H-1B visa Eligibility Predictions

Raquel Buezo	A20410771
Pablo Medrano	A20410758

Table of Contents

1. Introduction 3
2. Related Work 4
3. Approach 5
4. Results 6
5. Conclusions 6
6. References 8
7. Appendix: R code 9

1. Introduction

For this project, the problem that is trying to be solved is predicting whether an individual applying for an H-1B visa in the United States is eligible or not.

The H-1B visa permits foreigners working in specialty occupations to enter the country and stay with a non-immigrant status.

The application process for this highly desired visa is long and thorough and consists of the following steps [1]:

1. The U.S. employer must file the H-1B petition on behalf of the worker.
2. The prevailing and actual wages must be confirmed by the State Employment Security Agency.
3. File the Labor Condition Application (LCA).
4. Prepare the petition and file it at a USCIS office.
5. Wait for results.

The dataset that was analyzed for this project contains information since 2011 to 2016 about the H-1B petitions on the third step of the application process. It includes the result of the LCA prior to the lottery handled by USCIS.

The CASE_STATUS field of the dataset denotes this result and determines which applications can be filed and further processed for H-1B approval. The possible outcomes for CASE_STATUS in this third step are:

- Certified: The employer filed the LCA and it was approved by the Department of Labor (DOL).
- Certified – Withdrawn: The LCA was approved but later withdrawn by the employer.
- Withdrawn: The employer withdrew the LCA before approval.
- Denied: The LCA was denied by the DOL.

The rest of the attributes considered in the dataset are [2]:

- **CASE_NUMBER**
- **CASE STATUS**: If the petition was certified, denied or withdrawn.
- **EMPLOYER NAME**: Name of the company.
- **SOC NAME**: Occupational name associated with a code.
- **JOB TITLE**: What was the position held by the requester.
- **FULL TIME POSITION**: If position was full time or not.
- **PREVAILING WAGE**: salary.
- **YEAR**: What year was the petition made.
- **WORKSITE**: Where the person was working.

- **LON**: Longitude coordinate of the worksite.
- **LAT**: Latitude coordinate of the worksite

The reason this subject is of interest is that H-1B visas are the most common for highly skilled foreign workers and their demand had been increasing since 2013. Although this last year the number of received applications has decreased, it is still significantly over 85,000 which is the number of annually granted visas.

2. Related Work

The dataset is called "H-1B Visa Petitions 2011-2016" was obtained from Kaggle. Also on Kaggle and related to this dataset, there was a project titled "Likelihood of being eligible to file H-1B visa" [3].

This Kaggle solution attempted to build a model similar to the one proposed later in this report.

The steps that were followed were:

1. **Preliminary data processing.** This solution proposed keeping only the applications from 2016 where CASE STATUS was either CERTIFIED or DENIED. Then, rows with missing values were deleted. This solution proposed using the following predictors:
   a. Full time position
   b. Prevailing wage
   c. State, which was obtained from worksite, and kept only those with a high count (greater than 2,000).
   d. Occupation, which was obtained from SOC_NAME. The occupations considered were engineer, manager, technician, teacher, executive and accountant.

2. **Model building.** The data was divided into two training and test sets where the training set contained an 80% of the data and the remaining 20% was used for the test set. Highly correlated independent variables were removed. The methodologies that were used were Logistic Regression, Decision Tree and Boosting.

3. **Performance measures.** Performance of each method was measured by using the results of the Confusion Matrix and the ROC curve and AUC (Area Under the Curve) were used. Comparing these three modelling methods for this case, the best results were obtained by the decision tree.

3. Approach

The objective of this project was to create a model able to predict an individual's eligibility to file for an H-1B visa based on the data from previous applications.

This problem is a classification problem with and outcome of '1' for eligible or '0' for non-eligible. It was considered that CASE STATUS equal to CERTIFIED or CERTIFIED – WITHDRAWN resulted in an eligible outcome since they both imply that the LCA was approved, and all else was considered to give a non-eligible outcome.

The initial approach was to take into account all the features considered relevant for classifying applications into eligible or non-eligible. Afterwards, some simpler models with less number of features were considered to see the difference in performance by using these more "intuitive" predictors.

The hypothesis before carrying out the analysis was that PREVAILING WAGE and FULL TIME POSITION would be the most significant predictors. Since higher wages and full-time positions generally imply higher skilled and specialized jobs, it was expected that applications with these characteristics would more likely be eligible.

The process that was carried out for this analysis consisted of the following phases:

1. **Defining the objective.** Although maximum accuracy was a goal, it was considered more important to reduce the number of false positives since the H1-B visa application process is very scrutinous and visas are not easily obtained. The performance factors that were used to compare each of the methodologies were:
    - Specificity
    - ROC curve and AUC
    - Accuracy
2. **Preprocessing data and feature engineering.** Firstly, applications with missing data were eliminated. Secondly, outliers such as cases where CASE STATUS was REJECTED or INVALIDATED were also deleted from the dataset.

   Features such as CASE NUMBER, LONGITUDE and LATITUDE were not considered significant to the analysis and therefore eliminated.

   SOC NAME was used to create a predictor called occupation. Analyzing the data, 19 different major occupations were taken into account for this analysis.

3. **Building models.** Before building any models, it was decided to divide the data into testing and training set. The criteria followed was to use records from years 2014 and 2015 for the training set and those from 2016 as the test set.

   The baseline to which we compared our results was the "most frequent class" applied to our training data where the accuracy obtained was 98%.

   The predictive models chosen for fitting the model were Logistic Regression, LDA, QDA and Decision Trees.

4. Results

The first results that were obtained corresponded to the “default” threshold of 50% and can be summarized in the following table:

	LOGISTIC REGRESSION	TREE	LDA	QDA
ACCURACY	0.9881	0.9881	0.9875	0.7756
SPECIFICITY	0.0076	0.01179	0.01567	0.36233

Considering the objective mentioned earlier, which was to reduce the number of false positives, the 0.5 threshold was not considered appropriate and therefore it was increased.

After several tests, it was observed that as the threshold increased, the specificity increased as did the AUC. However, the accuracy obtained decreased, resulting in accuracies lower than the baseline (98%). The optimum threshold chosen was then 0.97 and from the results summarized in the table, it was concluded that the best algorithm for this case is Logistic Regression with an 85.94% accuracy, a specificity of 0.33 and an AUC of 0.598.

	LOGISTIC REGRESSION		TREE	
threshold	0.5	0.97	0.5	0.97
accuracy	0.9881	0.8594	0.9881	0.8352
specificity	0.0076	0.3302	0.01179	0.3274
AUC	0.5038	0.598	0.5058	0.584

Next, the best model (Logistic Regression with a threshold of 0.97) was applied to test the initial hypothesis that the most important predictors were PREVAILING WAGE and FULL TIME POSITION. The results that were obtained were much worse in terms of accuracy, they had a higher specificity but a lower AUC.

	Logistic Regression
accuracy	0.4742
specificity	0.53216
AUC	0.5028195

5. Conclusions

The model with the best performance for the chosen dataset is Logistic Regression with an 85.94% accuracy, a specificity of 0.33 and an AUC of 0.598. Although the accuracy is lower than the baseline, it is due to a high specificity which was the main objective of this project.

The four predictors used to calculate this model were prevailing wage, occupation, state, and full-time position.

Another reasonable algorithm that could have been used is SVM. However, due to the size of the dataset it resulted to be too complex computationally.

6. References

[1] Retrieved from http://www.immi-usa.com/h1b-application-process-step-by-step-guide/.

[2] Retrieved from https://www.foreignlaborcert.doleta.gov/docs/Performance_Data/Disclosure/FY15-FY16/H-1B_FY16_Record_Layout.pdf

[3] *Kaggle*. Retrieved from https://www.kaggle.com/gskhurana/h1b-visa-prediction.

