**Preprocessing the Data**
- Renamed column having numerical value for number of years of Education to 'Education 1'
- Encoded categorical features/discretized continuous features (age, Education1, Capital-gain, etc) from columns as numbers.
- Deleted cells from table with non-numerical values such as ? or n/a
- Scaled the features with a mean of 0 and variance of 1 using Standard Scaler in sci-kit library.
- The data was preprocessed using the same methods for all models.
- A cross validation of 10 was used for all models.

- The data was explored visually to see which feature was correlated to predicting whether Income would be greater than $50,000 US. The variable 'Education' was selected as the feature to be evaluated by each of the models to predict the dependent variable 'Income.'

**Tuned Hyperparameters**

1. All classifiers had a cross validation of 10 and training size of 0.80 unless otherwise noted.
2. Accuracy was used because it is the most commonly used evaluation metric for classification problems and is the number of correct predictions made as a ratio of all predictions made.

a) Support Vector Machine with C = 1, accuracy = 0.8446
Support Vector Machine with C = 5, accuracy = 0.8447
Support Vector Machine with C = 10, accuracy = 0.8414

b) Adaboost with tree depth = 1, accuracy = 0.8506
Adaboost with tree depth = 5, accuracy = 0.8561

Increasing the tree depth improved accuracy slightly.

c) Random Forest with tree depth = 1, accuracy = 0.7657
Random Forest with tree depth = 10, accuracy = 0.8535
Random Forest with tree depth = 15, accuracy = 0.8504
Random Forest with tree depth = 20, accuracy = 0.8477

The optimal tree depth parameter to yield the highest accuracy is between 10 and 15.

d) Adaboost with tree depth = 5, accuracy = 0.8561

e) Random Forest with tree depth = 10, accuracy = 0.8535

f) Support Vector Machine with C=5, accuracy = 0.8447

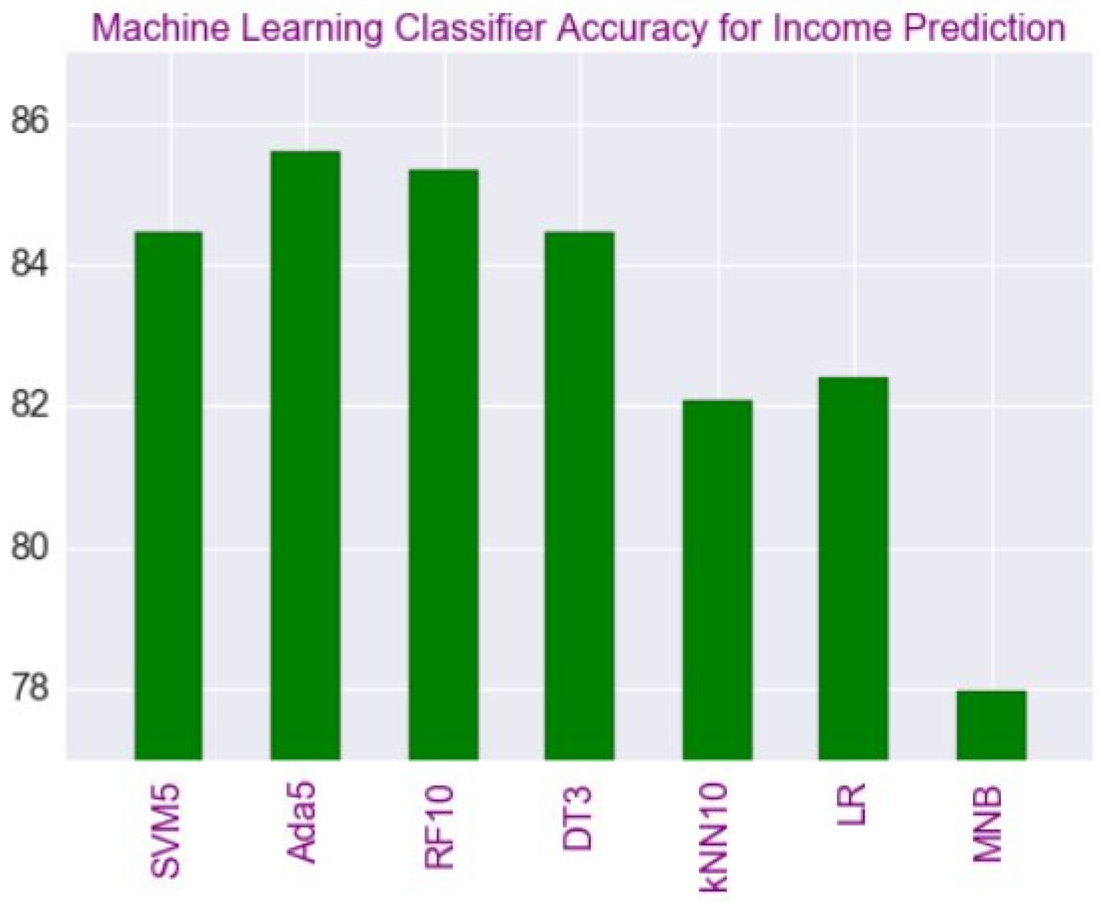
g) kNN when k=3, accuracy = 0.8005
kNN when k=10, accuracy = 0.8207
kNN when k=20, accuracy = 0.8205

h) Decision tree with depth = 3 and leaf size = 5 and cv=10, accuracy 0.8446
Decision tree with depth = 3 and leaf size = 5 and cv=5, accuracy decreased to 0.8442

i) Logistic regression model, accuracy = 0.8239

j) Multinomial Naive Bayes, accuracy = 0.7796

The top 7 most accurate classifiers were plotted visually, changing the y-axis and labels to be aesthetically pleasing.



*Why did the models perform differently? Compare and contrast the classifiers*.
In my case the Adaboost, Random Forest and Support Vector Machine were the top three performing classifiers in terms of accuracy of whether a person with a certain amount of education would have an income greater than $50,000. Other research suggests Adaboost is

also one of the best out-of-the-box classifiers. Adaboost is the most accurate because the model is run multiple times on reweighted training data. According to research by Fernandez-Delgado et al., in general random forest classifiers performed best from 179 classifiers on the entire UCI machine learning educational data set. Our random forest model was so accurate since it's not overly sensitive to the specific hyper-parameters in the data set used. Support Vector machine was also accurate because we selected a variable (Education) that impacted the resulting income within the relatively small data set. Since the instances of data is linearly separable (Income > 50K or Income <50K), the SVM method has a high accuracy.

*Why does changing c affect test data set*?
The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

*Why does changing tree depth affect test data set*?
More trees is always better with diminishing returns. Deeper trees are almost always better subject to requiring more trees for similar performance.

The above two points are directly a result of the bias-variance tradeoff. Deeper trees reduces the bias; more trees reduces the variance. The most important hyper-parameter is how many features to test for each split. The more useless features there are, the more features you should try.