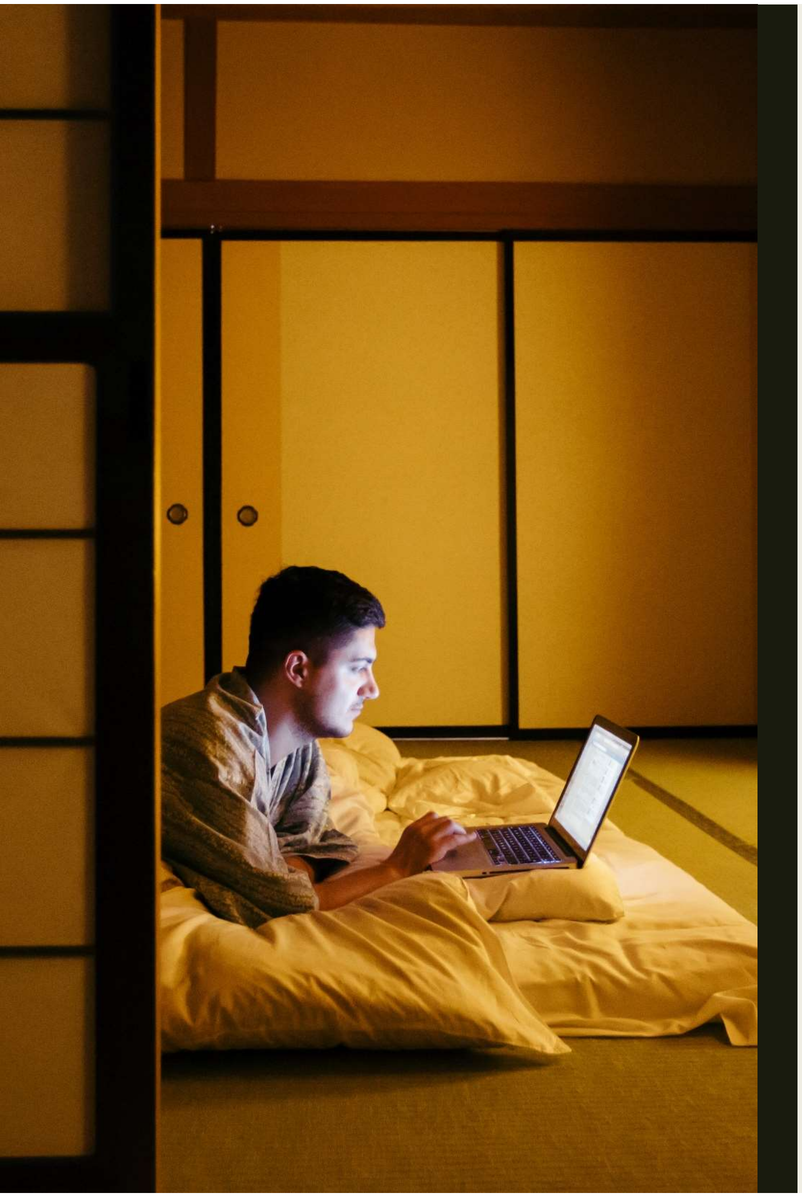
PHASE 3 PROJECT

Ryan S. Chung



Outline

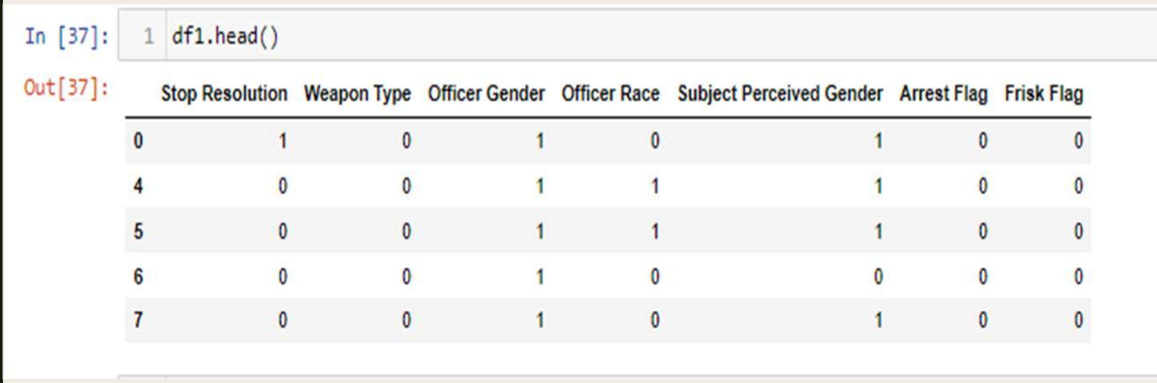
- Business Problem
- Data and Methods
- Conclusions

Business Problem

For this project, a data set containing information on stops made based on suspicious behavior was provided and our goal was to create models that could accurately predict whether an arrest was made or not given the rest of the information provided on each stop made.