7. Appendix: R code

libraries

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
library(dplyr)
```

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

```
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(tidyr)
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Loading required package: RSQLite
```

```
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

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

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```r
library(car)
```

```
## Warning: package 'car' was built under R version 3.4.3
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(e1071)
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```r
library(class)
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(readr)
library(tree)
library(readr)
h1b_kaggle <- read_csv("/Users/Pablo/Documents/IIT/Machine Learning/Project/h1b_kaggl
e.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   CASE_STATUS = col_character(),
##   EMPLOYER_NAME = col_character(),
##   SOC_NAME = col_character(),
##   JOB_TITLE = col_character(),
##   FULL_TIME_POSITION = col_character(),
##   PREVAILING_WAGE = col_double(),
##   YEAR = col_integer(),
##   WORKSITE = col_character(),
##   lon = col_double(),
##   lat = col_double()
## )
```

Preprocessing data

```
#find rows where conditions are true where CASE_STATUS is CERTIFIED, CERTIFIED-WITHDR
AWN that is treated as CERTIFIED or DENIED in the year 2014, 2015 and 2016

myData = filter(h1b_kaggle, h1b_kaggle$CASE_STATUS %in% c('CERTIFIED','DENIED', 'CERT
IFIED-WITHDRAWN')  & (h1b_kaggle$YEAR == 2016 | h1b_kaggle$YEAR == 2015 | h1b_kaggle$
YEAR == 2014))



#Keep only complete cases
myData = myData[complete.cases(myData),]

h1bData = myData

#Eliminate columns case#, employer, job title, long, lat
h1bData[,c(1,3,5,10,11)]=NULL

#Create a new column called worksite to keep only state
h1bData=separate(data = h1bData, col = WORKSITE, into = c("CITY", "STATE"), sep = ","
)

#Create a new column to save occupations
h1bData$occ=NA

#Keep occupations containing the keyword and set the new occupation
h1bData$occ[grep("engineer",h1bData$SOC_NAME, ignore.case = T)]="ENGINEER"
h1bData$occ[grep("manager",h1bData$SOC_NAME, ignore.case = T)]="MANAGER"
h1bData$occ[grep("technician",h1bData$SOC_NAME, ignore.case = T)]="TECHNICIAN"
h1bData$occ[grep("teacher",h1bData$SOC_NAME, ignore.case = T)]="TEACHER"
h1bData$occ[grep("executive",h1bData$SOC_NAME, ignore.case = T)]="EXECUTIVE"
h1bData$occ[grep("accountant",h1bData$SOC_NAME, ignore.case = T)]="ACCOUNTANT"
h1bData$occ[grep("actor",h1bData$SOC_NAME, ignore.case = T)]="ACTOR"
h1bData$occ[grep("advertising",h1bData$SOC_NAME, ignore.case = T)]="ADVERTISING"
h1bData$occ[grep("lawyer",h1bData$SOC_NAME, ignore.case = T)]="LAWYER"
h1bData$occ[grep("financial",h1bData$SOC_NAME, ignore.case = T)]="FINANCIAL"
h1bData$occ[grep("arquitect",h1bData$SOC_NAME, ignore.case = T)]="ARQUITECT"
h1bData$occ[grep("programmer",h1bData$SOC_NAME, ignore.case = T)]="SOFTWARE"
h1bData$occ[grep("software",h1bData$SOC_NAME, ignore.case = T)]="SOFTWARE"
h1bData$occ[grep("computer",h1bData$SOC_NAME, ignore.case = T)]="SOFTWARE"
h1bData$occ[grep("developer",h1bData$SOC_NAME, ignore.case = T)]="SOFTWARE"
h1bData$occ[grep("analyst",h1bData$SOC_NAME, ignore.case = T)]="ANALYST"
h1bData$occ[grep("scien",h1bData$SOC_NAME, ignore.case = T)]="SCIENTIST"
h1bData$occ[grep("specialist",h1bData$SOC_NAME, ignore.case = T)]="SPECIALIST"
h1bData$occ[grep("animal",h1bData$SOC_NAME, ignore.case = T)]="ANIMAL RELATED"
h1bData$occ[grep("athlet",h1bData$SOC_NAME, ignore.case = T)]="ATHLETE"
h1bData$occ[grep("cook",h1bData$SOC_NAME, ignore.case = T)]="COOK"
h1bData$occ[grep("chef",h1bData$SOC_NAME, ignore.case = T)]="COOK"
h1bData$occ[grep("admin",h1bData$SOC_NAME, ignore.case = T)]="ADMINISTRATIVE"

#Eliminate columns SOC_NAME and CITY
h1bData$SOC_NAME=NULL
h1bData$CITY= NULL

#Removing states with low count
a=sqldf("select count(*) cc, STATE from 'h1bData' group by STATE")
b=sqldf("select * from a where cc>2000 AND STATE <> ' NA'")
h1bData$STATE=ifelse(h1bData$STATE %in% b$STATE,h1bData$STATE,NA)
```

```
#Convert the dependent variable to binary
h1bData$CASE_STATUS=ifelse(h1bData$CASE_STATUS %in% c("CERTIFIED-WITHDRAWN", "CERTIFI
ED"),"1","0")

#Converting categorical variables into factors
h1bData[,c(-3)]= lapply(h1bData[,c(-3)], as.factor)
h1bData = h1bData[complete.cases(h1bData),]

#Using years 2014 and 2015 as training and 2016 as test
data.test = h1bData[0:557090, ]
data.train = h1bData[557091:1522982,]
data.test = data.test[, -4]
data.train = data.train[, -4]
```

LOGISTIC REGRESSION USING ALL PREDICTORS

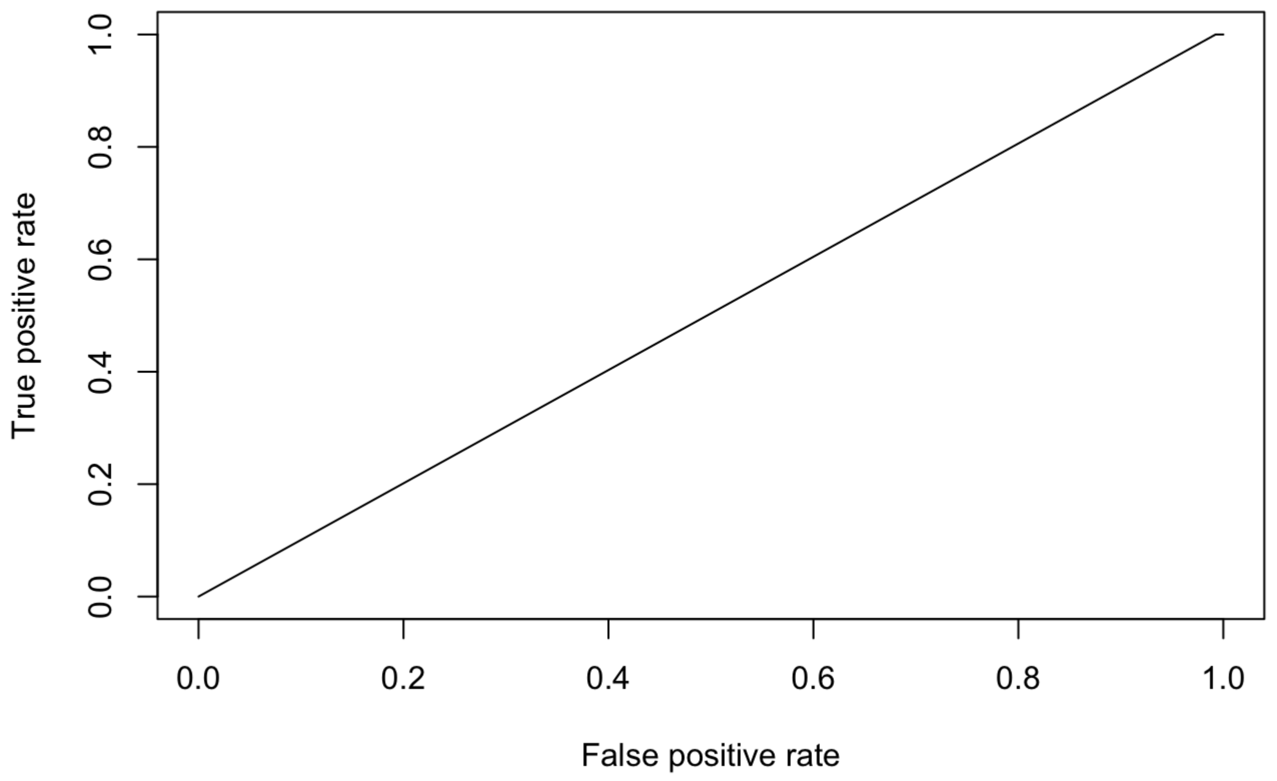
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~., family=binomial(link = logit), data = data.train
)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")
View(prediction)

#Threshold
pred_class = vector()
pred_class[prediction<0.5]=0
pred_class[prediction>=0.5]=1

#confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction       0      1
##          0      51      1
##          1    6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

```
#levels(data.test$CASE_STATUS)

#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5038045
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION AND PREVAILING_WAGE

