D698 Final Project

Coffy Andrews-Guo, Vyanna Hill

2023-12-05

```r
#Packages used
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
##
## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(ggpubr)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following objects are masked from 'package:MLmetrics':
##
##     MAE, RMSE
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(caTools)
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit https://docs.h2o.ai
##
## ----------------------------------------------------------------------
##
##
## Attaching package: 'h2o'
##
## The following object is masked from 'package:pROC':
##
##     var
##
## The following objects are masked from 'package:lubridate':
##
##     day, hour, month, week, year
##
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
##
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

#Uploading data sets for combination

```
elevation<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/USGS_CACounties_elevation_2023 (1).cs
slope<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/SlopePercentage_Calitracts_LF2020.csv")
whp<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/WHP2020_ZipCode_Summary - zipcode_summary.c

weather<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/NOAA_CACounties_AverageTemp_2022.csv")
rainfall<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/NOAA_CACounties_AveragePercipitation_2

LF_Vegdictonary<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/LF22_EVT_230 - LF22_EVT_230.csv
cali_vegtype<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/CalifornianTracts_VegType_2022LF -
#combining the weather data set first as both on the county lvl. Only want the averages of the year
cali_cweather<-weather%>%left_join(rainfall,by=join_by(ID))
cali_cweather<-cali_cweather%>%select(-c("Rank.x","Anomaly (1901-2000 base period).x","1901-2000 Mean.x
cali_cweather<-cali_cweather%>%rename(county_Id=ID,county_name=Name.x,State=State.x,avg_tempeture=Value
```

#onto topographic data, will need to combine by county and lat/long(if possible) #Need the county ID in Slope data for left combine with elevation

```
cali_topography<-slope%>%unite("countyID",1:2,remove = FALSE,sep = "")
```

#14 rows missing slope data, can be zero slope but using mice for imputation

```
cali_topography<-cali_topography%>%rename(tract_avgSlope="_mean",tract_countSlope="_count",tract_maxSlop
cali_topography<-complete(mice(cali_topography,method = "cart",seed = 333))

elevation<-elevation%>%rename(countyID="County FIPS Code")
```

#mapping the tract data in topography by county ID and the closest match by Longitude

```
cali_topography<-cali_topography%>%inner_join(elevation,by=join_by(countyID,closest(INTPTLON<=Longitude)
```

#removing unnecessary metrics and renaming columns for readability

```
cali_topography<-cali_topography%>%select(-c("Latitude","Longitude","Bgn Decision Date","Entry Date","Co
cali_topography<-cali_topography%>%rename(tractID=NAME,land_Area=ALAND,water_Area=AWATER,latitude=INTPTL
```

#Using LandFire's vegetation type dictionary to map tract's average vegetation type #Filtering for CA tracts only

```
cali_vegetation<-cali_vegtype%>%filter(STUSPS=="CA")
lf_small<-LF_Vegdictonary%>%select("VALUE","EVT_NAME","EVT_LF","EVT_CLASS")
cali_vegetation<-cali_vegetation%>%left_join(lf_small,by=join_by(closest("_mean">=VALUE)))
```

#Cleaning up new data set

```
cali_vegetation<-cali_vegetation%>%select(-c("STATEFP","COUNTYFP","TRACTCE","AFFGEOID","NAME","NAMELSAD"
```

#Combing weather, topography, and vegetation

```r
cali_features<-cali_topography%>%left_join(cali_cweather,by=join_by(County==county_name))
cali_features<-cali_features%>%select(-c("county_Id","State"))
cali_vegetation<-cali_vegetation%>% mutate(GEOID = paste("0", GEOID, sep = ""))
```

#Census Tract 9901 does not have vegetation as it is the shoreline, replacing NAs with Water

```r
cali_features<-cali_features%>%left_join(cali_vegetation,by=join_by(GEOID))
cali_features<-cali_features%>%mutate(EVT_LF=replace_na(EVT_LF,"Water"))
```

#saving export

```r
write.csv(cali_features,"caliTracts_features.csv")
```

#Reducing predictors from NRI before the combination

```r
nri_data<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/NRI_Table_CensusTracts_Subset.csv")
nri<-nri_data%>%select("STATE","STATEABBRV","STATEFIPS","COUNTY","COUNTYTYPE","COUNTYFIPS","STCOFIPS","
```

#only CA cases, converting categorical to binary

```r
nri<-nri%>%filter(STATEABBRV=="CA")
nri<-nri%>%mutate(WFRI_R=case_when(WFIR_RISKV<13000~0,WFIR_RISKV>13000~1))
```

#Removal of extra columns

```r
nri<-nri%>%select(-c(WFIR_EVNTS,WFIR_HLRR,WFIR_EALR))
```

#combination of cali features

```r
cali_features<-read_csv("C:/Users/walki/Documents/GitHub/D698/Datasets/caliTracts_features.csv")

nri_cali<-nri%>%inner_join(cali_features,by=join_by(TRACTFIPS==GEOID))
nri_cali<-nri_cali%>%select(-c(STATE,STATEABBRV,STATEFIPS,COUNTY,COUNTYTYPE,COUNTYFIPS,STCOFIPS,TRACT,.
```

#renaming columns for reference

```r
nri_cali<-nri_cali%>%rename(TRCT_WAREA=water_Area,TRCT_SLOPE=tract_avgSlope,CNTY_ELEV=county_avgElevati

write.csv(nri_cali,"nri_cali.csv")
```

###Data Exploration ##using nri.cali Dataset

```r
cali_data<-read_csv("C:/Users/walki/Documents/GitHub/D698/nri_cali.csv")
```

```
## New names:
## Rows: 9098 Columns: 34
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): TRACTFIPS, TRCT_VEGLF dbl (32): ...1, POPULATION, AREA, DRGT_EVNTS,
## DRGT_AFREQ, DRGT_HLRA, HWAV_EV...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

#Removing non important columns #Note: WFIR_RISKS AND WFIR_RISV holds actual percentages of the likelihood, cannot be included as predictor value

```
c_data<-cali_data%>%select(-c("...1","WFIR_RISKV","WFIR_RISKS","WFIR_HLRB"))
```

#Looking at the summary of all variables in the data set

```
summary(c_data)
```

```
##   TRACTFIPS           POPULATION         AREA             DRGT_EVNTS
##  Length:9098        Min.   :    0    Min.   :   0.008    Min.   :   0
##  Class :character   1st Qu.: 3209    1st Qu.:   0.379    1st Qu.:1148
##  Mode  :character   Median : 4194    Median :   0.693    Median :1372
##                     Mean   : 4340    Mean   :  16.190    Mean   :1330
##                     3rd Qu.: 5350    3rd Qu.:   1.644    3rd Qu.:1624
##                     Max.   :37562    Max.   :7024.460    Max.   :2261
##    DRGT_AFREQ       DRGT_HLRA          HWAV_EVNTS        HWAV_AFREQ
##  Min.   :  0.00   Min.   :0.0000121   Min.   :  0.00   Min.   : 0.000
##  1st Qu.: 52.18   1st Qu.:0.0000121   1st Qu.: 20.94   1st Qu.: 1.357
##  Median : 62.36   Median :0.0000121   Median : 35.00   Median : 2.167
##  Mean   : 60.47   Mean   :0.0006693   Mean   : 41.07   Mean   : 2.550
##  3rd Qu.: 73.82   3rd Qu.:0.0016279   3rd Qu.: 66.85   3rd Qu.: 4.139
##  Max.   :102.77   Max.   :0.0036879   Max.   :240.00   Max.   :14.861
##    HWAV_HLRA          LTNG_EVNTS       LTNG_AFREQ         SWND_EVNTS
##  Min.   :8.000e-10   Min.   :  0.0   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.:4.569e-05   1st Qu.: 12.0   1st Qu.: 0.5254   1st Qu.: 2.000
##  Median :7.618e-05   Median : 17.0   Median : 0.7727   Median : 5.000
##  Mean   :7.297e-05   Mean   : 21.6   Mean   : 0.9761   Mean   : 4.387
##  3rd Qu.:1.057e-04   3rd Qu.: 24.0   3rd Qu.: 1.0909   3rd Qu.: 6.000
##  Max.   :2.357e-04   Max.   :302.0   Max.   :13.7060   Max.   :11.000
##    SWND_AFREQ        SWND_HLRA           WFIR_AFREQ          WFIR_EXPA
##  Min.   :0.00000   Min.   :3.030e-08   Min.   :0.000000   Min.   :        0
##  1st Qu.:0.02971   1st Qu.:1.078e-06   1st Qu.:0.000000   1st Qu.:        0
##  Median :0.13369   Median :1.766e-06   Median :0.000000   Median :        0
##  Mean   :0.11547   Mean   :2.775e-05   Mean   :0.002021   Mean   :   563272
##  3rd Qu.:0.17430   3rd Qu.:2.114e-05   3rd Qu.:0.001067   3rd Qu.:        0
##  Max.   :1.46035   Max.   :8.467e-04   Max.   :0.063501   Max.   :258377478
##    WFIR_EXPT          WFIR_EXP_AREA        WFIR_HLRP            WFIR_HLRA
##  Min.   :0.000e+00   Min.   : 0.00000   Min.   :1.784e-06   Min.   :4.620e-07
##  1st Qu.:0.000e+00   1st Qu.: 0.00000   1st Qu.:6.677e-06   1st Qu.:6.600e-07
##  Median :0.000e+00   Median : 0.00000   Median :1.960e-05   Median :8.470e-07
##  Mean   :1.903e+09   Mean   : 0.15782   Mean   :4.423e-05   Mean   :9.784e-04
##  3rd Qu.:1.102e+09   3rd Qu.: 0.02316   3rd Qu.:1.973e-05   3rd Qu.:5.797e-05
##  Max.   :8.390e+10   Max.   :31.74707   Max.   :9.962e-04   Max.   :2.701e-02
##    WFIR_EALT          WFIR_EALS         WFIR_ALRA            WFRI_R
##  Min.   :       0   Min.   :  0.00   Min.   :0.000e+00   Min.   :0.0000
##  1st Qu.:       0   1st Qu.:  0.00   1st Qu.:0.000e+00   1st Qu.:0.0000
##  Median :       0   Median :  0.00   Median :0.000e+00   Median :0.0000
##  Mean   :  152785   Mean   : 29.78   Mean   :7.176e-07   Mean   :0.2071
##  3rd Qu.:    2518   3rd Qu.: 76.73   3rd Qu.:0.000e+00   3rd Qu.:0.0000
##  Max.   :29548172   Max.   :100.00   Max.   :1.377e-04   Max.   :1.0000
##    TRCT_WAREA          TRCT_SLOPE        CNTY_ELEV         CNTY_TEMP
##  Min.   :0.000e+00   Min.   :0.0000   Min.   : -84.0   Min.   :45.50
```

```
##  1st Qu.:0.000e+00   1st Qu.:0.4167   1st Qu.:  27.0   1st Qu.:60.50
##  Median :0.000e+00   Median :0.7900   Median :  96.0   Median :63.70
##  Mean   :8.072e+05   Mean   :1.6222   Mean   : 193.9   Mean   :62.85
##  3rd Qu.:4.108e+03   3rd Qu.:2.4142   3rd Qu.: 234.0   3rd Qu.:64.20
##  Max.   :1.098e+09   Max.   :9.0156   Max.   :2534.0   Max.   :75.50
##   CNTY_PRECIP     TRCT_VEGLF
##  Min.   : 2.17   Length:9098
##  1st Qu.: 7.64   Class :character
##  Median : 8.64   Mode  :character
##  Mean   :11.04
##  3rd Qu.:13.45
##  Max.   :52.90
```

#checking for null values

```
colSums(is.na(c_data))
```

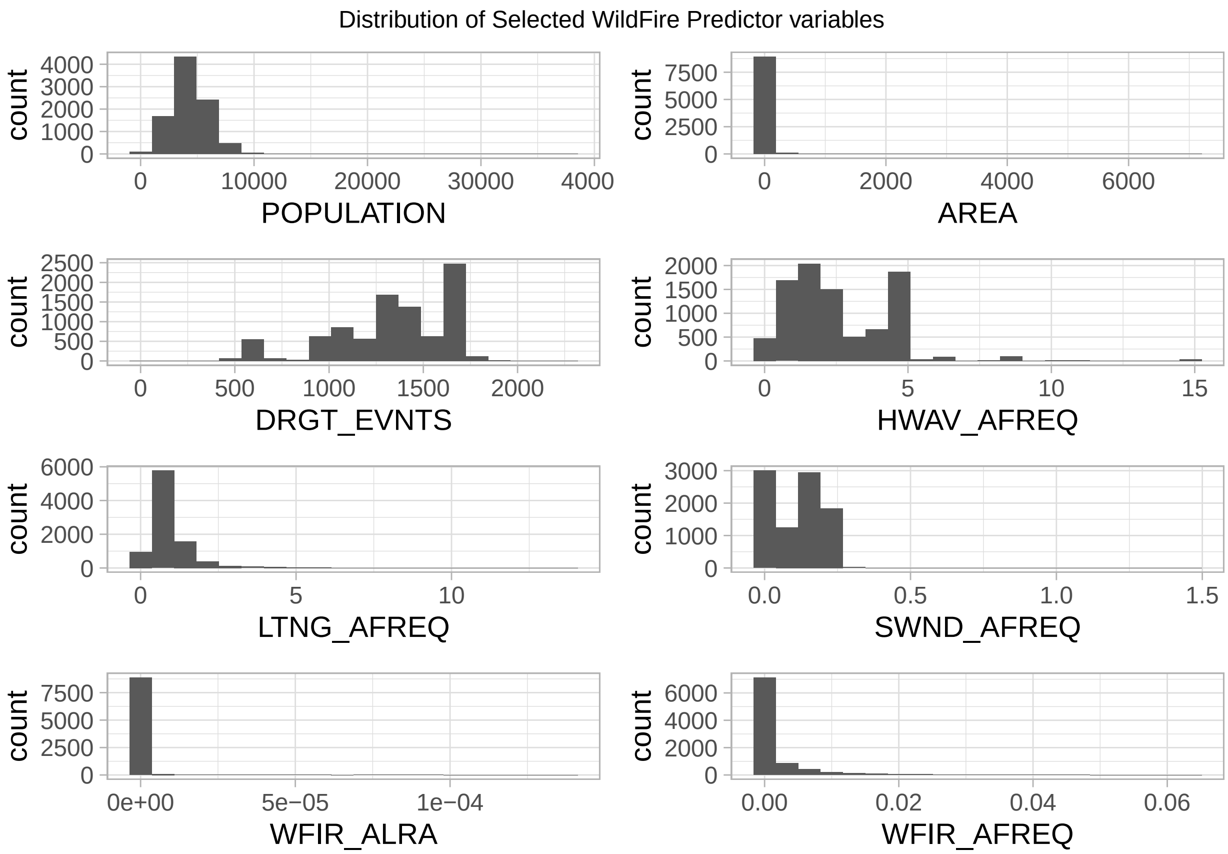
```
##     TRACTFIPS    POPULATION          AREA    DRGT_EVNTS    DRGT_AFREQ
##             0             0             0             0             0
##     DRGT_HLRA    HWAV_EVNTS    HWAV_AFREQ     HWAV_HLRA    LTNG_EVNTS
##             0             0             0             0             0
##    LTNG_AFREQ    SWND_EVNTS    SWND_AFREQ     SWND_HLRA    WFIR_AFREQ
##             0             0             0             0             0
##     WFIR_EXPA     WFIR_EXPT WFIR_EXP_AREA     WFIR_HLRP     WFIR_HLRA
##             0             0             0             0             0
##     WFIR_EALT     WFIR_EALS     WFIR_ALRA        WFRI_R    TRCT_WAREA
##             0             0             0             0             0
##    TRCT_SLOPE     CNTY_ELEV     CNTY_TEMP   CNTY_PRECIP    TRCT_VEGLF
##             0             0             0             0             0
```

#Reviewing the current distribution of the predictor values. See if there's future transformations

```
g1<-c_data%>%ggplot(aes(x=POPULATION))+geom_histogram(bins=20)+theme_light()
g2<-c_data%>%ggplot(aes(x=AREA))+geom_histogram(bins=20)+theme_light()
g3<-c_data%>%ggplot(aes(x=DRGT_EVNTS))+geom_histogram(bins=20)+theme_light()
g4<-c_data%>%ggplot(aes(x=HWAV_AFREQ))+geom_histogram(bins=20)+theme_light()
g7<-c_data%>%ggplot(aes(x=WFIR_AFREQ))+geom_histogram(bins=20)+theme_light()
g10<-c_data%>%ggplot(aes(x=WFIR_ALRA))+geom_histogram(bins=20)+theme_light()
g14<-c_data%>%ggplot(aes(x=LTNG_AFREQ))+geom_histogram(bins=20)+theme_light()
g22<-c_data%>%ggplot(aes(x=SWND_AFREQ))+geom_histogram(bins=20)+theme_light()
```

