Sean Ammirati

The Results of Stop and Frisk: What Contributes Most to Arrests?

I) Introduction

In this project I will be looking at a dataset available from the city of New York about the results of the Stop, Question and Frisk policy. What I am interested in is the following question: which factors lead to the highest likelihood of arrest?

The data is a collection of 22,564 cases of Stop, Question and Frisk in the City of New York in 2015. There are 112 variables which were measured. A majority of the variables in the dataset are categorical variables relating to the incident itself, while a few represent the age, race, height, weight, and sex of the person who was on the receiving end of the policy.

Stop and Frisk has been a controversial policy in New York City for quite some time. Many feel that it is a violation of privacy, that it is unconstitutional, or that the policy is inherently racist – that it disproportionally affects minorities and people of color. Meanwhile, there are others who support the policy, believing stop and frisk to keep people safe from terrorists and other wrongdoers.

As a result of this and other recent controversies involving police across the country, the NYPD has been getting a lot of negative press over the way it addresses these racial problems. What I aim to discover by analyzing this dataset are the main contributing factors to whether a person is arrested or not in a stop and frisk incident.

I will be using a few different methods of analysis to determine the best model to use in predicting whether a person will be arrested or not. Because the response variable, arstmade, is a qualitative, binary variable, I will be using categorical methods. These include logistic regression, K-Nearest Neighbors analysis and tree methods (like bagging, random forests, boosting).

II) Data

The data comes from the City of New York. It is a collection of 22,564 observations of 112 variables. Some of these variables had to be removed, as they did not contribute any information (for instance, they were the same value for all observations) or they had too many missing values. After removing these variables from the dataset, I had 75 variables to work with, with 22,563 observations.

For example, lineCM was a redundant variable, as it had 1's for all observations. I also removed apt number, state and zip, as many of these values were left blank. Overall, none of the variables I was very interested in had any missing values, so removing these variables likely had very little effect on the analysis, if any.

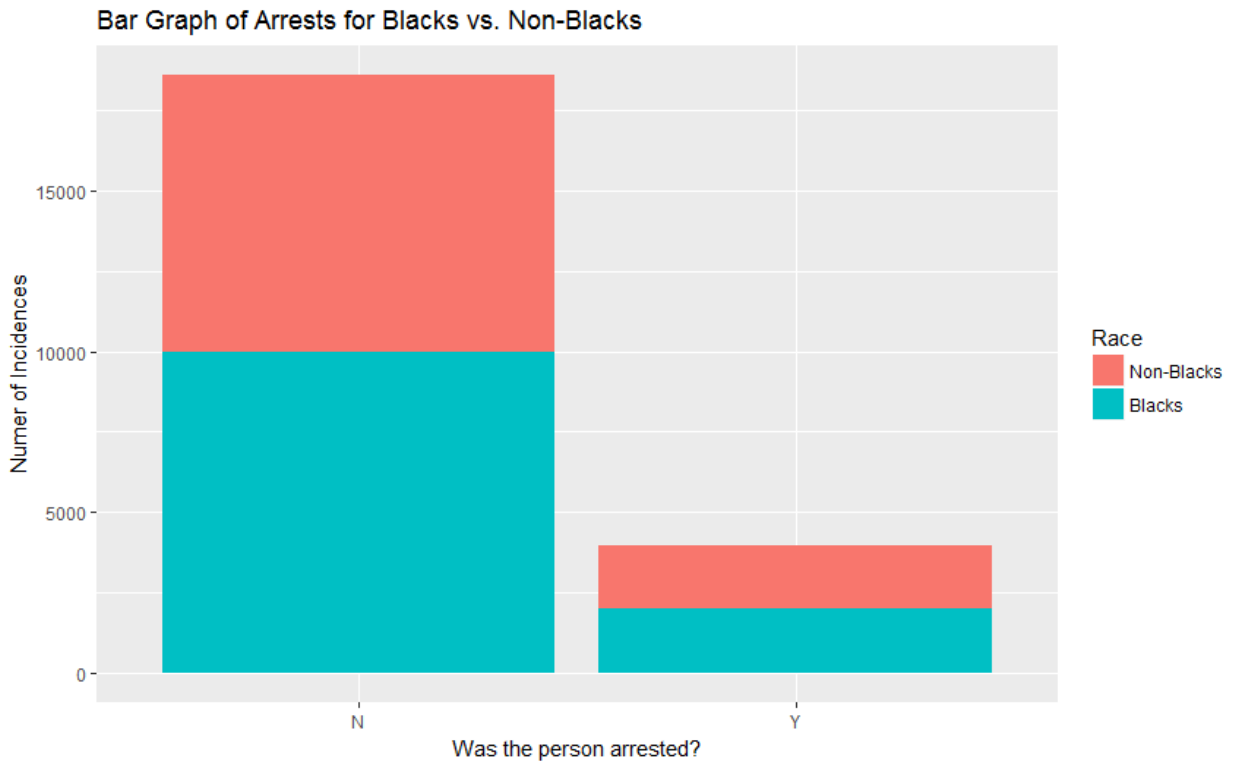
There were two variables for height (ht_inch and ht_feet) which I combined into one (height).

