# new_project_visa

## libraries

```r
library(randomForest)
```

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

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

```r
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
```

```r
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

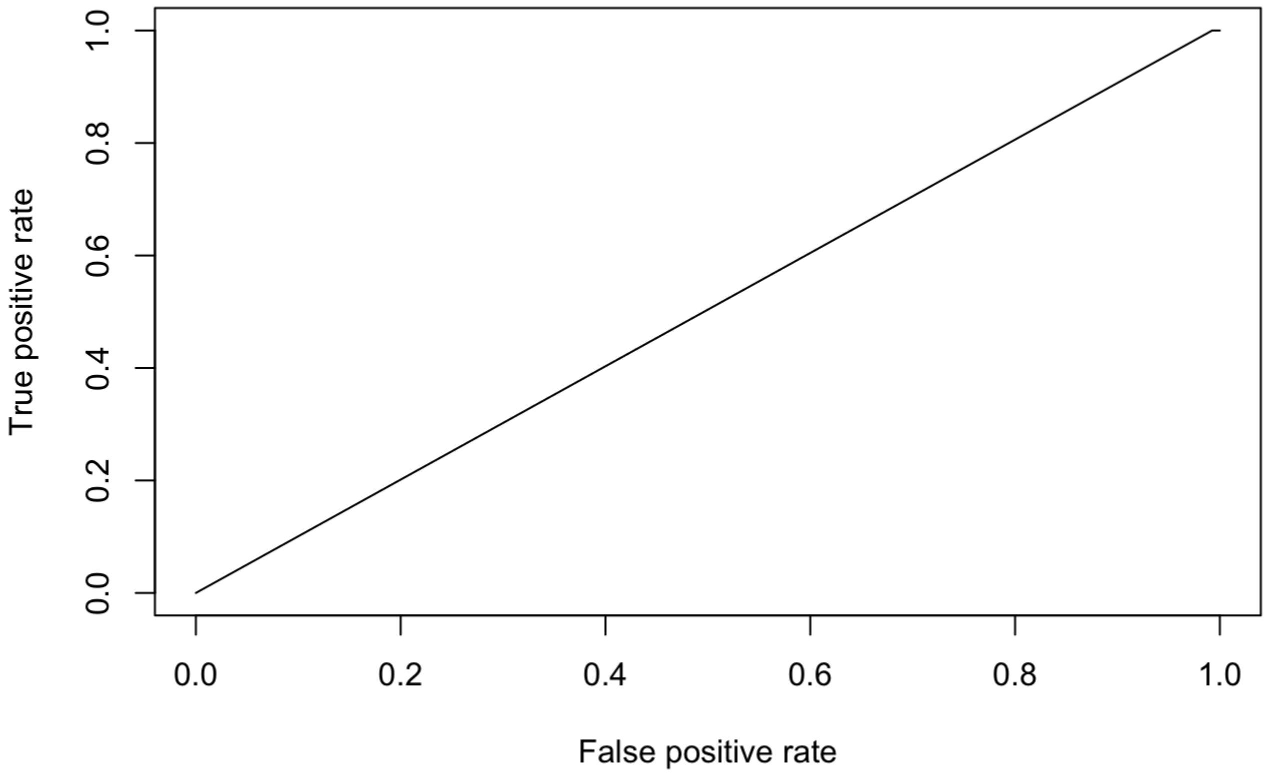
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~., family=binomial(link = logit), data = data.train
)
```

```
## 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

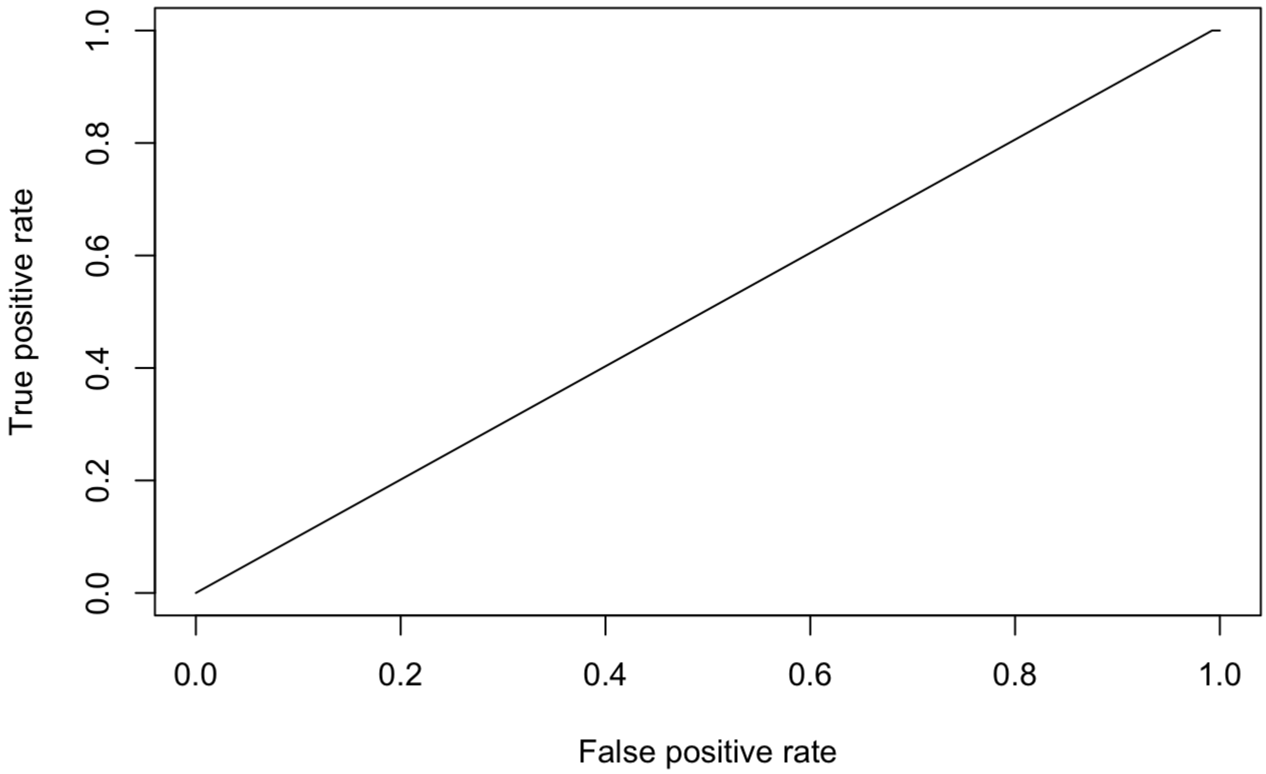
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE-occ, family=binomial(link = logit), data =
data.train)
```