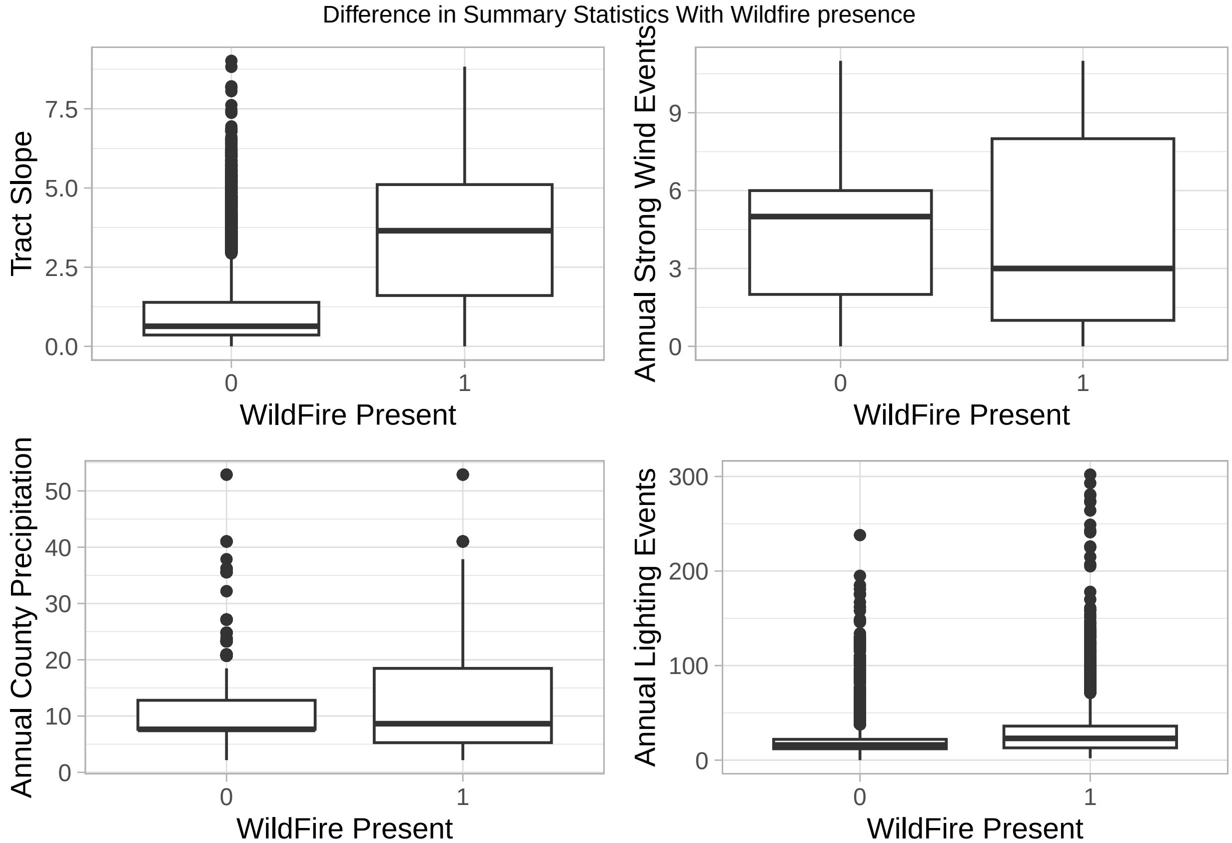
#Plot for project write up, only a selected few

```
plt1<-ggarrange(g1,g2,g3,g4,g14,g22,g10,g7,nrow =4,ncol =2,align="h",heights = 2,font.label = list(size

annotate_figure(plt1,top = text_grob("Distribution of Selected WildFire Predictor variables ",size=9))
```



#Reviewing a few variables on its boxplots in reflection with the response variable WFRI_R

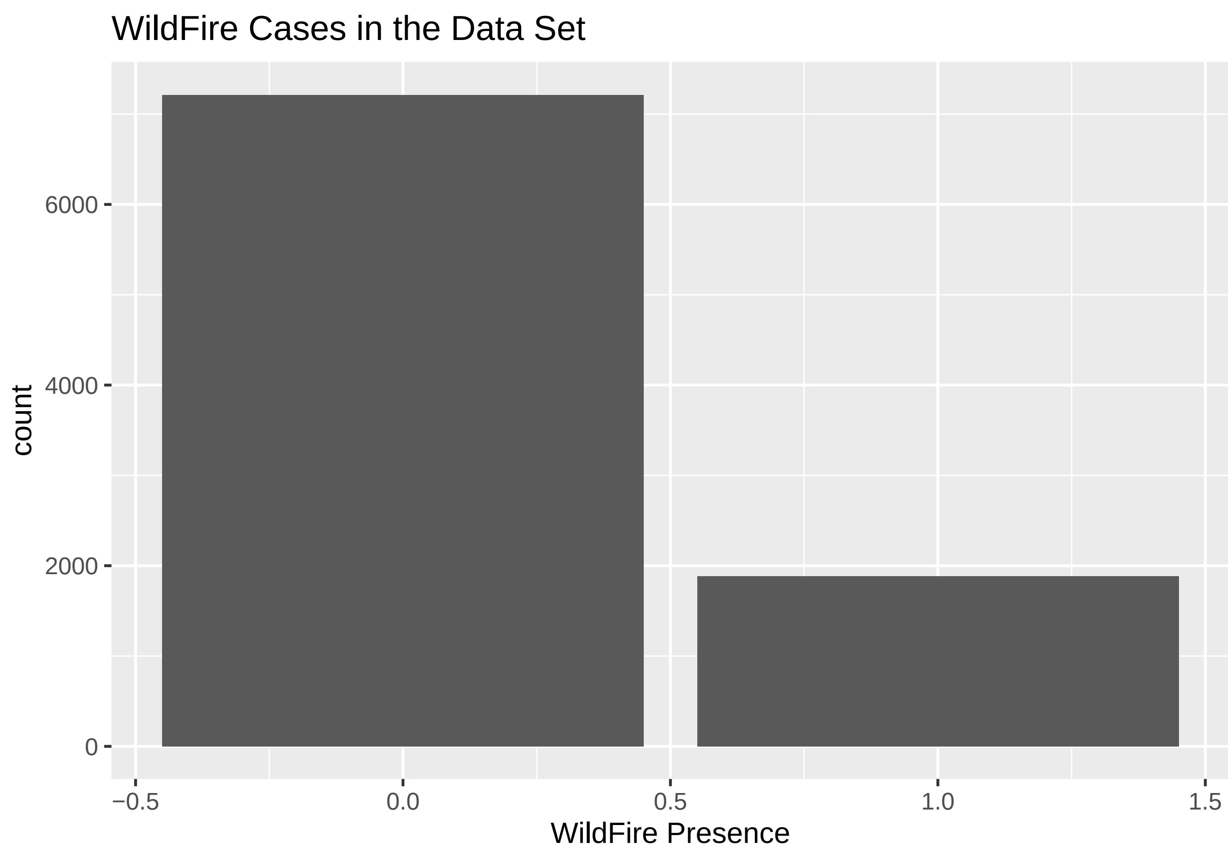
```
g1<-c_data%>%ggplot(aes(y=TRCT_SLOPE,x=factor(WFRI_R)))+geom_boxplot()+theme_light()+labs(x="WildFire Pr
g2<-c_data%>%ggplot(aes(y=SWND_EVNTS,x=factor(WFRI_R)))+geom_boxplot()+theme_light()+labs(x="WildFire Pr
g3<-c_data%>%ggplot(aes(y=CNTY_PRECIP,x=factor(WFRI_R)))+geom_boxplot()+theme_light()+labs(x="WildFire P
g4<-c_data%>%ggplot(aes(y=LTNG_EVNTS,x=factor(WFRI_R)))+geom_boxplot()+theme_light()+labs(x="WildFire Pr

plt1<-ggarrange(g1,g2,g3,g4,nrow =2,ncol =2,align="h",heights = 2,font.label = list(size =3, color = "bl

annotate_figure(plt1,top = text_grob("Difference in Summary Statistics With Wildfire presence",size=9))
```



#Checking if the data set is imbalanced

```
c_data%>%ggplot(aes(fill=WFRI_R))+geom_bar(aes(x=WFRI_R))+labs(title="WildFire Cases in the Data Set",x=
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a 'group' aesthetic or to convert a numerical
##   variable into a factor?
```



###Data Preparation

#transforming the tract vegetation life into binary dummy variables via mutate

```
unique(c_data$TRCT_VEGLF)
```

```
## [1] "Shrub"       "Tree"        "Developed"   "Herb"        "Agriculture"
## [6] "Sparse"      "Barren"      "Snow-Ice"    "Water"
```

```
c_data<-c_data%>%mutate(.isShrub=if_else(TRCT_VEGLF=="Shrub",1,0),
                        .isTree=if_else(TRCT_VEGLF=="Tree",1,0),
                        .isDeveloped=if_else(TRCT_VEGLF=="Developed",1,0),
                        .isHerb=if_else(TRCT_VEGLF=="Herb",1,0),
                        .isArgiculture=if_else(TRCT_VEGLF=="Argiculture",1,0),
                        .isSparse=if_else(TRCT_VEGLF=="Sparse",1,0),
                        .isBarren=if_else(TRCT_VEGLF=="Barren",1,0),
                        .isSnowIce=if_else(TRCT_VEGLF=="Snow-Ice",1,0),
                        )
c_data<-c_data%>%select(-c("TRCT_VEGLF"))
```

#Seeing if there's multi-collinearity in the current predictors #Drought events and drought frequency are highly correlated, Let's see with variable selection if one of the variables is dropped from the optimized predictor set

```
temp<-c_data%>%select(-c("WFRI_R","TRACTFIPS"))
temp<-cor(temp)
```

```
## Warning in stats::cor(x, ...): the standard deviation is zero
```

#setting the binary response and a few predictor variables as a factor before modeling

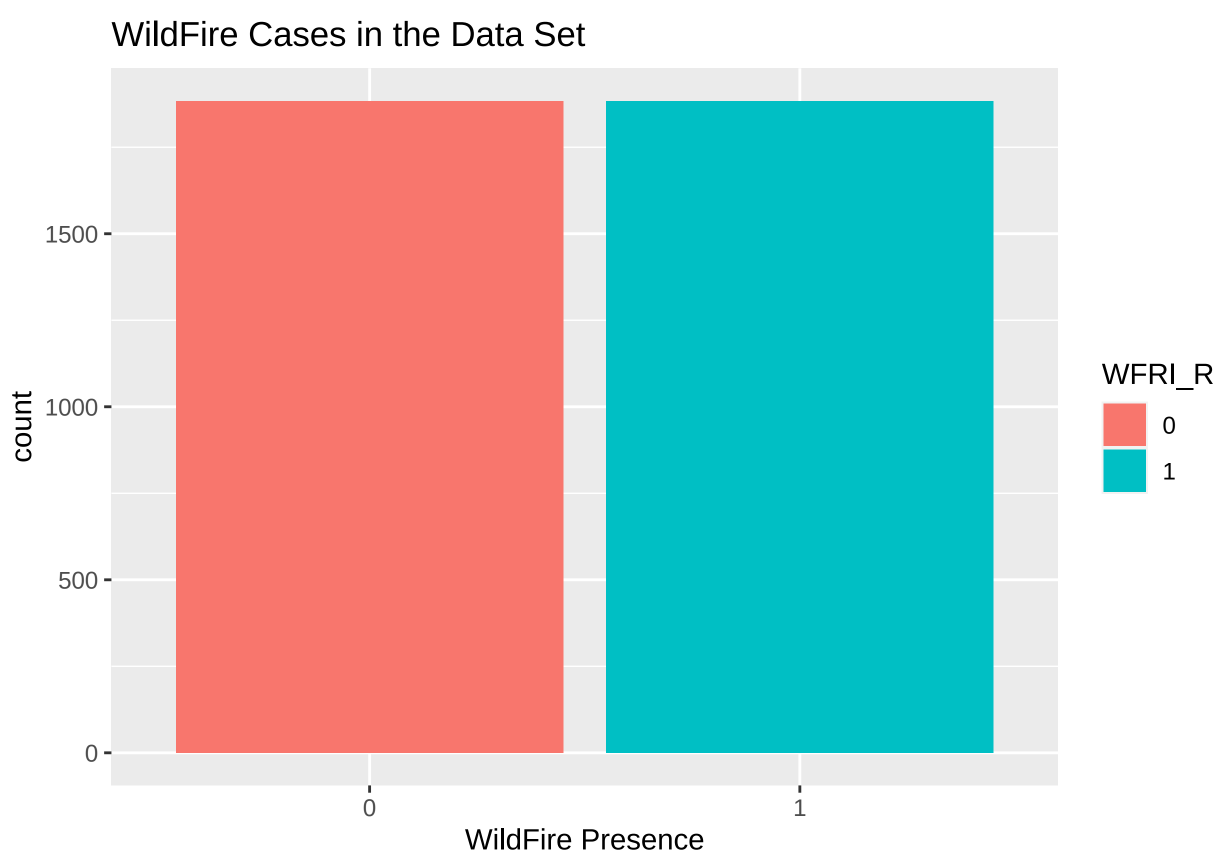
```
c_data<-c_data%>%mutate_at(c('WFRI_R',".isShrub",".isTree",".isDeveloped",".isHerb",".isArgiculture",".
```

#Downsampling the non wildfire cases so the models can predict more fire cases

```
dwn_data<-downSample(x=c_data[,-ncol(c_data)],y=c_data$WFRI_R)
```

#Seeing new distribution

```
dwn_data%>%ggplot(aes(fill=WFRI_R))+geom_bar(aes(x=WFRI_R))+labs(title="WildFire Cases in the Data Set"
```



#splitting data set into testing and training

```
temp<-sample.split(dwn_data$WFRI_R,SplitRatio = 0.7)
training_data<-subset(dwn_data,temp==TRUE)
test_data<-subset(dwn_data,temp==FALSE)
```

```r
names(test_data)
```

```
##  [1] "TRACTFIPS"      "POPULATION"     "AREA"           "DRGT_EVNTS"
##  [5] "DRGT_AFREQ"     "DRGT_HLRA"      "HWAV_EVNTS"     "HWAV_AFREQ"
##  [9] "HWAV_HLRA"      "LTNG_EVNTS"     "LTNG_AFREQ"     "SWND_EVNTS"
## [13] "SWND_AFREQ"     "SWND_HLRA"      "WFIR_AFREQ"     "WFIR_EXPA"
## [17] "WFIR_EXPT"      "WFIR_EXP_AREA"  "WFIR_HLRP"      "WFIR_HLRA"
## [21] "WFIR_EALT"      "WFIR_EALS"      "WFIR_ALRA"      "WFRI_R"
## [25] "TRCT_WAREA"     "TRCT_SLOPE"     "CNTY_ELEV"      "CNTY_TEMP"
## [29] "CNTY_PRECIP"    ".isShrub"       ".isTree"        ".isDeveloped"
## [33] ".isHerb"        ".isArgiculture" ".isSparse"      ".isBarren"
## [37] "Class"
```

```r
x_traindata <- setdiff(names(training_data), c("WFRI_R"))
names(training_data)
```

```
##  [1] "TRACTFIPS"      "POPULATION"     "AREA"           "DRGT_EVNTS"
##  [5] "DRGT_AFREQ"     "DRGT_HLRA"      "HWAV_EVNTS"     "HWAV_AFREQ"
##  [9] "HWAV_HLRA"      "LTNG_EVNTS"     "LTNG_AFREQ"     "SWND_EVNTS"
## [13] "SWND_AFREQ"     "SWND_HLRA"      "WFIR_AFREQ"     "WFIR_EXPA"
## [17] "WFIR_EXPT"      "WFIR_EXP_AREA"  "WFIR_HLRP"      "WFIR_HLRA"
## [21] "WFIR_EALT"      "WFIR_EALS"      "WFIR_ALRA"      "WFRI_R"
## [25] "TRCT_WAREA"     "TRCT_SLOPE"     "CNTY_ELEV"      "CNTY_TEMP"
## [29] "CNTY_PRECIP"    ".isShrub"       ".isTree"        ".isDeveloped"
## [33] ".isHerb"        ".isArgiculture" ".isSparse"      ".isBarren"
## [37] "Class"
```

Data Analysis

#connection to h20 server

```r
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         16 minutes 6 seconds
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.42.0.2
##     H2O cluster version age:    4 months and 10 days
##     H2O cluster name:           H2O_started_from_R_walki_kmz484
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.41 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.1.2 (2021-11-01)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is (4 months and 10 days) old. There may be a newer version available.
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.com/h2o/latest_
```