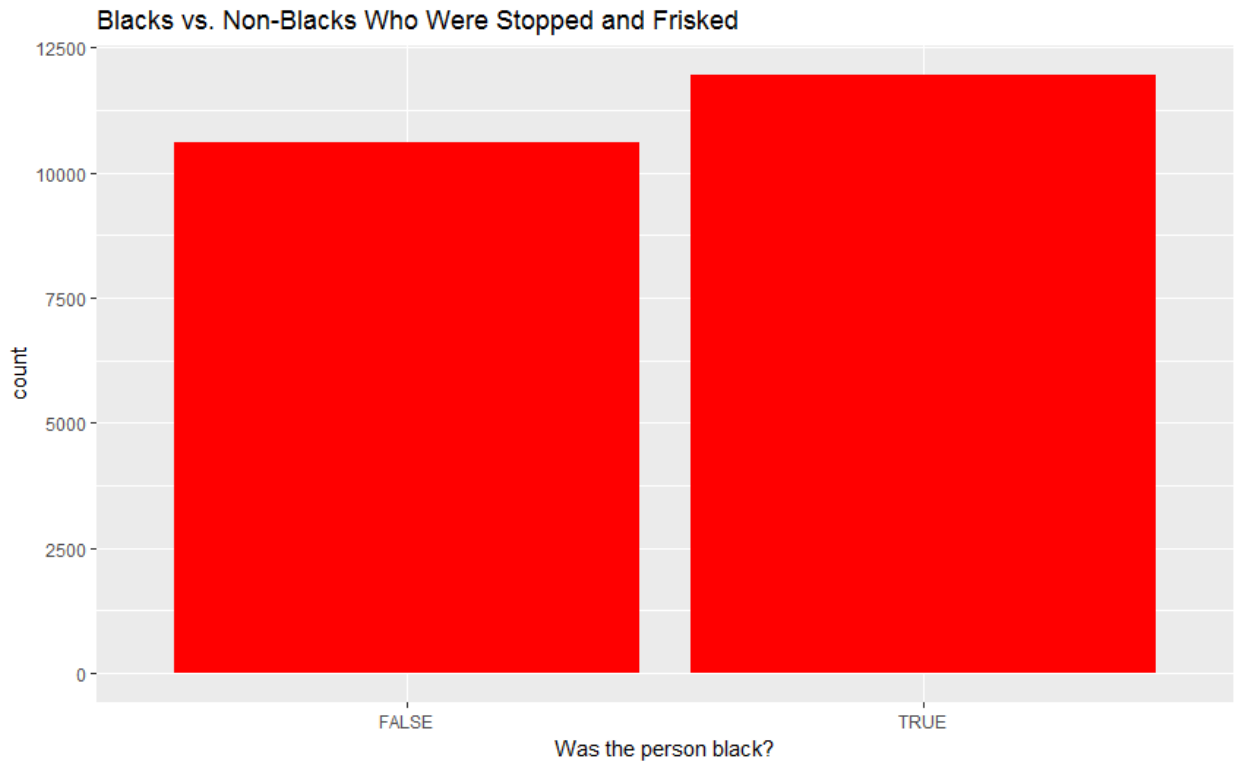
The response variable I used for all of the following analyses was arstmade, which was a categorical variable that had values of "Y" for yes and "N" for no. Many of the categorical variables in the dataset followed this pattern. The percentage of arrests in all cases was 0.18, meaning that around 18% of the stops resulted in an arrest.

The mean age of people who were stopped and frisked was around 29 years. Most of the people who were stopped were males (92.42%). Force was used in about 27% of the cases.

In investigating whether blacks were targetted more frequently and if they were arrested more frequently, I created a categorical variable that was 1 if the person was black and 0 otherwise. Below is a bar graph of the arrests, with the colorings indicating the percentage who were black and non-black.

It does not seem from these graphs alone that blacks were more likely to be arrested, since they have roughly half of the bars in both instances. However, it does seem like there are a disproportionate number of blacks who are stopped and frisked in the first place, which is shown in the second graph. With blacks making up roughly 23% of the population, it is surprising that almost double that (53%) of the people who were stopped were black.





Around 6.7 times more people were arrested than given summonses. That is, only 2% of the people were given summonses.

None of the people who were stopped had a machine gun, so I removed this variable from the data. I removed any variable which had levels that were exactly one, as this indicated that there was no variation between observations.

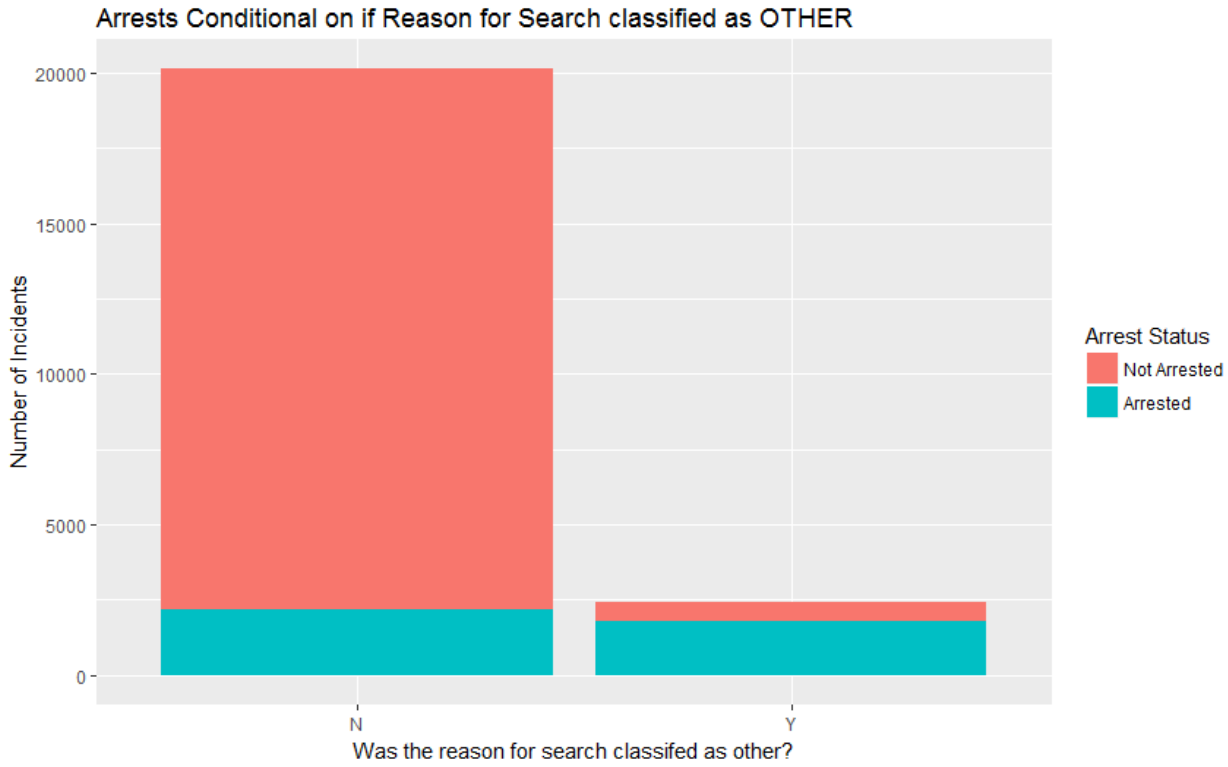
Although the suspected crime variable (crimsusp) was of interest to me in my analysis, I found the data to be inconsistent in how it was recorded. For instance, if it was a felony, the officer had data entered that had both "FEL" and "FELONY" in two different cases. Of course, R would not be able to recognize the fact that these are the same, and would treat them as separate variables.

As such, I decided to remove any variables which had levels larger than 10, as this would indicate that the data was too separated to be meaningful. Since much of this information is contained in the categorical variables (such as the rs variables, which had different values for different reasons for stopping the person), it was relatively inconsequential in my analysis to do so.

There were also groupings which could raise doubt as to whether the information recorded was accurate. Of primary concern to me was pf_other (physical force used: other), rs_other (reason for search: other) and ac_other (additional circumstances: other). These are not very informative, and could be raise a speculative point about whether these were truly extraordinary cases or if they were omitted for other reasons. Of course, that is merely speculation, but it is important to note that this is a source of potential inconsistency and unreliability from the dataset.