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

```r
#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
##
```

```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] 0.5038045
```

# LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND STATE

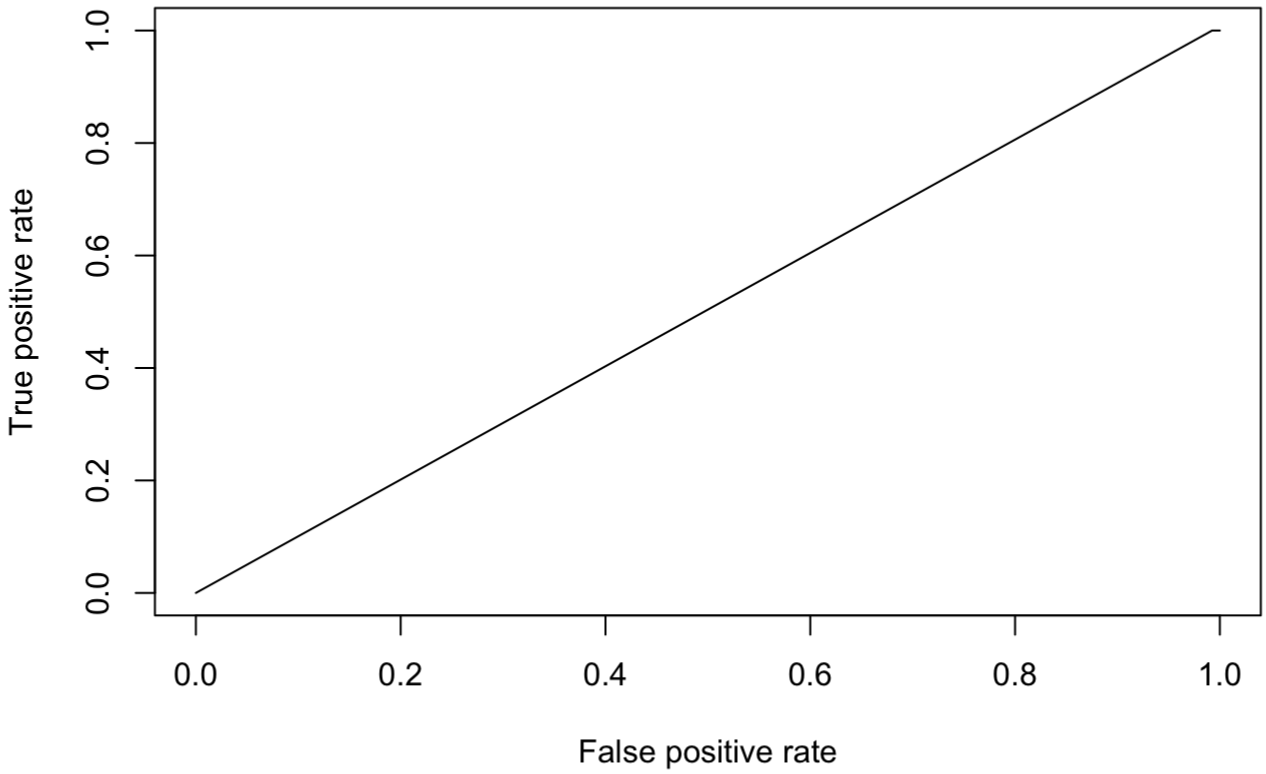
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-occ, family=binomial(link = logit), data = data.t
rain)
```

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