#Uploading the data set into h2o and splitting the data set into training/test. Choosing a 70/30 split and splitting testing for a validation test

```
train.h2o<-as.h2o(training_data)
```

```
##   |                                                                                |
```

```
test.h2o<-as.h2o(test_data)
```

```
##   |                                                                                |
```

#Model 1| Random Forest

```
#removed wildfire exposure features it's possibly tied to the  likelihood response(,"WFIR_EXPA","WFIR_E
features<-c("POPULATION","AREA","DRGT_AFREQ","DRGT_HLRA","HWAV_EVNTS","HWAV_AFREQ","HWAV_HLRA","LTNG_EV
response<-c("WFRI_R")
```

##Version 1 of Random Forest

```
#V1: Stopping metrics based on AUC score as preventing for overfitting and added Cross-validation with 
rf.model<-h2o.randomForest(x = features, y =response , training_frame = train.h2o, stopping_rounds = 5,
```

```
##   |                                                                                |
```

```
#see the first version of the tree structure
rf.model@model$model_summary
```

```
#look at cross-validation results from the model, Not much difference between cases
rf.cross<-rf.model@model$cross_validation_metrics_summary%>%select(-c(mean,sd))
#rf.cross
```

```
#review the feature importance of this random forest model and using the highest gini indexes in the fi
rf.features<-h2o.varimp(rf.model)
features_v2<-rf.features$variable[1:10]%>%as.vector()
```

##Version 2| RF

```
#Shortening max depth and trees for more conservative AUC. Applying highest gini index features to the 
rf.v2<-h2o.randomForest(x = features_v2, y =response , training_frame = train.h2o, stopping_rounds = 5,
```

```
##   |                                                                                |
```

RF Verison 2 - Model Performance

```r
rf2_perf <- h2o.performance(rf.v2, test.h2o)
rf2_perf
```

```
## H2OBinomialMetrics: drf
##
## MSE:  0.03868792
## RMSE:  0.1966925
## LogLoss:  0.1467657
## Mean Per-Class Error:  0.04424779
## AUC:  0.9882356
## AUCPR:  0.9863941
## Gini:  0.9764711
## R^2:  0.8452483
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##          0   1    Error      Rate
## 0      524  41 0.072566   =41/565
## 1        9 556 0.015929    =9/565
## Totals 533 597 0.044248  =50/1130
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                       max f1  0.532212   0.956971 259
## 2                       max f2  0.485115   0.976290 269
## 3                 max f0point5  0.720246   0.957605 209
## 4                 max accuracy  0.532212   0.955752 259
## 5                max precision  0.998871   1.000000   0
## 6                   max recall  0.128207   1.000000 302
## 7              max specificity  0.998871   1.000000   0
## 8             max absolute_mcc  0.532212   0.912970 259
## 9   max min_per_class_accuracy  0.653353   0.948673 230
## 10 max mean_per_class_accuracy  0.532212   0.955752 259
## 11                     max tns  0.998871 565.000000   0
## 12                     max fns  0.998871 562.000000   0
## 13                     max fps  0.007579 565.000000 399
## 14                     max tps  0.128207 565.000000 302
## 15                     max tnr  0.998871   1.000000   0
## 16                     max fnr  0.998871   0.994690   0
## 17                     max fpr  0.007579   1.000000 399
## 18                     max tpr  0.128207   1.000000 302
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

```r
rf2_pred <- h2o.predict(rf.v2,test.h2o)
```

```
##   |                                                                                |
```

```r
pred1 <- as.data.frame(rf2_pred$predict)
pred1$predict <- factor(pred1$predict, levels = c(0, 1))
mean(pred1$predict==test_data$WFRI_R)
```

```
## [1] 0.9495575
```

```r
#plotting logloss of the revised model
plot(rf.v2)
```

RF Version2 - Predictions

```r
#from the revised model, pull results from prediction against test set
rf.pred<-h2o.predict(rf.v2,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                              |
```

```r
rf.precsnprob<-h2o.predict(rf.v2,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                              |
```

```r
rf.reclprob<-h2o.predict(rf.v2,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                              |
```

RF Version 2 - confusion matrix of rf predictions

```r
rf.con<-confusionMatrix(rf.pred,test_data$WFRI_R,positive = "1",mode = "prec_recall")

rf.con
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 531  23
##          1  34 542
##
##                Accuracy : 0.9496
##                  95% CI : (0.9351, 0.9616)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8991
##
##  Mcnemar's Test P-Value : 0.1853
##
##               Precision : 0.9410
##                  Recall : 0.9593
##                      F1 : 0.9500
##              Prevalence : 0.5000
##          Detection Rate : 0.4796
##    Detection Prevalence : 0.5097
##       Balanced Accuracy : 0.9496
##
##        'Positive' Class : 1
##
```

RF Version 2 - storing confusion matrix results

```
confusionM<-rf.con$byClass%>%as.data.frame()%>%t()
confusionM<-as.data.frame(confusionM)
confusionM<-confusionM%>%rename(Pos_pred_value="Pos Pred Value",Neg_Pred_value="Neg Pred Value",Detecti
#confusionM
```

#creating a reference table w/ predicted probabilities, actual, and predictions all in one #see summary results of random forest model on the test data

```
rf.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-rf.pred,
      N<-rf.reclprob,
      Y<-rf.precsnprob
  )

rf.summary<-rf.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....rf.pred", N="N....rf.reclpr

rf.auc<-roc(rf.summary$obs,Y)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
test.results<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(rf.summary$pred)),y_true =as.numeric(as.character( rf
  t.mse<-MSE(as.numeric(as.character(rf.summary$pred)),as.numeric(as.character(rf.summary$obs))),
  t.RSME<-RMSE(as.numeric(as.character(rf.summary$pred)),as.numeric(as.character(rf.summary$obs))),
  t.AUC<-rf.auc$auc,
  t.ClassError<-mean(rf.summary$pred!=rf.summary$obs)

)
```

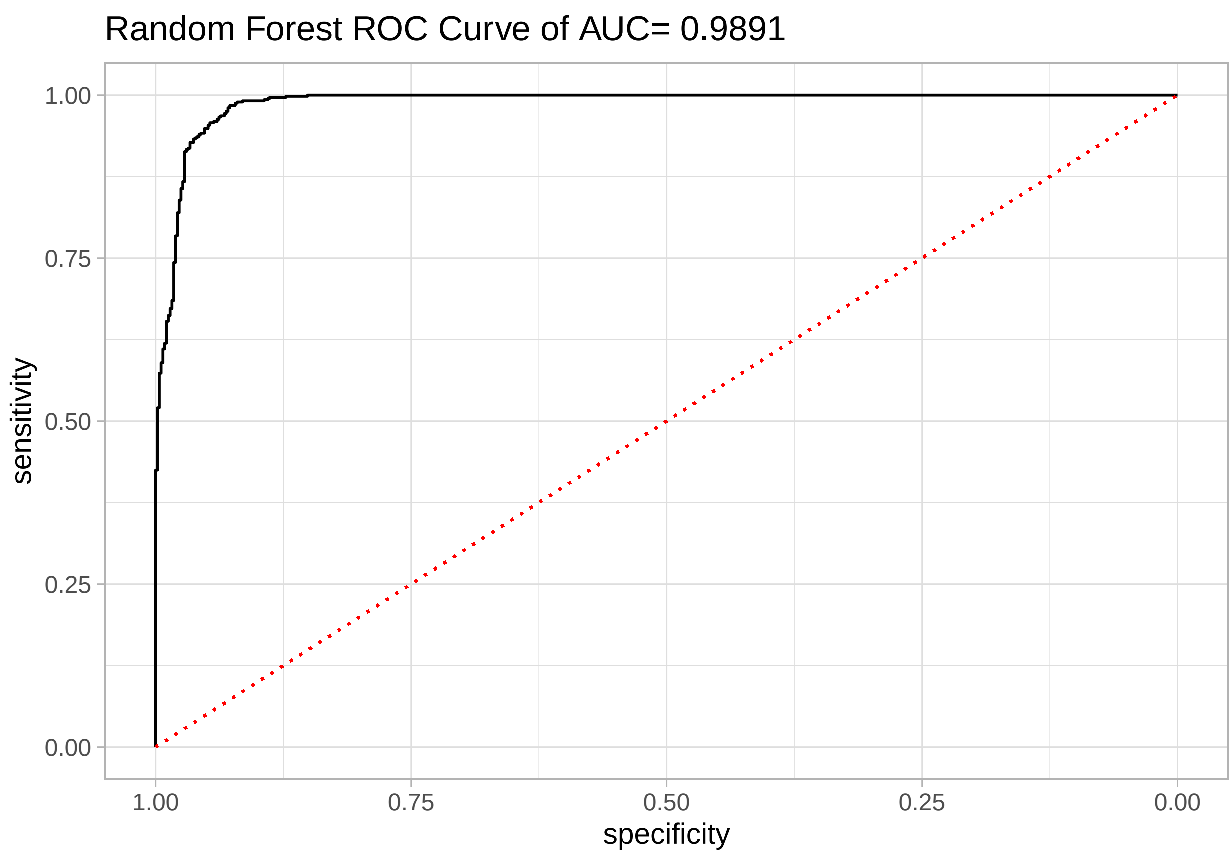
#Stores Test Performance of Models

```
test.results<-test.results%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.rf.summary.pr
test.results
```

```
##          R2        MSE     RMSE       AUC classError
## 1 0.7982301 0.05044248 0.224594 0.9882434 0.05044248
```

#plotting ROC curve of Random Forest - Version 2

```
ggroc(rf.auc)+ggtitle("Random Forest ROC Curve of AUC= 0.9891")+geom_segment(aes(x=1,y=0,xend=0,yend=1)
```



#Model 2| Gradient Boosting Decision Tree

#Version 1| #Higher learning rate, Stopping metric on AUC as logloss saw lower R^2, Cross-validation on training set

```
gb_model<-h2o.gbm(x=features,y=response,training_frame = train.h2o,learn_rate = 0.1,ntrees=1000,stoppin
```

```
##   |                                                                      |
```

#see the first version of the tree structure

```
gb_model@model$model_summary
gb.cross<-gb_model@model$cross_validation_metrics_summary%>%select(-c(mean,sd))
```

#Feature importance for gb_model

```
gb.features<-h2o.varimp(gb_model)
features_v2<-gb.features$variable[1:10]%>%as.vector()
```

#Version 2 #Lowering stopping rounds for a quicker review on the AUC score, Smaller tree size from v1

```
gb_v2<-h2o.gbm(x=features_v2,y=response,training_frame = train.h2o,learn_rate = 0.1,ntrees=45,stopping_
```

```
##   |                                                                      |
```

#see performance of prediction

```
gb.pred<-h2o.predict(gb_v2,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                              |
```

```
gb.precsnprob<-h2o.predict(gb_v2,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                              |
```

```
gb.reclprob<-h2o.predict(gb_v2,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                              |
```

#Print the confusion matrix

```
gb.con<-confusionMatrix(gb.pred,test_data$WFRI_R,positive = "1",mode = "prec_recall")

gb.con
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 530  34
##          1  35 531
##
##                Accuracy : 0.9389
##                  95% CI : (0.9234, 0.9522)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8779
##
##  Mcnemar's Test P-Value : 1
##
##               Precision : 0.9382
##                  Recall : 0.9398
##                      F1 : 0.9390
##              Prevalence : 0.5000
##          Detection Rate : 0.4699
##    Detection Prevalence : 0.5009
##       Balanced Accuracy : 0.9389
##
##        'Positive' Class : 1
##
```

#see summary results of random forest model on the test data

```r
gb.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-gb.pred,
      N<-gb.reclprob,
      Y<-gb.precsnprob
  )

gb.summary<-gb.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....gb.pred", N="N....gb.reclpr

gb.auc<-roc(gb.summary$obs,Y)
```

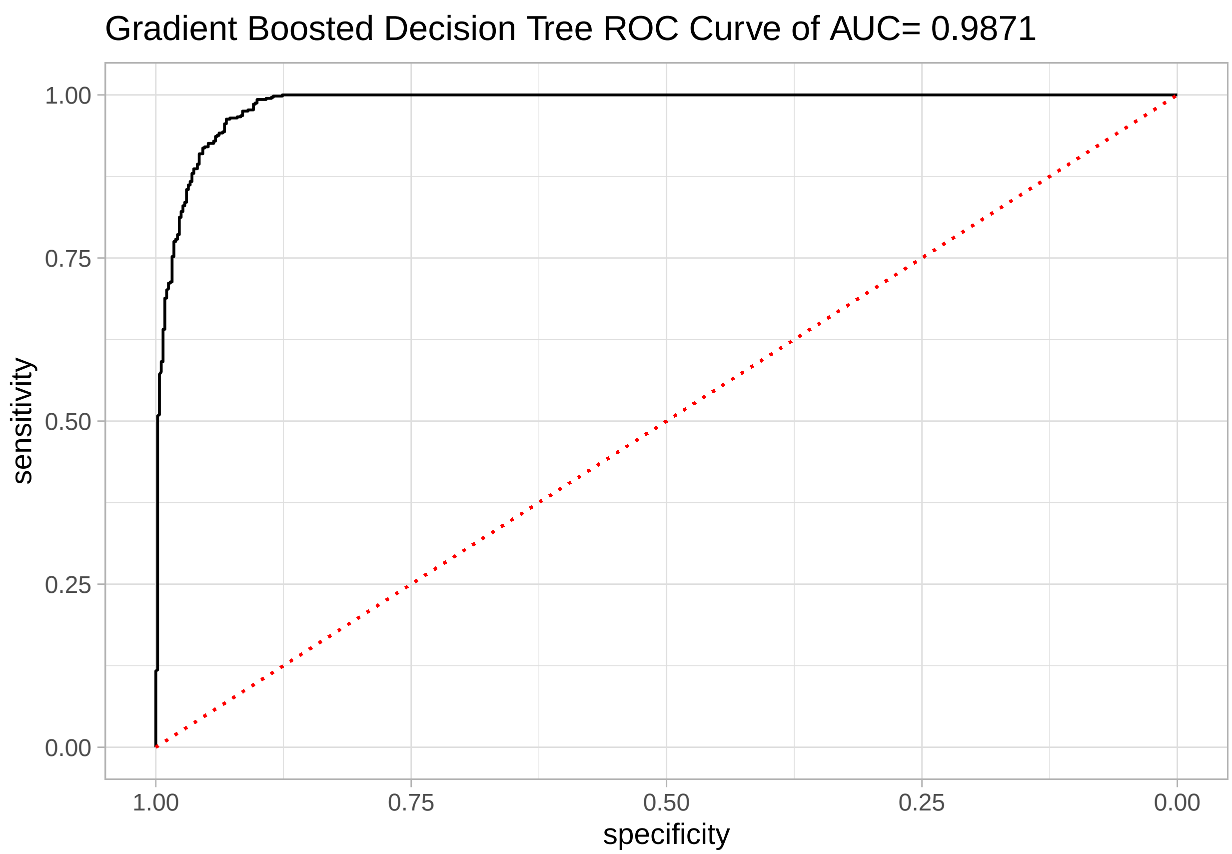
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
temp<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(gb.summary$pred)),y_true =as.numeric(as.character( gb
  t.mse<-MSE(as.numeric(as.character(gb.summary$pred)),as.numeric(as.character(gb.summary$obs))),
  t.RSME<-RMSE(as.numeric(as.character(gb.summary$pred)),as.numeric(as.character(gb.summary$obs))),
  t.AUC<-gb.auc$auc,
  t.ClassError<-mean(gb.summary$pred!=gb.summary$obs)

)
```

#plotting ROC curve of gradient boosted DT

```r
ggroc(gb.auc)+ggtitle("Gradient Boosted Decision Tree ROC Curve of AUC= 0.9871")+geom_segment(aes(x=1,y=
```




```
temp<-temp%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.gb.summary.pred....",MSE="t.m
,RMSE="t.RSME....RMSE.as.numeric.as.character.gb.summary.pred....as.numeric.as.character.gb.summary.obs
,AUC="t.AUC....gb.auc.auc",classError="t.ClassError....mean.gb.summary.pred....gb.summary.obs.")
```

#adding gradient boosting to the results data frame

```
test.results[2,]<-c(R2=temp$R2,MSE=temp$MSE,RMSE=temp$RMSE,AUC=temp$AUC,classError=temp$classError)
rownames(test.results)<-c("Random Forest","Gradient Boosting")
```

#storing confusion matrix testing results

```
temp<-gb.con$byClass%>%as.data.frame()%>%t()
temp<-as.data.frame(temp)

confusionM<-confusionM%>%add_row(Sensitivity= temp$Sensitivity,Specificity=temp$Specificity, Pos_pred_va

rownames(confusionM)<-c("Random Forest","Gradient Boosting")
```