Even worse, some of these variables contributed a lot to the models which I have fit. Below shows a bar graph of the number of arrests conditional on whether the reason for search was

classified as "other."



This confirms the results of the analysis below, as this variable has a large effect on whether the person was arrested or not. If the person was classified as being searched for other reasons not specified, they were arrested quite often. This leads to a limitation of the data and interpretation – we can not determine what the true reasons were. sb_other accounted for around 10.7% of all the cases, and 57.6% of all the people who were searched.

III) Analysis

Logistic Regression

I separated the data into two sets – test data and training data. The training data was randomly sampled from ¾ of the data.

I began my analysis by using a logistic regression model to predict arstmade on all of the other variables. This was because the response variable was categorical, so it was necessary to use a logistic model instead of a linear model with normal errors. I then used stepwise reduction to determine the best fit and to reduce the amount of variables used. Since there were a great deal of variables that came up insignificant in the full model, it reduced the model quite a lot.

I used the glm function using the binomial family to model the data.

Below is a summary of the results of the models:

```
Reduced Model Confusion Matrix
##       arrest.test
## fitted    N    Y
##      0 4513  397
##      1  126  590
Misclassification Error (Reduced Logistic Model)
## [1] 0.093
Full Logistic Model Confusion Matrix
##        arrest.test
## fitted2    N    Y
##       0 4508  401
##       1  131  586
Misclassification Error (Full Logistic Model)
## [1] 0.095
```

Using the stepwise regression model determined by the AIC produced a marginally better misclassification error, from .095 to .093 (or 9.5 % to 9.3%). This isn't a very big improvement, though.

Surprisingly, the model does not find any of the race factors to be significant in determining whether an arrest was made or not besides Q, which stands for the White-Hispanics. That is, all of the levels of the "race" variable other than "White-Hispanic" came up insignificant. All of these coefficients are positive besides "U", which stood for "unknown", which essentially means that if the person's race was known, the odds of arrest are higher. The full summary of the stepwise regression is included in the appendix(1).

When the stepwise regression was performed, we could see that contraband and knife/cutting instrument variables were highly significant in determining whether a person would get arrested or not. These variables are both very significant in the reduced logistic model.

The variables that represent the different boroughs were all found to be significant, all with negative values. This suggests that the variable that was not included in the factors (the Bronx) had a higher likelihood of being arrested than any of the other four boroughs. Since the largest coefficient is that of Staten Island it means that, all else equal, a person from Staten Island who was stopped and frisked was less likely to be arrested than a person from Manhattan.

The model also indicates that people who had a weapon were more likely to be arrested during a stop and frisk. This is logical, as if the police had found a weapon, it is quite likely that the person would be arrested. If the officer was in uniform, interestingly, a person would be less likely to be arrested, according to the model. If the police officer stopped the person because they knew them (indicated by rf_knowlY), the person was less likely to be arrested.

If the reason the officer used force was to defend themselves, the person was less likely to be arrested than if the reason for force was that the person was suspected of running away. If the person is searched, they had a higher chance of being arrested than if they were not searched.

KNN

The next model I used was the K-Nearest Neighbors approach. K-Nearest Neighbors is a non-parametric method, so it does not make assumptions about the underlying distribution of the variables. K-Nearest Neighbors works by finding the nearest data points to the ones you are testing them against and assigning a classification based on the nearest k points. I have used k=5 and k=10 in this model to see if they can accurately predict the chances of arrest.

Below are the tables for predicted vs actual arrest values for the test set, as well as the misclassification error, for the 10-nearest neighbor method (as this was the most successful.)

```
10-Nearest Neighbor Confusion Matrix
##             arrest.test
## knn.predict    N    Y
##           N 4479  749
##           Y  160  238
Misclassification Error (10 NN)
## [1] 0.162
```

The 10-Nearest Neighbor model was the most successful of the different k's I used (I tried 1, 5, 10, and 20). Still, it doesn't produce very good outcomes compared to the logistic model. In fact, if I were to guess that none of the people would get arrested, I would have an error of about 18%, so this model is only marginally better than the trivial method.

The failure of this method is due to the fact that the variables cannot exactly be scaled, and the distances are affected by this. As there are many categorical variables and few continuous ones, high numbers of the continuous variables may have influenced the predictions, since the KNN function in R uses Euclidian distances to predict the outcomes.

To rectify this, I perform the K-NN method using only the categorical variables. The predictions are slightly better in this model, as shown by the confusion matrix and misclassification rate below.

```
5-nearest Neighbor Confusion Matrix using Classification Vars
##              arrest.test
## knn.predict2    N    Y
##             N 4547  582
##             Y   92  405
5-Nearest Neighbor Classification Error
## [1] 0.12
```

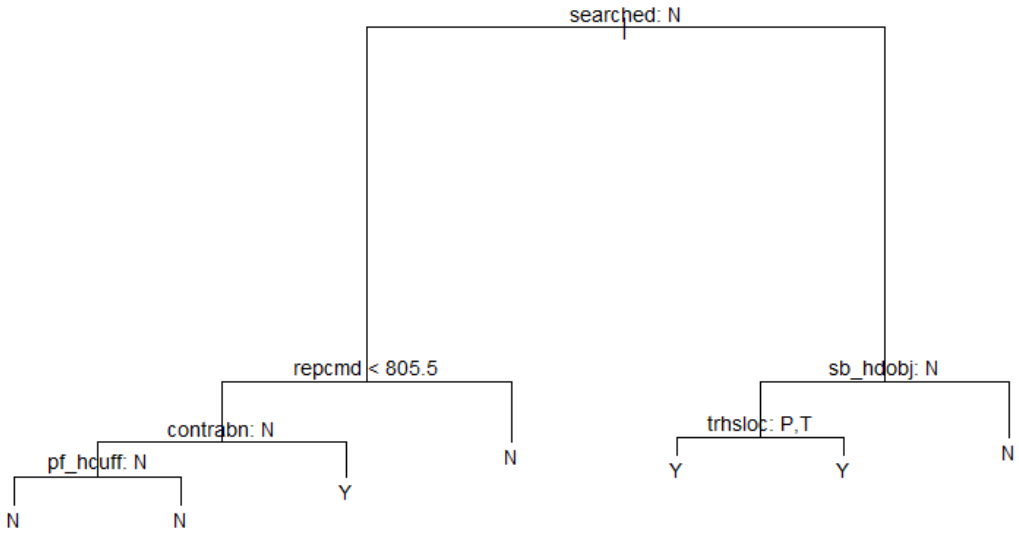
Interestingly, while the error is still larger than the logistic models, or in fact any of the models I used in the analysis, it is more accurate at predicting an arrest when an arrest was made than any of the earlier models, suggested by the second row of the confusion matrix above. That is, while it produces more overall error than any of the models I've used, it also produces the smallest type I error out of any of the models (the false positives).