```r
#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
##
```

```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] 0.5038045
```

# LOGISTIC REGRESSION USING ONLY FULL_TIME_POSITION, PREVAILING_WAGE AND OCCUPATION

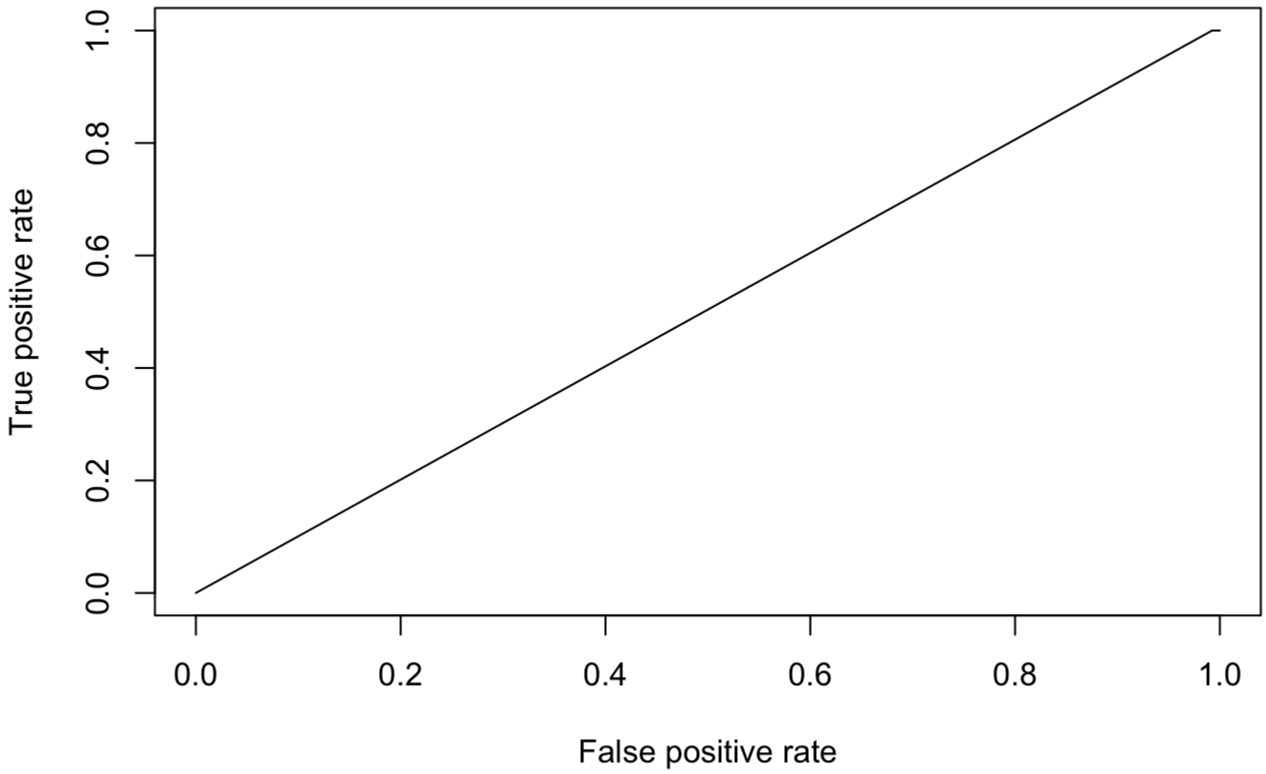
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE, family=binomial(link = logit), data = dat
a.train)
```

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

```r
#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
##
```

```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] 0.5038045
```

# 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

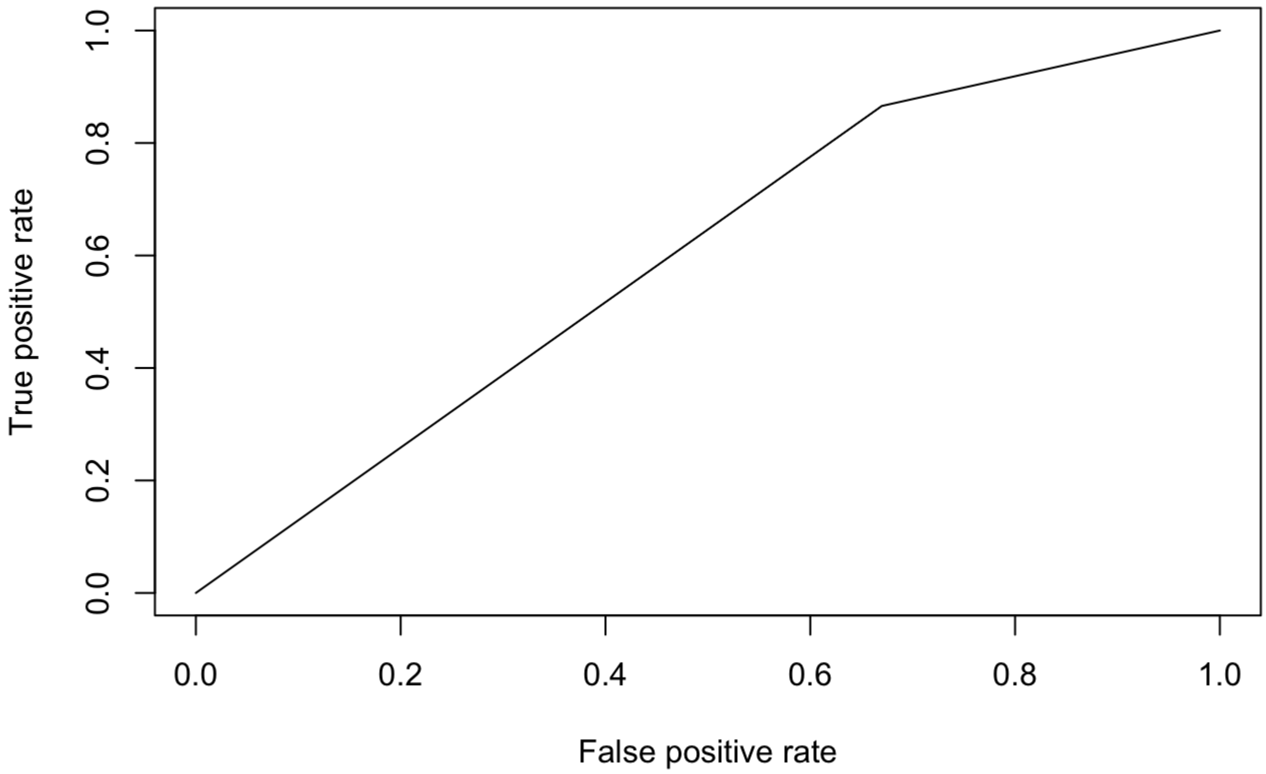
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~., family=binomial(link = logit), data = data.train
)
```