##Part 1 of analysis RF VS GB

#Current review on the two models

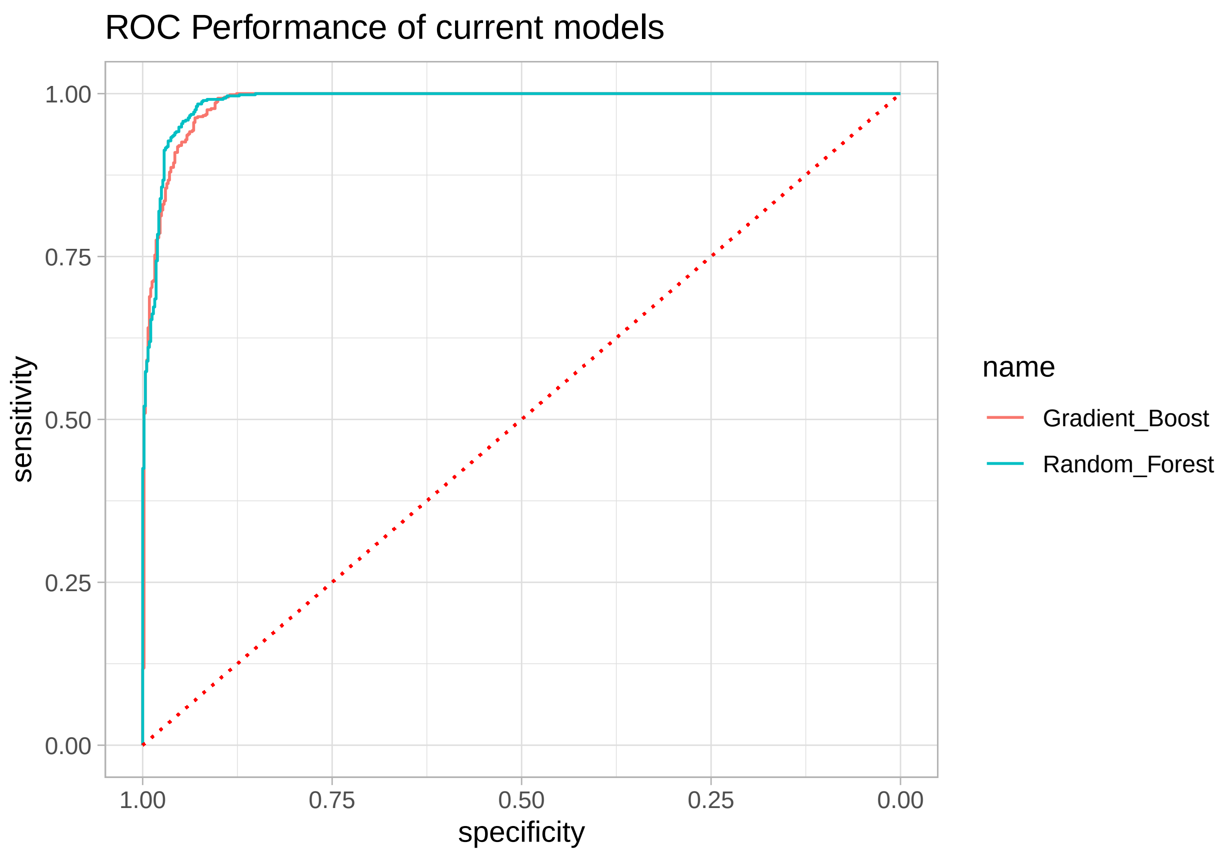
```
confusionM
```

```
##                    Sensitivity Specificity Pos_pred_value Neg_Pred_value
```

```
## Random Forest        0.959292   0.9398230      0.9409722      0.9584838
## Gradient Boosting    0.939823   0.9380531      0.9381625      0.9397163
##                   Precision   Recall         F1 Prevalence Detection_rate
## Random Forest     0.9409722 0.959292 0.9500438        0.5      0.4796460
## Gradient Boosting 0.9381625 0.939823 0.9389920        0.5      0.4699115
##                   Detection_prevalence Balanced_Accuracy
## Random Forest                0.5097345         0.9495575
## Gradient Boosting            0.5008850         0.9389381
```

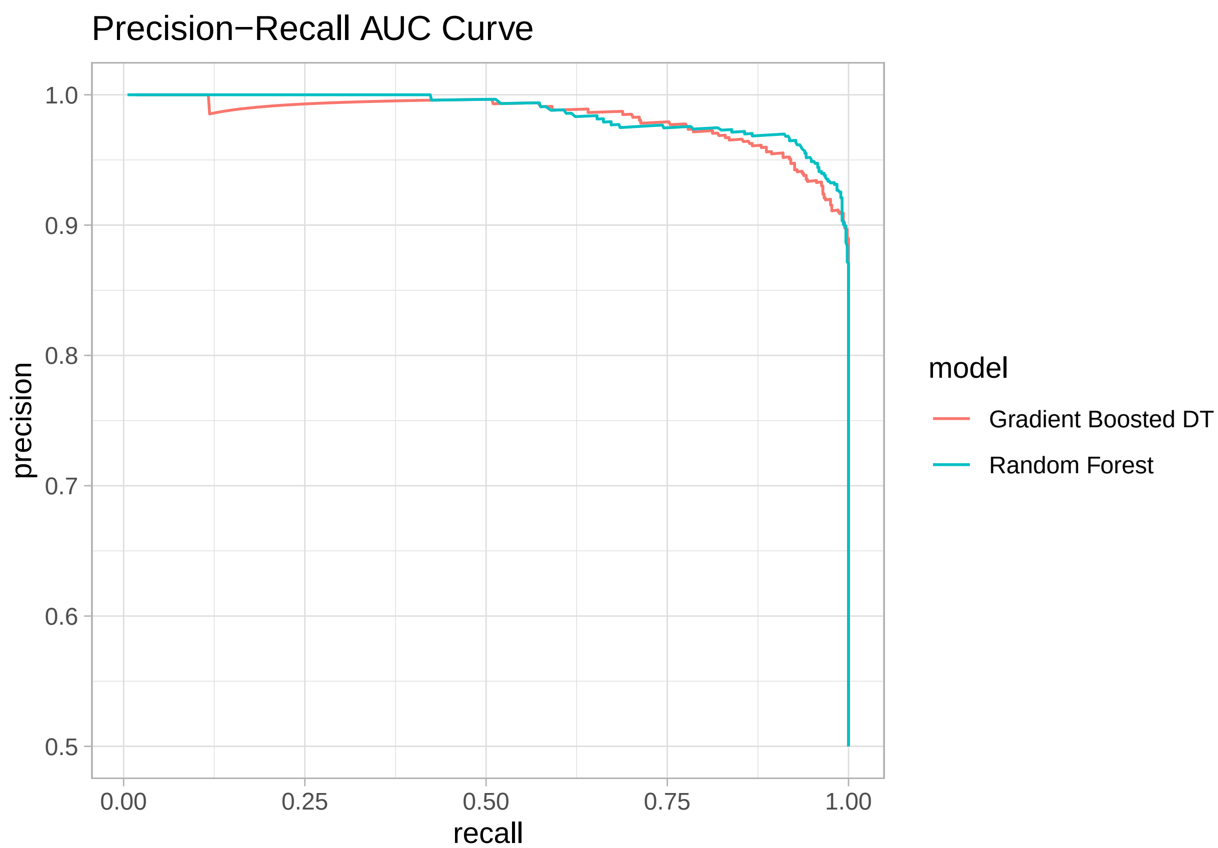
#see ROCs of gradient boosted DT and random forest

```
rocs<-list(Gradient_Boost=gb.auc,Random_Forest=rf.auc)
ggroc(rocs)+ggtitle("ROC Performance of current models")+geom_segment(aes(x=1,y=0,xend=0,yend=1),linety
```



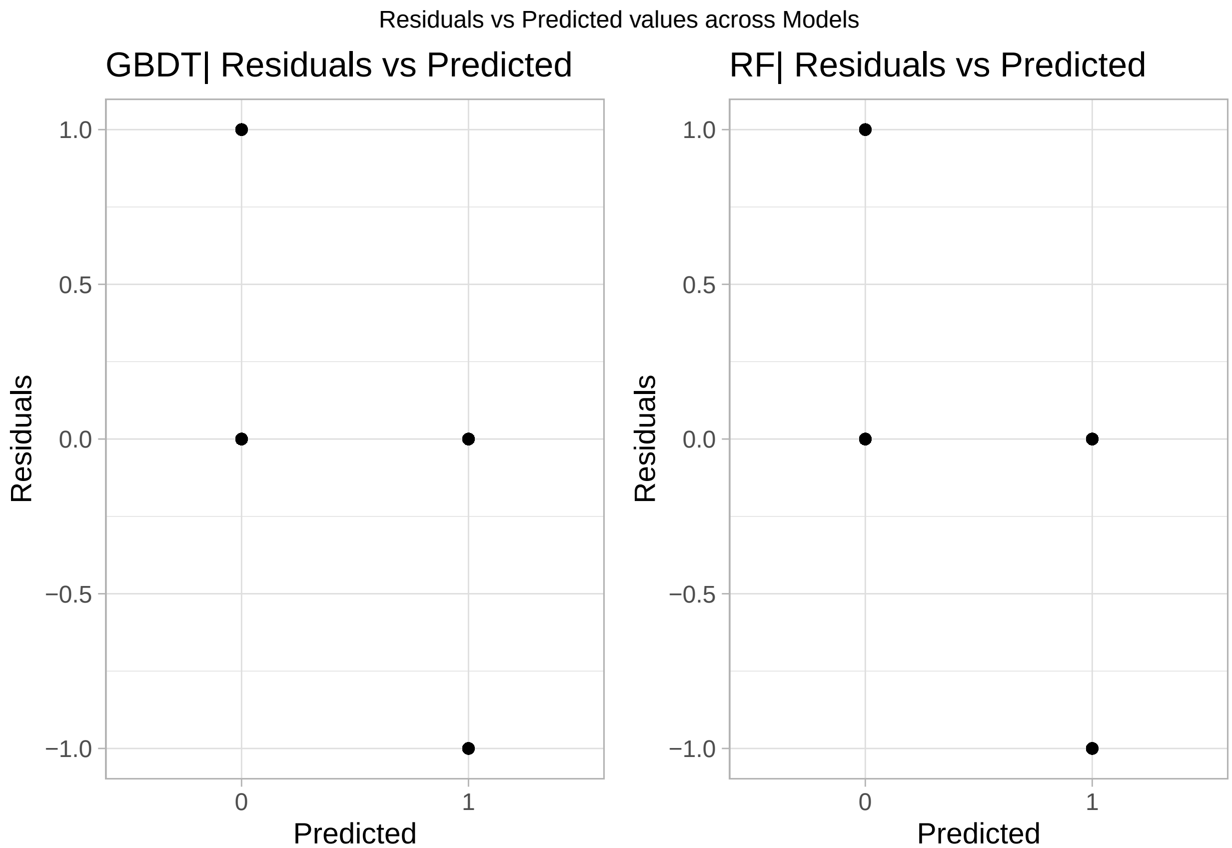
#see Precision-Recall Plots of the two models

```
rf.perf<-h2o.performance(rf.v2,test.h2o)%>%h2o.metric()%>%as.data.frame()%>%select(c(recall,precision))
rf.perf$model<-"Random Forest"
gb.perf<-h2o.performance(gb_v2,test.h2o)%>%h2o.metric()%>%as.data.frame()%>%select(c(recall,precision))
gb.perf$model<-"Gradient Boosted DT"
combine_rpplots<-rbind(rf.perf,gb.perf)

ggplot(combine_rpplots,aes(recall,precision,group=model,color=model))+geom_line()+labs(title ="Precisio
```



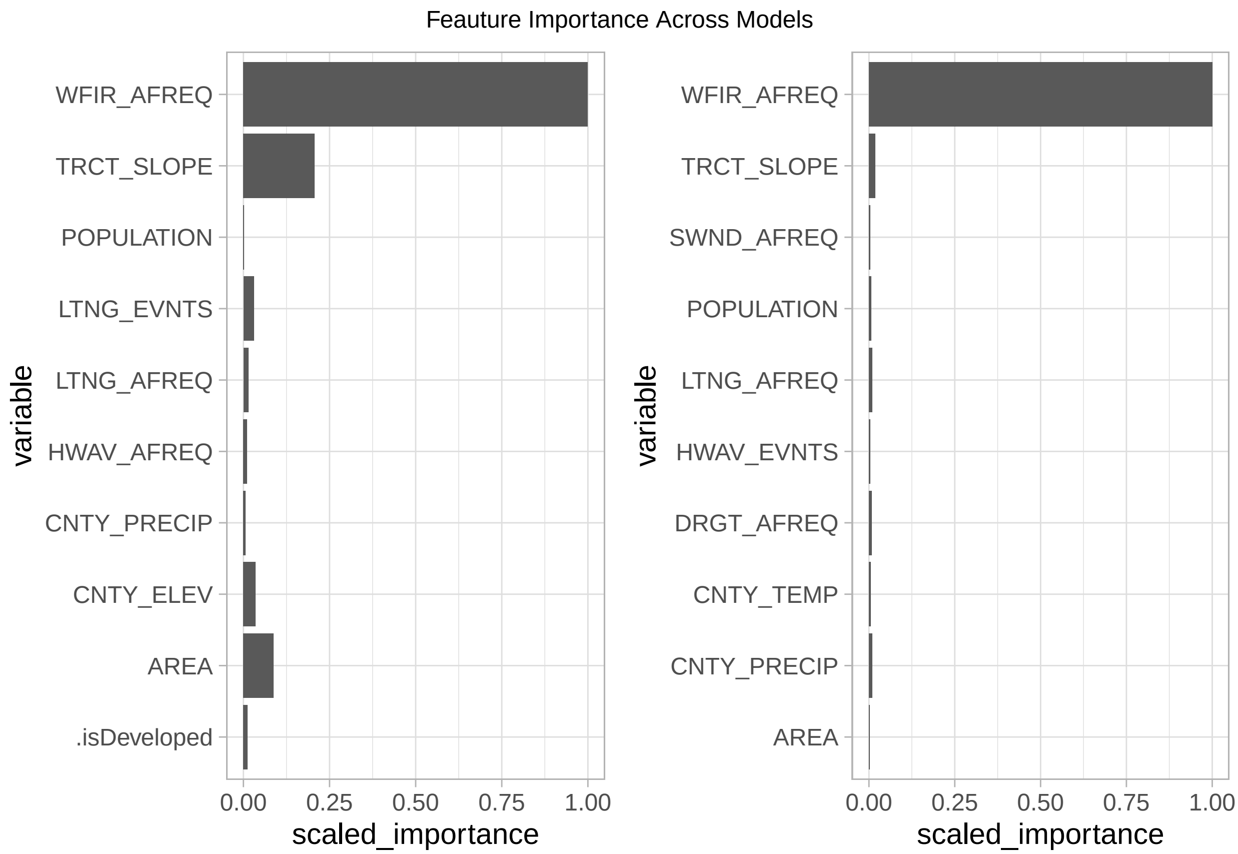
#Plotting residuals vs fitted

```
rf.summary<-rf.summary%>%mutate(resid=as.numeric(obs)-as.numeric(pred))
gb.summary<-gb.summary%>%mutate(resid=as.numeric(obs)-as.numeric(pred))
g1<-gb.summary%>%ggplot(aes(pred,resid))+geom_point()+labs(title="GBDT| Residuals vs Predicted",y="Resid
g2<-rf.summary%>%ggplot(aes(pred,resid))+geom_point()+labs(title="RF| Residuals vs Predicted",y="Residua

plt2<-ggarrange(g1,g2,ncol = 2)
annotate_figure(plt2,top = text_grob("Residuals vs Predicted values across Models",size=9))
```



#feature analysis. Plotting top ten feature by its gini score

```
rf.features<-h2o.varimp(rf.v2)%>%as.data.frame()
gb.features<-h2o.varimp(gb_v2)%>%as.data.frame()

g1<-rf.features%>%ggplot(aes(y=variable,x=scaled_importance))+geom_bar(stat="identity")+theme_light()
g2<-gb.features%>%ggplot(aes(y=variable,x=scaled_importance))+geom_bar(stat="identity")+theme_light()

plt3<-ggarrange(g1,g2,ncol = 2)
annotate_figure(plt3,top = text_grob("Feauture Importance Across Models",size=9))
```



Model 3 |Auto ML Models

Run AutoML for 5 base models using the "features" data.

```
aml <- h2o.automl(x = features, y = response,
                  training_frame = train.h2o,
                  max_models = 5,
                  seed = 1)
```

```
##   |                                                                              |
## 22:12:42.590: AutoML: XGBoost is not available; skipping it.  |
```