Tree-Based Methods

I will now use tree-based methods (a single tree, Random Forest, Bagging and Boosting) in order to predict whether a person will be arrested or not.

The first model I use will be an unpruned tree using all the variables. The tree selects groupings of the variables which produces the least errors in predictions, and lists decision trees based on the conditional results of each tree. It determines these factors by a popular vote in the case of classification trees, so that the most common outcome is the one which is predicted by the

tree. The tree is posted below.



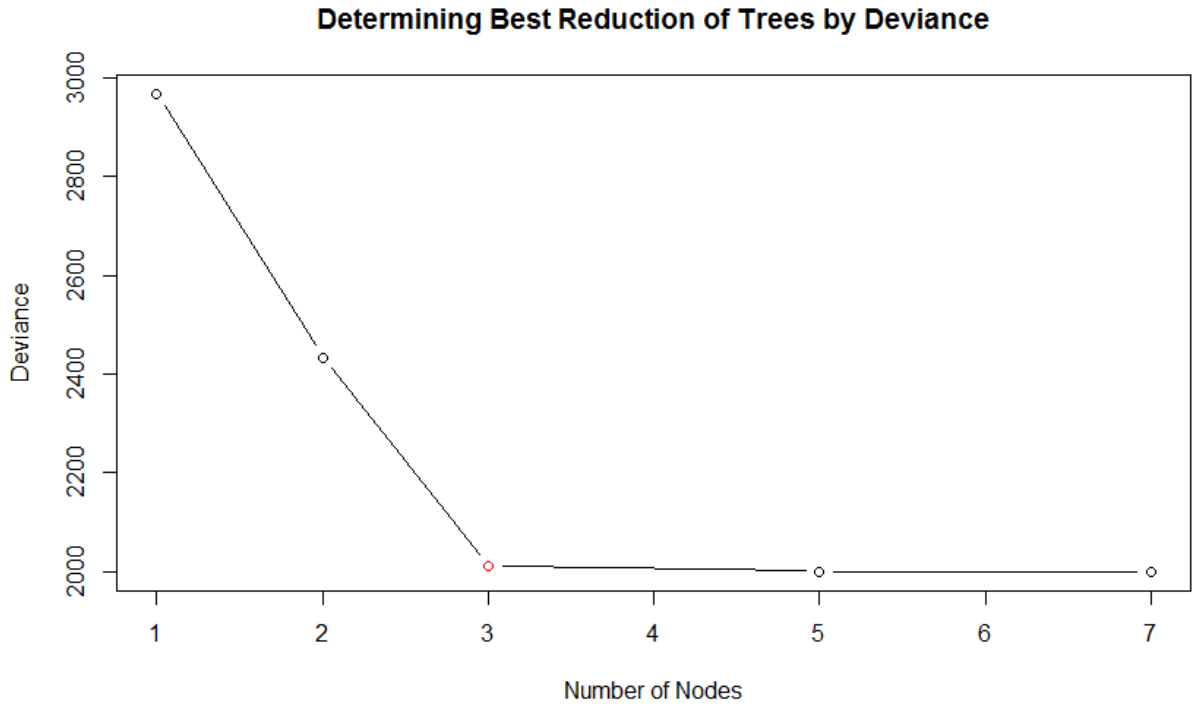
This tree has six nodes, so it is relatively simple. If the statement is true, you go to the right, and if it is false you go to the left. The top node is whether the person was searched or not. This makes sense, as if a person is searched, the police are more likely to find a reason to arrest the person (because of illegal goods, etc).

It is already evident that this tree can be pruned. For the trhsloc and pf_hcuff variables, either decision results in the same response. This means that the variable is redundant, and can be removed from the tree. The values that are included are: searched, repcmd (reporting officers command, which takes values from 1 to 999), contrabn (contraband), sb_hdobj (basis of search being a hard object), trhsloc P,T (whether the location was a transit location or housing location), and pf_hcuff(force used was handcuffing).

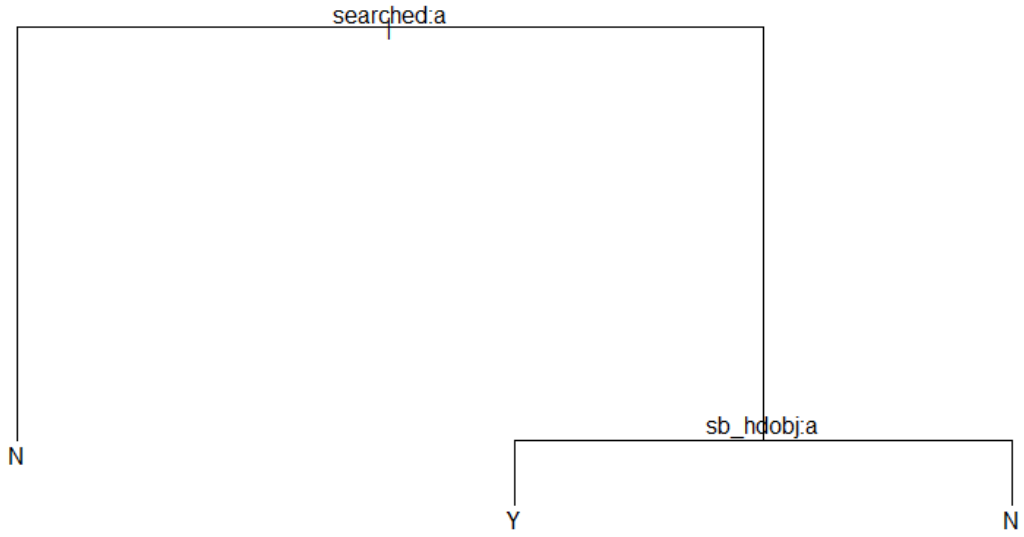
Some of these are quite interesting: the fact that the officers command has an influence on whether the person was arrested or not, and also the fact that regardless of whether the person was handcuffed or not, they were not reported as arrested if they reach the bottom left node. This defies our common sense – as we often expect someone who is handcuffed to be arrested. Based on the prior conditional factors, the tree says that if someone is handcuffed given they don't have contraband, they weren't searched, and the reporting officers command was less than 805, they would be classified as not being arrested.

Since we already have seen that we should prune this tree, I will find the correct number of nodes to

prune to by plotting the cross-validation errors.



