Adult Census Income Binary Classifier

Ujjawal Madan

14/02/2020

Contents

1 Introduction

In this project, I build a binary classifier that predicts whether a person's income is over $\50,000 or under $50,000 using the 1994 Cenusus Income Data Set. This dataset, originally extracted by Barry Becker from the 1994 US Census Database contains over 48,000 rows of data and 14 predictor variables. First, I explore each variable graphically stratified by income level after which the data is then preprocessed and the feature variables are selected. I then run several commonly-used machine learning algorithms with repeated cross validation and parameter tuning. Lastly, three versions of each model are created by optimizing the thresholds for Accuracy, F1-Score, and the Distance to Left Point - ROC Curve. The final three models that perform best in each of the three categories (Accuracy, F1, Distance to Left Point) are then tested with the final validation set.

Dataset can be found here: https://www.kaggle.com/uciml/adult-census-income

More details can be found here. http://www.cs.toronto.edu/~delve/data/adult/adultDetail.html

2 Examination of Variables

Let's examine the variables and the descriptions:

Table 1: Examination of Variables in Adult Income Census Dataset

Variable	Variable Description
age	Age of the individual
workclass	Employment status of the individual
fnlwgt	The number of people the census believes the entry represents
education	Highest level of education achieved by the individual
education_num	The duration one has been in school
marital_status	Marital status of the individual
occupation	General occupation of the individual
relationship	Relationship or family status of the individual. e.g. Own Child, Not in Family etc.
race	The individual's race or ethnic background
gender	Biological sex of the individual
capital_gain	Capital Gains reported by the individual
capital_loss	Capital Losses reported by the individual
hours_per_week	Hours per week spent at work
native_country	Individual's Country of Origin
income	Whether the individual makes more or less than $50,000 annually.

Intuitively, one can see that some variables may be more predictive than others. For example, it would make sense for fnlwgt to not be linked in any way with income level.

2.1 Data Formatting

There are several steps that need to performed initially to view our data:

Step 1 - Convert categorical variables into factored form.

Step 2: Examine the incomplete data and decide on course of action.

Table 2: Missing Valuables by Variable in Adult Income Census Dataset

Variable	Number of Values Missing
age	0

Variable	Number of Values Missing
workclass	2799
fnlwgt	0
education	0
education_num	0
marital_status	0
occupation	2809
relationship	0
race	0
gender	0
capital_gain	0
capital_loss	0
hours_per_week	0
native_country	857
income	0

Number of rows that contain a value in Occupation (not NA) when the Workclass field is NA.

```
## integer(0)
```

After examining the data, I discover that the issue lies with the native country, workclass, and occupation fields. Moreover, all the rows where the workclass entry is missing, the occupation entry is also missing. There are several options at this point:

Option 1: The NAs are left alone and the algorithms treat it as such.
As not all of the machine learning algorithms will be able to handle NA values, this option is not viable.

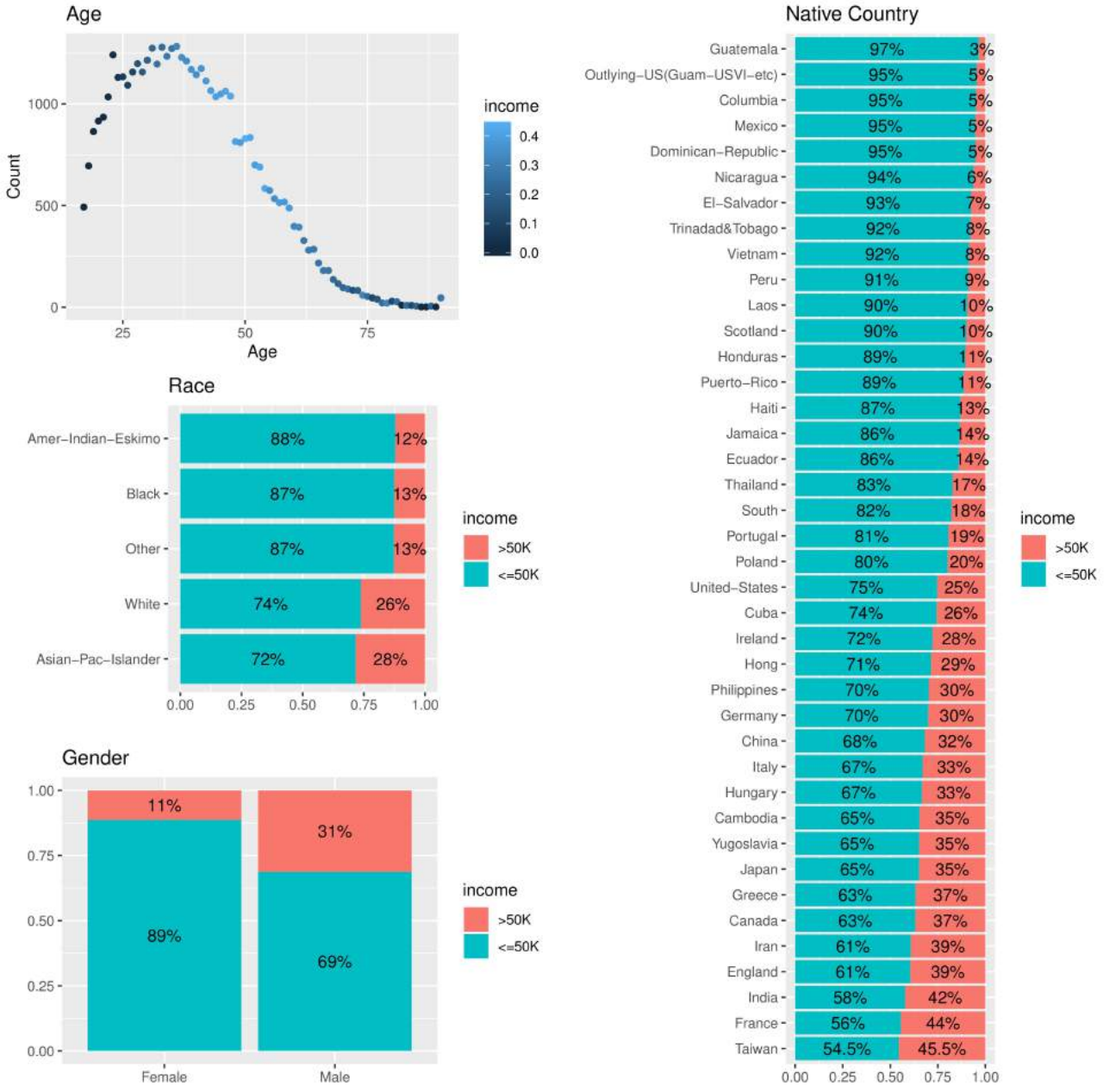
Option 2: The NAs are replaced with a value, either the mode of that variable or a random value.
This is an option, but could possibly tamper with our models.

Option 3: Remove the fields that contain NAs.

It certainly makes sense to remove the rows that are missing the occupation and workclass fields since there are two missing entries in those rows. While one might decide to not remove the rows with native_country missing, I choose to delete those rows as well as they only constitute a very small fraction of all the data and there are more than enough rows for training, testing and validation. In total, 3620 rows are removed altogether.

Result: All rows that contain one or more NA are removed.

2.2 Visualizing the Variables



The above plots are of individual personal characterstics: Age, Race, Gender, and Native Country.

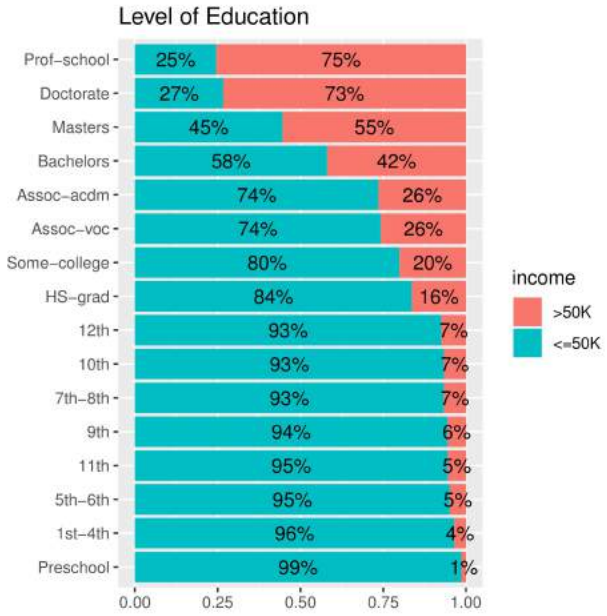
Age: Starting from the top left plot, Age, we can see that as one gets older, the odds of one having an income over $50,000 goes from almost 0 to 40 percent, peaking between 40 and 60 years of age, after which the probability slowly decrease with age.

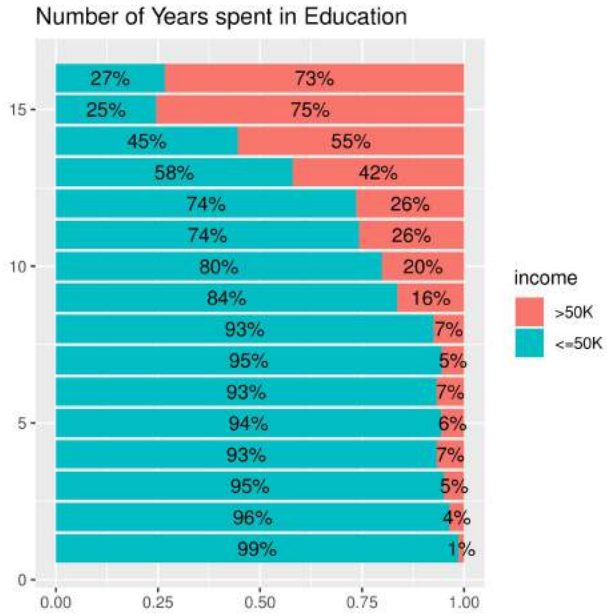
Race: An individual in this dataset falls into one of five categories. If one is either Amer-Indian_Eskimo, Black or Other, the odds of that individual making more than $50,000 are around 12 - 13 percent. If one is either White or Asian the odds are much greater - 26 to 28 percent.

Gender: There is a clear discrepancy in income level between males and females, with a randomly selected male having a three times greater chance of making more than $50,000 than a randomly selected female.

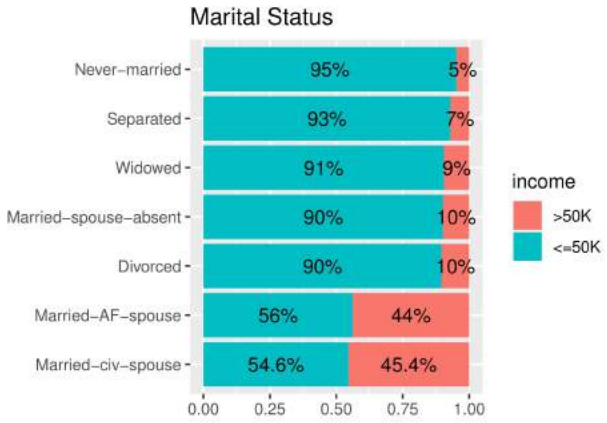
Native Country: Based on the rightmost plot, the native country of a randomly selected individual can noticeably

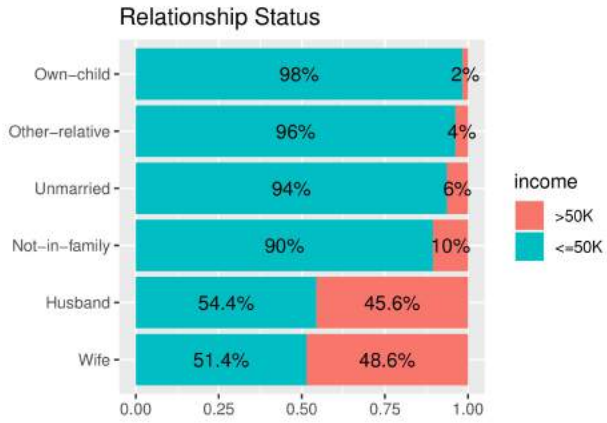
alter the chance of them making more than $50,000, with an randomly selected individual from Taiwan having almost five times as much chance of making moree than $50,000 than a randomly selected individual from Guatemala.





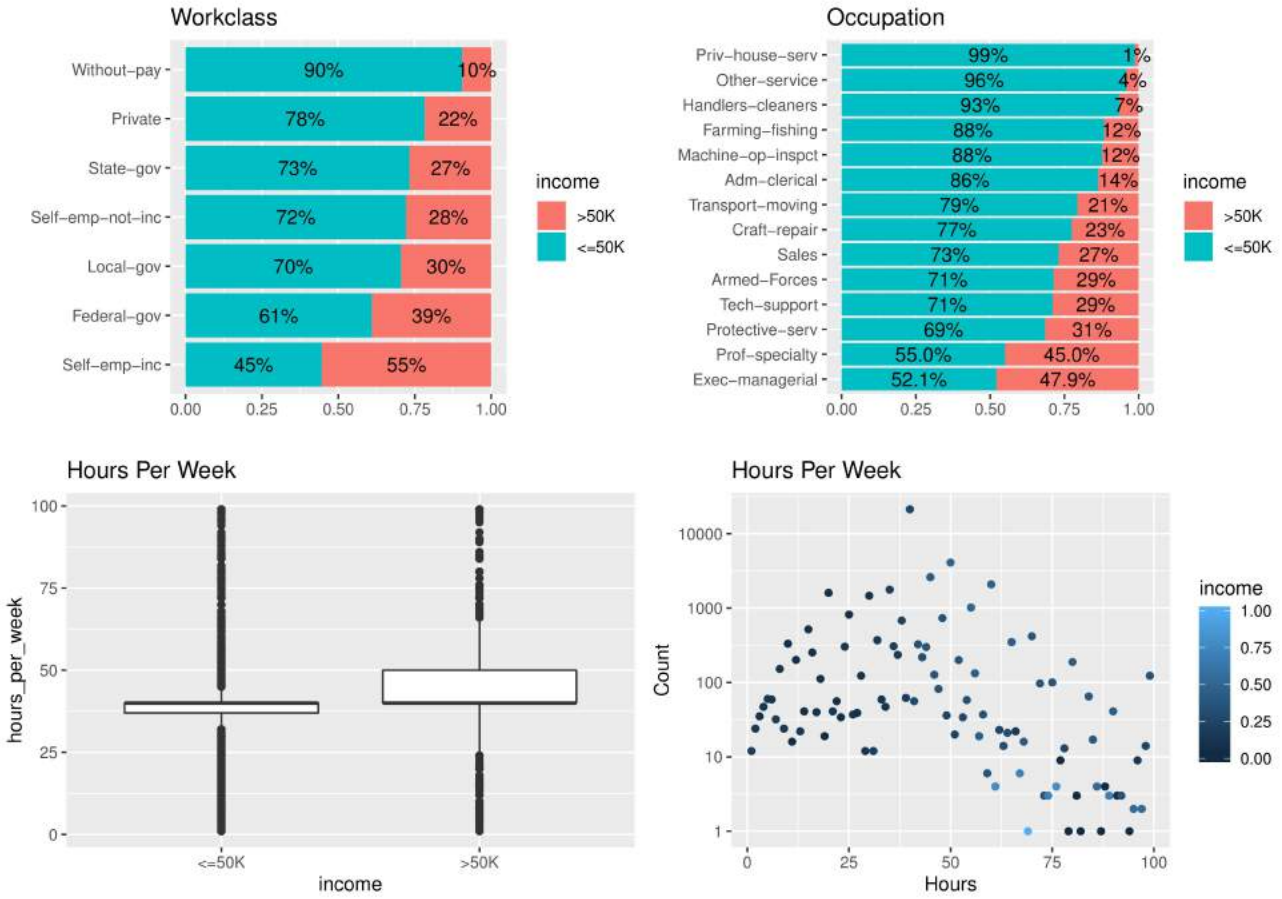
Level of Education and Number of Years Spent in Education The plots intutively seem correlated, given that the level of education often corresponds to the duration one has been in school. The above plots confirm the logical inference that higher education means there is greater chance of one's income being more than 50,000. A clear trend is visible that actually looks exponential in growth: the higher the number of years or level of education (linear), the higher the chance that individual makes more than $50,000 (exponential).





Marital Status It seems that one's marital status can have an effect on one's chance of having an income greater than $\50,000. If one is married, they have almost 5 to 10 times as much chance of having an income greater than $50,000 versus someone who is not married or not living with their spouse.

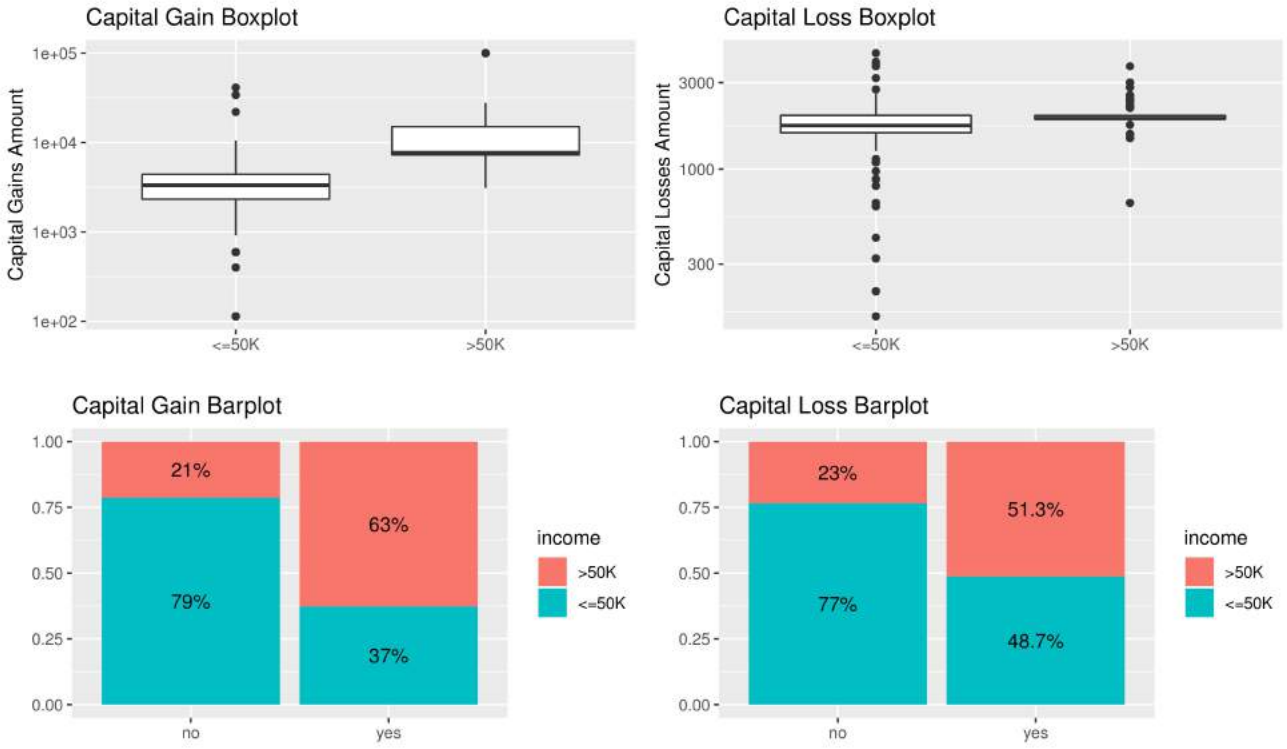
Relationship Status: This variable seems slightly similar to marital status although the categories are slightly different. Once again, if one has a spouse, one can have between 4 to 25 times as much chance of having an income over $50,000 than an individual who is otherwise unmarried.



Workclass - Starting fromm the top left plot again, depending on the employment status and type of employment, the odds of one's income level signficantly varies. It makes sense that one who is not being paid is not making more than $\50,000. Slightly surprising to me personally is that more than half of individuals who are self-employed make more than $50,000. Although this is not the feature selection stage, this variable seems like it would be a good predictor variable.

Occupation - The top right plot indicates that - depending on occupation - one's income level varies significantly. Once again, it logically makes sense that an executive manager would have a much higher chance of making more than $50,000 than a private house server. This also seems to be a good predictor variable.

Hours Per Week - The box plot on the left indicates that the median higher-income earner works a greater number of hours than the median lower-income earner. The plot on the left shows, although not very strongly, there seems to be a trend betwen hours worked and income level.



Capital Gains - The capital gains plots indicate both that one who has reported capital gains is more likely to have made an income over $\$50,000$, as well as that higher capital gain amounts are associated with those who make over $50,000.

Capital Loss - The capital loss plots indicate that, like the capital gains variable, one who has reported capital losses is more likely to have had a higher income level versus someone who has not reported any capital losses. However, the amount one reports does not seem to be strongly linked with one's income.

2.3 Preprocessing

Step 1: Normalize The first step is to normalize the variables, given that not doing so would not yield inaccurate models with algorithms such as the *k-nearest neighbours* algorithm.

Step 2: Seperate into training, testing and validation sets. As it is rather computationally expensive to use the standard 80:20 or 70:30 training to validation ratio, only 10,000 sample rows are used for training, with 5000 allotted for training purposes and the approximately 30,000 remaining rows allotted for the validation set which will not be utilized until final testing.

Step 3: Decide on a course of action for categorical variables. Ordinarily categorical variables would need to be converted to numeric form through some form of encoding or turned into dummy variables. However, the machine learning models will accept it in categorical form so they simply need to be put into factored form.

Result: The dataset is normalized and then split into train_set containing 10,050 rows, test_set containing 5024 rows and validation_set (which will not be used until the end) containing 30,148 rows.

2.4 Feature Engineering

Feature engineering is an important part of this process for several reasons. Firstly, the greater the number of variables means the greater the computation time and a leaner set of feature variables would reduce computation time considerably. Furthermore, variables that have very little predictive power would just create unncessary noise and correlated variables would also not perform well with each other. Intuitively, I discerned that education and education_num were both clearly correlated and so only the better performing variable of the two would be chosen.

While there are variety of techniques that one could utilize for feature selection - including filter methods, wrapper methods and embedded methods - for simplicity, I chose the RFE wrapper method using the random forest algorithm. Optimized for accuracy, the RFE wrapper recursively reduces the number of features by ranking them by variable importance and dropping the lowest performing feature after each call. Keeping track of accuracy, I then chose the set of features that maximized accuracy.

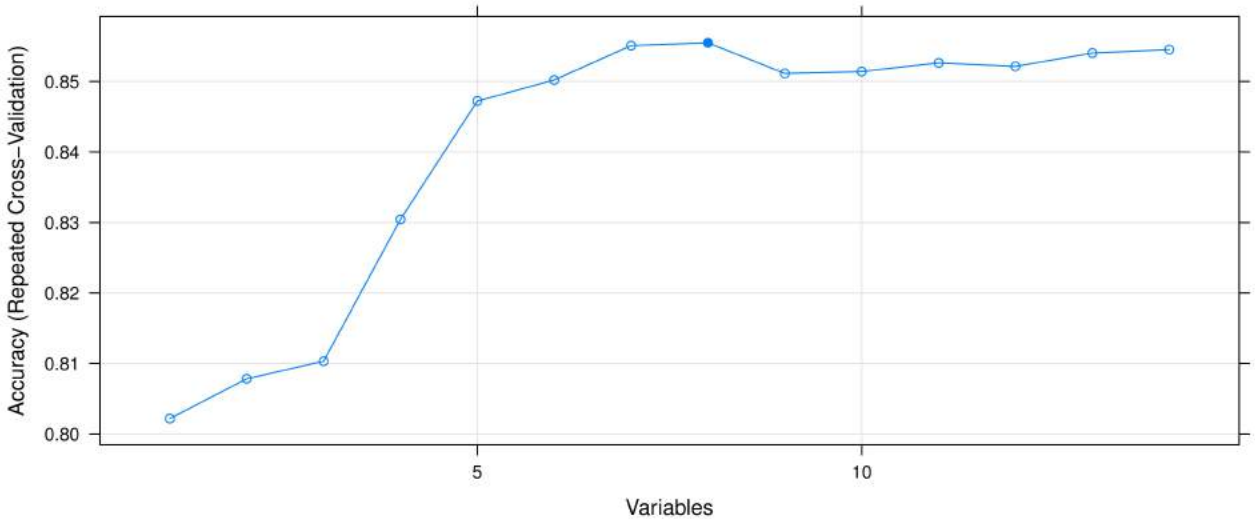


Table 3: Variable Importance

	Overall
capital_gain	94.37431
capital_loss	34.12511
occupation	33.66227
age	29.70439
marital_status	29.49700
relationship	24.59675
education	23.95239
hours_per_week	19.98718
education_num	19.92983

Given that education and education_num are quite similar and education is ranked higher, eduation_num is removed.

Based on the results above, the following variables (indicated below) are selected for predictor variables in the models.

```
frm <- as.formula(paste('income ~ capital_gain + capital_loss + marital_status + occupation +
                        age + relationship + education'))
#frm stands for forumla
```

3 Methods and Analysis

3.1 Hyperparameter Tuning

There are number of ways to tune each model. The five commonly used methods are:

- Manual Search
  - Manual Search may be the least computationally expensive and most efficient, especially if there is only one parameter. By visualizing the accuracy vs parameter, one can then narrow it down and find the exact value(s) to tune the model.
- Random Search
  - Random Search, while not the most effective, may be the most efficient in cases where there are several parameters. It may not yield the best-tuned model, but would still be very effective.
- Grid Search
  - Grid search, while thorough, is very computationally expensive given that it would have to run through every single combination of parameters and may not be the most efficient. It is however used for one model where grid search is the best option as there is only one parameter and manual search would be redundant.
- Automated Hyperparameter Tuning (Bayesian Optimization)
  - Bayesian Optimization is computationally expensive in R and so is not used in this project, although the code for it is attached if one wants to run it.
- Artificial Neural Networks (ANNs) Tuning
  - This project will not delve into neural networks although one could use it in this context.

Each model below is tuned with 5-fold cross validation repeated three times, to reduce computational time. Only the final models will be trained with 10 fold cross validation repeated 5 times.

3.2 Random Forest

Using the randomForest package, I chose to manually tune the mtry parameter as it would not very burdensome and would likely be less computationally expensive.

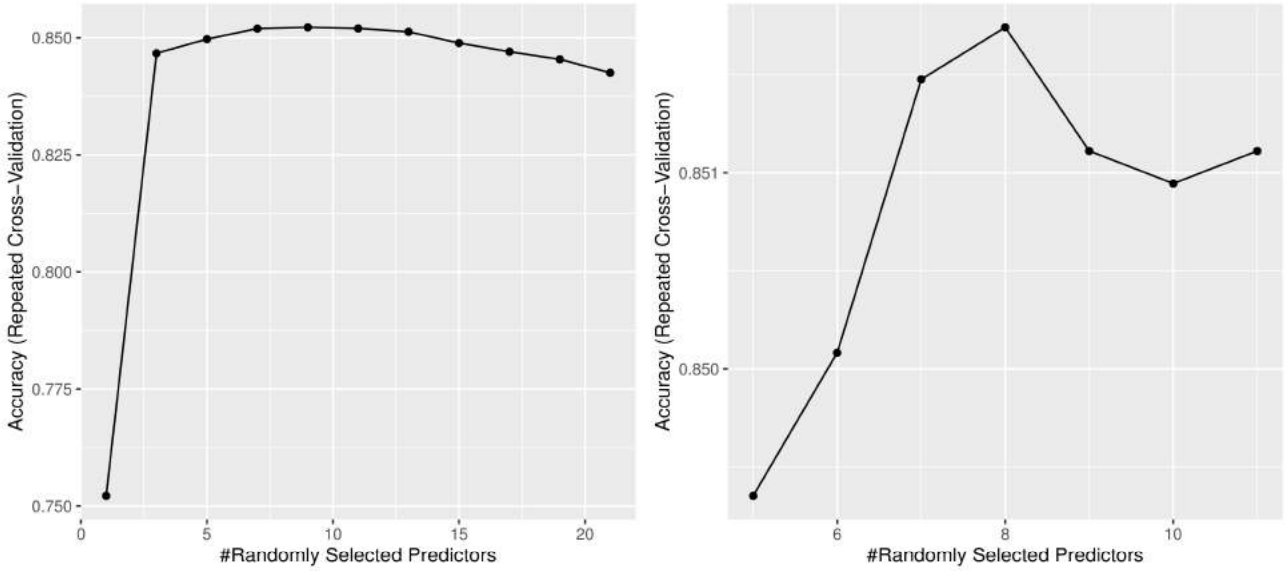


Figure 1: Random Forest Parameter Tuning - mtry VS Accuracy

3.3 Logistic Regression

Logistic Regression seems like an ideal choice given that is a binary classification. As there are no tuning parameters, the final model is trained with 5-fold cross validation repeated 3 times.

Bayes Logistic Regression is also explored as well with 5-fold cross validation repeated 3 times.

3.4 K-Nearest Neighbours

The K-Nearest Neigbours model is trained with 5-fold cross validation repeated three times and the tuning paramater, k (nearest number of neighbours), is tuned using a grid search.

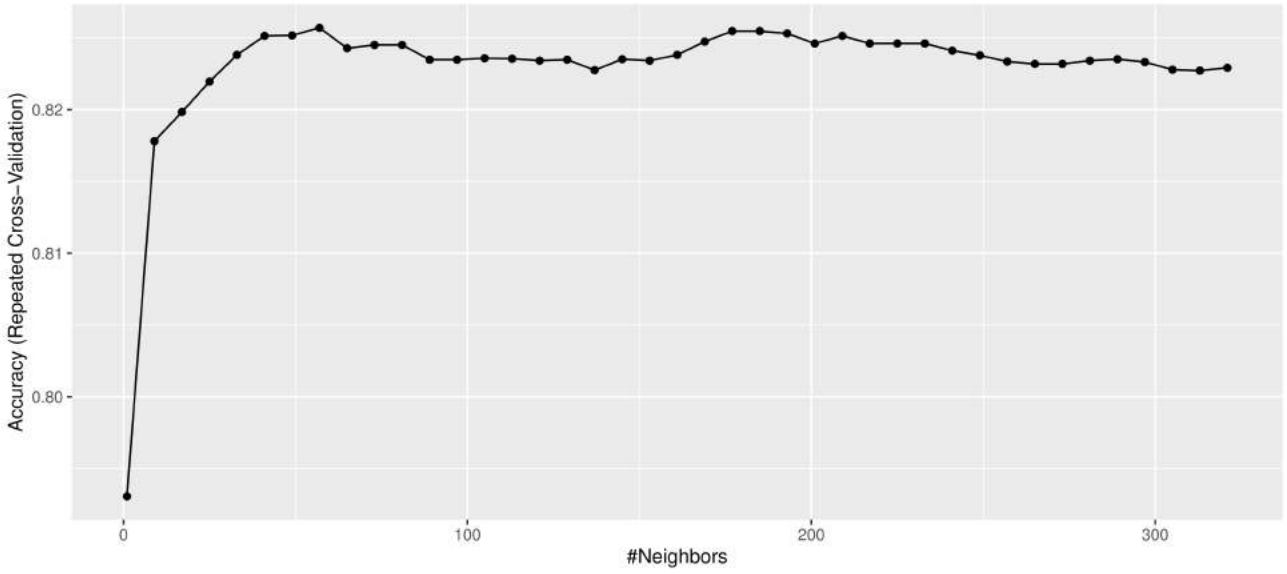


Figure 2: KNN Parameter Tuning - Number of Neighbours VS Accuracy

3.5 Naive Bayes

The Naive Bayes algorithm is also employed using a random search consisting of 25 iterations and with 5-fold cross validation repeated three times.

3.6 Support Vector Machine

There are two support vector machine algorithms I run: one with a linear kernel (svmLinear) and non-linear kernel (svm_radial).

Svm_Linear has only one parameter, C(Cost), and is manually tuned. The final model is trained with 5-fold cross validation repeated 3 times.

```
## line search fails -1.052172 -0.005471372 1.55829e-05 -3.842824e-06 -1.064385e-08 -4.146145e-09 -1.49929
```

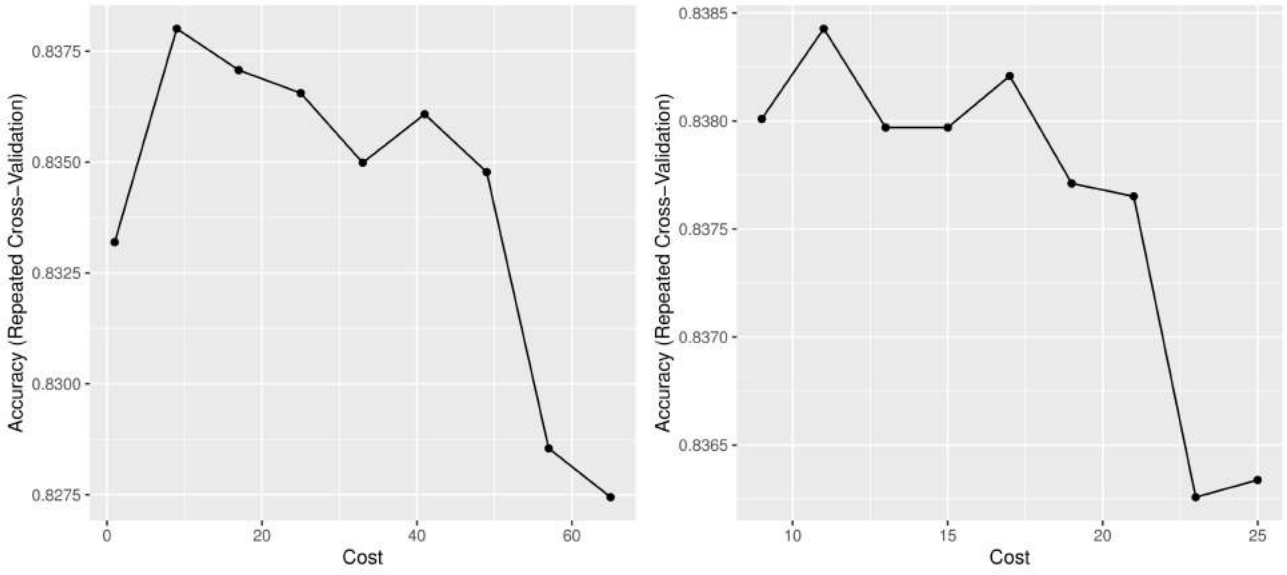


Figure 3: SVM Linear Parameter Tuning - Cost VS Accuracy

Svm_Radial has several parameters and so random search is employed with 25 iterations. One could also utilize the bayesian optimization method and use the provided code in the .R file.

3.7 XGBoost

The last algorithm employed is the XGBoost (Extreme Gradient Boosting). As the default method is to run 50 different combinations of parameters, there is no need to employ any form of parameter tuning.

4 Evaluation of Models

4.1 Choosing a Threshold

There are a number of ways to optimize the final model. The three methods that I examine are: Pure accuracy, F_1, or Distance to Corner of ROC curve, or a combination of all three.

Confusion Matrix Accuracy $Accuracy = TP/FP$

While this may be a good metric to optimize for, it is possible that the model is not accurately identifying the minor class sample, which in this case would be the those whose income is greater than $50,000. Context is key! If maximizing the ratio of total correctly predicted to total incorrectly predicted is the goal, then a model optimized for accuracy is clearly the right choice. However, it may not be the most suitable model depending on the context.

F1 $F1 = 2 * (Recall * Precision)/(Recall + Precision)$

The F1 score is the harmonic mean between precision and recall and is perhaps a more important measure than accuracy in a certain context. This score will measure how well it can detect an income greater than $\50,000 (Recall) as well as how often it will be correct when it does (Precision). For example, if the federal government is trying to get in touch with all individuals who earn more than $50,000 for tax reasons but wants to make sure it is only sending notices to the correct individuals, then a model optimized by the F1 score may be of more use to them given that context. If the focus is on better identifying the positive (smaller) class and if the classes are slightly imbalanced, which it is in this case, the F1 Score will be a better measure relative to other possible measures.

ROC TPR vs FPR

The ROC curve (Receving Operating Characterstic Curve) is often used to measure the overall performance of a classifier which is calculated by the true positive rate against the false positive rate. An optimal cutoff point (Distance to Left Corner) is calculated for each model where the point on the ROC curve that has the minimum distance to the upper left corner is chosen as the optimal cutoff. Again, if the context requires the accurate detection of both classes and both classes are equally important, then this may be the most appropriate measure in this particular situation.

4.2 ROC and PR Curve

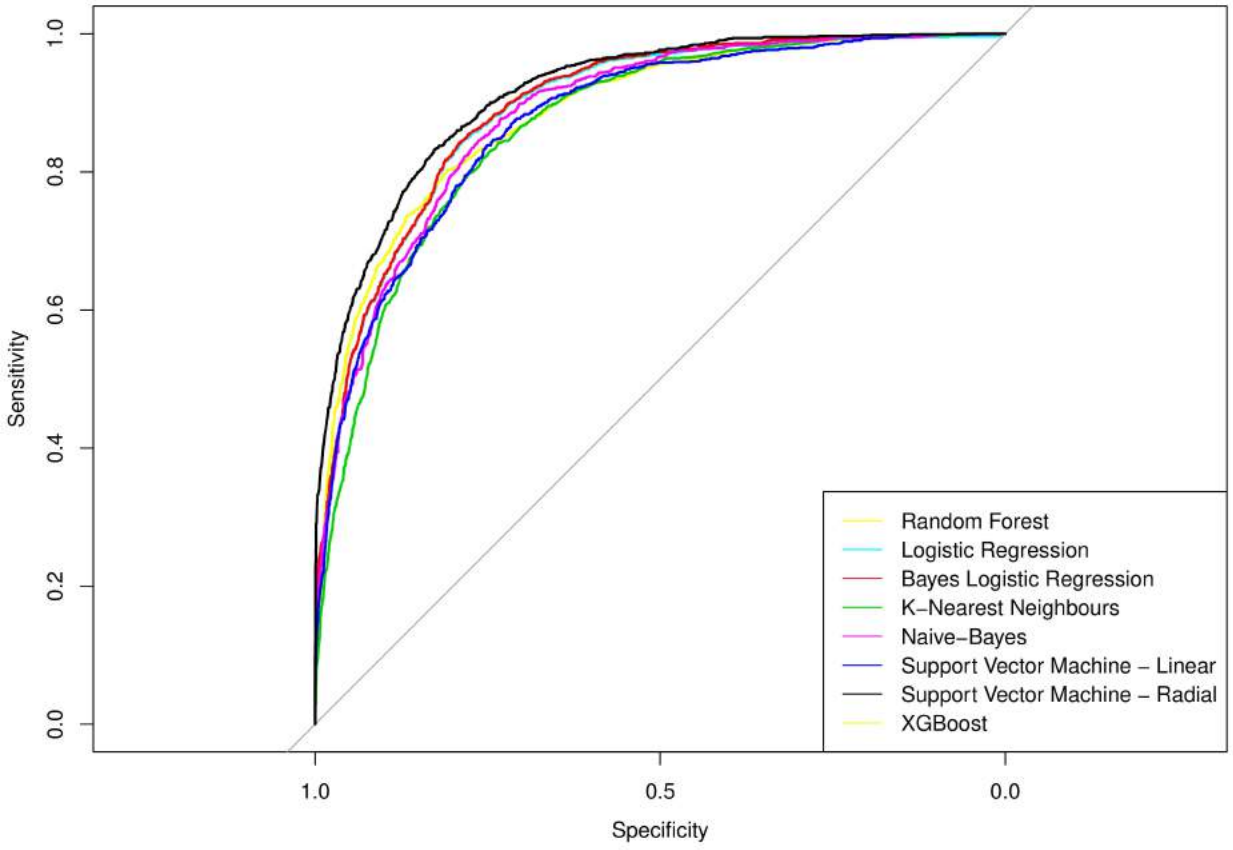


Figure 4: ROC Curve

One can see that in regards to sensitivity and specificity, the XGBoost algorithm performs quite well in relation to the other algorithms.

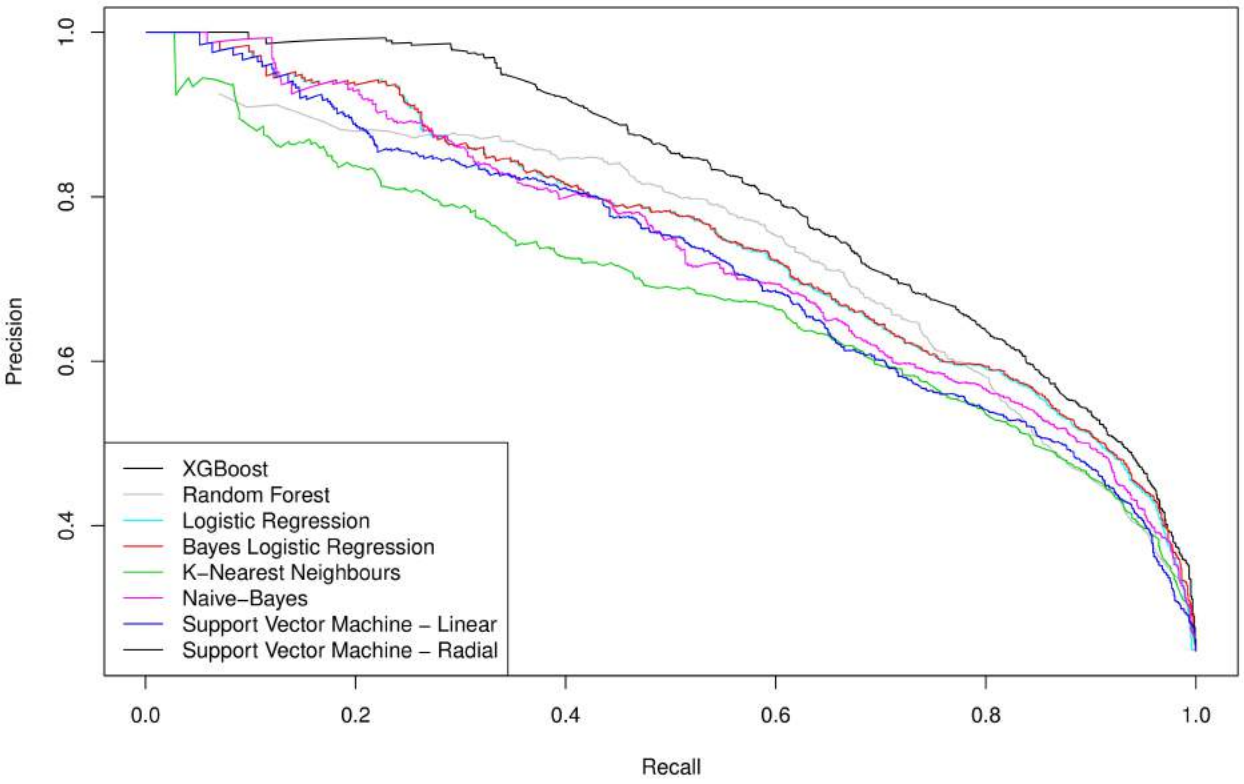


Figure 5: Precision vs Recall

Again, one can see that the XGBoost Algorithm performs quite well, except for a certain area where Recall is between 0.4 and 0.7 and the Random Forest algorithm outshines the XGBoost Algorithm.

4.3 Model Results

Table 4: All Models Optimized by Accuracy

	Accuracy	F1	ROC-AUC	Sensitivity (TPR)	Specificity (TNR)	Precision
Random Forest	0.85052	0.90318	0.88491	0.92696	0.61847	0.88059
Logistic Regression	0.84295	0.89844	0.89202	0.92352	0.59839	0.87469
Bayes Logistic Regression	0.84375	0.89904	0.89446	0.92485	0.59759	0.87462
K-Nearest Neighbours	0.82186	0.88524	0.86707	0.91347	0.54378	0.85871
Naive-Bayes	0.77010	0.86744	0.88909	1.00000	0.07229	0.76591
Support Vector Machine - Linear	0.83280	0.89407	0.88340	0.93808	0.51325	0.85401
Support Vector Machine - Radial	0.83639	0.89598	0.87365	0.93676	0.53173	0.85860
XGBoost	0.86027	0.90954	0.91389	0.93384	0.63695	0.88646

Table 5: All Models Optimized by F1 Score

	Accuracy	F1	ROC-AUC	Sensitivity (TPR)	Specificity (TNR)	Precision
Random Forest	0.85350	0.90634	0.88491	0.94231	0.58394	0.87301
Logistic Regression	0.84395	0.90136	0.89202	0.94787	0.52851	0.85920
Bayes Logistic Regression	0.84415	0.90150	0.89446	0.94813	0.52851	0.85923
K-Nearest Neighbours	0.81927	0.88701	0.86707	0.94311	0.44337	0.83721
Naive-Bayes	0.77010	0.86744	0.88909	1.00000	0.07229	0.76591
Support Vector Machine - Linear	0.83400	0.89668	0.88340	0.95766	0.45863	0.84300
Support Vector Machine - Radial	0.83678	0.89688	0.87365	0.94364	0.51245	0.85454
XGBoost	0.86226	0.91293	0.91389	0.96004	0.56546	0.87023

Table 6: All models Optimized by Distance to Left of ROC Curve

	Accuracy	F1	ROC-AUC	Sensitivity (TPR)	Specificity (TNR)	Precision
Random Forest	0.80693	0.86296	0.88491	0.80815	0.80321	0.92574
Logistic Regression	0.80314	0.85805	0.89202	0.79095	0.84016	0.93758
Bayes Logistic Regression	0.80394	0.85862	0.89446	0.79148	0.84177	0.93821
K-Nearest Neighbours	0.78304	0.84375	0.86707	0.77878	0.79598	0.92055
Naive-Bayes	0.80135	0.85767	0.88909	0.79571	0.81847	0.93010
Support Vector Machine - Linear	0.79319	0.85078	0.88340	0.78381	0.82169	0.93028
Support Vector Machine - Radial	0.77468	0.83460	0.87365	0.75576	0.83213	0.93181
XGBoost	0.82842	0.87883	0.91389	0.82720	0.83213	0.93733

It seems clear that the model that performs the best is the XGBoost Algorithm, in regards to Accuracy, F1 Score and Distance to Left Point of ROC Curve. As such, the XGBoost Algorithm is trained with 10-fold cross validation repeated 5 times and, after being optimized for Accuracy, F1 and Distance to Left Corner, is tested with the final validation set.

5 Conclusion

5.1 Final model and test with Validation

Table 7: Final Models Optimized by Accuracy, F1 Score, and Distance to Left of ROC Curve

	Accuracy	F1	ROC-AUC	Sensitivity (TPR)	Specificity (TNR)	Precision
Optimized by Accuracy	0.86467	0.91236	0.9184	0.93650	0.64668	0.88943
Optimized by F1	0.86410	0.91355	0.9184	0.95467	0.58927	0.87583
Optimized by Distance to Left	0.83004	0.87967	0.9184	0.82598	0.84234	0.94083

5.2 Final Results

The best classifier all in all was the XGBoost Algorithm, achieving an accuracy of 0.865 on the final validation set. Optimized for F1, it achieves a score of approximately 0.914 and optimized for Distance to Left Corner of ROC Curve, a sensitivity and specificity higher than 0.825.

In all respects, optimized for Accuracy, F1, and Distance to Left Corner, it was by far the best performing. Not far behind was the Random Forest Algorithm with the worst-performing algorithm being the Naive-Bayes algorithm. Further improvements include more precise parameter tuning of the XGBoost algorithm and possibly

more refinement of the random forest algorithm which possibly could surpass the XGBoost. Ensemble algorithms and other machine learning algorithms not explored in this project may also be an improvement on the base XGBoost Algorithm.

5.3 Resources

- Irizarry, R. A. (2020, February 24). Introduction to Data Science. Retrieved from https://rafalab.github.io/dsbook/caret.html
- Deng, K. (n.d.). https://www.cs.cmu.edu/~kdeng/thesis/feature.pdf . Retrieved from https://www.cs.cmu.edu/~kdeng/thesis/feature.pdf
- Lemon, C. (n.d.). http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf . Retrieved from http://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf
- Habibzadeh, F., Habibzadeh, P., & Yadollahie, M. (2016, October 15). On determining the most appropriate test cut-off value: the case of tests with continuous results. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5082211/
- Kabacoff, R. (n.d.). Missing Data. Retrieved from https://www.statmethods.net/input/missingdata.html
- Hosmer, D. & Lemeshow, S. (2000). Applied Logistic Regression (Second Edition). New York: John Wiley & Sons, Inc.
- Long, J. Scott (1997). Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks, CA: Sage Publications.