# INCOME CLASSIFICATION

Aravindh Boya

## 1. INTRODUCTION

The goal of this project to predict if the persons income will be over 50k$ a year or under 50k$. This data was collected by Barry Becker from the census database of the year 1994. The dataset is available here https://www.kaggle.com/lodetomasi1995/income-classification. This is a medium sized dataset with 32561 observations and 15 attributes. Of these 15 attributes 9 are categorical and 6 are numerical.

The dataset has various information about the person like the age, sex, gender, occupation, education level, hours per week, race, native country etc, these are used to predict if the person makes above or under 50k a year.

The dataset was found at kaggle.com, a data science and machine learning competitions platform. The dataset has reasonably clean records with minimal data pre processing needed.

### Motivation:

The motivation behind this project is to build a website where an individual can enter his details like age, gender, native country, work, education etc to see if he is getting paid reasonably. In lot of countries and lot of companies many employees are getting underpaid to what they should deserve. This project helps every individual to know their worth, right now its just a classification task of which says if he should earn more than 50k or less than 50k, but in future if the exact salary data, i would regression and try to predict the closest salary amount he should be paid. This project is just the first step, i'm looking forward to extend it full scale once i find the latest income data.

**Nature of data:**

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Categorical and Numerical
- Number of Records: 32561
- Number of Attributes: 15

**Attributes Info:**

| Attribute | Information |
|---|---|
| income | The income of the person. |
| age | continuous. |
| workclass | Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. |
| fnlwgt | continuous. |
| education | Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. |

| Attribute | Information |
|---|---|
| education-num | continuous. |
| marital-status | Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. |
| occupation | Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces. |
| relationship | Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. |
| race | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. |
| sex | Female, Male. |
| capital-gain | continuous. |
| capital-loss | continuous. |
| hours-per-week | continuous. |
| native-country | United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad & Tobago, Peru, Hong, Holand-Netherlands. |

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(smotefamily)
```

```
## Warning: package 'smotefamily' was built under R version 4.0.2
```

```
library("RColorBrewer")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(class)
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.2
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.2

## Loaded gbm 2.1.8
```

```
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.0.2

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.20
```

```
df = read.csv('C:/Users/aravi/OneDrive/Documents/special_topics/Datasets/income-evaluation.xls')

cat("The number of Null values in dataset is", sum(is.na(df)))
```

```
## The number of Null values in dataset is 0
```

```
summary(df)
```

```
##       age          workclass             fnlwgt          education
##  Min.   :17.00   Length:32561       Min.   :  12285   Length:32561
##  1st Qu.:28.00   Class :character   1st Qu.: 117827   Class :character
##  Median :37.00   Mode  :character   Median : 178356   Mode  :character
##  Mean   :38.58                      Mean   : 189778
##  3rd Qu.:48.00                      3rd Qu.: 237051
##  Max.   :90.00                      Max.   :1484705
##  education.num   marital.status      occupation        relationship
##  Min.   : 1.00   Length:32561       Length:32561       Length:32561
##  1st Qu.: 9.00   Class :character   Class :character   Class :character
##  Median :10.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :10.08
##  3rd Qu.:12.00
##  Max.   :16.00
##      race               sex             capital.gain    capital.loss
##  Length:32561       Length:32561       Min.   :    0   Min.   :   0.0
##  Class :character   Class :character   1st Qu.:    0   1st Qu.:   0.0
##  Mode  :character   Mode  :character   Median :    0   Median :   0.0
##                                        Mean   : 1078   Mean   :  87.3
##                                        3rd Qu.:    0   3rd Qu.:   0.0
##                                        Max.   :99999   Max.   :4356.0
##  hours.per.week  native.country        income
##  Min.   : 1.00   Length:32561       Length:32561
##  1st Qu.:40.00   Class :character   Class :character
##  Median :40.00   Mode  :character   Mode  :character
##  Mean   :40.44
##  3rd Qu.:45.00
##  Max.   :99.00
```

```
df$income = as.factor(df$income)
df$income = as.numeric(df$income)

df$income = df$income - 1

table(df$income)
```

### Data Preprocessing

```
##
##     0     1
## 24720  7841
```

# 2. METHODS

- Decision Tree
- Random Forests
- Support Vector Machine
- Boosting

```
###### set seed to ensure you always have same random numbers generated
set.seed(22)

train_ind = sample(seq_len(nrow(df)),size = 0.7 * nrow(df))
###### creates the training dataset with row numbers stored in train_ind
train =df[train_ind,]
test=df[-train_ind,]

table(train$income)
```

```
##
##     0     1
## 17253  5539
```

- **Split the data:** To address our problem, Above I have divided the data set into two samples, Training(70%) and Testing(30%). In our case we use for example 200 observations in the train data, 86 in the test data.

```
chisq.test(df$income, df$age)
chisq.test(df$income, df$workclass)
chisq.test(df$income, df$fnlwgt)
chisq.test(df$income, df$education)
chisq.test(df$income, df$education.num)
chisq.test(df$income, df$marital.status)
chisq.test(df$income, df$occupation)
chisq.test(df$income, df$relationship)
chisq.test(df$income, df$race)
chisq.test(df$income, df$sex)
chisq.test(df$income, df$capital.gain)
chisq.test(df$income, df$capital.loss)
chisq.test(df$income, df$native.country)
chisq.test(df$income, df$hours.per.week)
```

- **Dependent Variables:**

By using chi-squared test we came to know that dependent variables of income are all the other variables in the dataset except fnlwgt.The fnlwgt which is the final weight determined by the Census Organization is of no use in any of the analysis that we are doing henceforth and is removed. The educationnum if a repetitive variable which recodes the categorical variable education as a numeric variable but will be used in the analysis for decision trees, hence is not being removed.

- **Analysis:**

All the above classifiers are modeling the probability that an individual makes more than 50K annually. In another word, a response closer to 1 indicates higher chance of making over 50K while a response closer to 0

indicates a higher chance of making less than 50K. Thus, a threshold of 0.5 is used to determine whether an individual is predicted to make more than 50K annually or not. A confusion matrix is presented to evaluate how well the model predicts income.

## Classification Techniques

**Decision Tree** A decision tree classifies by choosing a threshold on a feature and splits the data according to a 'splitting rule'. Since the features need to be numerical, we had to discard certain features and change how we represented others. For example, we could not convert native.country into numerical values since this would cause an implicit feature ranking skewing our results.However, education is a feature that can be converted into a numerical value, as a certain level of education can be higher or lower than others in rank. For this reason, we chose only to consider limited attributes (This is represented as a binary feature with 1 being male and 0 being female). The tree is then built on the training set and used to predict the binary value of the label (whether or not an individual makes more that $50,000) on the test set.

```
decision_tree_fit <- rpart(income ~  workclass + education + marital.status  + occupation + relationshi
   method="class", data = train)

dt_predictions <-predict(decision_tree_fit, test, type = 'class')
```

**Random Forests** Random Forests is a supervised machine learning algorithm. It basically contains a large number of decision trees and are trained using the bagging method. It can be used for both regression and classification and its fairly easy to understand and implement, random forests generally perform better than most of the models. The random forests model takes random subset of features and builds the tree based on those thus adding the randomness to the model which results in a robust, diverse and generalized model. It has almost the same hyperparameters as decision trees. The best thing about the random forest models is it doesnt overfit, and the problem with random forest is that they can be very slow and time taking when the number of trees are large in number.

```
train$income = as.factor(train$income)


random_forest_fit <- randomForest(income ~  workclass + education + marital.status  + occupation + rela
    data = train)

rf_predictions <-predict(random_forest_fit, test)
```

**Support Vector Machine** Support Vector Machines also called as SVM is a supervised machine learning model. SVM objective is to find the hyperplane in the N-Dimensional space that can classify the data distinctly. The hyper plane is chosen such that they have maximum margin. SVM can be used for classification, regression and do detect the outliers. They are effective when the number of dimensions is high, they also are versatile as they have different types of kernels and you can also create custom kernels. The problem with the SVM is it can be very slow if the dataset is large.

```
svm_fit <- svm(income ~  workclass + education + marital.status  + occupation + relationship + race, da

svm_predictions <-predict(svm_fit, test)
```

**Boosting** Boosting algorithms are the ensembling models and they are set of algorithms that converts the weak learners to the strong learners. Boosting algorithms work by training the weak learners sequentially

and each of them trying to correct the predecessor. Boosting algorithms are generally used in top machine learning competitions and are generally better performing models than other ML models, its results are often compared to that of deep learning models. There are many types of boosting models like adaBoost, gradient boosting, lightgbm, xgboost etc.

```
train$workclass = as.factor(train$workclass)
train$education = as.factor(train$education)
train$marital.status = as.factor(train$marital.status)
train$occupation = as.factor(train$occupation)
train$relationship = as.factor(train$relationship)
train$race = as.factor(train$race)

test$workclass = as.factor(test$workclass)
test$education = as.factor(test$education)
test$marital.status = as.factor(test$marital.status)
test$occupation = as.factor(test$occupation)
test$relationship = as.factor(test$relationship)
test$race = as.factor(test$race)

train$income = as.character(train$income)

test$income = as.character(test$income)

gbm_fit <- gbm(income ~  workclass + education + marital.status  + occupation + relationship + race, da

gbm_predictions_prob<-predict(gbm_fit,newdata=test,n.trees=2000,type="response")


gbm_predictions <- levels(as.factor(train$income))[1+(gbm_predictions_prob>0.5)]
```

# 3.RESULTS

### Accuracy

Accuracy is one of the most common metrics in measuring the classification models performance. It is defined as the ratio of number of correct predictions over the total predictions made. So, obviously the more the accuracy is the better the model. We should always evaluate our models performance on the test data but on the train data since the model is built on the train data it usually performs better on the data it has seen, but test data is something which the model has not seen so we can trust the test data's metrics.

Accuracy = Number of correct predictions / Total number of predictions made

### Confusion Matrix:

We will use confusion matrix as our second metric as just in case if the accuracies of two models are relatively same we look at confusion matrix to identify the false negatives and false positives. Here in our case the primary goal is to predict if the person makes above 50k false positives and false negatives helps us in deciding the best model.

Confusion matrix as the name says can be really confusing to understand, it is the summary of the predictions where the correct and incorrect classifications are summarized. It is has 4 elements

- True positive - Observation is positive, and is predicted to be positive.
- True Negative - Observation is negative, and is predicted to be negative.

- False Positive - Observation is negative, but is predicted positive.
- False Negative - Observation is positive, but is predicted negative.

Here the goal for our project is to show that most number false positive through confusion matrix that means people has the income more than 50K$.

### Decision Tree

The following are the results of the Decision Tree Analysis. The accuracy using this model is **81.94%**. The sensitivity is 94.78% and the specificity is 40.31%. Here the specificity is little lower when compared to all other methods. The kappa rate is just over 40% so the error rate is quite low.

```
dt_cm = confusionMatrix(as.factor(dt_predictions), as.factor(test$income))
dt_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7077 1374
##          1  390  928
##
##                Accuracy : 0.8194
##                  95% CI : (0.8117, 0.827)
##     No Information Rate : 0.7644
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4118
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9478
##             Specificity : 0.4031
##          Pos Pred Value : 0.8374
##          Neg Pred Value : 0.7041
##              Prevalence : 0.7644
##          Detection Rate : 0.7244
##    Detection Prevalence : 0.8651
##       Balanced Accuracy : 0.6754
##
##        'Positive' Class : 0
##
```

```
dt_accuracy <- dt_cm$overall[1]

cat("The Decision Tree accuracy is", dt_accuracy)
```

```
## The Decision Tree accuracy is 0.8194288
```

### Random Forest

The following are the results of the Random Forest Analysis. The accuracy using this model is **83.25%**. The recall rate is 92% which is pretty high that mean all the positive values are correctly identified and

specificity is 54% which is pretty decent that mean above 50% of the negative values in the dataset are correctly identified by the model.

```
rf_cm = confusionMatrix(as.factor(rf_predictions), as.factor(test$income))
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6870 1039
##          1  597 1263
##
##                Accuracy : 0.8325
##                  95% CI : (0.825, 0.8399)
##     No Information Rate : 0.7644
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.502
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9200
##             Specificity : 0.5487
##          Pos Pred Value : 0.8686
##          Neg Pred Value : 0.6790
##              Prevalence : 0.7644
##          Detection Rate : 0.7032
##    Detection Prevalence : 0.8096
##       Balanced Accuracy : 0.7344
##
##        'Positive' Class : 0
##
```

```
rf_accuracy <- rf_cm$overall[1]

cat("The Random Forest accuracy is", rf_accuracy)
```

```
## The Random Forest accuracy is 0.8325315
```

### Support Vector Machine

The following are the results of the Support Vector Machine Analysis. The accuracy using this model is **82.64%**. Here the SVM records the highest negative predict value than any other method which is 68.43%.

```
svm_cm = confusionMatrix(as.factor(svm_predictions), as.factor(test$income))
svm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    0    1
##          0 6948 1177
##          1  519 1125
##
##                Accuracy : 0.8264
##                  95% CI : (0.8187, 0.8339)
##     No Information Rate : 0.7644
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4652
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9305
##             Specificity : 0.4887
##          Pos Pred Value : 0.8551
##          Neg Pred Value : 0.6843
##              Prevalence : 0.7644
##          Detection Rate : 0.7112
##    Detection Prevalence : 0.8317
##       Balanced Accuracy : 0.7096
##
##        'Positive' Class : 0
##
```

```
svm_accuracy <- svm_cm$overall[1]

cat("The SVM accuracy is", svm_accuracy)
```

```
## The SVM accuracy is 0.8263896
```

### Boosting

The following are the results of the GBM Analysis. The accuracy using this model is **82.91%**. This particular model records the highest number of "false negative" elements i.e 634 (type one errors) which is our main goal to predict with the minimal amount of the error rate.

```
gbm_cm = confusionMatrix(as.factor(gbm_predictions), as.factor(test$income))
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6846 1037
##          1  621 1265
##
##                Accuracy : 0.8303
##                  95% CI : (0.8227, 0.8377)
##     No Information Rate : 0.7644
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.4974
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9168
##             Specificity : 0.5495
##          Pos Pred Value : 0.8685
##          Neg Pred Value : 0.6707
##              Prevalence : 0.7644
##          Detection Rate : 0.7008
##    Detection Prevalence : 0.8069
##       Balanced Accuracy : 0.7332
##
##        'Positive' Class : 0
##
```

```
gbm_accuracy <- gbm_cm$overall[1]

cat("The GBM accuracy is", gbm_accuracy)
```

```
## The GBM accuracy is 0.8302795
```

**conclusion for most suitable method**

After considering the results all the methods implemented on this dataset, The most suitable method is ought to be ***GradientBoosting***. The stats for the Boosting are 82.91% Accuracy, Coefficient interval is 82.14% and 83.65% for 0 and 1 respectively and seems to be there is no class imbalance in this model. The positive predictive value is greater than 86%. Considering all these stats Random Forests model is said to be best suitable model fitted to these dataset.

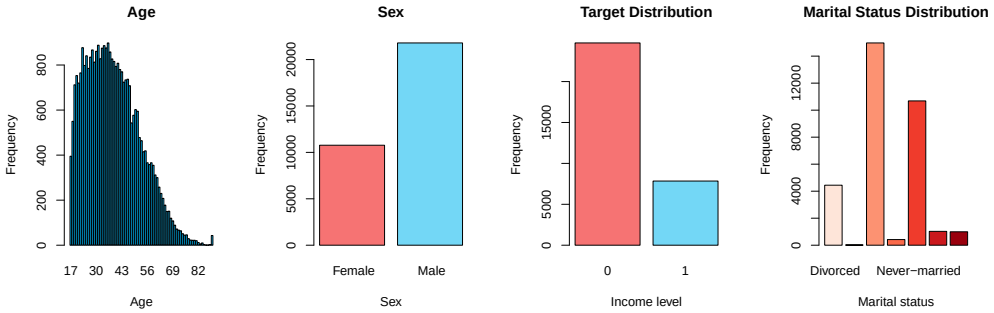
**Challenges faced:**

The dataset consists of most of the categorical data rather than the numerical data. So, applying the PCA and Clustering methods on this dataset is way too hard and very much challenging that made me to skip the implementation of those methods totally. Another challenge was constructing the plots for the SVM method. Apart from that i really did not face any other critical issues as the dataset is pretty clean and not much pre-processing is required.

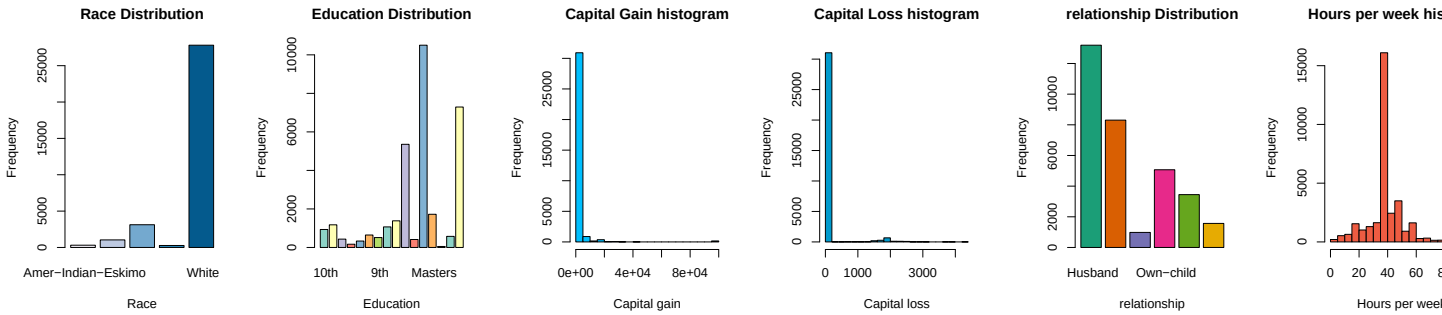
## Data Visualizations

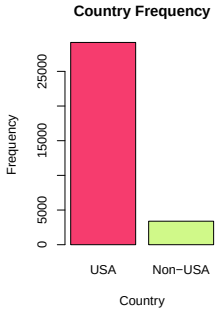
```
par(mfrow=c(1,2))

barplot(table(df$age), xlab = 'Age', ylab = 'Frequency', main = 'Age', col= "deepskyblue1")

barplot(table(as.factor(df$sex)), xlab = 'Sex', ylab = 'Frequency', main = 'Sex', col = c("#F67373", "#

barplot(table(as.factor(df$income)), xlab = 'Income level', ylab = 'Frequency', main = 'Target Distribu

barplot(table(as.factor(df$marital.status)), xlab = 'Marital status', ylab = 'Frequency', main = 'Marita
```

```
barplot(table(as.factor(df$race)), xlab = 'Race', ylab = 'Frequency', main = 'Race Distribution', col =

barplot(table(as.factor(df$education)), xlab = 'Education', ylab = 'Frequency', main = 'Education Distri

hist(df$capital.gain, xlab = 'Capital gain', ylab = 'Frequency', main = 'Capital Gain histogram', col =

hist(df$capital.loss, xlab = 'Capital loss', ylab = 'Frequency', main = 'Capital Loss histogram', col =

barplot(table(as.factor(df$relationship)), xlab = 'relationship', ylab = 'Frequency', main = 'relationsh

hist(df$hours.per.week, xlab = 'Hours per week', ylab = 'Frequency', main = 'Hours per week histogram',

country_names = c('USA','Non-USA')
country_freq = c(dim(df %>%filter(as.integer(as.factor(native.country)) %in% c(40)))[1], dim(df %>%filt

barplot(country_freq, main = "Country Frequency", xlab = "Country", ylab = 'Frequency', names.arg = cou
```

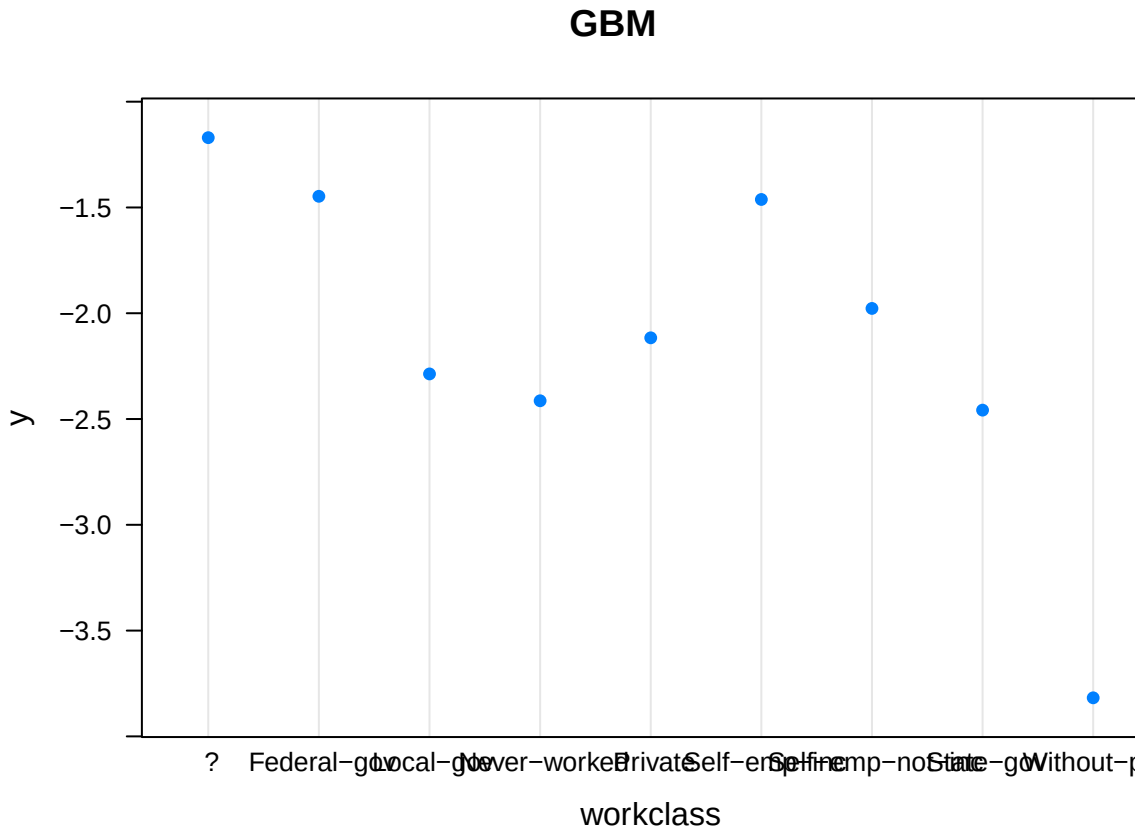



## Plots for the Data set



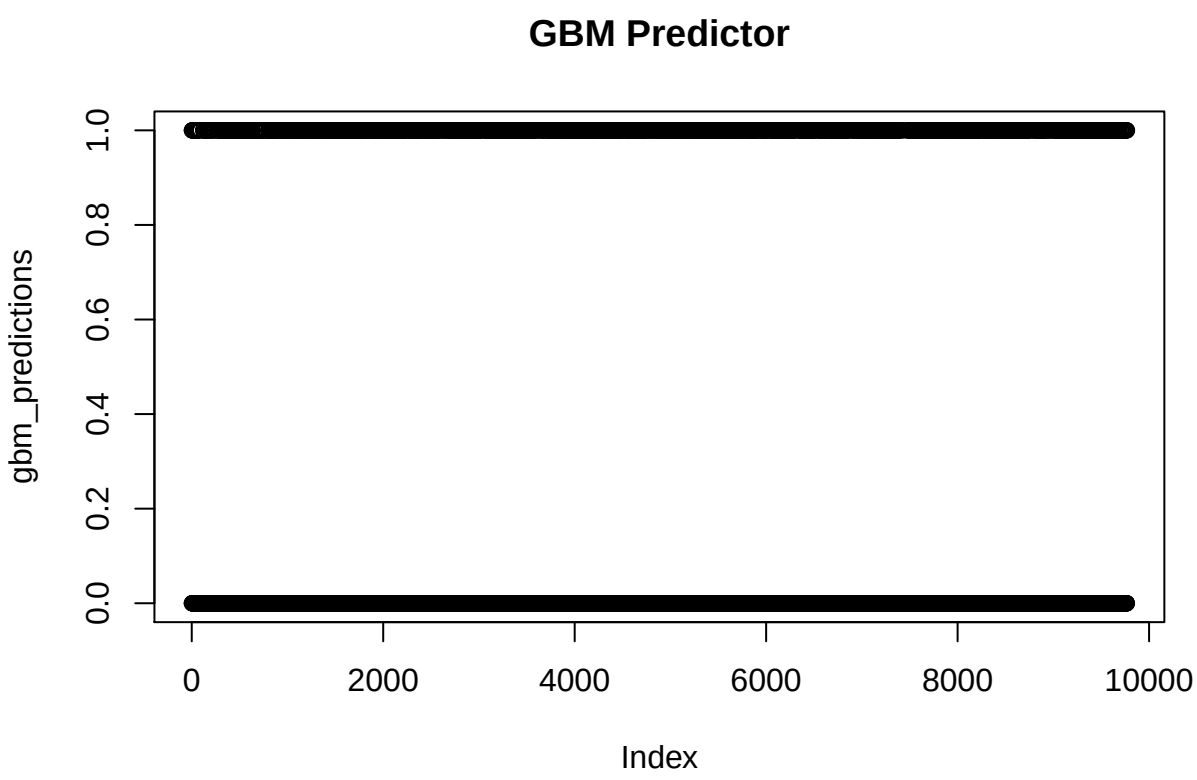



```
plot(gbm_fit ,i="workclass", main = "GBM")
```

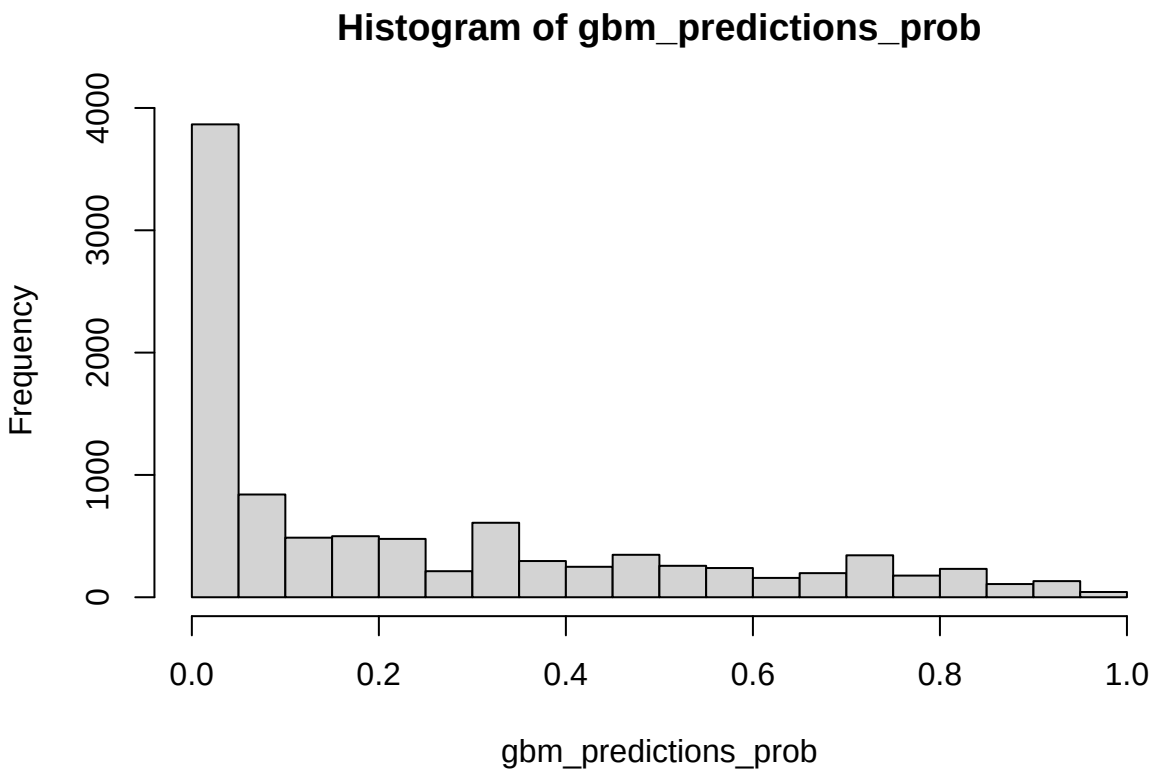



**Plots for the Results**

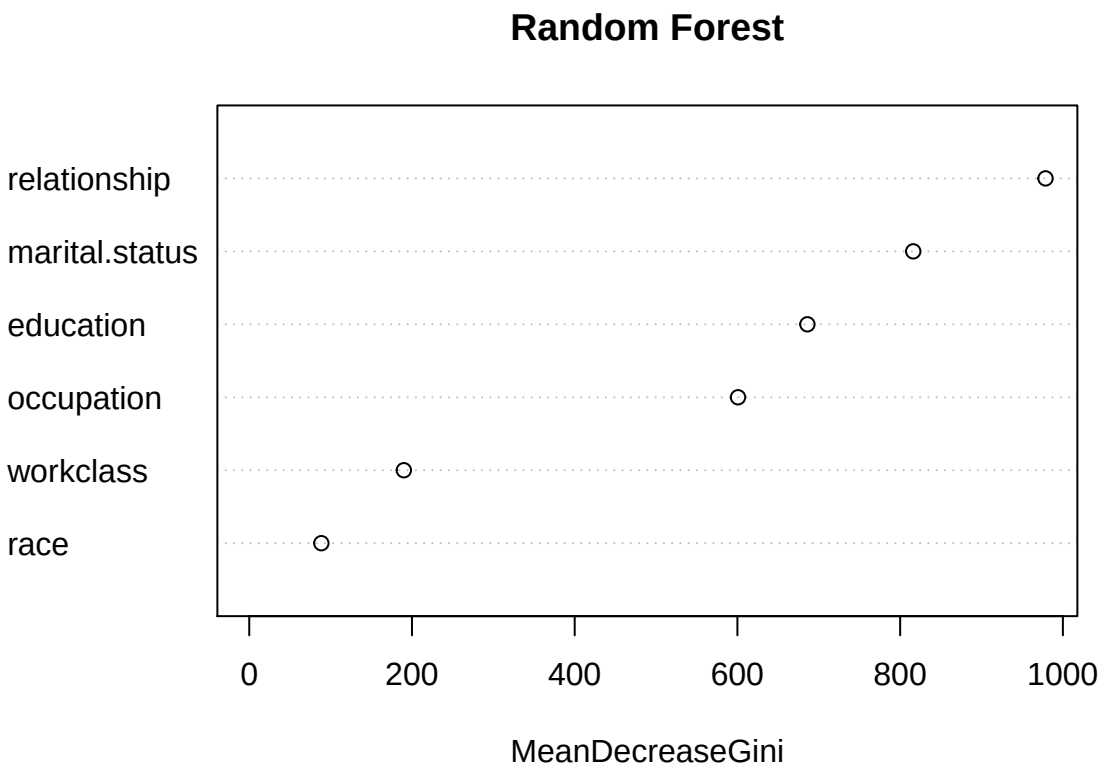
```
plot(gbm_predictions, main = "GBM Predictor")
```



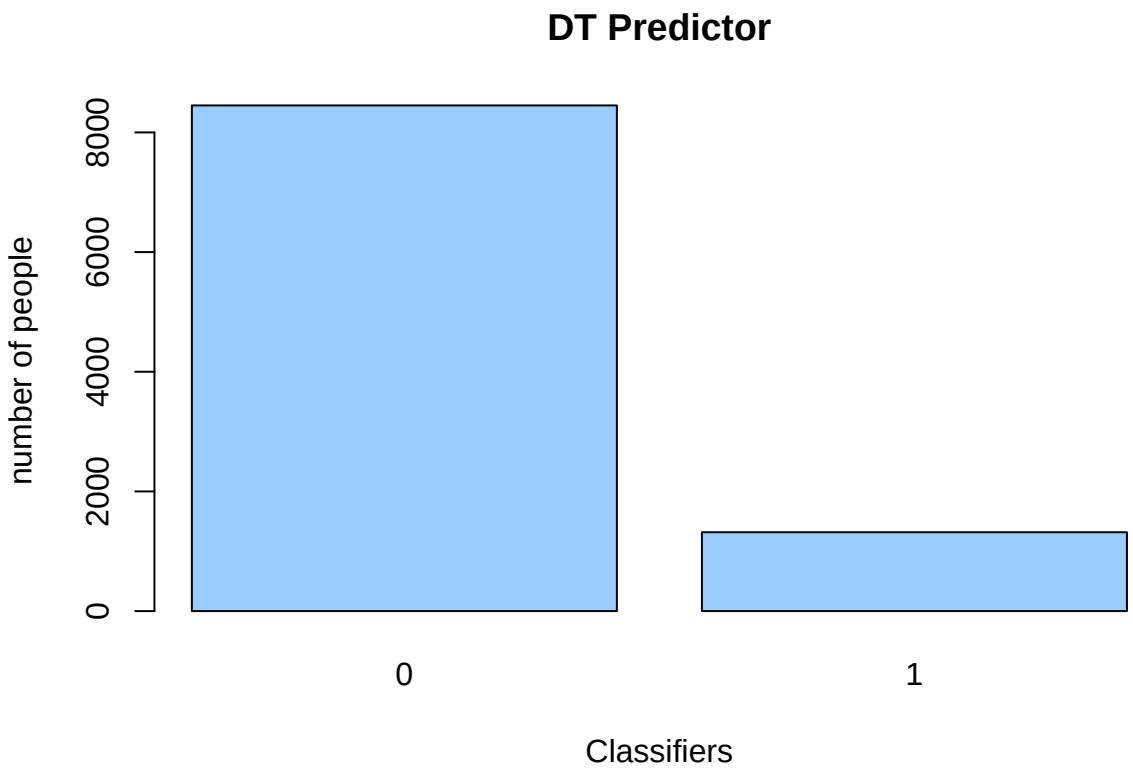

```
hist(gbm_predictions_prob)
```



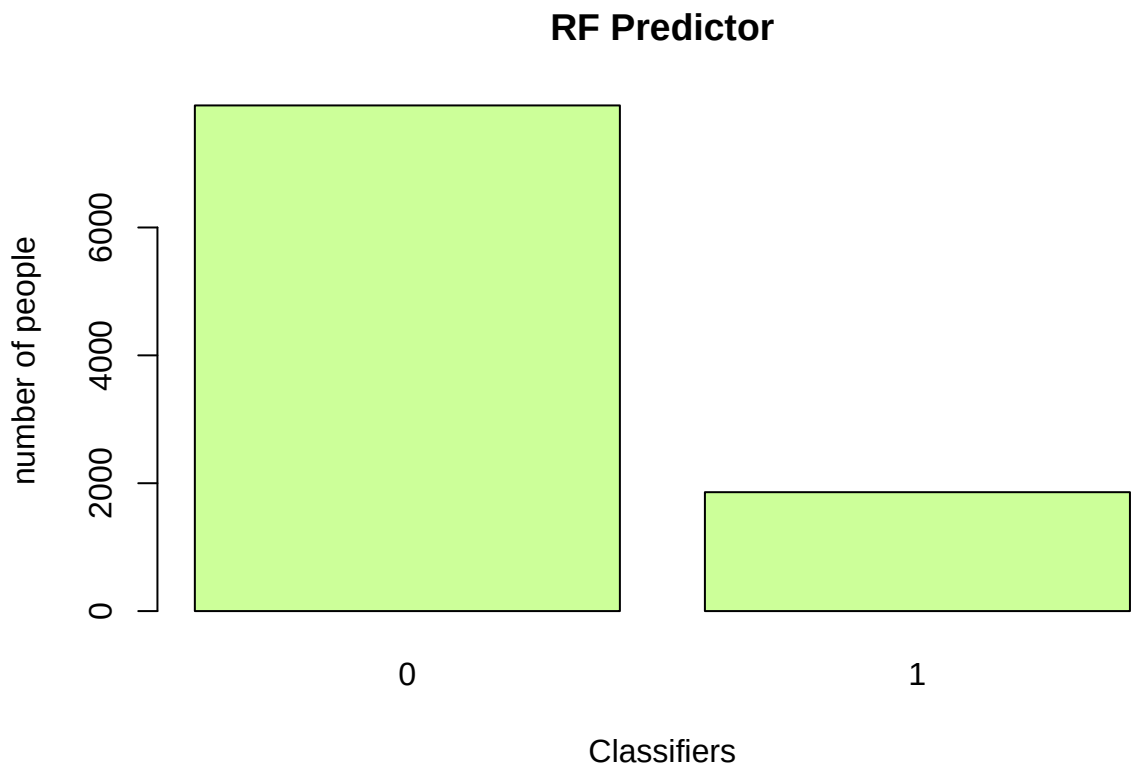

```
varImpPlot (random_forest_fit, main = "Random Forest")
```



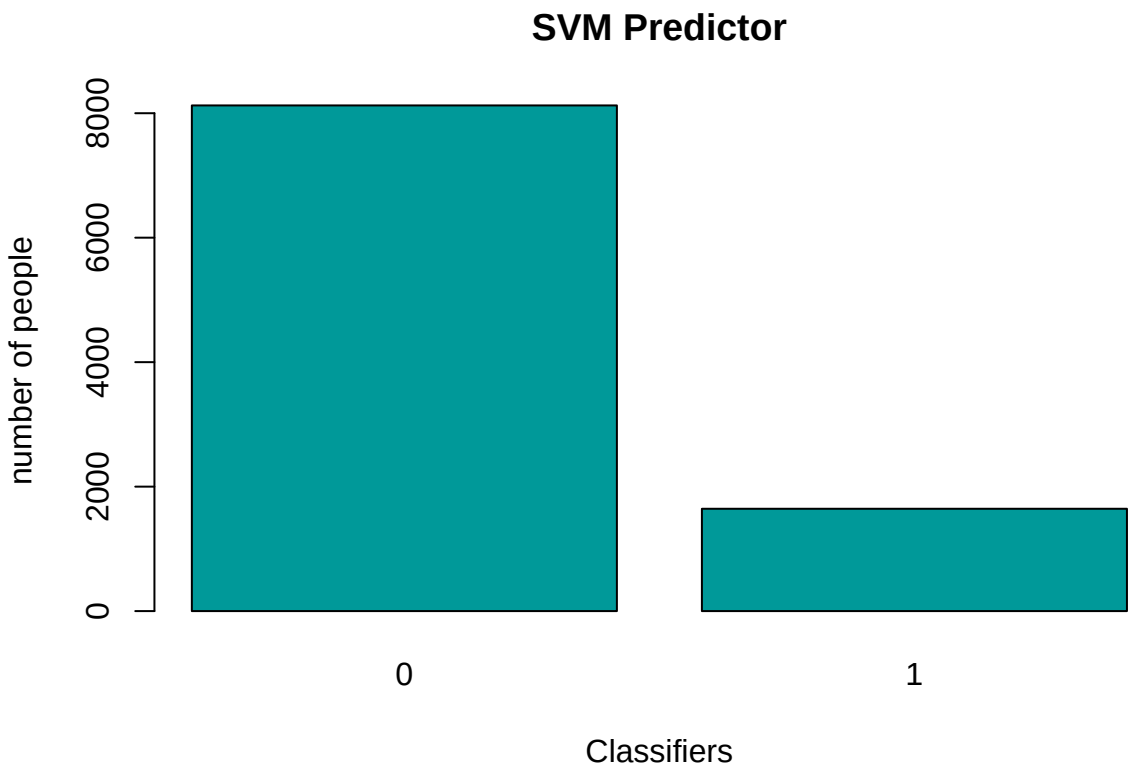

```
plot(dt_predictions, xlab = "Classifiers", ylab = "number of people", main = "DT Predictor", col = "#99
```



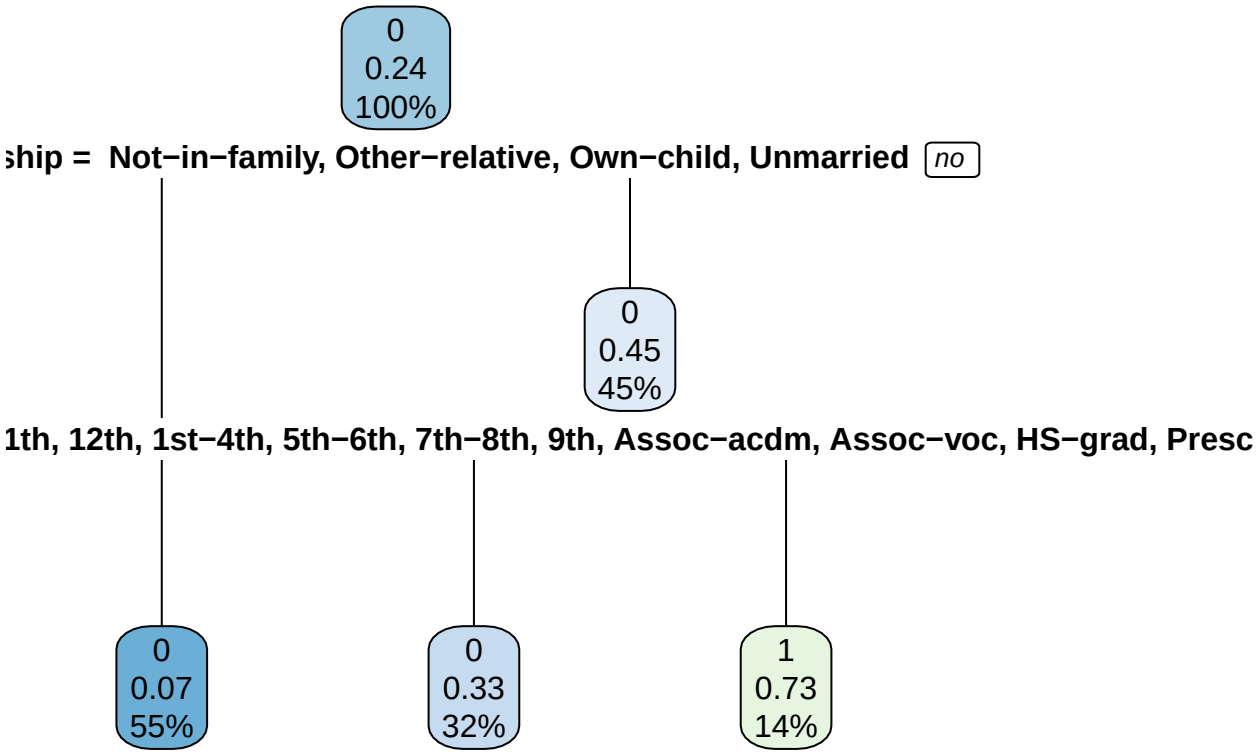

```
plot(rf_predictions, xlab = "Classifiers", ylab = "number of people", main = "RF Predictor", col = "#cc
```



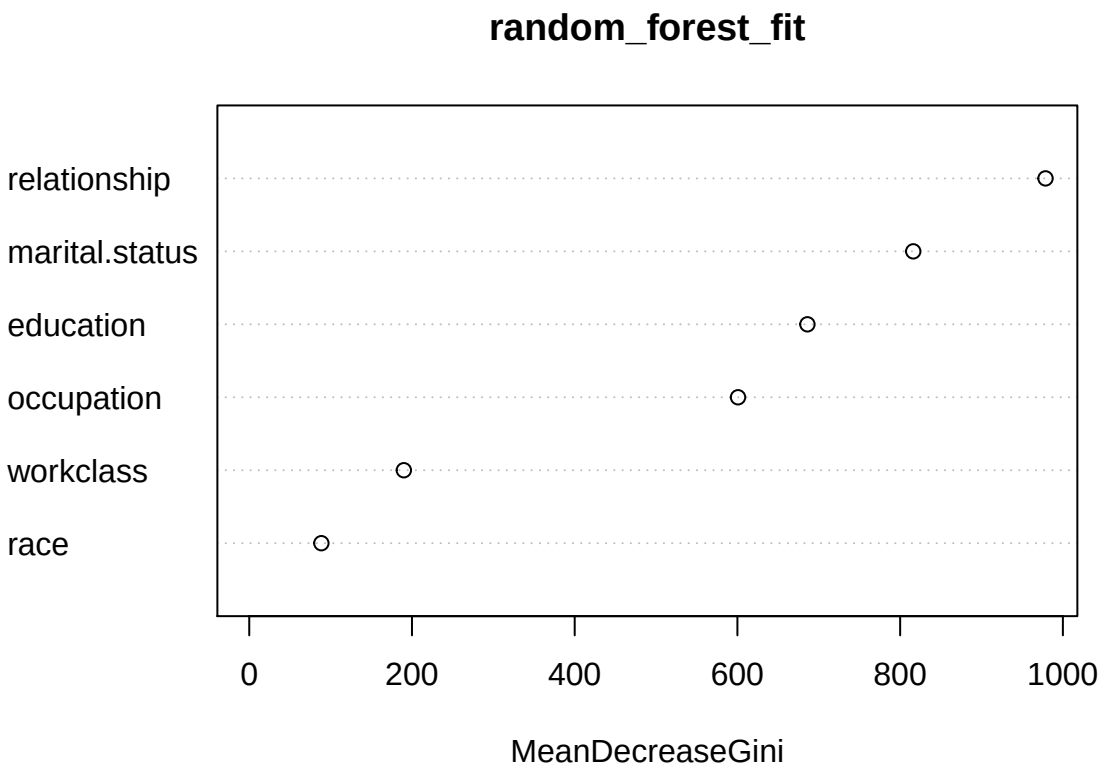

```
plot(svm_predictions, xlab = "Classifiers", ylab = "number of people", main = "SVM Predictor", col = "#
```



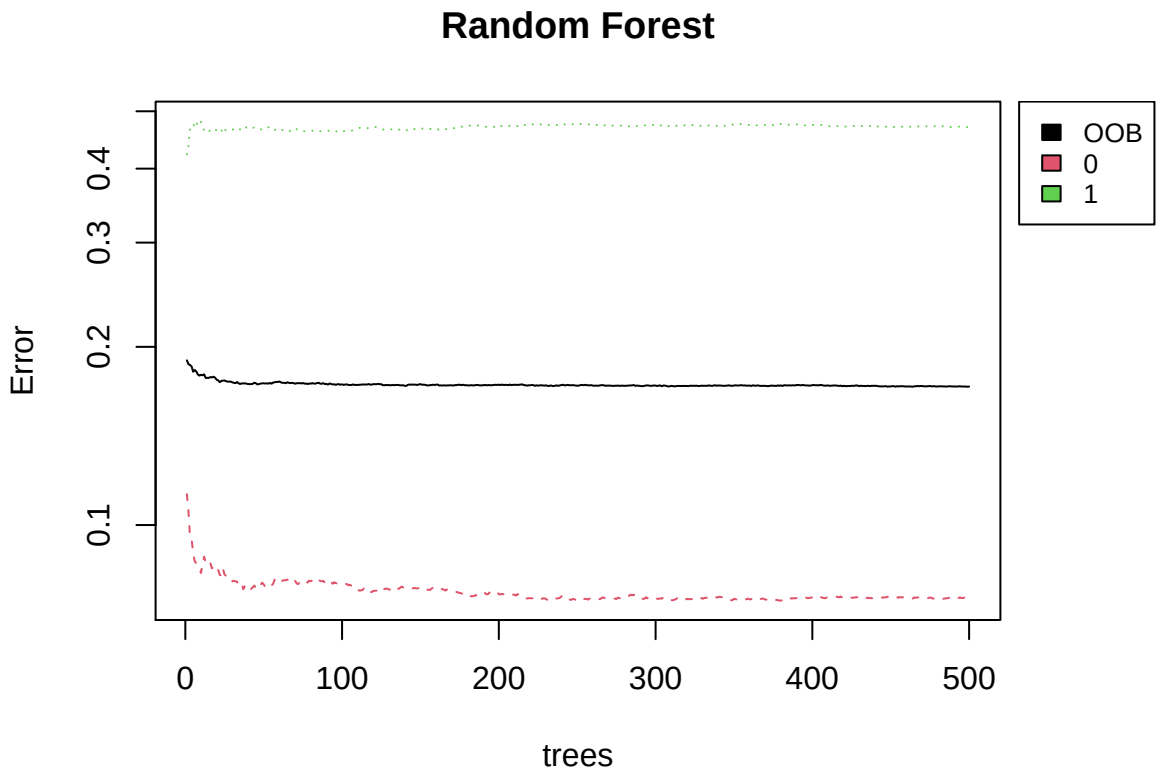

```
rpart.plot(decision_tree_fit)
```




```
##cat("The feature importances of predictors of random forest model")

varImpPlot(random_forest_fit, scale = FALSE)
```




```
layout(matrix(c(1,2),nrow=1),
       width=c(4,1))
par(mar=c(5,4,4,0))
plot(random_forest_fit ,log = "y", main = "Random Forest")
par(mar=c(5,0,4,2))
plot(c(0,1),type="n", axes=F, xlab="", ylab="")
legend("top", colnames(random_forest_fit$err.rate),col=1:3,cex=0.8,fill=1:3)
```




# 4. DISCUSSION

In this particular scenario the Gradient boosting and the Random Forests has out performed the other two classifiers. There was an huge debate to choose the best suitable method fitted to the dataset between GBM and RF. when it come to statistics of each model are fitted exceptinally to the data set but main purpose of the project is satisfied in the GBM. Though GBM takes a quite more time for the implementation but for this data set it can be negligible as it has very less number of observations.

**Might any of the methods we discussed in Summer I provide better performance on your dataset? Why?** Yes, to my surprise the Logistic Regression model provides an almost and slight better score than the models we used in Summer II. The main reason i think behind this is the size of the dataset, given the size of the dataset is small, logistic regression works better than tree models which often performs well on datasets with large size. The other reason might be data has good separability which the logit curve fits smooth. So, the logistic regression has slight better score because of the above mentioned two reasons.

**Logistic Regression** A logistic regression model will first be developed to compare the performance of various machine learning algorithms and is the most common classification model, though it has regression in its name. It is a statistical model which uses the logistic(sigmoid) function to model the binary dependent variable. It is used to describe the data and to understand the relationships between one dependent binary variable to the one or more independent variables.

```
logit_model = glm(
  as.factor(income) ~ workclass + education + marital.status  + occupation + relationship + race,
```

```
  data = train,
  family = "binomial"
)

logit_pred = predict(logit_model,newdata = test, type = "response")

logit_pred = ifelse(logit_pred > 0.5, 1, 0)

lr_cm = confusionMatrix(as.factor(logit_pred), as.factor(test$income))
lr_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6910 1091
##          1  557 1211
##
##                Accuracy : 0.8313
##                  95% CI : (0.8237, 0.8387)
##     No Information Rate : 0.7644
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4908
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9254
##             Specificity : 0.5261
##          Pos Pred Value : 0.8636
##          Neg Pred Value : 0.6850
##              Prevalence : 0.7644
##          Detection Rate : 0.7073
##    Detection Prevalence : 0.8190
##       Balanced Accuracy : 0.7257
##
##        'Positive' Class : 0
##
```

```
lr_accuracy <- lr_cm$overall[1]

cat("The Logistic Regression accuracy is", lr_accuracy)
```

```
## The Logistic Regression accuracy is 0.8313031
```

# 5. REFERENCES CITED

[1] http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf [2] https://www.kaggle.com/uciml/adult-census-income [3] https://www.rdocumentation.org/packages [4] https://rpubs.com/chidungkt, by Nguyen Chi Dung