```
## 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

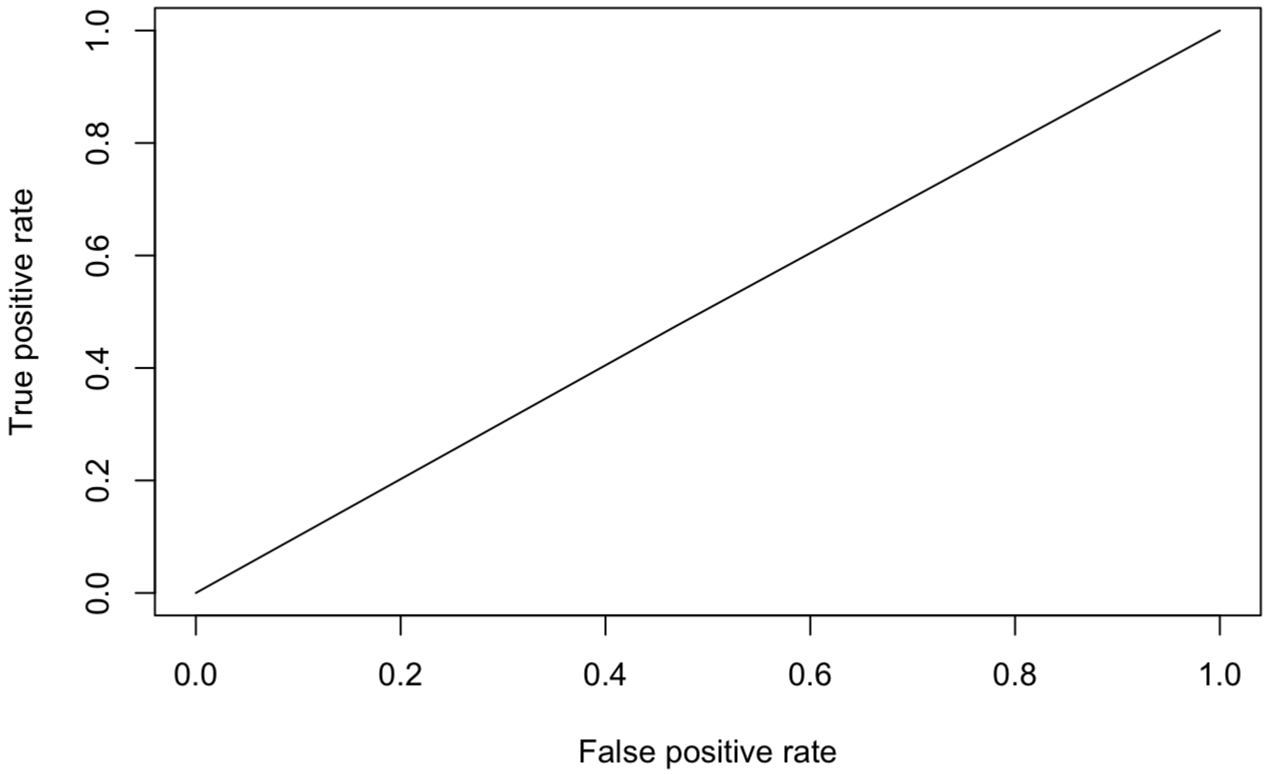
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE-occ, family=binomial(link = logit), data =
data.train)
```

```
## 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

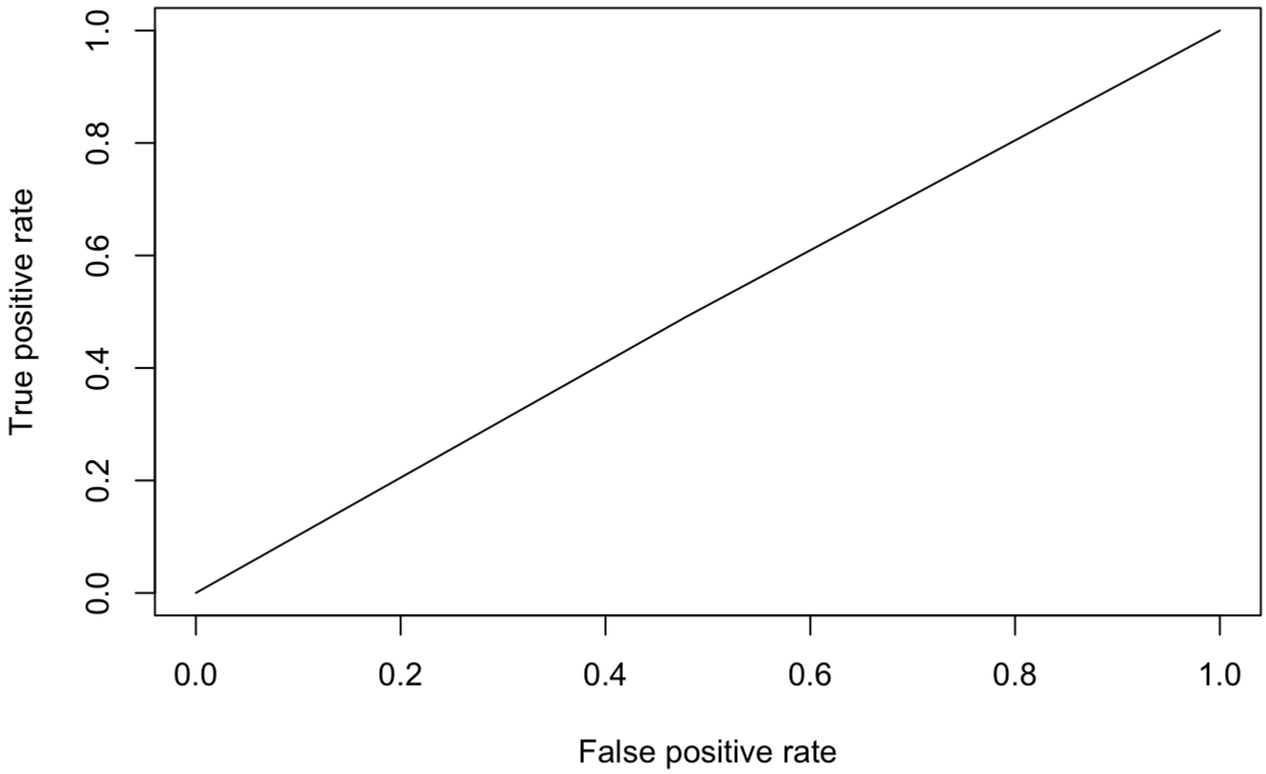
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-occ, family=binomial(link = logit), data = data.t
rain)
```

```
## 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

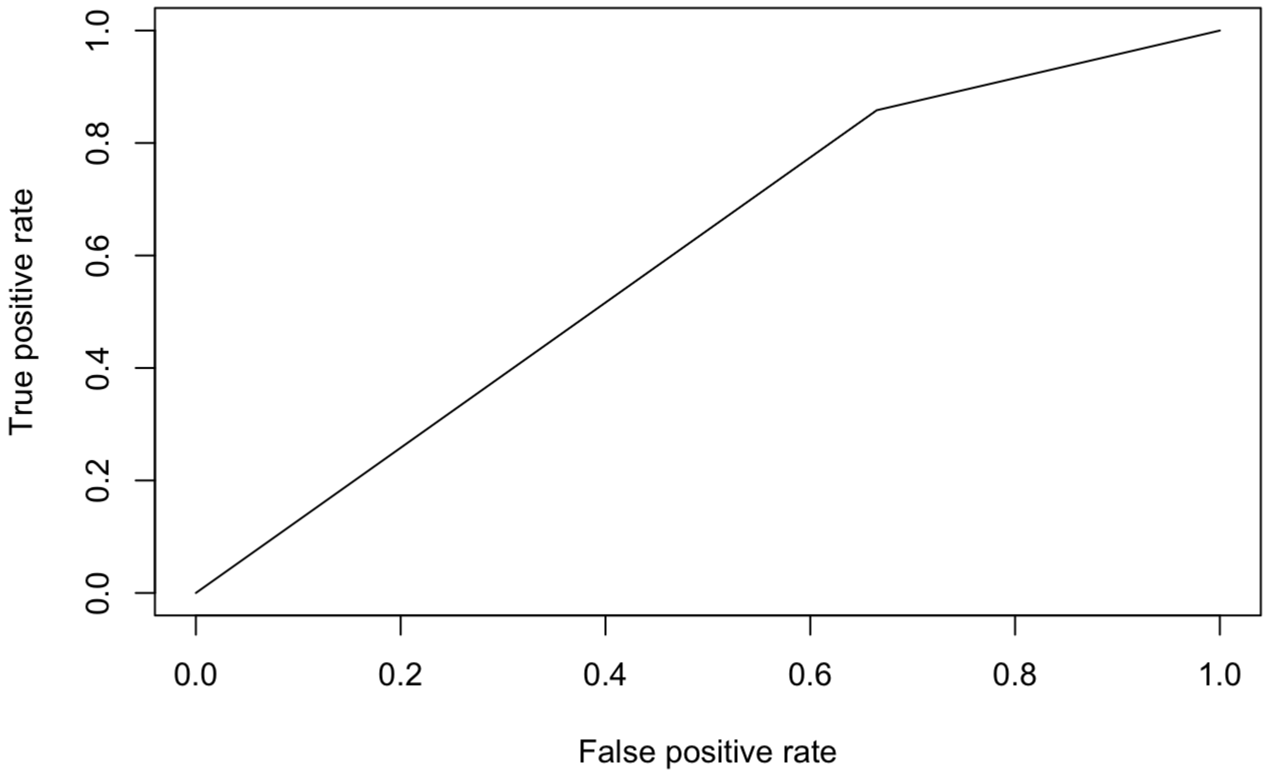
```
#Fitting the model on the training dataset
h1bglm.train.fit =glm(CASE_STATUS~.-STATE, family=binomial(link = logit), data = dat
a.train)
```

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

```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   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
##
```

```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
```