View the AutoML Leaderboard

```
lb <- aml@leaderboard
head(lb, n = nrow(lb))  # Print all rows instead of default (6 rows)
```

```
##                                                  model_id       auc   logloss
## 1 StackedEnsemble_BestOfFamily_1_AutoML_5_20231205_221242 0.9919661 0.1036018
## 2    StackedEnsemble_AllModels_1_AutoML_5_20231205_221242 0.9918423 0.1032283
## 3                       GBM_3_AutoML_5_20231205_221242 0.9917092 0.1083574
```

```
## 4                          GBM_2_AutoML_5_20231205_221242 0.9916690 0.1094926
## 5                          GBM_1_AutoML_5_20231205_221242 0.9912867 0.1050881
## 6                          DRF_1_AutoML_5_20231205_221242 0.9907154 0.1221969
## 7                          GLM_1_AutoML_5_20231205_221242 0.9870747 0.1694009
##       aucpr mean_per_class_error      rmse        mse
## 1 0.9909088           0.03790751 0.1724919 0.02975347
## 2 0.9905500           0.03563306 0.1719834 0.02957830
## 3 0.9905777           0.03904473 0.1750038 0.03062634
## 4 0.9905942           0.03790751 0.1764119 0.03112116
## 5 0.9890545           0.03563306 0.1725022 0.02975702
## 6 0.9891132           0.03866566 0.1810820 0.03279070
## 7 0.9821826           0.04776346 0.1996383 0.03985545
```

View the Leader Model

```r
leader_model <- aml@leader
leader_model
```

```
## Model Details:
## ==============
##
## H2OBinomialModel: stackedensemble
## Model ID:  StackedEnsemble_BestOfFamily_1_AutoML_5_20231205_221242
## Model Summary for Stacked Ensemble:
##                                      key            value
## 1                      Stacking strategy cross_validation
## 2  Number of base models (used / total)               3/3
## 3      # GBM base models (used / total)               1/1
## 4      # DRF base models (used / total)               1/1
## 5      # GLM base models (used / total)               1/1
## 6                  Metalearner algorithm              GLM
## 7    Metalearner fold assignment scheme           Random
## 8                     Metalearner nfolds                5
## 9                Metalearner fold_column               NA
## 10   Custom metalearner hyperparameters             None
##
##
## H2OBinomialMetrics: stackedensemble
## ** Reported on training data. **
##
## MSE:  0.00484105
## RMSE:  0.06957766
## LogLoss:  0.02698711
## Mean Per-Class Error:  0.001137225
## AUC:  0.9999632
## AUCPR:  0.9999627
## Gini:  0.9999264
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           0    1    Error     Rate
## 0      1317    2 0.001516  =2/1319
## 1         1 1318 0.000758  =1/1319
```

```
## Totals 1318 1320 0.001137  =3/2638
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold       value idx
## 1                       max f1  0.605071    0.998863 231
## 2                       max f2  0.525650    0.999242 235
## 3                 max f0point5  0.605071    0.998636 231
## 4                 max accuracy  0.605071    0.998863 231
## 5                max precision  0.999993    1.000000   0
## 6                   max recall  0.525650    1.000000 235
## 7              max specificity  0.999993    1.000000   0
## 8             max absolute_mcc  0.605071    0.997726 231
## 9   max min_per_class_accuracy  0.605939    0.998484 230
## 10 max mean_per_class_accuracy  0.605071    0.998863 231
## 11                     max tns  0.999993 1319.000000   0
## 12                     max fns  0.999993  659.000000   0
## 13                     max fps  0.000328 1319.000000 399
## 14                     max tps  0.525650 1319.000000 235
## 15                     max tnr  0.999993    1.000000   0
## 16                     max fnr  0.999993    0.499621   0
## 17                     max fpr  0.000328    1.000000 399
## 18                     max tpr  0.525650    1.000000 235
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F
##
## H2OBinomialMetrics: stackedensemble
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **
##
## MSE:  0.02975347
## RMSE:  0.1724919
## LogLoss:  0.1036018
## Mean Per-Class Error:  0.03790751
## AUC:  0.9919661
## AUCPR:  0.9909088
## Gini:  0.9839323
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           0    1    Error       Rate
## 0      1250   69 0.052312   =69/1319
## 1        31 1288 0.023503   =31/1319
## Totals 1281 1357 0.037908  =100/2638
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold       value idx
## 1                       max f1  0.468777    0.962631 245
## 2                       max f2  0.071618    0.981253 317
## 3                 max f0point5  0.723463    0.962579 190
## 4                 max accuracy  0.468777    0.962092 245
## 5                max precision  0.999992    1.000000   0
## 6                   max recall  0.071618    1.000000 317
## 7              max specificity  0.999992    1.000000   0
## 8             max absolute_mcc  0.468777    0.924569 245
## 9   max min_per_class_accuracy  0.622742    0.960576 215
```

```
## 10 max mean_per_class_accuracy  0.468777    0.962092 245
## 11                      max tns  0.999992 1319.000000   0
## 12                      max fns  0.999992  933.000000   0
## 13                      max fps  0.000009 1319.000000 399
## 14                      max tps  0.071618 1319.000000 317
## 15                      max tnr  0.999992    1.000000   0
## 16                      max fnr  0.999992    0.707354   0
## 17                      max fpr  0.000009    1.000000 399
## 18                      max tpr  0.071618    1.000000 317
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F
## Cross-Validation Metrics Summary:
##                mean       sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid
## accuracy   0.965853 0.005085   0.970205   0.962617   0.965714   0.971429
## auc        0.992218 0.002271   0.992883   0.991571   0.992871   0.994975
## err        0.034147 0.005085   0.029795   0.037383   0.034286   0.028571
## err_count 18.000000 2.549510  16.000000  20.000000  18.000000  15.000000
## f0point5   0.959790 0.004820   0.961680   0.957370   0.952738   0.965167
##           cv_5_valid
## accuracy    0.959302
## auc         0.988791
## err         0.040698
## err_count  21.000000
## f0point5    0.961995
##
## ---
##                         mean        sd cv_1_valid cv_2_valid cv_3_valid
## precision           0.955619  0.007621   0.956044   0.953237   0.944238
## r2                  0.881143  0.013881   0.883568   0.870539   0.883302
## recall              0.977125  0.014526   0.984906   0.974265   0.988327
## residual_deviance 109.191140 13.522715 109.657555 117.658590 106.122130
## rmse                0.172105  0.010243   0.170596   0.179878   0.170768
## specificity         0.954573  0.007967   0.955882   0.950570   0.944030
##                   cv_4_valid cv_5_valid
## precision           0.960289   0.964286
## r2                  0.901896   0.866411
## recall              0.985185   0.952941
## residual_deviance  88.469350 124.048080
## rmse                0.156544   0.182737
## specificity         0.956863   0.965517
```

AutoML Leader-Board Predictions

```r
# Make predictions on the validation set
pred <- h2o.predict(aml@leader, test.h2o)
```

```
##   |                                                                              |
```

```r
# Convert H2O frame to a data frame
predictions_df <- as.data.frame(pred)

head(predictions_df)
```

```
##   predict        p0           p1
## 1       0 0.9676080 0.0323919795
## 2       0 0.9840842 0.0159158428
## 3       0 0.9996577 0.0003422652
## 4       0 0.8036897 0.1963102573
## 5       0 0.9707758 0.0292241819
## 6       0 0.9995839 0.0004161366
```

Top model with the "AUC" metric

```r
# Get the best model using a non-default metric
m <- h2o.get_best_model(aml, criterion = "auc")
m
```

```
## Model Details:
## ==============
##
## H2OBinomialModel: stackedensemble
## Model ID:  StackedEnsemble_BestOfFamily_1_AutoML_5_20231205_221242
## Model Summary for Stacked Ensemble:
##                                     key            value
## 1                     Stacking strategy cross_validation
## 2  Number of base models (used / total)              3/3
## 3      # GBM base models (used / total)              1/1
## 4      # DRF base models (used / total)              1/1
## 5      # GLM base models (used / total)              1/1
## 6                 Metalearner algorithm              GLM
## 7    Metalearner fold assignment scheme           Random
## 8                    Metalearner nfolds                5
## 9               Metalearner fold_column               NA
## 10   Custom metalearner hyperparameters             None
##
##
## H2OBinomialMetrics: stackedensemble
## ** Reported on training data. **
##
## MSE:  0.00484105
## RMSE:  0.06957766
## LogLoss:  0.02698711
## Mean Per-Class Error:  0.001137225
## AUC:  0.9999632
## AUCPR:  0.9999627
## Gini:  0.9999264
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           0    1    Error     Rate
## 0      1317    2 0.001516  =2/1319
## 1         1 1318 0.000758  =1/1319
## Totals 1318 1320 0.001137  =3/2638
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold       value idx
## 1                       max f1  0.605071    0.998863 231
## 2                       max f2  0.525650    0.999242 235
```

```
## 3                 max f0point5  0.605071    0.998636 231
## 4                 max accuracy  0.605071    0.998863 231
## 5                max precision  0.999993    1.000000   0
## 6                   max recall  0.525650    1.000000 235
## 7              max specificity  0.999993    1.000000   0
## 8             max absolute_mcc  0.605071    0.997726 231
## 9   max min_per_class_accuracy  0.605939    0.998484 230
## 10 max mean_per_class_accuracy  0.605071    0.998863 231
## 11                     max tns  0.999993 1319.000000   0
## 12                     max fns  0.999993  659.000000   0
## 13                     max fps  0.000328 1319.000000 399
## 14                     max tps  0.525650 1319.000000 235
## 15                     max tnr  0.999993    1.000000   0
## 16                     max fnr  0.999993    0.499621   0
## 17                     max fpr  0.000328    1.000000 399
## 18                     max tpr  0.525650    1.000000 235
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F
##
## H2OBinomialMetrics: stackedensemble
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout predictions) **
##
## MSE:  0.02975347
## RMSE:  0.1724919
## LogLoss:  0.1036018
## Mean Per-Class Error:  0.03790751
## AUC:  0.9919661
## AUCPR:  0.9909088
## Gini:  0.9839323
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           0    1    Error       Rate
## 0      1250   69 0.052312   =69/1319
## 1        31 1288 0.023503   =31/1319
## Totals 1281 1357 0.037908  =100/2638
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold       value idx
## 1                       max f1  0.468777    0.962631 245
## 2                       max f2  0.071618    0.981253 317
## 3                 max f0point5  0.723463    0.962579 190
## 4                 max accuracy  0.468777    0.962092 245
## 5                max precision  0.999992    1.000000   0
## 6                   max recall  0.071618    1.000000 317
## 7              max specificity  0.999992    1.000000   0
## 8             max absolute_mcc  0.468777    0.924569 245
## 9   max min_per_class_accuracy  0.622742    0.960576 215
## 10 max mean_per_class_accuracy  0.468777    0.962092 245
## 11                     max tns  0.999992 1319.000000   0
## 12                     max fns  0.999992  933.000000   0
## 13                     max fps  0.000009 1319.000000 399
## 14                     max tps  0.071618 1319.000000 317
## 15                     max tnr  0.999992    1.000000   0
```

```
## 16                        max fnr  0.999992    0.707354   0
## 17                        max fpr  0.000009    1.000000 399
## 18                        max tpr  0.071618    1.000000 317
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F
## Cross-Validation Metrics Summary:
##                mean       sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid
## accuracy   0.965853 0.005085   0.970205   0.962617   0.965714   0.971429
## auc        0.992218 0.002271   0.992883   0.991571   0.992871   0.994975
## err        0.034147 0.005085   0.029795   0.037383   0.034286   0.028571
## err_count 18.000000 2.549510  16.000000  20.000000  18.000000  15.000000
## f0point5   0.959790 0.004820   0.961680   0.957370   0.952738   0.965167
##           cv_5_valid
## accuracy    0.959302
## auc         0.988791
## err         0.040698
## err_count  21.000000
## f0point5    0.961995
##
## ---
##                         mean        sd cv_1_valid cv_2_valid cv_3_valid
## precision           0.955619  0.007621   0.956044   0.953237   0.944238
## r2                  0.881143  0.013881   0.883568   0.870539   0.883302
## recall              0.977125  0.014526   0.984906   0.974265   0.988327
## residual_deviance 109.191140 13.522715 109.657555 117.658590 106.122130
## rmse                0.172105  0.010243   0.170596   0.179878   0.170768
## specificity         0.954573  0.007967   0.955882   0.950570   0.944030
##                   cv_4_valid cv_5_valid
## precision           0.960289   0.964286
## r2                  0.901896   0.866411
## recall              0.985185   0.952941
## residual_deviance  88.469350 124.048080
## rmse                0.156544   0.182737
## specificity         0.956863   0.965517
```

AutoML Performance

```r
# Extract actual values from the test set
perf <- h2o.performance(leader_model, test.h2o )
perf
```

```
## H2OBinomialMetrics: stackedensemble
##
## MSE:  0.03926883
## RMSE:  0.1981637
## LogLoss:  0.1372975
## Mean Per-Class Error:  0.04867257
## AUC:  0.9872692
## AUCPR:  0.9836436
## Gini:  0.9745383
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
```

```
##          0   1    Error      Rate
## 0      532  33 0.058407   =33/565
## 1       22 543 0.038938   =22/565
## Totals 554 576 0.048673  =55/1130
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                       max f1  0.530993   0.951797 265
## 2                       max f2  0.048111   0.976833 320
## 3                 max f0point5  0.645673   0.949592 252
## 4                 max accuracy  0.574580   0.951327 259
## 5                max precision  0.991132   0.995781  24
## 6                   max recall  0.048111   1.000000 320
## 7              max specificity  0.999994   0.998230   0
## 8             max absolute_mcc  0.530993   0.902826 265
## 9   max min_per_class_accuracy  0.633232   0.948673 254
## 10 max mean_per_class_accuracy  0.574580   0.951327 259
## 11                     max tns  0.999994 564.000000   0
## 12                     max fns  0.999994 404.000000   0
## 13                     max fps  0.000316 565.000000 399
## 14                     max tps  0.048111 565.000000 320
## 15                     max tnr  0.999994   0.998230   0
## 16                     max fnr  0.999994   0.715044   0
## 17                     max fpr  0.000316   1.000000 399
## 18                     max tpr  0.048111   1.000000 320
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

```r
#Generate predictions on a test set, you can make predictions directly on the `H2OAutoML` object
aml_pred <- h2o.predict(leader_model, test.h2o)
```

```
##   |                                                                              |
```

```r
#Accuracy measure on the test data
amlpred_df <- as.data.frame(aml_pred$predict)
amlpred_df$predict <- factor(amlpred_df$predict, levels = c(0,1))
```

#Print the confusion matrix

```r
#Accuracy measure on the test data
aml_cm<-confusionMatrix(amlpred_df$predict,test_data$WFRI_R,positive = "1",mode = "prec_recall")

aml_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 528  22
##          1  37 543
##
##                Accuracy : 0.9478
```

```
##                  95% CI : (0.9332, 0.96)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8956
##
##  Mcnemar's Test P-Value : 0.06836
##
##               Precision : 0.9362
##                  Recall : 0.9611
##                      F1 : 0.9485
##              Prevalence : 0.5000
##          Detection Rate : 0.4805
##    Detection Prevalence : 0.5133
##       Balanced Accuracy : 0.9478
##
##        'Positive' Class : 1
##
```

Model 4 - Naive-Bayes

```r
# Build and train the model:
pros_nb <- h2o.naiveBayes(x = features,
                          y = response,
                          training_frame = train.h2o,
                          laplace = 0,
                          nfolds = 5,
                          seed = 1234,
                          keep_cross_validation_predictions = TRUE)
```

```
##   |                                                                              |
```

```r
nb_perf <- h2o.performance(pros_nb, test.h2o)
nb_perf
```

```
## H2OBinomialMetrics: naivebayes
##
## MSE:  0.1384367
## RMSE:  0.3720708
## LogLoss:  1.207416
## Mean Per-Class Error:  0.1106195
## AUC:  0.9436307
## AUCPR:  0.929643
## Gini:  0.8872613
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##          0   1    Error       Rate
## 0      478  87 0.153982    =87/565
## 1       38 527 0.067257    =38/565
## Totals 516 614 0.110619  =125/1130
```

```
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                       max f1  0.001184   0.893978 258
## 2                       max f2  0.000361   0.929234 294
## 3                 max f0point5  0.017826   0.890688 185
## 4                 max accuracy  0.001581   0.889381 250
## 5                max precision  1.000000   0.958730   0
## 6                   max recall  0.000000   1.000000 399
## 7              max specificity  1.000000   0.976991   0
## 8             max absolute_mcc  0.001184   0.781706 258
## 9   max min_per_class_accuracy  0.006278   0.879646 211
## 10 max mean_per_class_accuracy  0.001581   0.889381 250
## 11                     max tns  1.000000 552.000000   0
## 12                     max fns  1.000000 263.000000   0
## 13                     max fps  0.000000 565.000000 399
## 14                     max tps  0.000000 565.000000 399
## 15                     max tnr  1.000000   0.976991   0
## 16                     max fnr  1.000000   0.465487   0
## 17                     max fpr  0.000000   1.000000 399
## 18                     max tpr  0.000000   1.000000 399
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

```r
#Generate predictions on a test set, you can make predictions directly on the `H2OAutoML` object
nb_pred <- h2o.predict(pros_nb, test.h2o)
```

```
##   |                                                                              |
```

```r
#Accuracy measure on the test data
nbpred_df <- as.data.frame(nb_pred$predict)
nbpred_df$predict <- factor(nbpred_df$predict, levels = c(0,1))
```

```r
#Accuracy measure on the test data
nb_cm<-confusionMatrix(nbpred_df$predict,test_data$WFRI_R,positive = "1",mode = "prec_recall")

nb_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 483  45
##          1  82 520
##
##                Accuracy : 0.8876
##                  95% CI : (0.8677, 0.9054)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7752
##
```

```
## Mcnemar's Test P-Value : 0.001401
##
##               Precision : 0.8638
##                  Recall : 0.9204
##                      F1 : 0.8912
##              Prevalence : 0.5000
##          Detection Rate : 0.4602
##    Detection Prevalence : 0.5327
##       Balanced Accuracy : 0.8876
##
##        'Positive' Class : 1
##
```

Model 5 - SVM Model

```r
# Build and train the model:
svm_model <- h2o.psvm(gamma = 0.01,
                      rank_ratio = 0.1,
                      x = features,
                      y = response,
                      training_frame = train.h2o,
                      disable_training_metrics = FALSE,
                      seed = 1)
```

```
##   |                                                                              |
```

```r
svm_perf <- h2o.performance(svm_model, test.h2o)
svm_perf
```

```
## H2OBinomialMetrics: psvm
##
## MSE:  0.4433628
## RMSE:  0.665855
## LogLoss:  NaN
## Mean Per-Class Error:  0.4433628
## AUC:  NaN
## AUCPR:  NaN
## Gini:  NaN
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##         0    1    Error       Rate
## 0      74  491 0.869027   =491/565
## 1      10  555 0.017699    =10/565
## Totals 84 1046 0.443363  =501/1130
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                            metric threshold      value idx
## 1                          max f1  1.000000   0.689013   0
```

```
## 2                       max f2  1.000000   0.839383   0
## 3                 max f0point5  1.000000   0.584334   0
## 4                 max accuracy  1.000000   0.556637   0
## 5                max precision  1.000000   0.530593   0
## 6                   max recall  1.000000   0.982301   0
## 7              max specificity  1.000000   0.130973   0
## 8             max absolute_mcc  1.000000   0.215911   0
## 9   max min_per_class_accuracy  1.000000   0.130973   0
## 10 max mean_per_class_accuracy  1.000000   0.556637   0
## 11                     max tns  1.000000  74.000000   0
## 12                     max fns  1.000000  10.000000   0
## 13                     max fps  1.000000 491.000000   0
## 14                     max tps  1.000000 555.000000   0
## 15                     max tnr  1.000000   0.130973   0
## 16                     max fnr  1.000000   0.017699   0
## 17                     max fpr  1.000000   0.869027   0
## 18                     max tpr  1.000000   0.982301   0
##
## Gains/Lift Table: Extract with 'h2o.gainsLift(<model>, <data>)' or 'h2o.gainsLift(<model>, valid=<T/F
```

```
#Generate predictions on a test set, you can make predictions directly on the `H2OAutoML` object
svm_pred <- h2o.predict(svm_model, test.h2o)
```

```
##   |                                                                              |
```

```
#Accuracy measure on the test data
svmpred_df <- as.data.frame(svm_pred$predict)
svmpred_df$predict <- factor(svmpred_df$predict, levels = c(0,1))
```

```
#Accuracy measure on the test data
svm_cm<-confusionMatrix(svmpred_df$predict,test_data$WFRI_R,positive = "1",mode = "prec_recall")

svm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  74  10
##          1 491 555
##
##                Accuracy : 0.5566
##                  95% CI : (0.5271, 0.5859)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 7.787e-05
##
##                   Kappa : 0.1133
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.5306
##                  Recall : 0.9823
```

```
##                       F1 : 0.6890
##               Prevalence : 0.5000
##           Detection Rate : 0.4912
##     Detection Prevalence : 0.9257
##        Balanced Accuracy : 0.5566
##
##         'Positive' Class : 1
##
```

Model 6 - Deep Learning

```r
# Build and train the model:
dl <- h2o.deeplearning(x = features,
                       y = response,
                       distribution = "AUTO",
                       hidden = c(1),
                       epochs = 1000,
                       train_samples_per_iteration = -1,
                       reproducible = TRUE,
                       activation = "Tanh",
                       single_node_mode = FALSE,
                       balance_classes = FALSE,
                       force_load_balance = FALSE,
                       seed = 23123,
                       score_training_samples = 0,
                       score_validation_samples = 0,
                       training_frame = train.h2o,
                       stopping_rounds = 0,
                       keep_cross_validation_predictions = TRUE)
```

```
##   |                                                                              |
```

```r
dl_perf <- h2o.performance(dl, test.h2o)
dl_perf
```

```
## H2OBinomialMetrics: deeplearning
##
## MSE:  0.04250999
## RMSE:  0.2061795
## LogLoss:  0.1624409
## Mean Per-Class Error:  0.04778761
## AUC:  0.9781392
## AUCPR:  0.9665388
## Gini:  0.9562785
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##          0   1    Error      Rate
## 0      521  44 0.077876   =44/565
```

```
## 1       10 555 0.017699   =10/565
## Totals 531 599 0.047788  =54/1130
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                       max f1  0.294898   0.953608 216
## 2                       max f2  0.088587   0.973958 237
## 3                 max f0point5  0.803380   0.944868 165
## 4                 max accuracy  0.294898   0.952212 216
## 5                max precision  0.984779   0.979424   2
## 6                   max recall  0.001645   1.000000 385
## 7              max specificity  0.984812   0.991150   0
## 8             max absolute_mcc  0.294898   0.906067 216
## 9   max min_per_class_accuracy  0.678508   0.939823 182
## 10 max mean_per_class_accuracy  0.294898   0.952212 216
## 11                     max tns  0.984812 560.000000   0
## 12                     max fns  0.984812 378.000000   0
## 13                     max fps  0.000525 565.000000 399
## 14                     max tps  0.001645 565.000000 385
## 15                     max tnr  0.984812   0.991150   0
## 16                     max fnr  0.984812   0.669027   0
## 17                     max fpr  0.000525   1.000000 399
## 18                     max tpr  0.001645   1.000000 385
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

```r
#Generate predictions on a test set, you can make predictions directly on the `H2OAutoML` object
dl_pred <- h2o.predict(dl, test.h2o)
```

```
##   |                                                                               |
```

```r
#Accuracy measure on the test data
dlpred_df <- as.data.frame(dl_pred$predict)
dlpred_df$predict <- factor(dlpred_df$predict, levels = c(0,1))
```

```r
#Accuracy measure on the test data
dl_cm<-confusionMatrix(dlpred_df$predict,test_data$WFRI_R,positive = "1",mode = "prec_recall")

dl_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 521  10
##          1  44 555
##
##                Accuracy : 0.9522
##                  95% CI : (0.9381, 0.9639)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9044
##
##  Mcnemar's Test P-Value : 7.098e-06
##
##               Precision : 0.9265
##                  Recall : 0.9823
##                      F1 : 0.9536
##              Prevalence : 0.5000
##          Detection Rate : 0.4912
##    Detection Prevalence : 0.5301
##       Balanced Accuracy : 0.9522
##
##        'Positive' Class : 1
##
```

Part 2 Analysis| Comparison of Top LM model compared to other two models

#Retrieve Top performing model and save separately #saving name: GBM_1_AutoML_1_20231119_133450

```
winining_aml<-aml@leader
winining_aml@model$model_summary
```

#Retrieving Top performing Auto model and save its prediction separately for performance

```
automl.pred<-h2o.predict(leader_model,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                              |
```

```
automl.precsnprob<-h2o.predict(leader_model,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                              |
```

```
automl.reclprob<-h2o.predict(aml@leader,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                              |
```

```
#Print the confusion matrix
automl.con<-confusionMatrix(automl.pred,test_data$WFRI_R,positive = "1",mode = "prec_recall")

automl.con
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 528  22
##          1  37 543
##
```

```
##                Accuracy : 0.9478
##                  95% CI : (0.9332, 0.96)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8956
##
##  Mcnemar's Test P-Value : 0.06836
##
##               Precision : 0.9362
##                  Recall : 0.9611
##                      F1 : 0.9485
##              Prevalence : 0.5000
##          Detection Rate : 0.4805
##    Detection Prevalence : 0.5133
##       Balanced Accuracy : 0.9478
##
##        'Positive' Class : 1
##
```

#see summary results of automled ensemble model on the test data

```
automl.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-automl.pred,
      N<-automl.reclprob,
      Y<-automl.precsnprob
  )


automl.summary<-automl.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....automl.pred", N="N.

automl.auc<-roc(automl.summary$obs,Y)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
temp<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(automl.summary$pred)),y_true =as.numeric(as.character
  t.mse<-MSE(as.numeric(as.character(automl.summary$pred)),as.numeric(as.character(automl.summary$obs))
  t.RSME<-RMSE(as.numeric(as.character(automl.summary$pred)),as.numeric(as.character(automl.summary$obs
  t.AUC<-automl.auc$auc,
  t.ClassError<-mean(automl.summary$pred!=automl.summary$obs)

)

automl.summary
```

```
##      obs pred            N            Y
## 1      0    0 9.676080e-01 0.0323919795
## 2      0    0 9.840842e-01 0.0159158428
```

```
## 3       0     0 9.996577e-01 0.0003422652
## 4       0     0 8.036897e-01 0.1963102573
## 5       0     0 9.707758e-01 0.0292241819
## 6       0     0 9.995839e-01 0.0004161366
## 7       0     0 9.856802e-01 0.0143198363
## 8       0     0 9.923463e-01 0.0076536986
## 9       0     0 9.862869e-01 0.0137130539
## 10      0     0 9.974926e-01 0.0025073802
## 11      0     0 9.744232e-01 0.0255768364
## 12      0     0 9.996485e-01 0.0003515084
## 13      0     0 9.996668e-01 0.0003331694
## 14      0     0 9.860773e-01 0.0139226702
## 15      0     0 9.996854e-01 0.0003145621
## 16      0     0 9.996933e-01 0.0003066597
## 17      0     0 9.996725e-01 0.0003274555
## 18      0     0 9.995553e-01 0.0004446769
## 19      0     0 9.995701e-01 0.0004298915
## 20      0     0 9.996274e-01 0.0003725508
## 21      0     0 9.993222e-01 0.0006777669
## 22      0     0 9.996437e-01 0.0003562592
## 23      0     0 9.996655e-01 0.0003344614
## 24      0     0 9.918938e-01 0.0081061854
## 25      0     0 9.995671e-01 0.0004328536
## 26      0     0 6.029191e-01 0.3970808866
## 27      0     0 9.996310e-01 0.0003689695
## 28      0     0 9.994906e-01 0.0005093917
## 29      0     0 9.995973e-01 0.0004026567
## 30      0     0 9.996279e-01 0.0003720755
## 31      0     0 9.996857e-01 0.0003143422
## 32      0     0 9.995905e-01 0.0004094675
## 33      0     0 9.996466e-01 0.0003533962
## 34      0     0 9.782340e-01 0.0217660268
## 35      0     0 9.877464e-01 0.0122535908
## 36      0     0 9.996230e-01 0.0003769521
## 37      0     0 9.995594e-01 0.0004405636
## 38      0     0 9.996834e-01 0.0003165704
## 39      0     0 9.996271e-01 0.0003728943
## 40      0     0 9.996497e-01 0.0003502803
## 41      0     0 9.879470e-01 0.0120530447
## 42      0     0 9.996689e-01 0.0003310733
## 43      0     0 9.886028e-01 0.0113971969
## 44      0     0 9.996297e-01 0.0003703129
## 45      0     0 9.995902e-01 0.0004098262
## 46      0     0 9.996362e-01 0.0003637611
## 47      0     0 9.996523e-01 0.0003476515
## 48      0     0 9.865789e-01 0.0134210654
## 49      0     0 9.996535e-01 0.0003464768
## 50      0     0 9.996737e-01 0.0003262943
## 51      0     0 9.996166e-01 0.0003834114
## 52      0     0 9.831708e-01 0.0168292363
## 53      0     0 9.996848e-01 0.0003151729
## 54      0     0 9.996570e-01 0.0003429885
## 55      0     0 9.879099e-01 0.0120901157
## 56      0     0 9.994980e-01 0.0005019878
```

```
## 57      0     0 9.994423e-01 0.0005576557
## 58      0     1 1.789829e-01 0.8210171254
## 59      0     0 9.937827e-01 0.0062172879
## 60      0     0 9.996384e-01 0.0003615615
## 61      0     0 9.996293e-01 0.0003706842
## 62      0     0 9.996743e-01 0.0003256606
## 63      0     0 9.995120e-01 0.0004880377
## 64      0     0 9.809187e-01 0.0190812635
## 65      0     0 9.995904e-01 0.0004095780
## 66      0     0 9.927333e-01 0.0072667209
## 67      0     0 9.947376e-01 0.0052624392
## 68      0     0 9.996459e-01 0.0003541294
## 69      0     0 9.996621e-01 0.0003378616
## 70      0     0 9.996833e-01 0.0003167118
## 71      0     0 9.996345e-01 0.0003655256
## 72      0     0 8.600808e-01 0.1399192235
## 73      0     0 9.995767e-01 0.0004233353
## 74      0     0 9.996919e-01 0.0003080952
## 75      0     0 9.997207e-01 0.0002792912
## 76      0     0 9.898861e-01 0.0101138701
## 77      0     0 9.995807e-01 0.0004193034
## 78      0     0 9.995803e-01 0.0004196838
## 79      0     0 9.996066e-01 0.0003934363
## 80      0     0 9.996592e-01 0.0003407707
## 81      0     0 9.388218e-01 0.0611781977
## 82      0     0 9.996713e-01 0.0003286667
## 83      0     0 9.878803e-01 0.0121197445
## 84      0     0 9.996667e-01 0.0003332917
## 85      0     0 9.996256e-01 0.0003743552
## 86      0     0 9.893974e-01 0.0106025701
## 87      0     0 9.996662e-01 0.0003337957
## 88      0     1 1.961270e-01 0.8038730295
## 89      0     0 9.996195e-01 0.0003805429
## 90      0     0 9.996534e-01 0.0003466319
## 91      0     0 9.996790e-01 0.0003209502
## 92      0     0 9.996376e-01 0.0003623995
## 93      0     0 9.935250e-01 0.0064750342
## 94      0     0 9.911894e-01 0.0088106403
## 95      0     0 9.996907e-01 0.0003093053
## 96      0     0 9.438134e-01 0.0561866327
## 97      0     0 9.915869e-01 0.0084130944
## 98      0     0 9.996935e-01 0.0003064620
## 99      0     0 9.996458e-01 0.0003542437
## 100     0     0 9.996347e-01 0.0003653063
## 101     0     0 9.996193e-01 0.0003807245
## 102     0     1 1.668056e-02 0.9833194430
## 103     0     0 9.996280e-01 0.0003719722
## 104     0     0 9.996046e-01 0.0003953651
## 105     0     1 3.317095e-02 0.9668290466
## 106     0     0 8.525988e-01 0.1474011654
## 107     0     0 9.906594e-01 0.0093406087
## 108     0     0 9.994302e-01 0.0005698128
## 109     0     0 9.996362e-01 0.0003638432
## 110     0     0 9.586980e-01 0.0413019744
```

```
## 111    0    1 1.199968e-01 0.8800032260
## 112    0    0 9.996380e-01 0.0003620193
## 113    0    0 9.995777e-01 0.0004223433
## 114    0    0 9.996462e-01 0.0003538196
## 115    0    0 9.996321e-01 0.0003678782
## 116    0    1 9.719600e-02 0.9028039965
## 117    0    0 9.996241e-01 0.0003758527
## 118    0    1 4.310503e-01 0.5689497150
## 119    0    0 9.996398e-01 0.0003602086
## 120    0    0 6.002852e-01 0.3997148271
## 121    0    0 9.914087e-01 0.0085912761
## 122    0    0 9.995771e-01 0.0004229154
## 123    0    0 9.996865e-01 0.0003135268
## 124    0    0 9.928256e-01 0.0071744196
## 125    0    0 9.947271e-01 0.0052728611
## 126    0    1 1.638317e-01 0.8361682998
## 127    0    0 9.495087e-01 0.0504912866
## 128    0    0 9.945508e-01 0.0054492406
## 129    0    0 9.150151e-01 0.0849849493
## 130    0    0 9.994627e-01 0.0005372856
## 131    0    0 9.848147e-01 0.0151852626
## 132    0    1 1.912070e-01 0.8087930306
## 133    0    0 9.996624e-01 0.0003375569
## 134    0    0 9.997047e-01 0.0002952630
## 135    0    0 9.994455e-01 0.0005544841
## 136    0    0 9.912845e-01 0.0087154892
## 137    0    0 9.996351e-01 0.0003649435
## 138    0    0 9.996361e-01 0.0003638894
## 139    0    1 4.717634e-01 0.5282365815
## 140    0    0 9.996810e-01 0.0003189924
## 141    0    0 9.892676e-01 0.0107323844
## 142    0    0 9.871989e-01 0.0128010507
## 143    0    0 9.996539e-01 0.0003461481
## 144    0    0 9.936914e-01 0.0063085797
## 145    0    1 5.426896e-01 0.4573104228
## 146    0    0 9.939432e-01 0.0060568406
## 147    0    0 9.996707e-01 0.0003292789
## 148    0    0 9.996951e-01 0.0003048572
## 149    0    0 9.650974e-01 0.0349025540
## 150    0    0 9.996371e-01 0.0003628940
## 151    0    0 9.810283e-01 0.0189717250
## 152    0    0 9.996377e-01 0.0003622936
## 153    0    0 9.996523e-01 0.0003477271
## 154    0    0 9.996536e-01 0.0003464006
## 155    0    0 9.996803e-01 0.0003196832
## 156    0    0 9.996866e-01 0.0003134131
## 157    0    0 9.996048e-01 0.0003952155
## 158    0    0 9.930375e-01 0.0069624586
## 159    0    0 9.996807e-01 0.0003193303
## 160    0    0 9.996338e-01 0.0003661648
## 161    0    0 9.941400e-01 0.0058600046
## 162    0    1 1.269195e-01 0.8730804540
## 163    0    0 9.996675e-01 0.0003325457
## 164    0    0 9.803693e-01 0.0196306666
```

```
## 165    0     0 9.996073e-01 0.0003927284
## 166    0     0 9.996208e-01 0.0003791986
## 167    0     0 9.921710e-01 0.0078290393
## 168    0     0 9.995835e-01 0.0004164526
## 169    0     0 9.996252e-01 0.0003747504
## 170    0     0 9.996797e-01 0.0003202925
## 171    0     0 9.952714e-01 0.0047286064
## 172    0     0 9.995708e-01 0.0004292478
## 173    0     1 9.213139e-02 0.9078686098
## 174    0     0 9.996314e-01 0.0003685824
## 175    0     0 9.996092e-01 0.0003907613
## 176    0     0 9.995742e-01 0.0004258099
## 177    0     0 9.996315e-01 0.0003684890
## 178    0     0 9.887419e-01 0.0112580822
## 179    0     0 9.995809e-01 0.0004191004
## 180    0     0 9.878056e-01 0.0121943832
## 181    0     0 9.996238e-01 0.0003762068
## 182    0     0 9.873788e-01 0.0126211898
## 183    0     1 9.282104e-03 0.9907178961
## 184    0     0 9.948538e-01 0.0051462180
## 185    0     0 9.996332e-01 0.0003668028
## 186    0     0 9.631778e-01 0.0368221909
## 187    0     0 9.996551e-01 0.0003448545
## 188    0     0 9.924850e-01 0.0075149877
## 189    0     0 9.996650e-01 0.0003349913
## 190    0     0 9.996520e-01 0.0003479801
## 191    0     0 9.996495e-01 0.0003504616
## 192    0     0 9.996173e-01 0.0003827061
## 193    0     0 9.996135e-01 0.0003865073
## 194    0     0 9.726624e-01 0.0273376156
## 195    0     0 9.859668e-01 0.0140331951
## 196    0     0 9.786228e-01 0.0213771664
## 197    0     0 9.931058e-01 0.0068942057
## 198    0     0 9.996534e-01 0.0003465672
## 199    0     0 9.995796e-01 0.0004203624
## 200    0     0 9.995742e-01 0.0004258414
## 201    0     0 9.763645e-01 0.0236354623
## 202    0     0 9.996143e-01 0.0003856794
## 203    0     0 9.995669e-01 0.0004331447
## 204    0     0 9.996409e-01 0.0003590794
## 205    0     0 7.638197e-01 0.2361802664
## 206    0     0 9.995997e-01 0.0004003022
## 207    0     0 9.996188e-01 0.0003812446
## 208    0     0 9.995608e-01 0.0004392111
## 209    0     0 9.932409e-01 0.0067591500
## 210    0     0 9.939630e-01 0.0060369763
## 211    0     0 6.480496e-01 0.3519503992
## 212    0     0 8.226613e-01 0.1773387151
## 213    0     0 9.856649e-01 0.0143350973
## 214    0     0 9.914443e-01 0.0085557460
## 215    0     0 9.596125e-01 0.0403875256
## 216    0     0 9.996912e-01 0.0003087646
## 217    0     0 9.995759e-01 0.0004240851
## 218    0     0 9.464860e-01 0.0535139964
```

```
## 219    0     0 9.938637e-01 0.0061362848
## 220    0     0 6.038563e-01 0.3961437449
## 221    0     0 9.996856e-01 0.0003144280
## 222    0     0 9.800519e-01 0.0199481391
## 223    0     0 9.996323e-01 0.0003676890
## 224    0     0 9.995194e-01 0.0004805697
## 225    0     0 9.995902e-01 0.0004098117
## 226    0     0 9.699245e-01 0.0300755063
## 227    0     0 9.996786e-01 0.0003213588
## 228    0     0 9.996925e-01 0.0003074896
## 229    0     0 9.996767e-01 0.0003232974
## 230    0     0 6.625523e-01 0.3374476547
## 231    0     0 9.996766e-01 0.0003234230
## 232    0     0 9.996248e-01 0.0003751534
## 233    0     0 9.996437e-01 0.0003563183
## 234    0     0 9.996738e-01 0.0003261905
## 235    0     0 9.996539e-01 0.0003461265
## 236    0     0 9.890763e-01 0.0109237012
## 237    0     0 9.995568e-01 0.0004431946
## 238    0     0 9.789292e-01 0.0210708318
## 239    0     0 9.996250e-01 0.0003750398
## 240    0     0 6.047984e-01 0.3952016109
## 241    0     0 9.996523e-01 0.0003476737
## 242    0     1 2.083388e-01 0.7916612393
## 243    0     1 1.024292e-01 0.8975707985
## 244    0     0 5.904204e-01 0.4095795894
## 245    0     0 9.995518e-01 0.0004482338
## 246    0     0 9.996631e-01 0.0003369275
## 247    0     0 9.995435e-01 0.0004564916
## 248    0     0 9.590395e-01 0.0409605181
## 249    0     0 9.908281e-01 0.0091719384
## 250    0     0 9.950122e-01 0.0049877687
## 251    0     0 9.996574e-01 0.0003425665
## 252    0     0 9.821070e-01 0.0178930372
## 253    0     0 9.996490e-01 0.0003510104
## 254    0     0 9.995960e-01 0.0004039935
## 255    0     0 9.995960e-01 0.0004039898
## 256    0     0 9.996539e-01 0.0003460982
## 257    0     1 1.991681e-01 0.8008319354
## 258    0     0 9.932200e-01 0.0067800038
## 259    0     0 9.996069e-01 0.0003931013
## 260    0     0 9.996072e-01 0.0003927830
## 261    0     1 1.493408e-01 0.8506591996
## 262    0     0 9.651388e-01 0.0348611793
## 263    0     0 9.996020e-01 0.0003979625
## 264    0     0 9.996687e-01 0.0003312778
## 265    0     0 7.255612e-01 0.2744387788
## 266    0     0 9.996867e-01 0.0003132880
## 267    0     0 9.996760e-01 0.0003240187
## 268    0     0 9.942445e-01 0.0057554619
## 269    0     0 9.996784e-01 0.0003216159
## 270    0     0 5.722179e-01 0.4277820986
## 271    0     0 9.799101e-01 0.0200899270
## 272    0     0 9.996361e-01 0.0003638563
```

```
## 273    0     1 3.830526e-01 0.6169473647
## 274    0     0 9.996743e-01 0.0003256517
## 275    0     0 9.996502e-01 0.0003498457
## 276    0     0 9.856522e-01 0.0143477756
## 277    0     0 9.995563e-01 0.0004436666
## 278    0     0 9.874848e-01 0.0125151691
## 279    0     0 9.948381e-01 0.0051619067
## 280    0     0 9.928777e-01 0.0071222754
## 281    0     0 9.746697e-01 0.0253303296
## 282    0     0 9.939319e-01 0.0060681111
## 283    0     0 7.220895e-01 0.2779104913
## 284    0     0 9.887028e-01 0.0112972152
## 285    0     0 9.912351e-01 0.0087649057
## 286    0     1 3.557890e-01 0.6442110294
## 287    0     0 9.952099e-01 0.0047901185
## 288    0     0 9.996530e-01 0.0003470343
## 289    0     0 9.996526e-01 0.0003473891
## 290    0     0 9.995599e-01 0.0004401032
## 291    0     0 9.996536e-01 0.0003464267
## 292    0     0 9.996233e-01 0.0003767428
## 293    0     0 9.996518e-01 0.0003481505
## 294    0     0 9.230273e-01 0.0769727292
## 295    0     0 9.941057e-01 0.0058943050
## 296    0     0 9.994472e-01 0.0005528077
## 297    0     0 9.945510e-01 0.0054489705
## 298    0     0 9.996337e-01 0.0003663498
## 299    0     0 9.996980e-01 0.0003020002
## 300    0     0 9.996007e-01 0.0003993418
## 301    0     1 1.566329e-05 0.9999843367
## 302    0     0 9.996437e-01 0.0003563062
## 303    0     0 9.348219e-01 0.0651780557
## 304    0     0 9.996731e-01 0.0003268774
## 305    0     0 9.924045e-01 0.0075955080
## 306    0     0 9.996188e-01 0.0003812159
## 307    0     0 9.943349e-01 0.0056651253
## 308    0     0 9.995907e-01 0.0004093391
## 309    0     0 9.877627e-01 0.0122372821
## 310    0     0 9.851887e-01 0.0148113185
## 311    0     0 9.996416e-01 0.0003583672
## 312    0     0 9.996300e-01 0.0003700116
## 313    0     0 9.918415e-01 0.0081584556
## 314    0     0 9.995869e-01 0.0004131307
## 315    0     0 9.996654e-01 0.0003346340
## 316    0     0 9.947875e-01 0.0052125156
## 317    0     0 9.996422e-01 0.0003577962
## 318    0     0 9.995760e-01 0.0004240304
## 319    0     0 9.397293e-01 0.0602707019
## 320    0     0 9.996969e-01 0.0003030911
## 321    0     0 9.996772e-01 0.0003228117
## 322    0     0 9.995469e-01 0.0004531245
## 323    0     0 9.996403e-01 0.0003596783
## 324    0     0 9.872106e-01 0.0127893920
## 325    0     0 9.996484e-01 0.0003516155
## 326    0     0 9.996272e-01 0.0003727848
```

```
## 327    0    1 2.275233e-01 0.7724767486
## 328    0    0 9.996689e-01 0.0003310647
## 329    0    0 8.430075e-01 0.1569924657
## 330    0    0 9.996637e-01 0.0003363021
## 331    0    0 9.995625e-01 0.0004374662
## 332    0    0 9.994060e-01 0.0005939965
## 333    0    0 9.935764e-01 0.0064236439
## 334    0    0 9.948529e-01 0.0051470551
## 335    0    0 9.919393e-01 0.0080607119
## 336    0    0 9.939921e-01 0.0060078877
## 337    0    1 4.276722e-01 0.5723278022
## 338    0    1 5.833325e-02 0.9416667490
## 339    0    0 9.996303e-01 0.0003697371
## 340    0    0 9.946062e-01 0.0053938067
## 341    0    0 9.996188e-01 0.0003812475
## 342    0    0 9.995875e-01 0.0004125169
## 343    0    0 9.996675e-01 0.0003325436
## 344    0    0 9.996167e-01 0.0003833306
## 345    0    0 9.996261e-01 0.0003739347
## 346    0    0 9.995853e-01 0.0004147404
## 347    0    0 9.812260e-01 0.0187739925
## 348    0    0 6.781836e-01 0.3218163576
## 349    0    0 9.926780e-01 0.0073220111
## 350    0    0 9.922966e-01 0.0077034412
## 351    0    0 9.996217e-01 0.0003783301
## 352    0    0 9.996715e-01 0.0003284901
## 353    0    1 1.967265e-01 0.8032734592
## 354    0    0 9.996571e-01 0.0003429260
## 355    0    1 5.158414e-01 0.4841585981
## 356    0    0 9.996478e-01 0.0003521560
## 357    0    0 9.995887e-01 0.0004112685
## 358    0    0 9.996646e-01 0.0003353964
## 359    0    0 6.072527e-01 0.3927473008
## 360    0    0 9.995941e-01 0.0004058768
## 361    0    0 9.996427e-01 0.0003573093
## 362    0    0 9.857424e-01 0.0142575707
## 363    0    0 9.995736e-01 0.0004264282
## 364    0    0 9.996345e-01 0.0003655045
## 365    0    0 9.917287e-01 0.0082712509
## 366    0    0 9.996228e-01 0.0003772175
## 367    0    0 9.996970e-01 0.0003029707
## 368    0    0 9.996431e-01 0.0003569087
## 369    0    0 9.996306e-01 0.0003694334
## 370    0    0 9.947415e-01 0.0052584548
## 371    0    1 5.284546e-02 0.9471545392
## 372    0    0 9.917017e-01 0.0082982607
## 373    0    0 9.870693e-01 0.0129307387
## 374    0    0 9.996312e-01 0.0003687851
## 375    0    0 9.996223e-01 0.0003776832
## 376    0    0 9.996613e-01 0.0003387425
## 377    0    0 9.996685e-01 0.0003315330
## 378    0    1 4.303511e-01 0.5696488800
## 379    0    0 9.579621e-01 0.0420378726
## 380    0    0 9.858823e-01 0.0141176603
```

```
## 381    0    0 9.996399e-01 0.0003600624
## 382    0    0 9.941223e-01 0.0058777224
## 383    0    0 9.996478e-01 0.0003521925
## 384    0    0 9.995623e-01 0.0004376724
## 385    0    0 9.861760e-01 0.0138240367
## 386    0    0 9.951588e-01 0.0048411615
## 387    0    0 9.994225e-01 0.0005775018
## 388    0    0 9.904266e-01 0.0095734040
## 389    0    0 9.996266e-01 0.0003733958
## 390    0    0 9.896100e-01 0.0103899634
## 391    0    1 4.289213e-02 0.9571078697
## 392    0    0 9.899671e-01 0.0100328892
## 393    0    0 9.996288e-01 0.0003711739
## 394    0    0 9.996160e-01 0.0003840435
## 395    0    0 9.607176e-01 0.0392823675
## 396    0    0 9.996425e-01 0.0003575145
## 397    0    0 9.995330e-01 0.0004670024
## 398    0    0 9.996207e-01 0.0003792558
## 399    0    0 9.774655e-01 0.0225345098
## 400    0    0 9.996761e-01 0.0003239333
## 401    0    0 9.906970e-01 0.0093029661
## 402    0    0 9.996090e-01 0.0003909686
## 403    0    0 9.996283e-01 0.0003716842
## 404    0    0 9.996801e-01 0.0003199273
## 405    0    1 3.103702e-01 0.6896297832
## 406    0    0 9.996585e-01 0.0003415374
## 407    0    0 9.996708e-01 0.0003292308
## 408    0    0 9.934512e-01 0.0065488362
## 409    0    0 9.996564e-01 0.0003436288
## 410    0    0 9.996311e-01 0.0003689142
## 411    0    0 9.862889e-01 0.0137110822
## 412    0    0 9.996369e-01 0.0003631266
## 413    0    0 9.994754e-01 0.0005246193
## 414    0    0 9.996271e-01 0.0003729054
## 415    0    0 9.996518e-01 0.0003482019
## 416    0    0 9.996224e-01 0.0003776000
## 417    0    0 9.996381e-01 0.0003618560
## 418    0    0 9.996864e-01 0.0003135696
## 419    0    0 9.996170e-01 0.0003830449
## 420    0    0 9.911884e-01 0.0088115995
## 421    0    0 9.996511e-01 0.0003488732
## 422    0    0 9.944865e-01 0.0055135034
## 423    0    0 9.996640e-01 0.0003359722
## 424    0    0 9.807576e-01 0.0192424476
## 425    0    0 9.996874e-01 0.0003126000
## 426    0    0 9.933791e-01 0.0066209019
## 427    0    0 9.995371e-01 0.0004628647
## 428    0    0 9.996313e-01 0.0003687010
## 429    0    1 8.984148e-02 0.9101585176
## 430    0    0 9.995159e-01 0.0004840951
## 431    0    0 9.996395e-01 0.0003604630
## 432    0    1 6.894470e-02 0.9310553017
## 433    0    0 9.709416e-01 0.0290584430
## 434    0    0 9.996127e-01 0.0003873322
```