Data Used

The data set used was obtained from data.gov called Terry Stops. It contains records of police reported under the supreme court case Terry v. Ohio 1967. Each data point contains various aspects of each stop made such as time reported, officer gender, stop resolution, perceived subject race, etc.



```
In [37]:  df1.head()
```

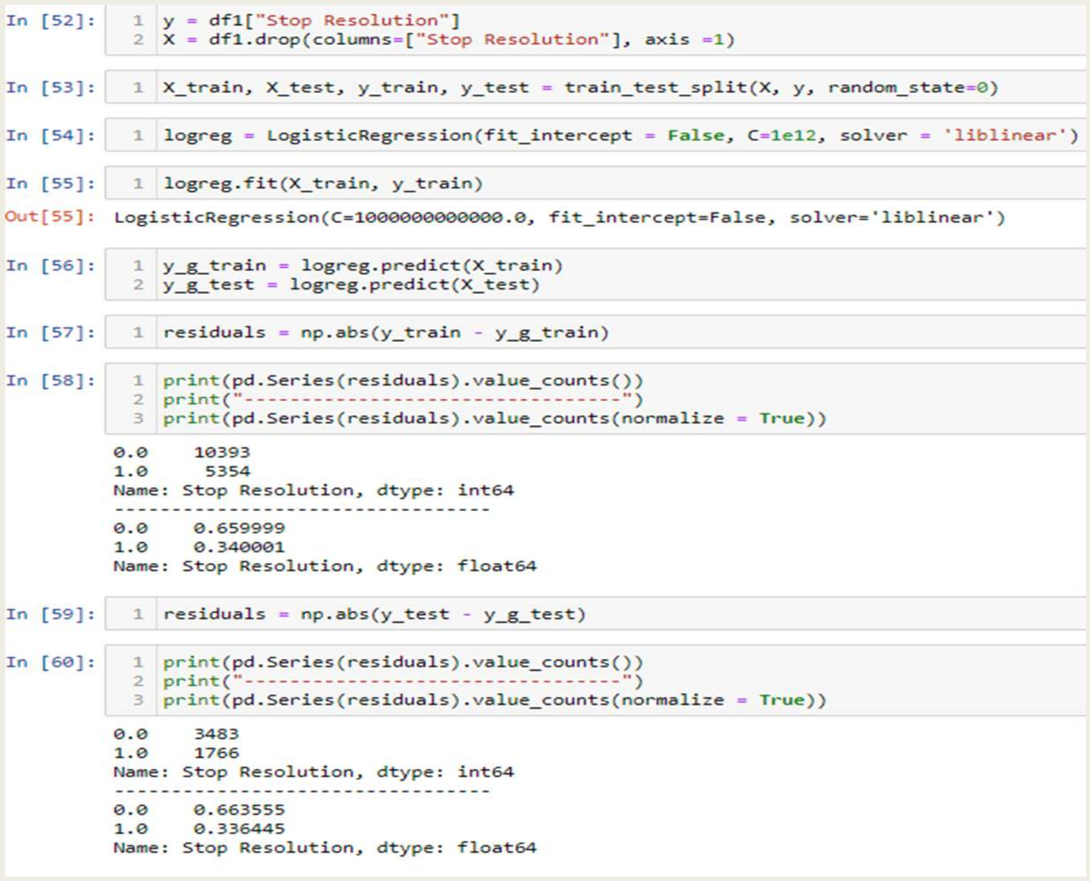
Out[37]:

	Stop Resolution	Weapon Type	Officer Gender	Officer Race	Subject Perceived Gender	Arrest Flag	Frisk Flag
0	1	0	1	0	1	0	0
4	0	0	1	1	1	0	0
5	0	0	1	1	1	0	0
6	0	0	1	0	0	0	0
7	0	0	1	0	1	0	0

The cleaned data set used for the creation of each model is comprised of each feature being assigned a value of 0 or 1. (Reference code for context of what 0 or 1 indicates).

First Model

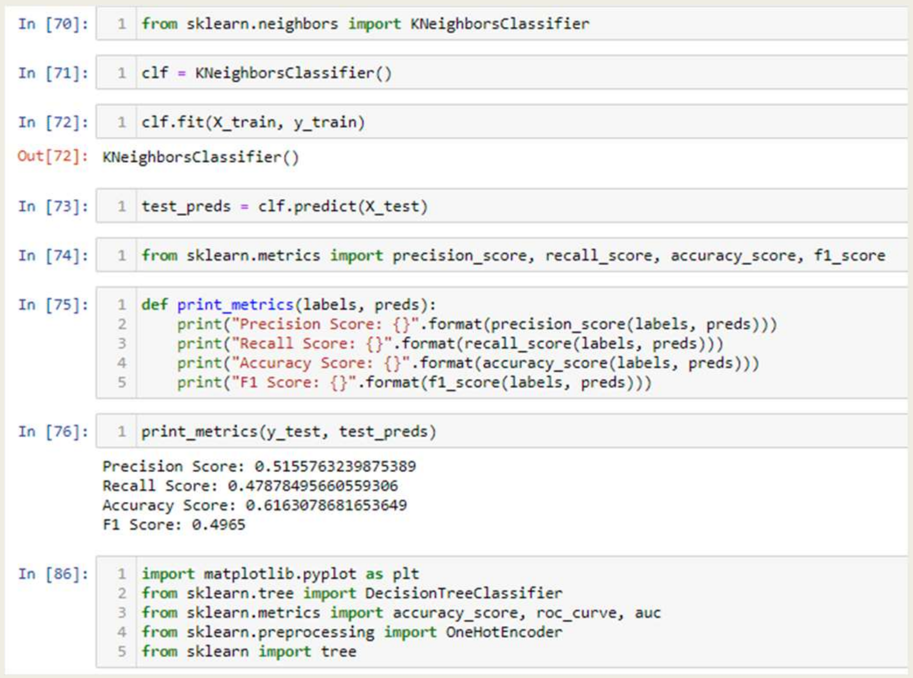
A Logistic Regression Model was then created and the model had an accuracy of 65.9% on the training set with an accuracy of 66.4% on the test set.



```
In [52]: y = df1["Stop Resolution"]
         X = df1.drop(columns=["Stop Resolution"], axis =1)

In [53]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

In [54]: logreg = LogisticRegression(fit_intercept = False, C=1e12, solver = 'liblinear')

In [55]: logreg.fit(X_train, y_train)

Out[55]: LogisticRegression(C=1000000000000.0, fit_intercept=False, solver='liblinear')

In [56]: y_g_train = logreg.predict(X_train)
         y_g_test = logreg.predict(X_test)

In [57]: residuals = np.abs(y_train - y_g_train)

In [58]: print(pd.Series(residuals).value_counts())
         print("---------------------------------")
         print(pd.Series(residuals).value_counts(normalize = True))

         0.0    10393
         1.0     5354
         Name: Stop Resolution, dtype: int64
         ---------------------------------
         0.0    0.659999
         1.0    0.340001
         Name: Stop Resolution, dtype: float64

In [59]: residuals = np.abs(y_test - y_g_test)

In [60]: print(pd.Series(residuals).value_counts())
         print("---------------------------------")
         print(pd.Series(residuals).value_counts(normalize = True))

         0.0    3483
         1.0    1766
         Name: Stop Resolution, dtype: int64
         ---------------------------------
         0.0    0.663555
         1.0    0.336445
         Name: Stop Resolution, dtype: float64
```

Second Model

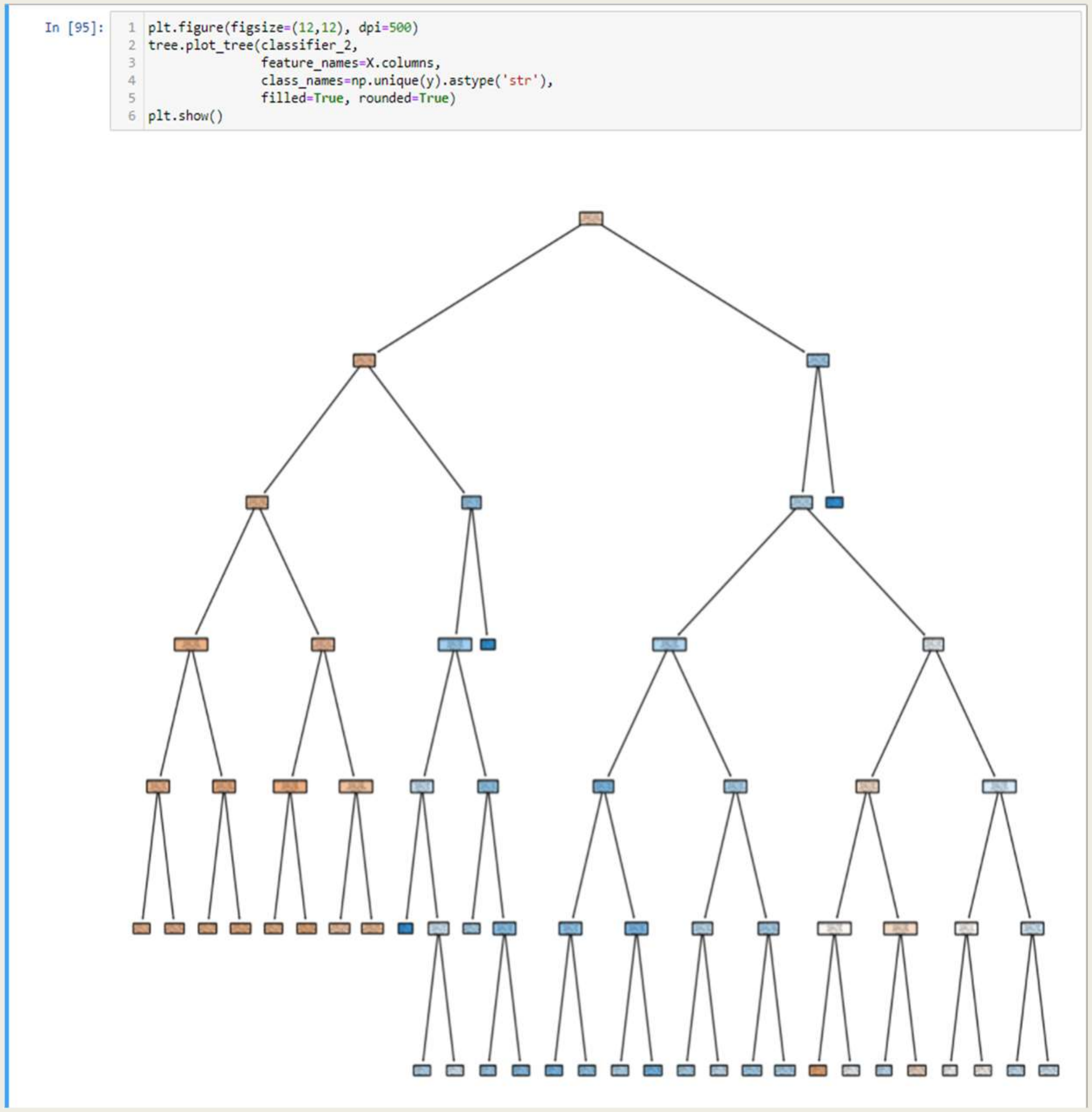
The second model created was a K Neighbors Classifier Model. This model was found to have an accuracy score of 61.6% with an F1 Score of 0.4965.



```
In [70]: from sklearn.neighbors import KNeighborsClassifier

In [71]: clf = KNeighborsClassifier()

In [72]: clf.fit(X_train, y_train)

Out[72]: KNeighborsClassifier()

In [73]: test_preds = clf.predict(X_test)

In [74]: from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

In [75]: def print_metrics(labels, preds):
             print("Precision Score: {}".format(precision_score(labels, preds)))
             print("Recall Score: {}".format(recall_score(labels, preds)))
             print("Accuracy Score: {}".format(accuracy_score(labels, preds)))
             print("F1 Score: {}".format(f1_score(labels, preds)))

In [76]: print_metrics(y_test, test_preds)

         Precision Score: 0.5155763239875389
         Recall Score: 0.47878495660559306
         Accuracy Score: 0.6163078681653649
         F1 Score: 0.4965

In [86]: import matplotlib.pyplot as plt
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import accuracy_score, roc_curve, auc
         from sklearn.preprocessing import OneHotEncoder
         from sklearn import tree
```

Third Model

The third model created was a Decision Tree Classifier Model which was found to have an accuracy score of 66.1%. The AUC was 0.61.

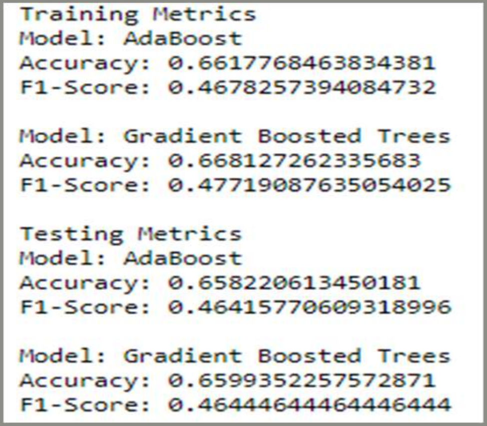


```
In [95]:
plt.figure(figsize=(12,12), dpi=500)
tree.plot_tree(classifier_2,
               feature_names=X.columns,
               class_names=np.unique(y).astype('str'),
               filled=True, rounded=True)
plt.show()
```

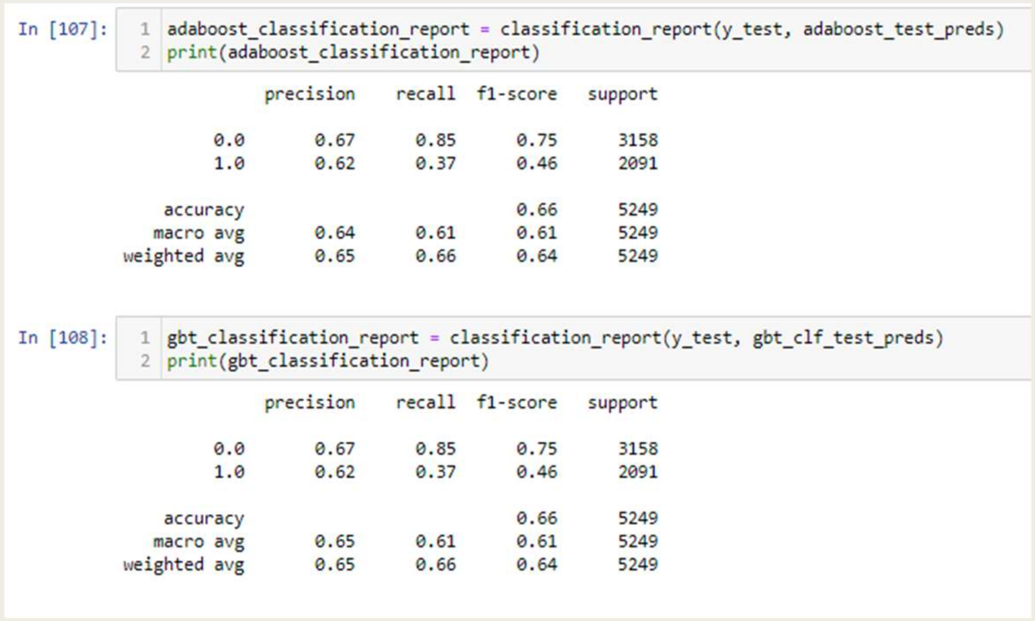
Final Models

The final models created were an Adaboost and Gradient Boosting Classifier.

The accuracy and F1 scores of both models on the training and testing metrics can be seen below.



```
Training Metrics
Model: AdaBoost
Accuracy: 0.6617768463834381
F1-Score: 0.4678257394084732

Model: Gradient Boosted Trees
Accuracy: 0.668127262335683
F1-Score: 0.47719087635054025

Testing Metrics
Model: AdaBoost
Accuracy: 0.658220613450181
F1-Score: 0.46415770609318996

Model: Gradient Boosted Trees
Accuracy: 0.6599352257572871
F1-Score: 0.4644644464446444
```

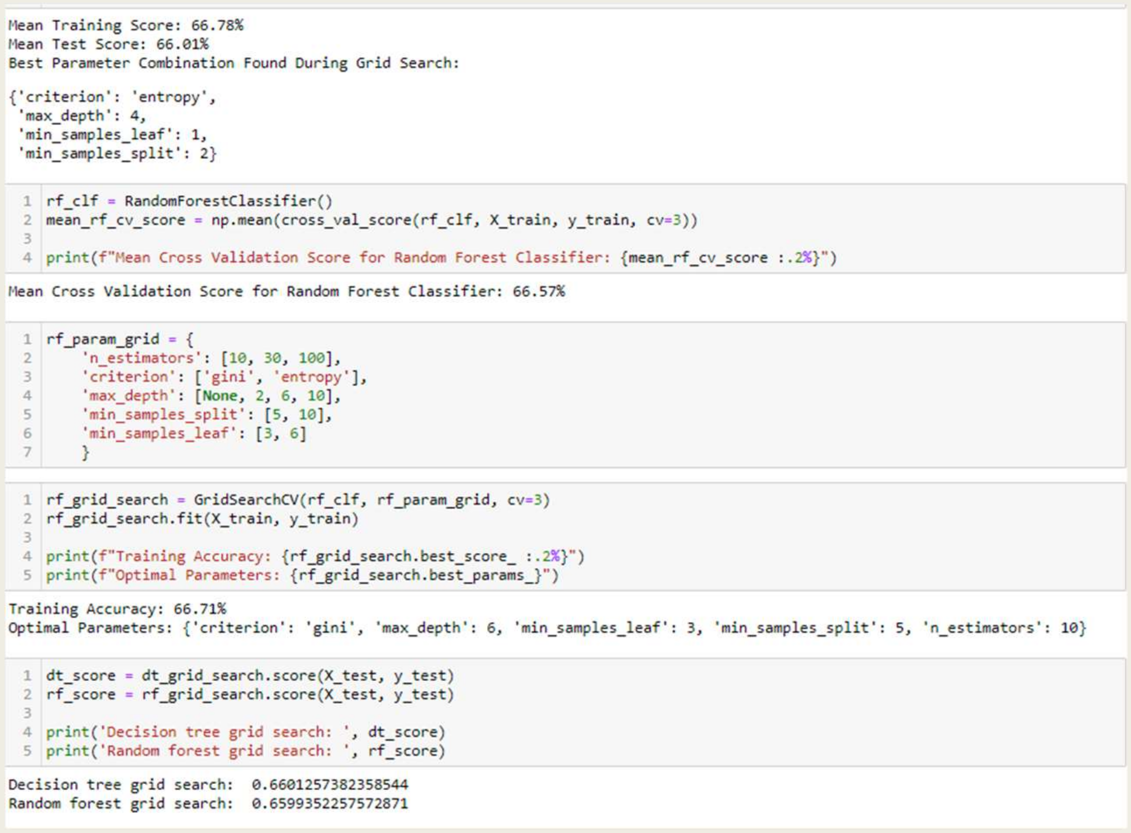


```
In [107]: adaboost_classification_report = classification_report(y_test, adaboost_test_preds)
          print(adaboost_classification_report)

              precision    recall  f1-score   support

         0.0       0.67      0.85      0.75      3158
         1.0       0.62      0.37      0.46      2091

    accuracy                           0.66      5249
   macro avg       0.64      0.61      0.61      5249
weighted avg       0.65      0.66      0.64      5249
```

```
In [108]: gbt_classification_report = classification_report(y_test, gbt_clf_test_preds)
          print(gbt_classification_report)

              precision    recall  f1-score   support

         0.0       0.67      0.85      0.75      3158
         1.0       0.62      0.37      0.46      2091

    accuracy                           0.66      5249
   macro avg       0.65      0.61      0.61      5249
weighted avg       0.65      0.66      0.64      5249
```

Grid Search

After reviewing all models and their accuracy ratings. The decision tree classifier model was chosen to be run through grid search. The mean cross validation score was 66.55% and the grid search was made to run through 648 different permutations. It was found that the random forest grid search slightly underperformed compared to the baseline model.



```
Mean Training Score: 66.78%
Mean Test Score: 66.01%
Best Parameter Combination Found During Grid Search:

{'criterion': 'entropy',
 'max_depth': 4,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

```
rf_clf = RandomForestClassifier()
mean_rf_cv_score = np.mean(cross_val_score(rf_clf, X_train, y_train, cv=3))

print(f"Mean Cross Validation Score for Random Forest Classifier: {mean_rf_cv_score :.2%}")
```

```
Mean Cross Validation Score for Random Forest Classifier: 66.57%
```

```
rf_param_grid = {
    'n_estimators': [10, 30, 100],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 2, 6, 10],
    'min_samples_split': [5, 10],
    'min_samples_leaf': [3, 6]
    }
```

```
rf_grid_search = GridSearchCV(rf_clf, rf_param_grid, cv=3)
rf_grid_search.fit(X_train, y_train)

print(f"Training Accuracy: {rf_grid_search.best_score_ :.2%}")
print(f"Optimal Parameters: {rf_grid_search.best_params_}")
```

```
Training Accuracy: 66.71%
Optimal Parameters: {'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 10}
```

```
dt_score = dt_grid_search.score(X_test, y_test)
rf_score = rf_grid_search.score(X_test, y_test)

print('Decision tree grid search: ', dt_score)
print('Random forest grid search: ', rf_score)
```

```
Decision tree grid search:  0.6601257382358544
Random forest grid search:  0.6599352257572871
```

CONCLUSIONS

Looking back at our initial models we can see most of them were able to have an accuracy of around 60% on predicting whether an arrest or not was made based off of the data from the cleaned set. It is important to keep in mind that the aspects of each arrest used in these models included weapon type, officer gender, officer race, subject perceived gender, arrest flag, and frisk flag. Keeping in mind how each feature was categorized to be used in each model would be necessary in understanding what the models actually look at. When running the decision tree classifier through grid search it seems it could be further tuned for better accuracy.

THANK YOU