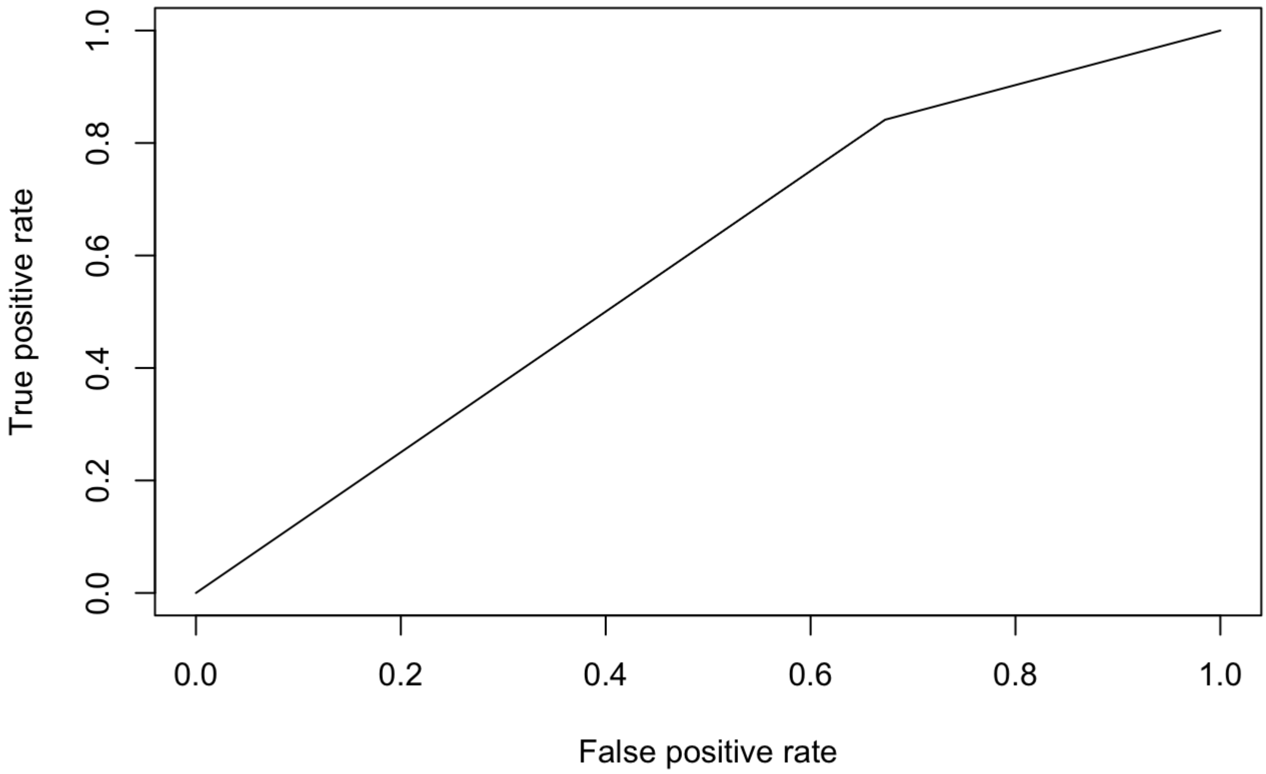
```
## [1] 0.5967016
```

# 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
```

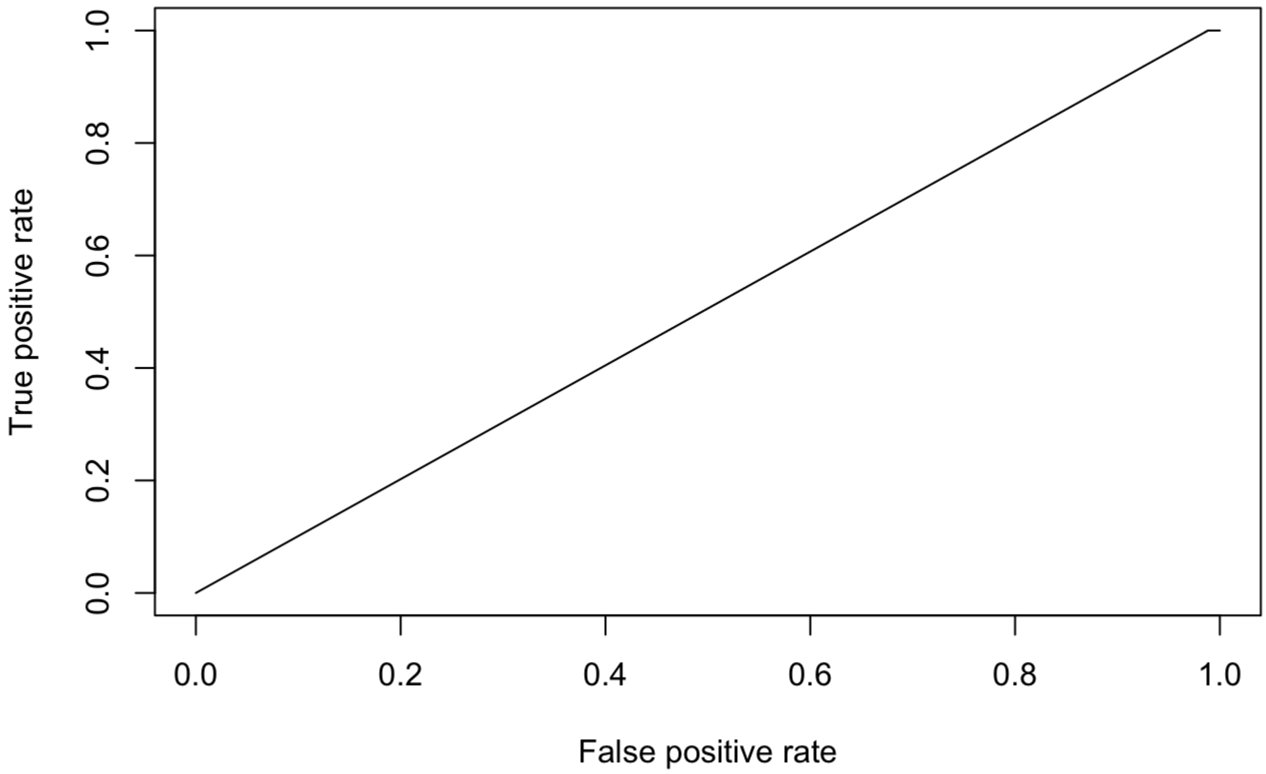
```
## [1] 0.5843848
```

# 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
```

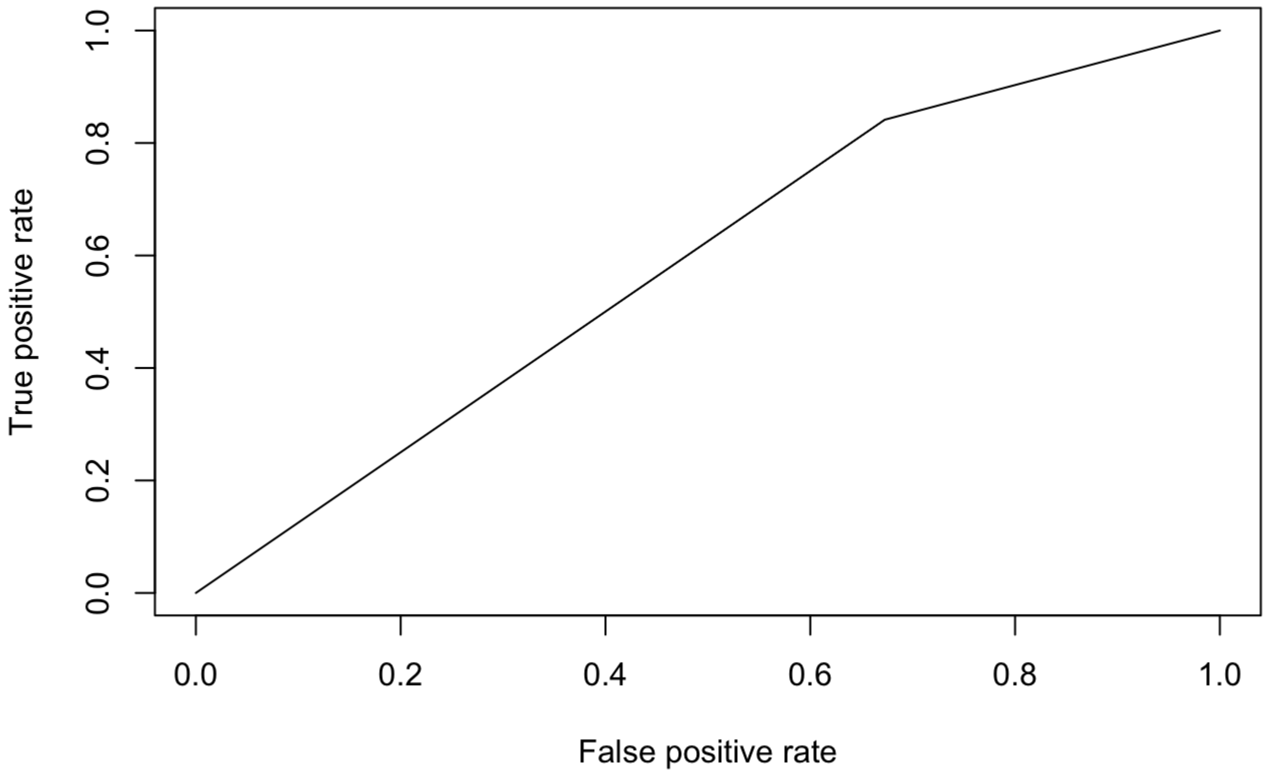
```
## [1] 0.505881
```

# 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
```

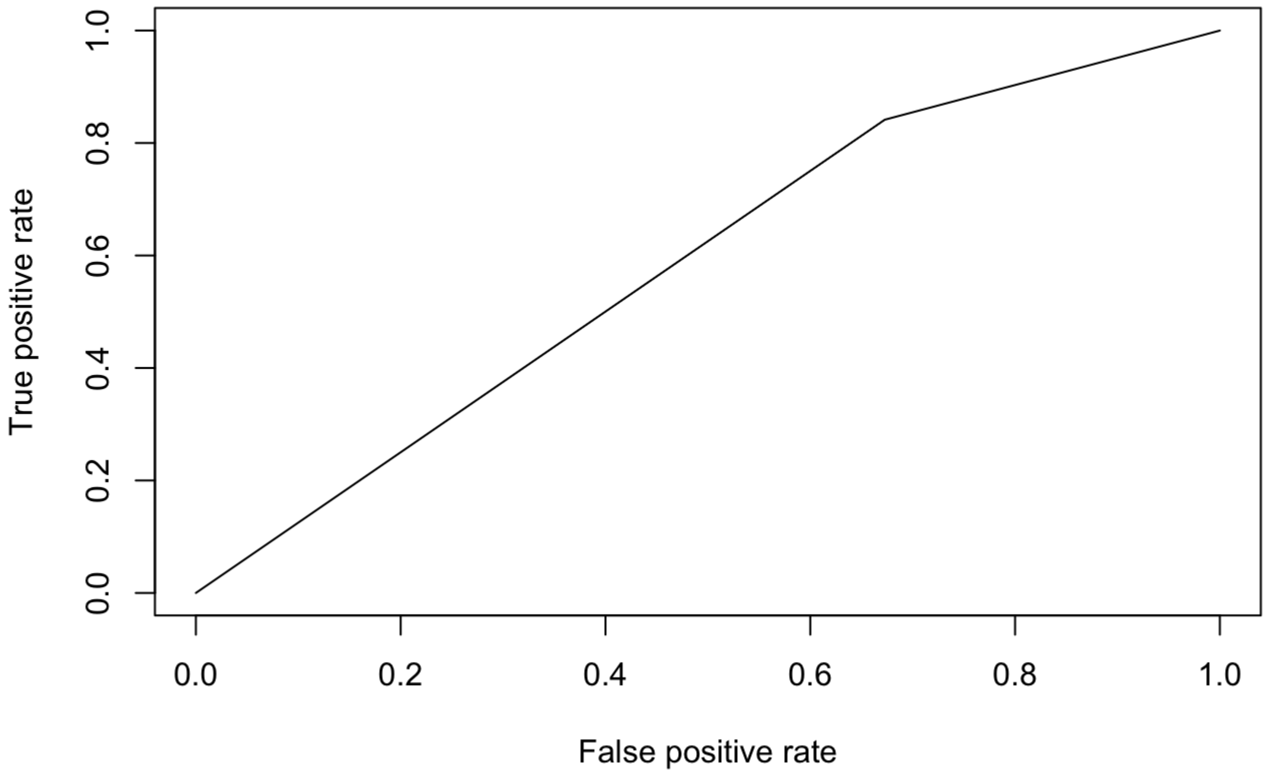
```
## [1] 0.5843848
```