There is a clear leveling off at 3, marked with a red point, so we will prune the next tree to three nodes.



This tree is extremely small, and only considers two variables – whether a person was searched and whether the reason for the search was that they had a hard object.

Below, I post the tables for the pruned and unpruned trees, along with their misclassification errors.

```
Pruned Tree Confusion Matrix
##            arrest.test
## pruned.pred    N    Y
##           N 4466  486
##           Y  173  501
Misclassification Error (Pruned Tree)
## [1] 0.117
Unpruned Tree Confusion Matrix
##          arrest.test
## tree.pred    N    Y
##         N 4426  447
##         Y  213  540
Misclassification Error (Unpruned Tree)
## [1] 0.117
```

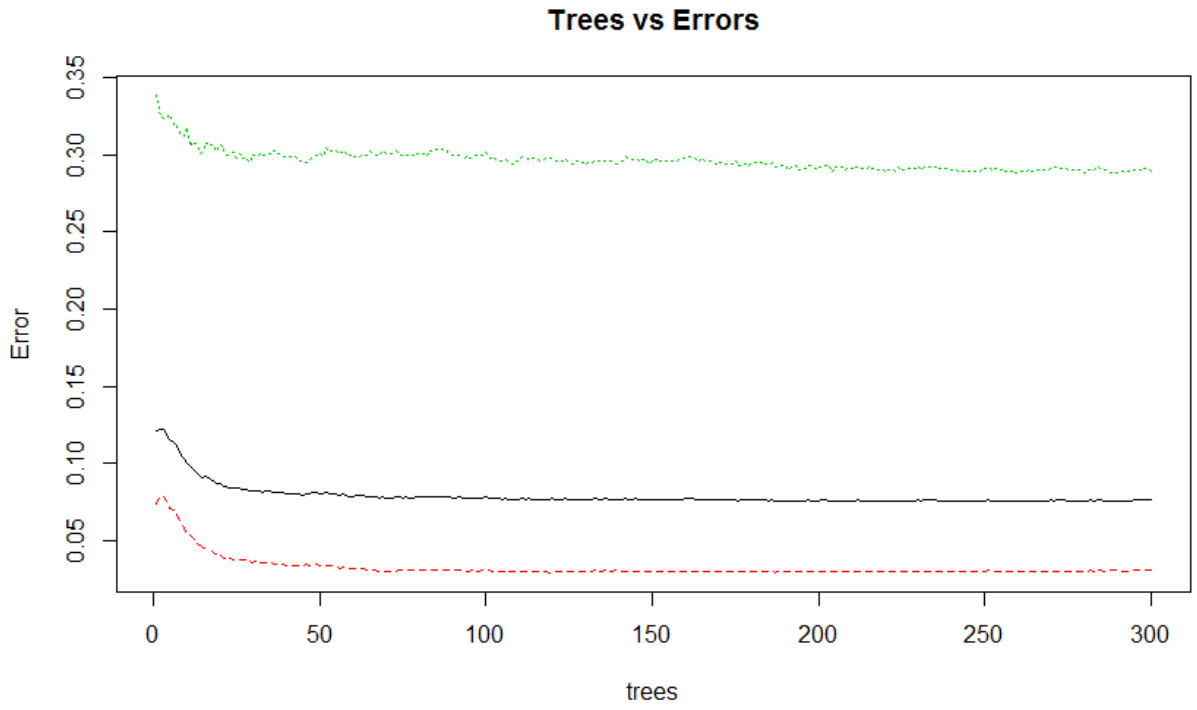
The misclassification errors are quite high for the single trees, which is expected since one tree will rarely be sufficient in predicting the outcome. They perform roughly the same on the data, which indicates that the pruning was effective – we were able to reduce the nodes without significantly effecting the accuracy of the models. It is notable that the pruned tree reduced Type I error, which is more desirable in this case. Still, both trees performed better on the overall error than the KNN approach.

Next, I will use the bagging method. The bagging method, or bootstrap aggregation, uses bootstrap samples repeatedly to create many trees, and then averages these trees to make predictions. In the case of a classification problem such as this one, bagging will predict by the majority vote.

Performing bagging with 300 trees reports the following confusion matrix and misclassification error:

```
Bagging (n=300) Confusion Matrix
##          arrest.test
## bag.pred    N    Y
##        N 4505  285
##        Y  134  702
Misclassification Error (Bagging, n=300)
## [1] 0.074
```

This is a large improvement over any of the previous methods we have used. In order to make the results more easily interpretable and prevent overfitting, I will reduce the number of trees used. Looking at the plot below can help us determine the correct number of trees to use :

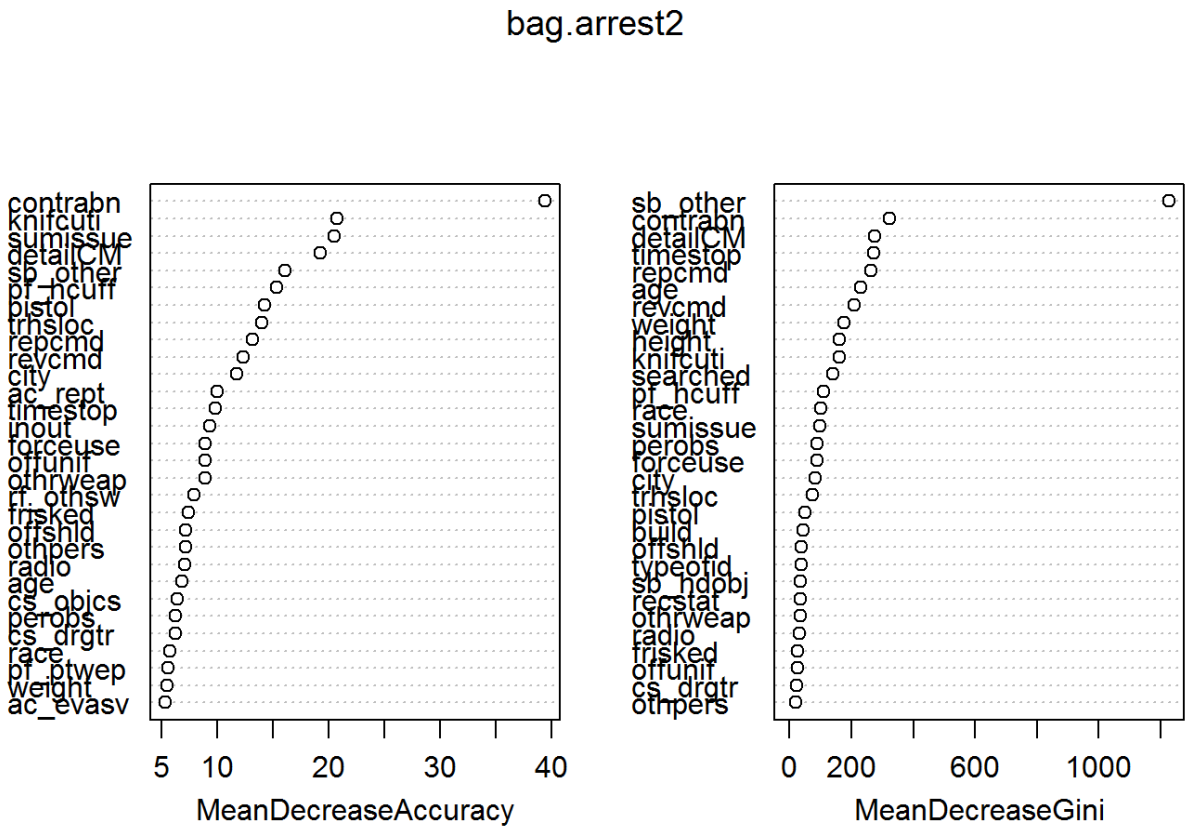


This plot determines the errors produced by each amount of trees. Using this, we can see visually where the line appears to stabilize and use this information to minimize the loss in accuracy from removing the trees. The green line shows the errors for the affirmative case (which is more likely), the red line for the negative case, and the black line for the overall error (the same as the misclassification error above).

The plot appears to level off at around n=25. When I recreated the model using only 25 trees, I found the misclassification error to be 0.076. The error and confusion matrix will be

included in the appendix. Although this is slightly higher than the model using 300 trees, it it a very small reduction in accuracy for a rather large increase in the interpretability and utility of the model.

Now that we have determined a good amount of trees, we can look at an importance plot to see which variables create the most variations in the errors. Since this is a categorical variable, it will be most useful to use the right graph, which uses the Gini index.



The variable with the most importance is sb_other, which is reason for search:other. This is a troubling thing, as described in the data section, because this is not a very informative variable. We can see that race now does have some importance in determining the arrests, as well as some other variables that were ommited from the logistic regression model. Interestingly, age, weight and height are included quite highly as well. Once again, whether the person was searched, had contraband, or was handcuffed is determined as important, which is consistent with our findings from the logistic regression.

By using a sample of the variables in each tree and performing the bagging method on a different subset for each iteration, we can implement a Random Forest method. This method is useful since it allows some variation in the trees, since they will not only be dominated by the most important variables. Using two numbers for the number of variables used, I produced two random forest models. Below are their confusion matrices and misclassification errors.

```
Random Forest (sqroot(p)), n=300 Confusion Matrix
##         arrest.test
## RF.pred    N    Y
##       N 4531  308
##       Y  108  679
Misclassification Error (Random Forest I)
## [1] 0.074
Random Forest (p/2), n=300 Confusion Matrix
##          arrest.test
## RF2.pred    N    Y
##        N 4522  291
##        Y  117  696
Misclassification Error (Random Forest II)
## [1] 0.073
```

The misclassification errors for the two are nearly identical. They perform very similarly to the bagging method. I would prefer the second model, as it predicts better in the case that the prediction and observed value are both yes than they are both no.

By using a similar method to the bagging method earlier, we determine a good reduction in the amount of trees is to 40. Using 40 trees, we get the following confusion matrix and misclassification error.

```
Random Forest (p/2, ntree=40)
##          arrest.test
## RF3.pred    N    Y
##        N 4516  284
##        Y  123  703
Random Forest 3 Misclassification Error
## [1] 0.072
```

In this case, the error actually was reduced. This can be accounted for by the fact that it simplified the model, which resulted in the random forest model with 40 trees actually predicting better than the more complex model. This suggests that the increase in bias by using a simpler model with 40 trees was less than the decrease in variance of the model on the testing data. This

model performs the best of all the models I have used, and is the most successful at predicting whether a person is arrested or not when they are stopped and frisked, with about a 7.2% misclassification rate.

Conclusions

Below is a table summarizing all of the models that I've used, and their misclassifcation errors on the test data.

Test Error for Methods Used

	Method	TestError	FalsePositive	FalseNegative
1	Full Logistic Model	0.095	0.176	0.081
2	Reduced Logistic Model	0.093	0.183	0.082
3	10-NN Full Model	0.162	0.402	0.143
4	5-NN Reduced Model	0.120	0.185	0.113
5	Pruned Tree	0.117	0.283	0.092
6	Unpruned Tree	0.117	0.257	0.098
7	Bagging (n=300)	0.074	0.160	0.059
8	Bagging (n=25)	0.076	0.173	0.058
9	Random Forest I (n=300)	0.074	0.137	0.064
10	Random Forest II (n=300)	0.073	0.144	0.060
11	Random Forest III (n=40)	0.072	0.149	0.059

I have also included the false positive and false negative results, as these can help to determine which models fit the best.

The Random Forest model which used 40 trees and 39 variables performed the best out of all the methods used. It is clear that the bagging and random forest methods were superior to the logistic and KNN methods. Of the simpler methods, the reduced Logistic Model using stepwise reduction with the AIC performed the best.

Despite my original inclinations, we can see from the importance plots on the bagging model and the summaries of the logistic model that race was not a very significant factor in determining whether a person would be arrested or not. However, as we saw before from the bar graphs, there was a significantly larger portion of blacks who were stopped and frisked than the general population. This suggests that a black person is more likely to be stopped, but not significantly more or less likely than other races to be arrested after they are stopped. Which raises the question: why stop more black people if they are no more likely to be arrested after the frisk than other races?

Another variable of interest was sb_other. This variable indicates that there was an "other reason for stopping the subject." As such, this variable is difficult to interpret. However, the variable was quite significant with a high coefficient in the logistic model, and was rated as the most important variable by a large margin in the bagging models. This is a point of difficulty in the interpretations. This may indicate that there needs to be more disclosure about the reason a person was searched, since the sb variables in their current state fail to capture much of the reasons for arrest.

In both the logisitic and bagging methods, the variables for contraband and knife/cutting object came up as significant, and had a significant effect on whether the person was arrested or not. This is logical: if the officer found a weapon on contraband on the person, they were much more likely to be arrested. This was far more significant than other factors about the person, in particular the cs variables, which list the reasons why the person was stopped, or the age, weight, gender and race variables, which are the physical attributes of the person. This suggests that the physical attributes of a person are not nearly as important as criminal possession in predicting an arrest, which is in support of the stop and frisk's usage.

