is already will trained on those data instances. However, the changes that occur by ignoring the data will small gradient may reduce the accuracy of the model once it is learned. Therefore GOSS examines all of the high gradient data and a random sample of the small gradient data. This method can maximize the amount of relevent data used in the training of the model without handicapping the accuracy completely.

Exclusive Feature Bundling This is the second novel technique and its goal is to reduce the number of features. This technique relies on the fact that high dimensional data tend to be sparse,

and therefore there are likely large bundles of features that are mutually exclusive are nearly mutually exclusive. These features can be bundled into a single feature and their histograms can be combined. This reduces the complexity of building histograms from $O(\#data \times \#feature)$ to $O(\#data \times \#bundle)$ and if $\#bundle << \#feature$ then the total complexity is greatly reduced.

Note that a max depth <= 0 indicates that the depth is unbounded. Searching over this grid of size 4000 is something that I would have never even tried for XGBoost or any other GBDT method, though it still did take slightly more than 20 hours to fit.

After this I did a grid search on the following grid

`param_grid = { 'reg_alpha': np.linspace(.1,1,10), 'reg_lambda': np.linspace(.1,1,10) }`

and found ———

6 Results

6.1 Random Forest Classifier

The random forest classifier was an easy place to begin in my attempt to find a successful classifier. It is a simple method and would likely give me a good lower bound on the success that I would have.

In using the random forest classifier I did a gridsearch for the best parameters. The parameters I decided to seach over are the number of trees in the classifier and the maximum depth of the trees in the classifier. I chose this because having more trees in the classifier will improve the accuracy, however if the depth of each of the trees in unbounded then overfitting of the individual trees may become an issue.

In my gridsearch I found the best parameters to be the following

My results with the random forest classifier were rather disappointing, especially given the amount of time it took to train which was 5.32 hours on the following parameter grid

`param_grid = { 'n_estimators': np.arange(100,5100,500), 'max_depth': np.arange(5,100,10) }`

which amounts to a 100 parameter grid.

The out-of-box score was rather promising at .70 however the method scored barely better than chance. Here is a scoring of the model I ran:

```
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to un
  UserWarning)
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to un
  UserWarning)


score on a test size of 500000 is 0.286044
```

The scoring here is lackluster to say the least though it did give me hope for more complex methods to preform better.

6.2 Gradient Descent Boosted Classification

This method is an obvious next step after a random forest. With the ability to alter the subsample rate and the learning rate, I expect to get better results with this model. I began by doing another

grid search, but I used some of the results from the previous search on the random forest model to save time. Fitting the following grid

```
   param_grid = {     'learning_rate': np.linspace(.01,1,10),     'subsample':
np.linspace(.05,1,10) }
```

I found the following parameters

```
In [ ]: with open('./pickles/GradientBoostedClf1.pickle',"rb") as f:
            clf = pickle.load(f)
            print('Max Depth = {}'.format(clf.get_params()['max_depth']))
            print('Number of Estimators = {}.'.format(clf.get_params()['n_estimators']))

/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to un
  UserWarning)
/home/ethan/anaconda3/lib/python3.7/site-packages/sklearn/base.py:318: UserWarning: Trying to un
  UserWarning)



        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-28-69247c951368> in <module>
         11 with open('./pickles/GradientBoostedClf.pickle',"rb") as f:
         12     clf = pickle.load(f)
    ---> 13     score = clf.score(samp_X,samp_y)
         14     print(f'score on a test size of {size} is {score}.')


        ~/anaconda3/lib/python3.7/site-packages/sklearn/base.py in score(self, X, y, sample_weig
        367         """
        368         from .metrics import accuracy_score
    --> 369         return accuracy_score(y, self.predict(X), sample_weight=sample_weight)
        370
        371


        ~/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/_gb.py in predict(self, X)
       2163             The predicted values.
       2164         """
    -> 2165         raw_predictions = self.decision_function(X)
       2166         encoded_labels = \
       2167             self.loss_._raw_prediction_to_decision(raw_predictions)


        ~/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/_gb.py in decision_function(sel
       2119         """
       2120         X = check_array(X, dtype=DTYPE, order="C", accept_sparse='csr')
```

```
-> 2121         raw_predictions = self._raw_predict(X)
   2122         if raw_predictions.shape[1] == 1:
   2123             return raw_predictions.ravel()


~/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/_gb.py in _raw_predict(self, X)
   1653     def _raw_predict(self, X):
   1654         """Return the sum of the trees raw predictions (+ init estimator)."""
-> 1655         raw_predictions = self._raw_predict_init(X)
   1656         predict_stages(self.estimators_, X, self.learning_rate,
   1657                        raw_predictions)


~/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/_gb.py in _raw_predict_init(sel
   1647                                           dtype=np.float64)
   1648         else:
-> 1649             raw_predictions = self.loss_.get_init_raw_predictions(
   1650                 X, self.init_).astype(np.float64)
   1651         return raw_predictions


AttributeError: 'MultinomialDeviance' object has no attribute 'get_init_raw_predictions'
```

```
In [ ]:
```

6.3 XGBoost

XGBoost was the best ensemble classifier of which I was aware when I began this project, so it was the obvious conclusion to my exploration of ensemble classifiers.

The grid I used was the following

```
param_grid = {     'reg_alpha':np.linspace(.01,1,10),     'reg_lambda':np.linspace(.01,1,10),
'gamma':np.linspace(.01,1,10) }
```

and I found the following parameters

```
In [ ]: with open('./pickles/XGBoostClf.pickle', "rb") as f:
            clf = pickle.load(f)
            print(clf.feature_importances_)
            print(clf.get_params())
```

6.4 LightGBM

Because I was very new to LightGBM when I began, I did a grid search on the following grid

```
param_grid = {     'boosting_type': ['gbdt','dart','goss'],     'learning_rate':
np.linspace(.01,1,10),     'n_estimators': np.arange(100,1100,100),
'max_depth': np.arange(0,10) }
```

and I found the following best parameters

6.5 Feature Importances

The only thing left to report about the results of my research is to examine the different assigned feature importances.

7 Analysis

8 Conclusion

9 References

[1] Statement Of Concern About Predictive Policing By Aclu and 16 Civil Rights Privacy, Racial Justice, and Technology Organizations https://www.aclu.org/other/statement-concern-about-predictive-policing-aclu-and-16-civil-rights-privacy-racial-justice

[2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems 30 (NIPS 2017). Retrieved from http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf