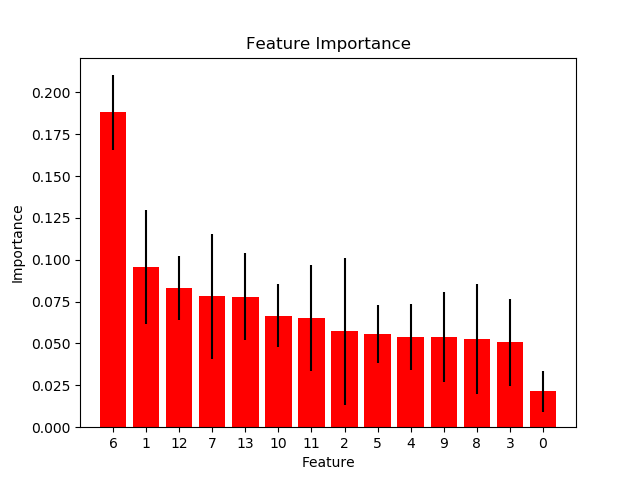
# Features

1. left\_corner, right\_corner
   1. What: Control of the lower corner of the board
   2. How: Returns 0 if the corner is empty, 1 if it has a Player 1 piece, and -1 if it has a Player 2 piece
   3. Why: Control of the corners can be a good strategy according to Connect 4 experts. We use 1 and -1 for the players instead of 1 and 2 for more meaningful results when added to the other corner or other features
2. center (for each player)
   1. What: Control of the center of the board
   2. How: The player receives 2 points for each piece it has in the center column and 1 point for each piece it has in the columns next to center
   3. Why: Control of the center is a favorite strategy of Connect 4 experts because it gives the player the option to form 4-in-a-row in any direction
3. connections (for each player)
   1. What: The number of connections the player has made, i.e. pairs of adjacent pieces
   2. How: Each pair of adjacent spaces on the board is checked to see if they both contain the specified player’s piece. Returns the total number of such connections
   3. Why: Although we don’t know the exact nature of the connections, the more connections there are, the greater the likelihood that they will form 4-in-a-row
4. rows (for each player)
   1. What: The number of rows that contain a player piece
   2. How: Checks each row for the given player’s pieces and returns the number of rows that contain at least 1 piece
   3. Why: A 4-in-a-row can be formed horizontally, vertically, diagonally (up, right) or diagonally (up, left). Except for horizontal, this will take up 4/6 rows, so occupying more rows may place the player closer to the goal. Also, occupying more rows gives the player more options to build a 4-in-a-row
5. columns (for each player)
   1. What: The number of columns that contain a player piece
   2. How: Checks each column for the given player’s pieces and returns the number of columns that contain at least 1 piece
   3. Why: Similar to rows
6. Pairwise combinations (+, -, \*)
   1. What: Pairwise combinations of the 10 base features using addition, subtraction, and multiplication
   2. How: Add, subtract, or multiply each pair of the 10 base features
   3. Why: This creates new features including net totals (e.g. center(1) – center(2)) and composite features (e.g. left\_corner + right\_corner reflects different states of the corners as a whole) as well as new features we wouldn’t have thought of ourselves

# Feature Selection



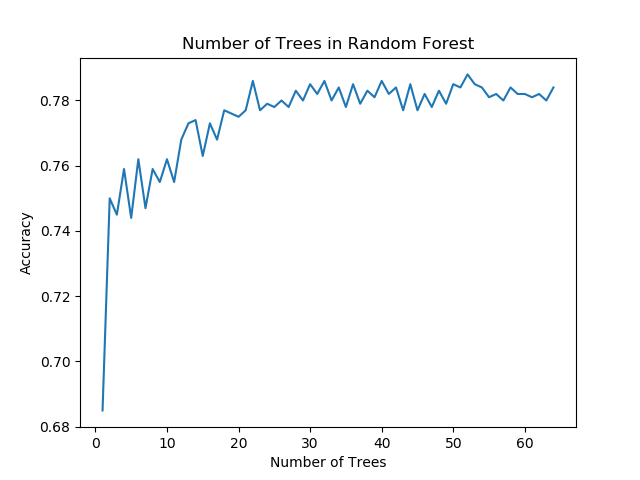
Our 10 base features along with the 3 combinations of each pair creates 145 total features that the decision trees can use to split data. However, many of those features are totally or nearly meaningless, so we select the best features using an sklearn.feature\_selection.SelectKBest object. We select the k best features for k = 1, 2, 3, …, 145 and select the k that gives us the best accuracy on a basic decision tree classifier.

This feature selection strategy chose k = 14 and picked the following features (in order of importance):

1. center(1) – center(2)
2. left\_corner + right\_corner
3. center(2) – columns(2)
4. center(2) – rows(1)
5. rows(1) – rows(2)
6. center(2) + columns(1)
7. center(2) \* columns(1)
8. left\_corner + center(1)
9. right\_corner + center(2)
10. left\_corner – rows(1)
11. center(2) \* rows(2)
12. center(2) + rows(2)
13. left\_corner \* center(1)
14. center(2)

This list has some expected features such as net totals for center (#1) and rows (#5), total corners (#2), and one of the original features (#14). Some of the other features make intuitive sense, like #13 is large when Player 1 controls the center and left corner, but becomes negative if Player 2 controls the left corner. Simple combinations like #11 and #12 may not form a visually identifiable board characteristic, but since they positively combine features that are good for Player 2 when large, it makes sense that the combination could have a stronger relationship than a single base feature. However, some features like #8 don’t have a clear meaning.

# Experiment 1 – Number of trees in a random forest



Probability that the accuracies are the same between:

n = 1, n = 64: 5.017407534224816e-07

n = 2, n = 64: 0.07217523878233857

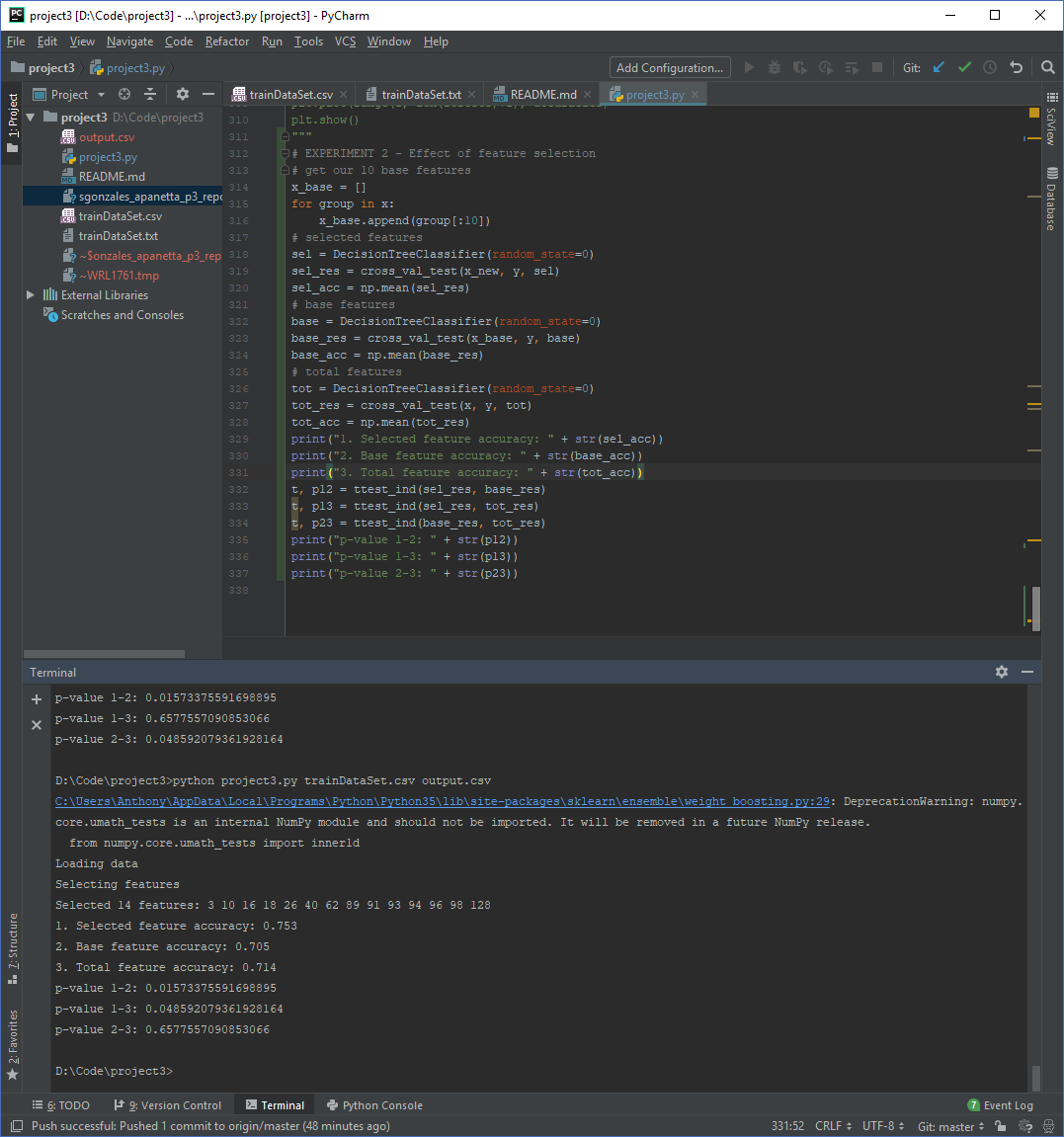
n = 3, n = 64: 0.03987374533215778

n = 4, n = 64: 0.18322878153738742

n = 5, n = 64: 0.035181842697043185

Our data set and features did not produce enough accuracy to consistently tell apart forests of different sizes, but 3 out of the 5 smallest forests did have a p-value <0.05 compared to the largest forest with 64 trees. Therefore, we can conclude that the number of trees in a forest does have a statistically significant effect on accuracy.

# Experiment 2 – Effect of feature selection

We verified that our feature selection improved results on a standard decision tree by calculating accuracy and p-values when using our 14 selected features, our 10 base features, or the 145 total features generated. The accuracy using our selected features was greater than when using either our base or total features, and since the p-value is <0.05 we can reject the null hypothesis and conclude our feature selection does improve accuracy.