For instance, being stopped due to the clothing you wear (cs_cloth) or being perceived as a lookout (cs_lkout) were not nearly as significant as carrying a suspicious object (cs_object) or if they were searched (searched), and actually had negative coefficients. So it appears that, most of the time, the probability of being arrested was largely reliant on whether the person was searched or not, or found to have an object. This matches up with the significance and postive coefficient of cs_drgtr, which was the variable representing the cause of search being for a suspected drug transaction. One would conclude that if a person was being suspected for a drug transaction, they would also be more likely to have contraband or perhaps a weapon than if they were not.

Interestingly, the frisked variable was not found to be significant in the logistic regression model, suggesting that frisking, as compared to searching, was not very significant in arrest, and therefore not too effective a measure to stop criminal activity. In both models, searching had a stronger effect than frisking in determining whether a person was arrested or not.

The success of the random forest/bagging methods as compared to the simpler methods suggests that the relationship between whether a person is arrested or not and the various variables in the dataset is quite complex, and that it cannot be estimated as well using the simpler techniques.

Appendix I (referenced items):

Coefficients of the stepwise logistic model:

```
Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.1394455  0.3563078  -3.198 0.001384 **
## recstat1       0.3575529  0.1023959   3.492 0.000480 ***
## recstat9      -2.0037225  2.3393201  -0.857 0.391699
## recstatA      -0.1811463  0.1180715  -1.534 0.124978
## inoutO        -0.8198646  0.0803864 -10.199  < 2e-16 ***
## trhslocP      -0.8738981  0.1249270  -6.995 2.65e-12 ***
## trhslocT      -1.4117556  0.1472958  -9.584  < 2e-16 ***
## perobs         0.0085851  0.0033665   2.550 0.010767 *
## typeofidP      0.0553679  0.2048434   0.270 0.786934
## typeofidR     -1.2301359  0.3365018  -3.656 0.000257 ***
## typeofidV     -0.1420737  0.2072662  -0.685 0.493051
## sumissueY     -2.9274054  0.2337174 -12.525  < 2e-16 ***
## offunifY      -0.2300162  0.0709762  -3.241 0.001192 **
## friskedY       0.1596321  0.0999233   1.598 0.110144
## searchedY      1.8727029  0.1564972  11.966  < 2e-16 ***
## contrabnY      3.1108031  0.1387840  22.415  < 2e-16 ***
## pistolY        2.9397144  0.3037615   9.678  < 2e-16 ***
## knifcutiY      2.0781429  0.1408105  14.758  < 2e-16 ***
## othrweapY      1.9085458  0.2061663   9.257  < 2e-16 ***
## pf_handsY      0.1829775  0.0958719   1.909 0.056318 .
## pf_wallY      -0.2684473  0.1426597  -1.882 0.059872 .
## pf_grndY       0.7069531  0.2111336   3.348 0.000813 ***
## pf_drwepY     -0.7202203  0.2427153  -2.967 0.003004 **
## pf_hcuffY      1.3038209  0.0958149  13.608  < 2e-16 ***
## pf_pepspY      3.6381377  1.1662366   3.120 0.001811 **
## pf_otherY     -0.2736084  0.1936878  -1.413 0.157766
## radioY        -0.4540335  0.0686508  -6.614 3.75e-11 ***
## ac_reptY       0.3203196  0.0716227   4.472 7.74e-06 ***
## rf_vcrimY     -0.2302783  0.0885105  -2.602 0.009276 **
## rf_othswY     -0.2195720  0.0938505  -2.340 0.019305 *
```

```
## ac_proxmY      0.1764237  0.0639188   2.760 0.005778 **
## rf_attirY     -0.3508092  0.1347952  -2.603 0.009254 **
## cs_objcsY      0.5277556  0.1306460   4.040 5.35e-05 ***
## cs_casngY     -0.1567943  0.0897938  -1.746 0.080783 .
## cs_lkoutY     -0.4306545  0.1190220  -3.618 0.000297 ***
## cs_clothY     -0.3454175  0.1787932  -1.932 0.053367 .
## cs_drgtrY      0.3362326  0.1183804   2.840 0.004507 **
## ac_evasvY      0.2346666  0.0809086   2.900 0.003727 **
## ac_assocY     -0.5555065  0.1340808  -4.143 3.43e-05 ***
## rf_rfcmpY     -0.2040480  0.1014848  -2.011 0.044365 *
## rf_verblY     -0.4083410  0.2922657  -1.397 0.162366
## cs_vcrimY     -0.3965895  0.1264841  -3.135 0.001716 **
## cs_bulgeY     -0.4881904  0.1440797  -3.388 0.000703 ***
## cs_otherY     -0.1770344  0.0673632  -2.628 0.008587 **
## rf_knowlY     -0.4071034  0.1555024  -2.618 0.008845 **
## ac_otherY     -0.3129862  0.0982278  -3.186 0.001441 **
## sb_hdobjY     -0.7305307  0.1589738  -4.595 4.32e-06 ***
## sb_otherY      1.1601221  0.1583057   7.328 2.33e-13 ***
## revcmd         0.0004988  0.0001588   3.142 0.001681 **
## rf_furtY      -0.1420974  0.0805927  -1.763 0.077874 .
## rf_bulgY      -0.2908946  0.1356321  -2.145 0.031974 *
## forceuseDO    -0.7594367  0.4329658  -1.754 0.079425 .
## forceuseDS    -0.6950447  0.1298365  -5.353 8.64e-08 ***
## forceuseOR    -0.0864302  0.2400623  -0.360 0.718823
## forceuseOT     0.3274358  0.1251247   2.617 0.008874 **
## forceuseSF    -0.0552919  0.1300915  -0.425 0.670820
## forceuseSW    -0.4997659  0.2145224  -2.330 0.019824 *
## sexM          -0.2124173  0.1120586  -1.896 0.058014 .
## sexZ          -0.6879402  0.4163837  -1.652 0.098498 .
## raceB          0.0408446  0.1577469   0.259 0.795693
## raceI          0.0878653  0.5519119   0.159 0.873510
## raceP          0.2519675  0.1881200   1.339 0.180441
## raceQ          0.3582209  0.1613654   2.220 0.026423 *
## raceU          0.2148683  0.4095903   0.525 0.599866
```

```
## raceW          0.1083503  0.1798309   0.603 0.546833
## raceZ          0.5206662  0.3238037   1.608 0.107842
## age            0.0073152  0.0026085   2.804 0.005042 **
## weight        -0.0012546  0.0008145  -1.540 0.123469
## cityBROOKLYN  -0.8072589  0.0847163  -9.529  < 2e-16 ***
## cityMANHATTAN -0.6764624  0.0904685  -7.477 7.58e-14 ***
## cityQUEENS    -0.5136538  0.0932366  -5.509 3.61e-08 ***
## citySTATEN IS -1.0791310  0.1548114  -6.971 3.16e-12 ***
## detailCM       0.0090240  0.0011467   7.870 3.56e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15712.6  on 16875  degrees of freedom
## Residual deviance:  8297.6  on 16803  degrees of freedom
## AIC: 8443.6
##
## Number of Fisher Scoring iterations: 6
```

Confusion Matrix/Error for Bagging with 25 trees:

```
Confusion Matrix
```

```
##          arrest.test
## bag.pred2    N    Y
##         N 4490  277
##         Y  149  710
```

```
Misclassification Error
```

```
## [1] 0.076
```

Apendix II, key calls:

Code for Full Logistic Model

```
logistfullmodel<-glm(arstmade~.,data=mydata[train,],family=binomial)
summary(logistfullmodel)

fit2<-predict(logistfullmodel,newdata=mydata.test,type="response")

fitted2<-ifelse(fit2>.5,1,0)
```

Code for Stepwise Logistic Model:

```
stepwiselogmodel<-glm(formula = arstmade ~ recstat + inout + trhsloc + perobs
+
    typeofid + sumissue + offunif + frisked + searched + contrabn +
    pistol + knifcuti + othrweap + pf_hands + pf_wall + pf_grnd +
    pf_drwep + pf_hcuff + pf_pepsp + pf_other + radio + ac_rept +
    rf_vcrim + rf_othsw + ac_proxm + rf_attir + cs_objcs + cs_casng +
    cs_lkout + cs_cloth + cs_drgtr + ac_evasv + ac_assoc + rf_rfcmp +
    rf_verbl + cs_vcrim + cs_bulge + cs_other + rf_knowl + ac_other +
    sb_hdobj + sb_other + revcmd + rf_furt + rf_bulg + forceuse +
    sex + race + age + weight + city + detailCM, family = binomial,
    data = mydata[train, ])


summary(stepwiselogmodel)
fit1<-predict(stepwiselogmodel,newdata=mydata.test,type="response")
fitted<-ifelse(fit1>.5,1,0)
```

Code for K-NN tests:

```
train1<-model.matrix(arstmade~.,data=mydata[train,])
test1<-model.matrix(arstmade~.,data=mydata.test)
knn.predict<-knn(train1,test1,arstmade[train],k=10)
train2<-model.matrix(arstmade~.,data=mydata[train,])[,7:60]
test2<-model.matrix(arstmade~.,data=mydata.test)[,7:60]
knn.predict2<-knn(train2,test2,arstmade[train],k=5)
```

Code for Tree-Based Methods

```
tree.arrest <- tree(arstmade ~ ., mydata[train,])##unpruned
```

```
cv.arrest<-cv.tree(tree.arrest,FUN=prune.misclass)##for pruning
prune.arrest<-prune.misclass(tree.arrest, best=3)##pruned tree

tree.pred<-predict(tree.arrest,newdata=mydata.test,type="class")##predictions
, unpruned

pruned.pred<-predict(prune.arrest,newdata=mydata.test,type="class") #predicti
ons pruned


bag.arrest<- randomForest(arstmade~ ., data=mydata[train,], mtry=(ncol(mydata
)-1), importance=TRUE, ntree=300)
bag.pred<-predict(bag.arrest,newdata=mydata.test,type="class")


varImpPlot(bag.arrest)##importance plot

bag.arrest2<- randomForest(arstmade~ ., data=mydata[train,], mtry=(ncol(mydat
a)-1), importance=TRUE, ntree=25)

bag.pred2<-predict(bag.arrest2,newdata=mydata.test,type="class") #bagging wit
h less trees

RF.arrest<-randomForest(arstmade~ ., data=mydata[train,], mtry=9, importance=
TRUE, ntree=300)

RF.pred<-predict(RF.arrest,newdata=mydata.test,type="class") ##random forest,
p=9

RF2.arrest<-randomForest(arstmade~ ., data=mydata[train,], mtry=(ncol(mydata)
-1)/2, importance=TRUE, ntree=300)

RF2.pred<-predict(RF2.arrest,newdata=mydata.test,type="class")##random forest
, p=74/2

RF3.arrest<-randomForest(arstmade~ ., data=mydata[train,], mtry=(ncol(mydata)
-1)/2, importance=TRUE, ntree=40)

RF3.pred<-predict(RF3.arrest,newdata=mydata.test,type="class") #RF with less
treesp
```

Collecting the Results

```
Method<-c("Full Logistic Model","Reduced Logistic Model","10-NN Full Model","
5-NN Reduced Model","Pruned Tree","Unpruned Tree","Bagging (n=300)","Bagging
(n=25)","Random Forest I (n=300)","Random Forest II (n=300)","Random Forest I
II (n=40)")


TestError<-c(misclasserrorsfull,misclasserrorstepwise,KNNMisclass,KNNMisclass
2,misclassificationprune,misclassificationunprune,misclassificationbagging,mi
sclassificationbagging2,misclassRF,misclassRF2,misclassRF3)


#error: Yes when No

Er1<-table1[2,1]/sum(table1[2,])
```

```
Er2<-table2[2,1]/sum(table2[2,])
Er3<-table3[2,1]/sum(table3[2,])
Er4<-table4[2,1]/sum(table4[2,])
Er5<-table5[2,1]/sum(table5[2,])
Er6<-table6[2,1]/sum(table6[2,])
Er7<-table7[2,1]/sum(table7[2,])
Er8<-table8[2,1]/sum(table8[2,])
Er9<-table9[2,1]/sum(table9[2,])
Er10<-table10[2,1]/sum(table10[2,])
Er11<-table11[2,1]/sum(table11[2,])

#error: No when Yes
Er21<-table1[1,2]/sum(table1[1,])
Er22<-table2[1,2]/sum(table2[1,])
Er23<-table3[1,2]/sum(table3[1,])
Er24<-table4[1,2]/sum(table4[1,])
Er25<-table5[1,2]/sum(table5[1,])
Er26<-table6[1,2]/sum(table6[1,])
Er27<-table7[1,2]/sum(table7[1,])
Er28<-table8[1,2]/sum(table8[1,])
Er29<-table9[1,2]/sum(table9[1,])
Er210<-table10[1,2]/sum(table10[1,])
Er211<-table11[1,2]/sum(table11[1,])

FalsePositive<-round(c(Er1,Er2,Er3,Er4,Er5,Er6,Er7,Er8,Er9,Er10,Er11),3)
FalseNegative<-round(c(Er21,Er22,Er23,Er24,Er25,Er26,Er27,Er28,Er29,Er210,Er2
11),3)

data.frame(Method,TestError,FalsePositive,FalseNegative)
```