# 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
```

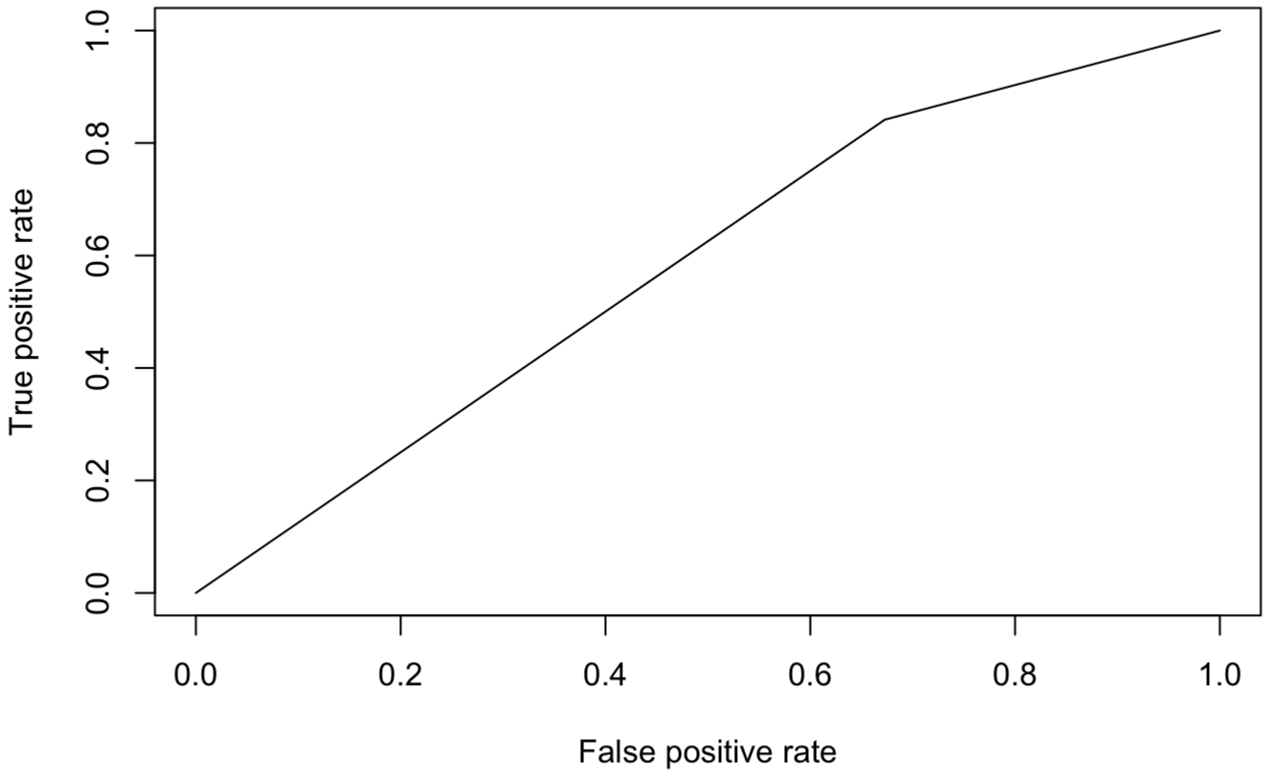
```
## [1] 0.5843848
```

# 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
```