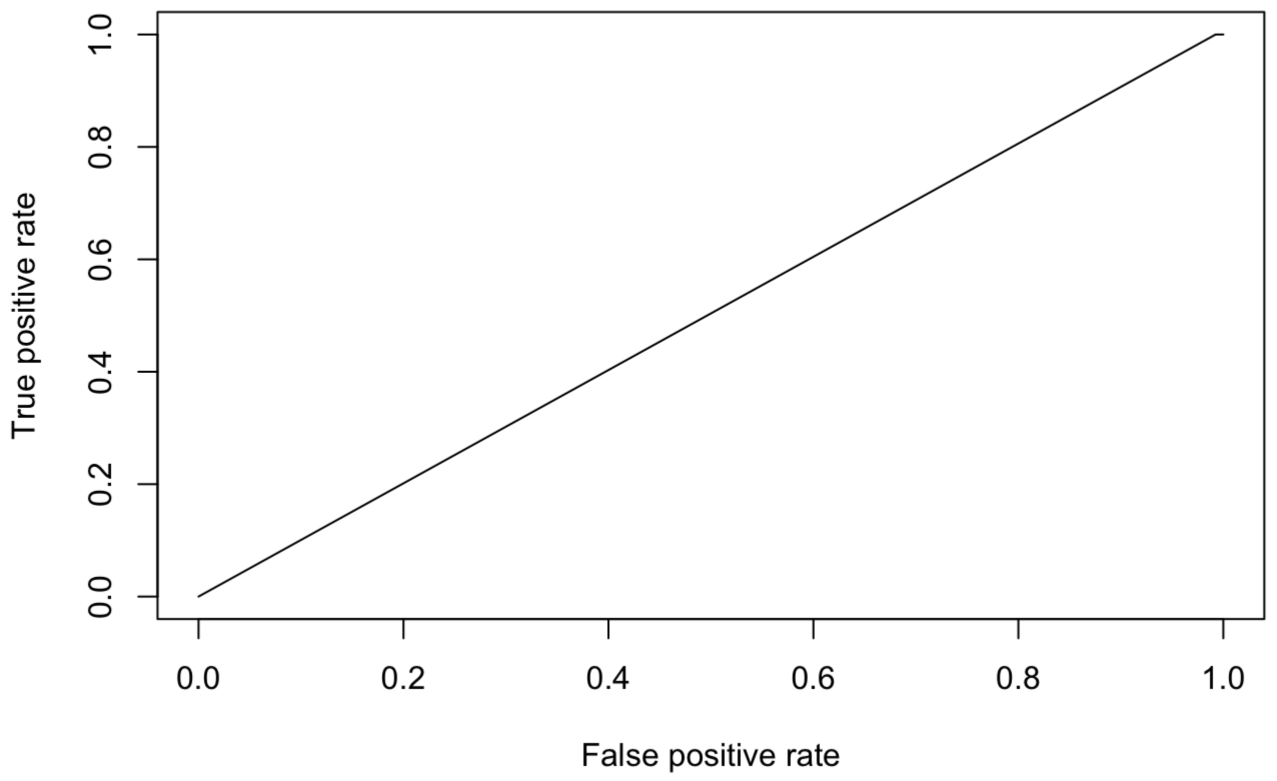
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE-occ, family=binomial(link = logit), data =
data.train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.5]=0
pred_class[prediction>=0.5]=1

#confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

```
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5038045
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

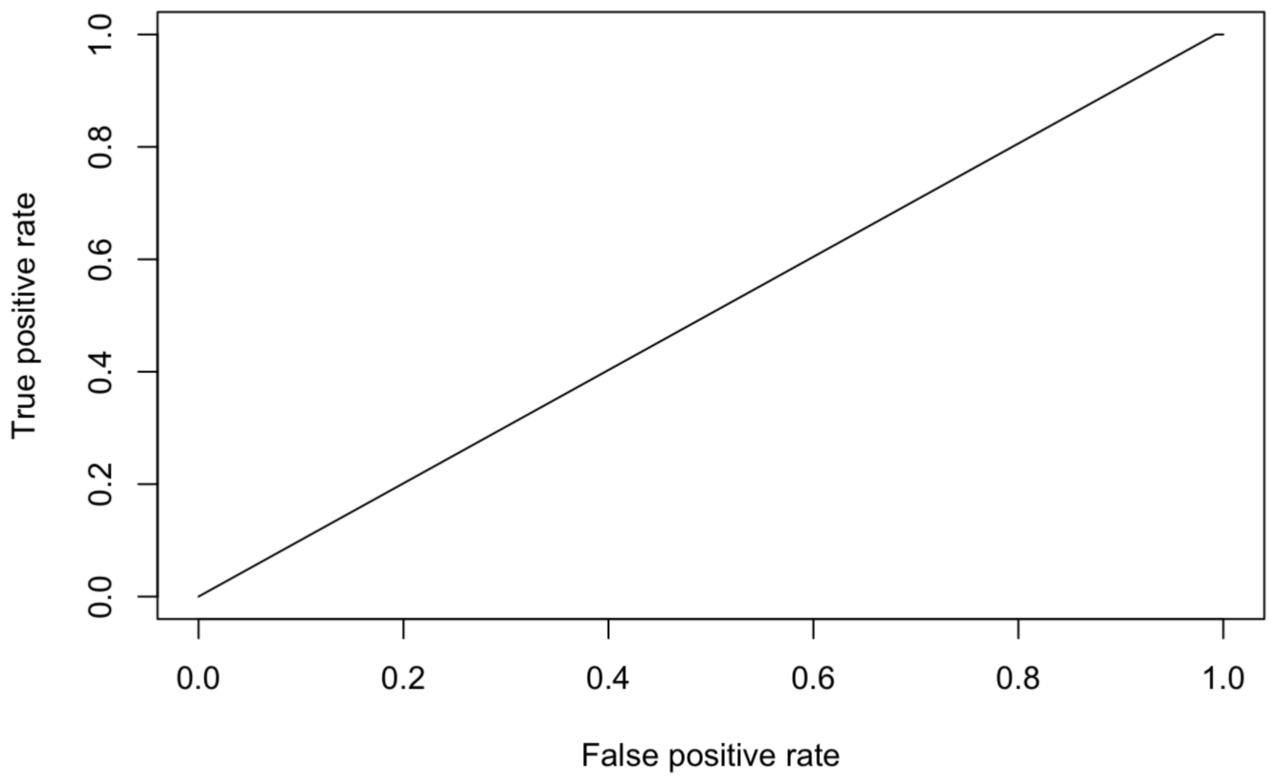
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-occ, family=binomial(link = logit), data = data.t
rain)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.5]=0
pred_class[prediction>=0.5]=1

#confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

```
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5038045
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

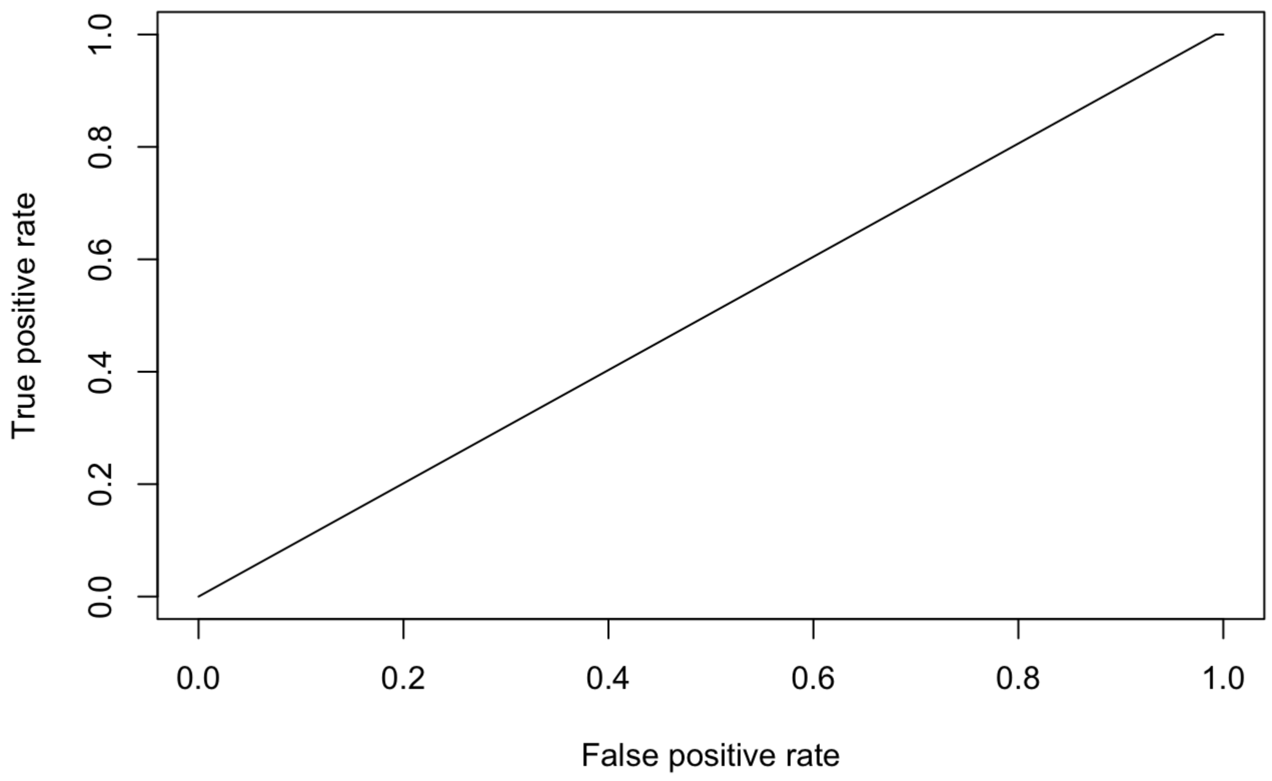
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE, family=binomial(link = logit), data = dat
a.train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.5]=0
pred_class[prediction>=0.5]=1

#confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

```
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5038045
```

LDA USING USING ALL PREDICTORS

```
lda.fit=lda(CASE_STATUS~., data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0    105    360
##          1   6596 550029
##
##                Accuracy : 0.9875
##                  95% CI : (0.9872, 0.9878)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.9991
##
##                   Kappa : 0.0278
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99935
##             Specificity : 0.01567
##          Pos Pred Value : 0.98815
##          Neg Pred Value : 0.22581
##              Prevalence : 0.98797
##          Detection Rate : 0.98733
##    Detection Prevalence : 0.99917
##       Balanced Accuracy : 0.50751
##
##        'Positive' Class : 1
##
```

LDA USING ONLY FULL_TIME_POSITION AND PREVAILING_WAGE

```
lda.fit=lda(CASE_STATUS~.-STATE-occ, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

LDA USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

```
lda.fit=lda(CASE_STATUS~.-STATE, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     99    318
##          1   6602 550071
##
##                Accuracy : 0.9876
##                  95% CI : (0.9873, 0.9879)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.9964
##
##                   Kappa : 0.0264
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99942
##             Specificity : 0.01477
##          Pos Pred Value : 0.98814
##          Neg Pred Value : 0.23741
##              Prevalence : 0.98797
##          Detection Rate : 0.98740
##    Detection Prevalence : 0.99925
##       Balanced Accuracy : 0.50710
##
##        'Positive' Class : 1
##
```

LDA USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

```
lda.fit=lda(CASE_STATUS~.-occ, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

QDA USING USING ALL PREDICTORS

```
qda.fit=qda(CASE_STATUS~., data=data.train)
qda.predict = predict(qda.fit, data.test)
View(qda.predict)
confusionMatrix(qda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2428 120754
##          1   4273 429635
##
##                Accuracy : 0.7756
##                  95% CI : (0.7745, 0.7767)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.78060
##             Specificity : 0.36233
##          Pos Pred Value : 0.99015
##          Neg Pred Value : 0.01971
##              Prevalence : 0.98797
##          Detection Rate : 0.77121
##    Detection Prevalence : 0.77888
##       Balanced Accuracy : 0.57147
##
##        'Positive' Class : 1
##
```

QDA USING ONLY FULL_TIME_POSITION AND PREVAILING_WAGE

```
lda.fit=lda(CASE_STATUS~.-STATE-occ, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

QDA USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

```
lda.fit=lda(CASE_STATUS~.-STATE, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     99    318
##          1   6602 550071
##
##                Accuracy : 0.9876
##                  95% CI : (0.9873, 0.9879)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.9964
##
##                   Kappa : 0.0264
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99942
##             Specificity : 0.01477
##          Pos Pred Value : 0.98814
##          Neg Pred Value : 0.23741
##              Prevalence : 0.98797
##          Detection Rate : 0.98740
##    Detection Prevalence : 0.99925
##       Balanced Accuracy : 0.50710
##
##        'Positive' Class : 1
##
```

QDA USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

```
lda.fit=lda(CASE_STATUS~.-occ, data=data.train)
lda.predict = predict(lda.fit, data.test, type = "prob")
confusionMatrix(lda.predict$class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     51      1
##          1   6650 550388
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9883)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2719
##
##                   Kappa : 0.0149
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999998
##             Specificity : 0.007611
##          Pos Pred Value : 0.988062
##          Neg Pred Value : 0.980769
##              Prevalence : 0.987971
##          Detection Rate : 0.987970
##    Detection Prevalence : 0.999907
##       Balanced Accuracy : 0.503804
##
##        'Positive' Class : 1
##
```

LOGISTIC REGRESSION USING ALL PREDICTORS and threshold 0.97

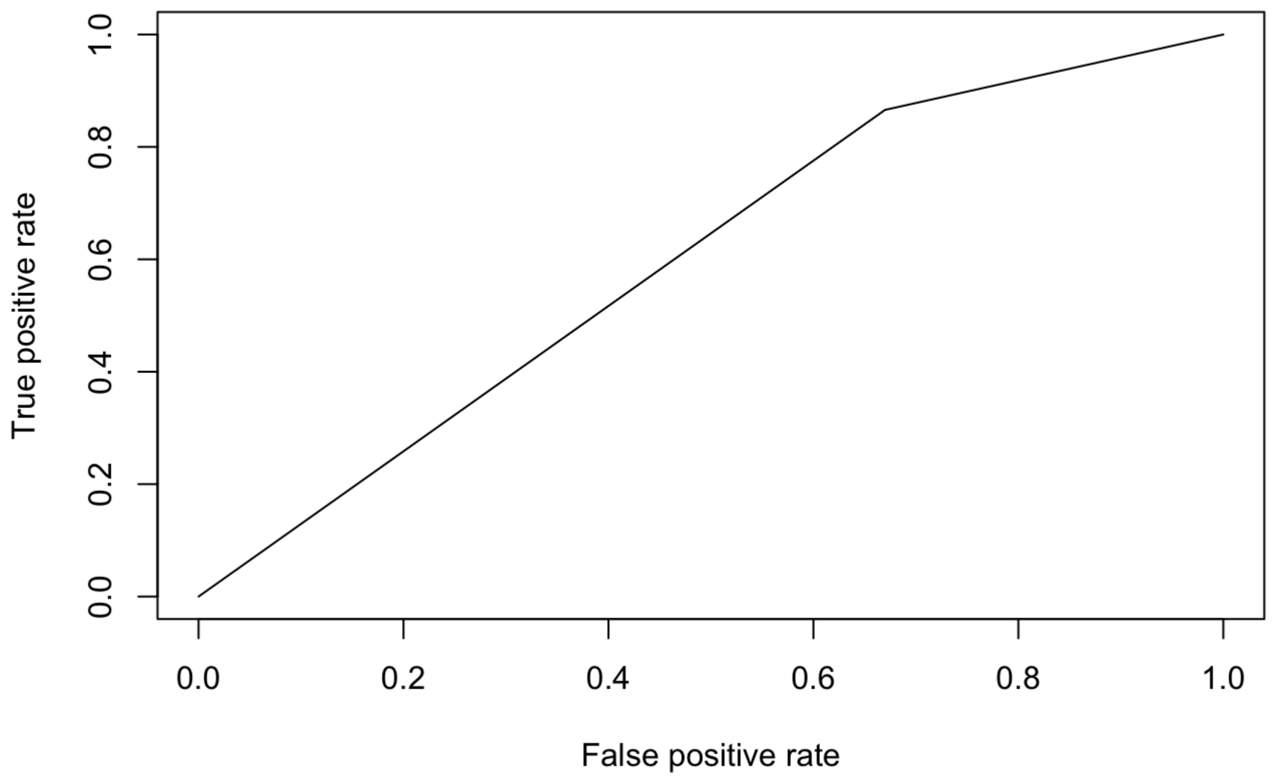
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~., family=binomial(link = logit), data = data.train
)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Predicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.97]=0
pred_class[prediction>=0.97]=1

##confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2213  73858
##          1   4488 476531
##
##                Accuracy : 0.8594
##                  95% CI : (0.8584, 0.8603)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0321
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.86581
##             Specificity : 0.33025
##          Pos Pred Value : 0.99067
##          Neg Pred Value : 0.02909
##              Prevalence : 0.98797
##          Detection Rate : 0.85539
##    Detection Prevalence : 0.86345
##       Balanced Accuracy : 0.59803
##
##        'Positive' Class : 1
##
```

```
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5980284
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION AND PREVAILING_WAGE

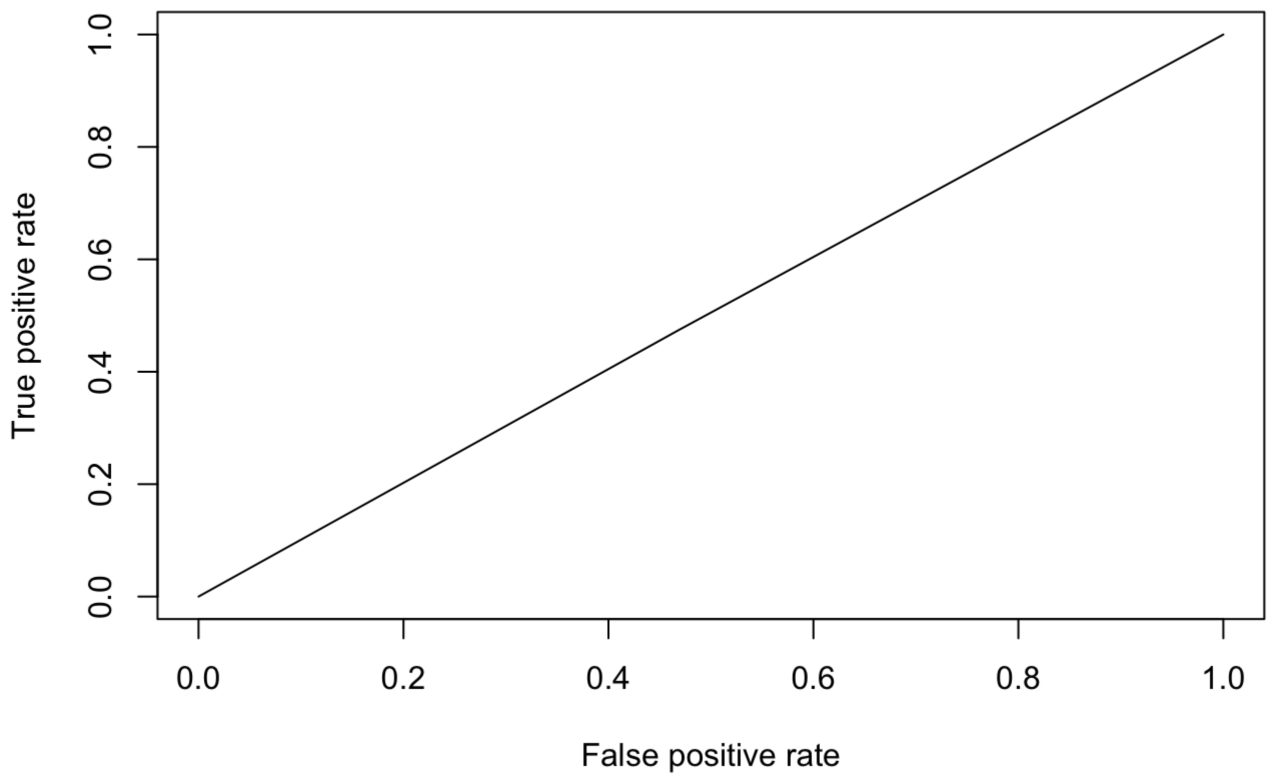
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE-occ, family=binomial(link = logit), data =
data.train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.97]=0
pred_class[prediction>=0.97]=1

##confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction      0      1
##          0   3566 289791
##          1   3135 260598
## 
##                Accuracy : 0.4742
##                  95% CI : (0.4729, 0.4755)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
## 
##                   Kappa : 3e-04
##  Mcnemar's Test P-Value : <2e-16
## 
##             Sensitivity : 0.47348
##             Specificity : 0.53216
##          Pos Pred Value : 0.98811
##          Neg Pred Value : 0.01216
##              Prevalence : 0.98797
##          Detection Rate : 0.46778
##    Detection Prevalence : 0.47341
##       Balanced Accuracy : 0.50282
## 
##        'Positive' Class : 1
## 
```

```r
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5028195
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

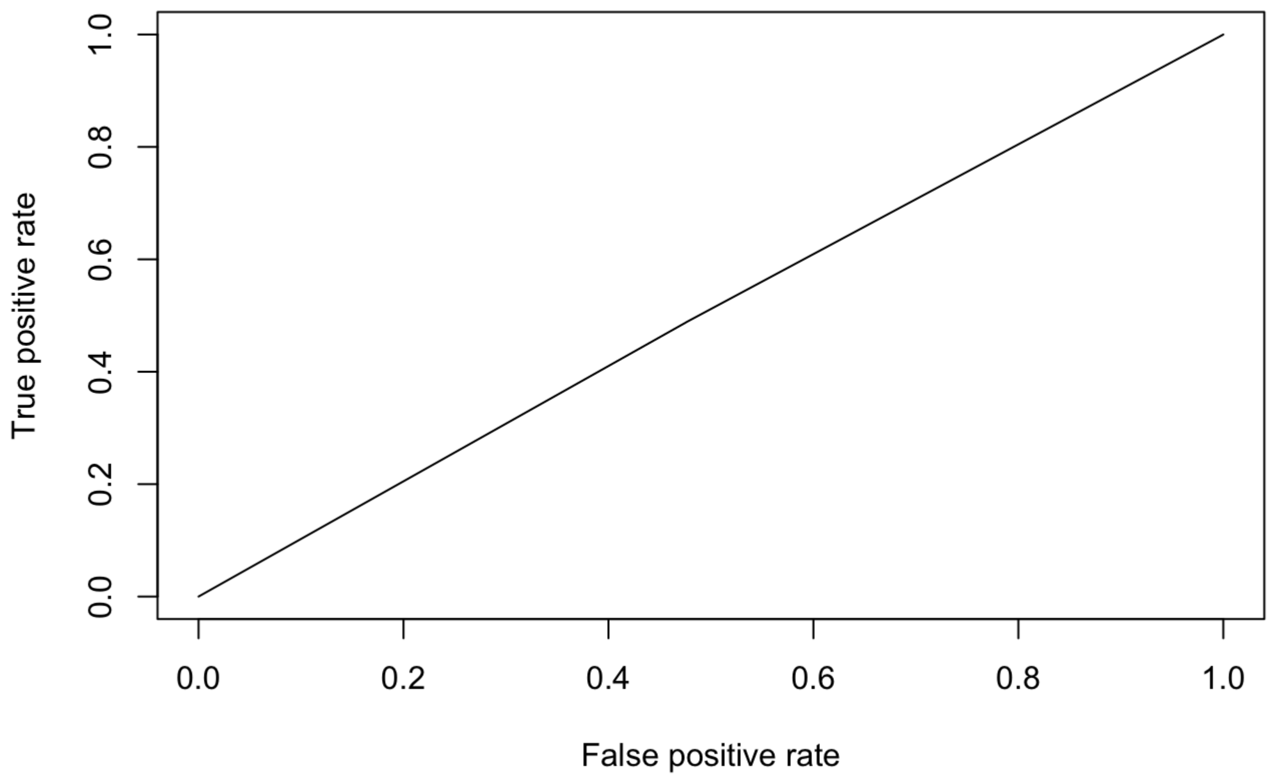
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-occ, family=binomial(link = logit), data = data.t
rain)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.97]=0
pred_class[prediction>=0.97]=1

##confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   3512 281733
##          1   3189 268656
##
##                Accuracy : 0.4886
##                  95% CI : (0.4872, 0.4899)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 6e-04
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.48812
##             Specificity : 0.52410
##          Pos Pred Value : 0.98827
##          Neg Pred Value : 0.01231
##              Prevalence : 0.98797
##          Detection Rate : 0.48225
##    Detection Prevalence : 0.48797
##       Balanced Accuracy : 0.50611
##
##        'Positive' Class : 1
##
```

```r
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

```
## [1] 0.5061106
```

LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

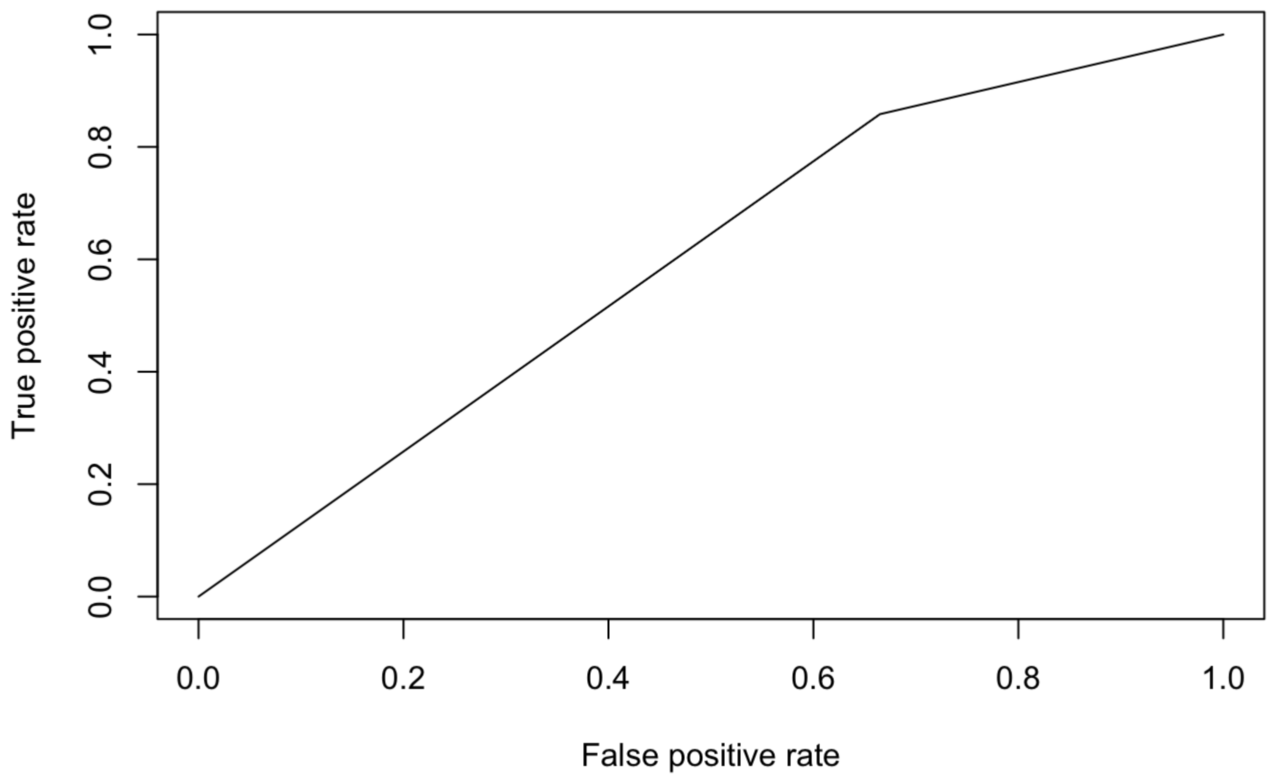
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE, family=binomial(link = logit), data = dat
a.train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Finding Prdicitons on Testing set
prediction=predict(h1bglm.train.fit, newdata=data.test, type="response")

#Threshold
pred_class = vector()
pred_class[prediction<0.97]=0
pred_class[prediction>=0.97]=1

#confusion matrix
confusionMatrix(pred_class, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2246  78029
##          1   4455 472360
##
##                Accuracy : 0.8519
##                  95% CI : (0.851, 0.8529)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0301
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85823
##             Specificity : 0.33517
##          Pos Pred Value : 0.99066
##          Neg Pred Value : 0.02798
##              Prevalence : 0.98797
##          Detection Rate : 0.84791
##    Detection Prevalence : 0.85590
##       Balanced Accuracy : 0.59670
##
##        'Positive' Class : 1
##
```

```
#ROC Curve
pred = prediction(pred_class, data.test$CASE_STATUS)
perf = performance(pred,"tpr","fpr")
plot(perf)
```



```
#Area Under the Curve
auc.tmp = performance(pred,"auc");
auc = as.numeric(auc.tmp@y.values)
auc
```

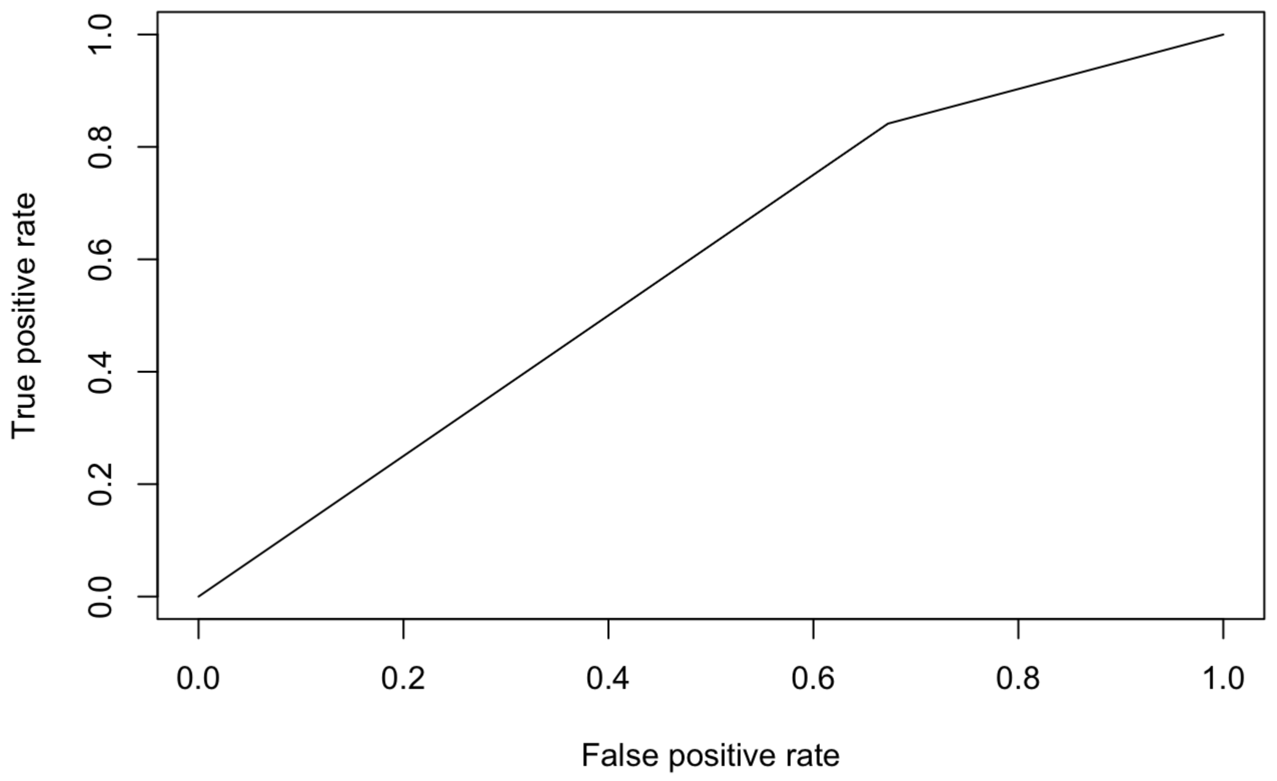
```
## [1] 0.5967016
```

DECISION TREE USING ALL PREDICTORS AND THRESHOLD 0.97

```
ctree1=ctree(CASE_STATUS~., data=data.train)
probabilities = 1-unlist(treeresponse(ctree1,newdata=data.test), use.names=F)[seq(1,n
row(data.test)*2,2)]

probabilities[probabilities<0.97]=0
probabilities[probabilities>=0.97]=1

confusionMatrix(probabilities, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2194  87316
##          1   4507 463073
##
##                Accuracy : 0.8352
##                  95% CI : (0.8342, 0.8361)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0238
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84136
##             Specificity : 0.32741
##          Pos Pred Value : 0.99036
##          Neg Pred Value : 0.02451
##              Prevalence : 0.98797
##          Detection Rate : 0.83124
##    Detection Prevalence : 0.83933
##       Balanced Accuracy : 0.58438
##
##        'Positive' Class : 1
##
```

```
#ROC CURVE
pred1 = prediction( probabilities, data.test$CASE_STATUS)
perf1 = performance(pred1,"tpr","fpr")
plot(perf1)
```



```
#AREA Under the Curve
auc.tmp1 = performance(pred1,"auc");
auc1 = as.numeric(auc.tmp1@y.values)
auc1
```

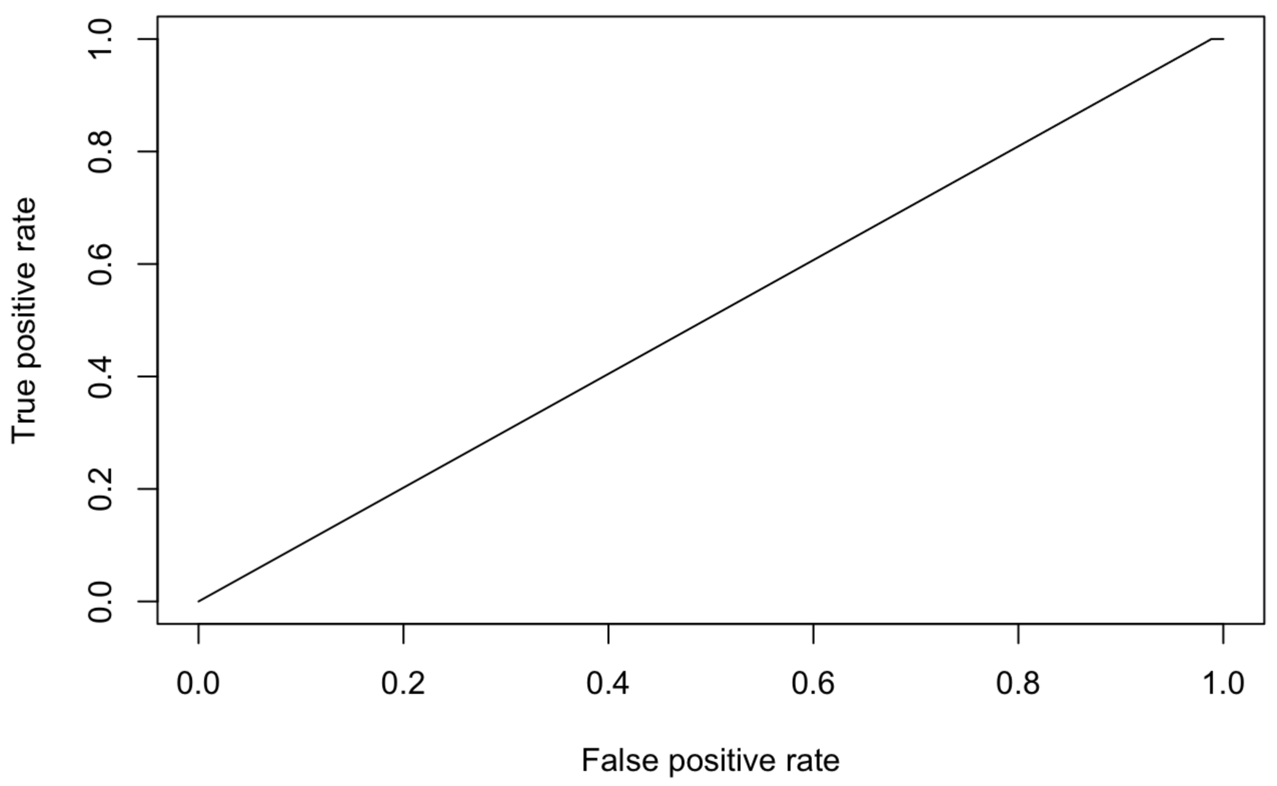
```
## [1] 0.5843848
```

DECISION TREE USING ALL PREDICTORS AND THRESHOLD 0.5

```
ctree1=ctree(CASE_STATUS~., data=data.train)
probabilities = 1-unlist(treeresponse(ctree1,newdata=data.test), use.names=F)[seq(1,n
row(data.test)*2,2)]

probabilities[probabilities<0.5]=0
probabilities[probabilities>=0.5]=1

confusionMatrix(probabilities, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0     79     15
##          1   6622 550374
##
##                Accuracy : 0.9881
##                  95% CI : (0.9878, 0.9884)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 0.2178
##
##                   Kappa : 0.0229
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99997
##             Specificity : 0.01179
##          Pos Pred Value : 0.98811
##          Neg Pred Value : 0.84043
##              Prevalence : 0.98797
##          Detection Rate : 0.98794
##    Detection Prevalence : 0.99983
##       Balanced Accuracy : 0.50588
##
##        'Positive' Class : 1
##
```

```
#ROC CURVE
pred1 = prediction( probabilities, data.test$CASE_STATUS)
perf1 = performance(pred1,"tpr","fpr")
plot(perf1)
```



```
#AREA Under the Curve
auc.tmp1 = performance(pred1,"auc");
auc1 = as.numeric(auc.tmp1@y.values)
auc1
```

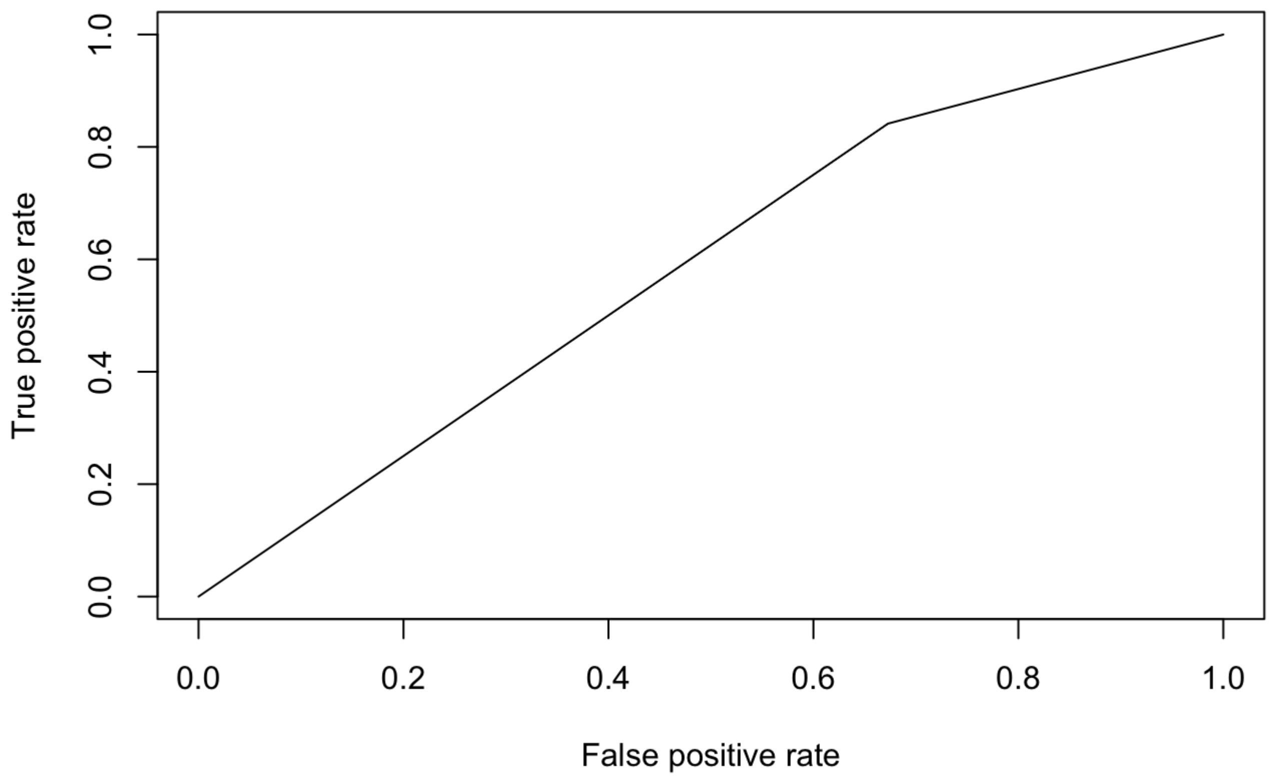
```
## [1] 0.505881
```

DECISION TREE USING ONLY FULL_TIME_POSITION AND PREVAILING_WAGE

```
ctree1=ctree(CASE_STATUS~.-STATE-occ, data=data.train)
probabilities = 1-unlist(treeresponse(ctree1,newdata=data.test), use.names=F)[seq(1,n
row(data.test)*2,2)]

probabilities[probabilities<0.97]=0
probabilities[probabilities>=0.97]=1

confusionMatrix(probabilities, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2194  87316
##          1   4507 463073
##
##                Accuracy : 0.8352
##                  95% CI : (0.8342, 0.8361)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0238
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84136
##             Specificity : 0.32741
##          Pos Pred Value : 0.99036
##          Neg Pred Value : 0.02451
##              Prevalence : 0.98797
##          Detection Rate : 0.83124
##    Detection Prevalence : 0.83933
##       Balanced Accuracy : 0.58438
##
##        'Positive' Class : 1
##
```

```
#ROC CURVE
pred1 = prediction( probabilities, data.test$CASE_STATUS)
perf1 = performance(pred1,"tpr","fpr")
plot(perf1)
```



```
#AREA Under the Curve
auc.tmp1 = performance(pred1,"auc");
auc1 = as.numeric(auc.tmp1@y.values)
auc1
```

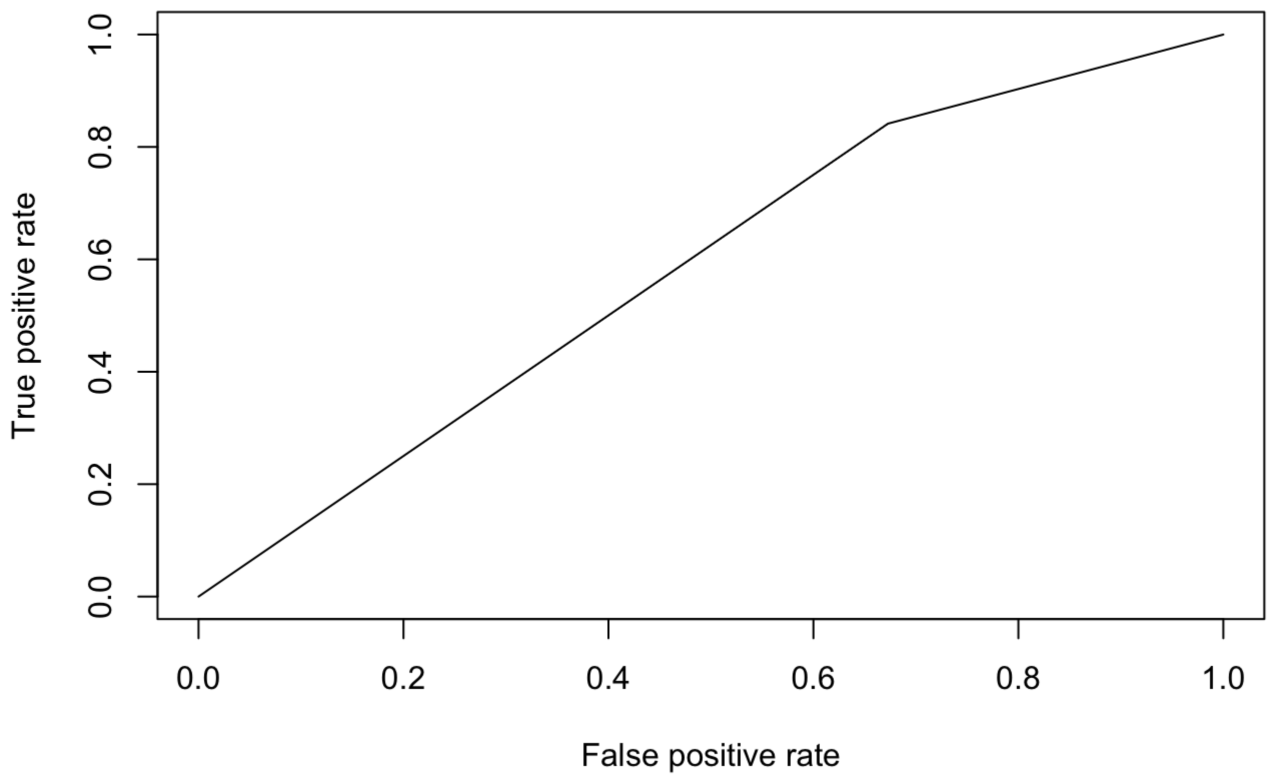
```
## [1] 0.5843848
```

DECISION TREE USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

```
ctree1=ctree(CASE_STATUS~.-occ, data=data.train)
probabilities = 1-unlist(treeresponse(ctree1,newdata=data.test), use.names=F)[seq(1,n
row(data.test)*2,2)]

probabilities[probabilities<0.97]=0
probabilities[probabilities>=0.97]=1

confusionMatrix(probabilities, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2194  87316
##          1   4507 463073
##
##                Accuracy : 0.8352
##                  95% CI : (0.8342, 0.8361)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0238
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84136
##             Specificity : 0.32741
##          Pos Pred Value : 0.99036
##          Neg Pred Value : 0.02451
##              Prevalence : 0.98797
##          Detection Rate : 0.83124
##    Detection Prevalence : 0.83933
##       Balanced Accuracy : 0.58438
##
##        'Positive' Class : 1
##
```

```
#ROC CURVE
pred1 = prediction( probabilities, data.test$CASE_STATUS)
perf1 = performance(pred1,"tpr","fpr")
plot(perf1)
```



```
#AREA Under the Curve
auc.tmp1 = performance(pred1,"auc");
auc1 = as.numeric(auc.tmp1@y.values)
auc1
```

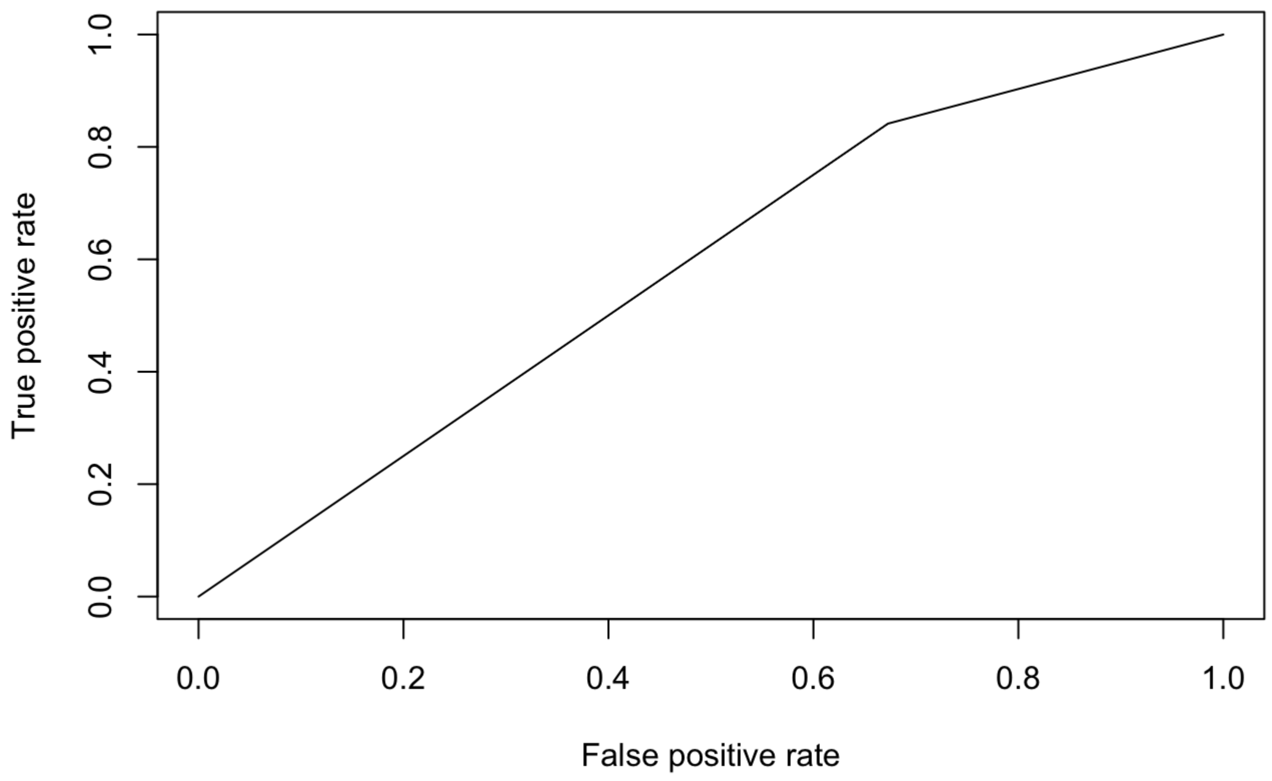
```
## [1] 0.5843848
```

DECISION TREE USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

```
ctree1=ctree(CASE_STATUS~.-STATE, data=data.train)
probabilities = 1-unlist(treeresponse(ctree1,newdata=data.test), use.names=F)[seq(1,n
row(data.test)*2,2)]

probabilities[probabilities<0.97]=0
probabilities[probabilities>=0.97]=1

confusionMatrix(probabilities, data.test$CASE_STATUS,positive="1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction      0      1
##          0   2194  87316
##          1   4507 463073
## 
##                Accuracy : 0.8352
##                  95% CI : (0.8342, 0.8361)
##     No Information Rate : 0.988
##     P-Value [Acc > NIR] : 1
## 
##                   Kappa : 0.0238
##  Mcnemar's Test P-Value : <2e-16
## 
##             Sensitivity : 0.84136
##             Specificity : 0.32741
##          Pos Pred Value : 0.99036
##          Neg Pred Value : 0.02451
##              Prevalence : 0.98797
##          Detection Rate : 0.83124
##    Detection Prevalence : 0.83933
##       Balanced Accuracy : 0.58438
## 
##        'Positive' Class : 1
## 
```

```
#ROC CURVE
pred1 = prediction( probabilities, data.test$CASE_STATUS)
perf1 = performance(pred1,"tpr","fpr")
plot(perf1)
```



```
#AREA Under the Curve
auc.tmp1 = performance(pred1,"auc");
auc1 = as.numeric(auc.tmp1@y.values)
auc1
```

```
## [1] 0.5843848
```