```
## 435    0    0 9.947533e-01 0.0052467131
## 436    0    0 9.996336e-01 0.0003663801
## 437    0    0 9.936199e-01 0.0063801486
## 438    0    0 9.874796e-01 0.0125203667
## 439    0    0 9.996701e-01 0.0003299235
## 440    0    0 9.996591e-01 0.0003409141
## 441    0    0 9.937485e-01 0.0062515441
## 442    0    0 9.996603e-01 0.0003396977
## 443    0    0 9.996533e-01 0.0003467060
## 444    0    0 9.996512e-01 0.0003487933
## 445    0    0 9.948527e-01 0.0051472913
## 446    0    0 9.996801e-01 0.0003199216
## 447    0    0 9.937803e-01 0.0062196940
## 448    0    0 9.921761e-01 0.0078238899
## 449    0    0 9.996956e-01 0.0003044406
## 450    0    0 9.996391e-01 0.0003609196
## 451    0    0 9.996343e-01 0.0003657477
## 452    0    0 9.996732e-01 0.0003267591
## 453    0    0 9.856587e-01 0.0143413187
## 454    0    0 9.947342e-01 0.0052657879
## 455    0    0 9.996407e-01 0.0003592702
## 456    0    0 9.951069e-01 0.0048930525
## 457    0    0 9.996591e-01 0.0003409196
## 458    0    0 9.996218e-01 0.0003782129
## 459    0    0 9.941154e-01 0.0058846009
## 460    0    0 9.996477e-01 0.0003523333
## 461    0    0 9.996056e-01 0.0003943508
## 462    0    0 9.996677e-01 0.0003323473
## 463    0    0 9.996394e-01 0.0003605637
## 464    0    0 9.996308e-01 0.0003692494
## 465    0    0 9.995718e-01 0.0004281821
## 466    0    0 9.996501e-01 0.0003499171
## 467    0    0 9.941273e-01 0.0058726998
## 468    0    0 9.917040e-01 0.0082960255
## 469    0    0 9.996674e-01 0.0003325554
## 470    0    0 8.830252e-01 0.1169747847
## 471    0    0 9.947310e-01 0.0052690494
## 472    0    0 9.871313e-01 0.0128687124
## 473    0    1 5.074243e-01 0.4925757290
## 474    0    0 9.917348e-01 0.0082651957
## 475    0    1 1.621628e-01 0.8378371906
## 476    0    0 9.996540e-01 0.0003460369
## 477    0    0 9.996078e-01 0.0003921694
## 478    0    0 9.996466e-01 0.0003534301
## 479    0    0 9.996557e-01 0.0003443146
## 480    0    0 9.997001e-01 0.0002998613
## 481    0    0 9.996696e-01 0.0003304239
## 482    0    1 4.771565e-02 0.9522843508
## 483    0    0 9.996508e-01 0.0003491668
## 484    0    0 9.946471e-01 0.0053528861
## 485    0    0 9.996671e-01 0.0003329329
## 486    0    0 9.996118e-01 0.0003882217
## 487    0    0 9.996753e-01 0.0003246599
## 488    0    0 9.996376e-01 0.0003623562
```

```
## 489     0     0 9.996755e-01 0.0003244930
## 490     0     0 9.996573e-01 0.0003426847
## 491     0     0 9.924392e-01 0.0075607829
## 492     0     0 9.995136e-01 0.0004864299
## 493     0     0 9.996558e-01 0.0003441701
## 494     0     0 9.797259e-01 0.0202741201
## 495     0     0 9.940510e-01 0.0059490255
## 496     0     0 9.861670e-01 0.0138330327
## 497     0     0 9.995563e-01 0.0004437397
## 498     0     0 9.945946e-01 0.0054054344
## 499     0     0 9.770864e-01 0.0229135922
## 500     0     0 9.996779e-01 0.0003221274
## 501     0     0 9.996951e-01 0.0003048512
## 502     0     0 9.855808e-01 0.0144191750
## 503     0     0 9.996910e-01 0.0003089742
## 504     0     0 9.996314e-01 0.0003686301
## 505     0     0 9.996229e-01 0.0003770684
## 506     0     0 9.942809e-01 0.0057191352
## 507     0     0 9.996835e-01 0.0003165271
## 508     0     0 9.879622e-01 0.0120378173
## 509     0     0 9.923645e-01 0.0076354963
## 510     0     0 9.890656e-01 0.0109343689
## 511     0     0 9.926519e-01 0.0073480550
## 512     0     0 9.807427e-01 0.0192572846
## 513     0     0 7.431244e-01 0.2568756393
## 514     0     0 9.900953e-01 0.0099046977
## 515     0     0 9.996074e-01 0.0003925943
## 516     0     0 9.927675e-01 0.0072324740
## 517     0     0 9.936636e-01 0.0063364289
## 518     0     0 9.996940e-01 0.0003059547
## 519     0     0 9.996031e-01 0.0003968815
## 520     0     0 9.996659e-01 0.0003340631
## 521     0     0 9.996312e-01 0.0003688367
## 522     0     0 9.904883e-01 0.0095117059
## 523     0     0 9.996279e-01 0.0003720909
## 524     0     0 9.951096e-01 0.0048903903
## 525     0     0 9.996723e-01 0.0003277258
## 526     0     1 2.018225e-01 0.7981774881
## 527     0     0 7.403162e-01 0.2596838241
## 528     0     0 9.911843e-01 0.0088157392
## 529     0     0 9.995432e-01 0.0004568421
## 530     0     0 9.938111e-01 0.0061888999
## 531     0     0 9.927974e-01 0.0072026195
## 532     0     0 9.901433e-01 0.0098566935
## 533     0     0 9.996539e-01 0.0003460799
## 534     0     0 9.996651e-01 0.0003348734
## 535     0     0 9.910913e-01 0.0089086713
## 536     0     0 9.923767e-01 0.0076233476
## 537     0     1 2.255805e-01 0.7744194843
## 538     0     0 9.996239e-01 0.0003761164
## 539     0     0 9.538025e-01 0.0461975464
## 540     0     0 9.946332e-01 0.0053668131
## 541     0     0 9.996471e-01 0.0003528778
## 542     0     0 9.995975e-01 0.0004025399
```

```
## 543    0     0 9.995455e-01 0.0004544915
## 544    0     0 9.996756e-01 0.0003243854
## 545    0     0 9.915530e-01 0.0084470041
## 546    0     0 9.860936e-01 0.0139064259
## 547    0     0 9.996058e-01 0.0003941895
## 548    0     0 6.652497e-01 0.3347503244
## 549    0     0 9.996577e-01 0.0003423422
## 550    0     0 9.868945e-01 0.0131054823
## 551    0     0 9.823631e-01 0.0176369050
## 552    0     0 9.994877e-01 0.0005122766
## 553    0     0 9.945484e-01 0.0054516134
## 554    0     0 9.964960e-01 0.0035040241
## 555    0     0 9.996706e-01 0.0003293975
## 556    0     0 9.996321e-01 0.0003678658
## 557    0     0 9.856133e-01 0.0143866661
## 558    0     0 9.996976e-01 0.0003024141
## 559    0     0 9.996176e-01 0.0003823861
## 560    0     0 9.929020e-01 0.0070980393
## 561    0     0 9.905897e-01 0.0094103360
## 562    0     0 9.898113e-01 0.0101886707
## 563    0     0 9.942459e-01 0.0057540993
## 564    0     0 9.996830e-01 0.0003169633
## 565    0     0 9.996482e-01 0.0003517887
## 566    1     1 8.737093e-06 0.9999912629
## 567    1     0 5.677887e-01 0.4322112584
## 568    1     1 1.659978e-01 0.8340022255
## 569    1     1 6.777954e-03 0.9932220456
## 570    1     1 3.056888e-02 0.9694311247
## 571    1     1 2.244395e-02 0.9775560484
## 572    1     1 2.368731e-06 0.9999976313
## 573    1     1 7.373757e-06 0.9999926262
## 574    1     1 1.795620e-01 0.8204380448
## 575    1     1 2.136436e-01 0.7863563575
## 576    1     1 9.659124e-03 0.9903408760
## 577    1     1 6.051368e-06 0.9999939486
## 578    1     1 2.686443e-06 0.9999973136
## 579    1     1 3.408562e-06 0.9999965914
## 580    1     1 1.911443e-02 0.9808855680
## 581    1     1 5.394782e-03 0.9946052179
## 582    1     1 2.566987e-02 0.9743301281
## 583    1     1 6.206341e-06 0.9999937937
## 584    1     1 2.883469e-01 0.7116530719
## 585    1     1 6.175206e-02 0.9382479410
## 586    1     1 4.170124e-06 0.9999958299
## 587    1     1 2.387002e-02 0.9761299787
## 588    1     1 4.052260e-02 0.9594774032
## 589    1     1 1.153117e-02 0.9884688337
## 590    1     1 7.818274e-02 0.9218172564
## 591    1     1 2.944585e-03 0.9970554150
## 592    1     1 8.172617e-02 0.9182738278
## 593    1     0 8.406214e-01 0.1593786252
## 594    1     1 4.049329e-06 0.9999959507
## 595    1     1 4.053989e-06 0.9999959460
## 596    1     1 1.225997e-01 0.8774002807
```

```
## 597    1    1 7.272393e-06 0.9999927276
## 598    1    1 1.012183e-02 0.9898781673
## 599    1    1 1.166225e-01 0.8833775416
## 600    1    1 3.994706e-03 0.9960052943
## 601    1    1 2.722315e-02 0.9727768537
## 602    1    1 2.389436e-01 0.7610564061
## 603    1    1 1.873662e-05 0.9999812634
## 604    1    1 3.524823e-01 0.6475176787
## 605    1    1 2.480032e-06 0.9999975200
## 606    1    1 6.765705e-06 0.9999932343
## 607    1    1 1.737592e-02 0.9826240793
## 608    1    1 5.363641e-02 0.9463635914
## 609    1    1 6.477129e-06 0.9999935229
## 610    1    1 1.096523e-05 0.9999890348
## 611    1    1 3.957517e-06 0.9999960425
## 612    1    1 3.411179e-01 0.6588820758
## 613    1    1 6.818786e-06 0.9999931812
## 614    1    1 9.508913e-03 0.9904910872
## 615    1    1 2.026451e-02 0.9797354868
## 616    1    1 2.943017e-01 0.7056982683
## 617    1    1 1.416042e-02 0.9858395850
## 618    1    1 5.028667e-06 0.9999949713
## 619    1    1 1.012150e-01 0.8987850066
## 620    1    1 1.560715e-02 0.9843928450
## 621    1    1 7.920159e-03 0.9920798414
## 622    1    1 1.544611e-01 0.8455389301
## 623    1    1 1.540995e-02 0.9845900529
## 624    1    1 6.332683e-03 0.9936673171
## 625    1    1 2.320976e-02 0.9767902395
## 626    1    1 7.871854e-06 0.9999921281
## 627    1    1 4.104078e-02 0.9589592187
## 628    1    1 7.645041e-02 0.9235495932
## 629    1    1 1.136373e-02 0.9886362738
## 630    1    1 5.133216e-06 0.9999948668
## 631    1    1 3.543270e-01 0.6456730147
## 632    1    1 9.692308e-03 0.9903076921
## 633    1    1 2.044740e-05 0.9999795526
## 634    1    1 1.824183e-02 0.9817581664
## 635    1    1 5.692392e-03 0.9943076082
## 636    1    1 3.302826e-01 0.6697174337
## 637    1    1 3.065520e-03 0.9969344796
## 638    1    0 9.518889e-01 0.0481111399
## 639    1    1 1.531764e-02 0.9846823564
## 640    1    1 7.376164e-03 0.9926238359
## 641    1    1 5.473886e-06 0.9999945261
## 642    1    0 8.429623e-01 0.1570376515
## 643    1    1 1.476681e-05 0.9999852332
## 644    1    1 1.323085e-01 0.8676915086
## 645    1    1 4.291216e-06 0.9999957088
## 646    1    1 3.393925e-02 0.9660607518
## 647    1    1 3.639080e-06 0.9999963609
## 648    1    1 7.643260e-06 0.9999923567
## 649    1    0 7.012495e-01 0.2987504851
## 650    1    1 1.799621e-02 0.9820037922
```

```
## 651    1     1 2.353533e-02 0.9764646697
## 652    1     1 5.896066e-02 0.9410393438
## 653    1     1 1.802389e-02 0.9819761087
## 654    1     1 1.425422e-01 0.8574578110
## 655    1     1 4.362055e-06 0.9999956379
## 656    1     1 1.234619e-02 0.9876538112
## 657    1     1 1.029876e-01 0.8970124156
## 658    1     1 1.550772e-02 0.9844922790
## 659    1     1 1.473667e-02 0.9852633324
## 660    1     1 2.981121e-02 0.9701887934
## 661    1     1 6.722494e-02 0.9327750643
## 662    1     1 4.062188e-02 0.9593781233
## 663    1     1 6.070395e-03 0.9939296046
## 664    1     1 1.550813e-02 0.9844918681
## 665    1     1 3.716040e-03 0.9962839598
## 666    1     1 1.887570e-01 0.8112430202
## 667    1     1 2.536279e-02 0.9746372069
## 668    1     1 5.234367e-03 0.9947656335
## 669    1     1 2.938410e-01 0.7061589539
## 670    1     1 9.308918e-02 0.9069108208
## 671    1     1 1.121544e-01 0.8878455796
## 672    1     1 5.092188e-03 0.9949078123
## 673    1     1 7.938089e-06 0.9999920619
## 674    1     1 7.286548e-06 0.9999927135
## 675    1     1 4.466380e-03 0.9955336197
## 676    1     1 2.894454e-01 0.7105545706
## 677    1     1 4.865543e-02 0.9513445747
## 678    1     1 2.599124e-06 0.9999974009
## 679    1     1 3.679456e-02 0.9632054382
## 680    1     1 7.711020e-03 0.9922889802
## 681    1     1 8.185238e-02 0.9181476206
## 682    1     1 5.896941e-02 0.9410305936
## 683    1     1 1.983012e-01 0.8016988214
## 684    1     1 1.806835e-01 0.8193164872
## 685    1     1 9.748585e-02 0.9025141533
## 686    1     1 1.256813e-02 0.9874318663
## 687    1     1 3.174431e-01 0.6825569319
## 688    1     1 1.006309e-01 0.8993690786
## 689    1     1 1.222079e-05 0.9999877792
## 690    1     1 1.705107e-02 0.9829489325
## 691    1     1 2.014132e-02 0.9798586781
## 692    1     1 2.261410e-01 0.7738589890
## 693    1     1 8.468749e-02 0.9153125106
## 694    1     1 3.258434e-02 0.9674156582
## 695    1     1 7.540982e-06 0.9999924590
## 696    1     1 9.623481e-06 0.9999903765
## 697    1     1 8.041309e-02 0.9195869149
## 698    1     1 6.885335e-06 0.9999931147
## 699    1     1 5.308709e-06 0.9999946913
## 700    1     1 3.281694e-02 0.9671830550
## 701    1     1 8.925161e-03 0.9910748385
## 702    1     1 9.838484e-02 0.9016151621
## 703    1     1 4.998497e-03 0.9950015027
## 704    1     1 1.199008e-01 0.8800991690
```

```
## 705    1    1 4.095989e-01 0.5904010822
## 706    1    1 8.393400e-02 0.9160659982
## 707    1    1 1.796652e-02 0.9820334847
## 708    1    1 3.620496e-03 0.9963795040
## 709    1    1 3.672843e-03 0.9963271571
## 710    1    1 1.040714e-02 0.9895928627
## 711    1    1 6.560048e-06 0.9999934400
## 712    1    1 5.429675e-03 0.9945703248
## 713    1    1 8.496535e-02 0.9150346544
## 714    1    1 2.098064e-02 0.9790193597
## 715    1    1 6.362191e-06 0.9999936378
## 716    1    1 6.211578e-02 0.9378842155
## 717    1    1 5.597160e-02 0.9440283982
## 718    1    1 5.470453e-06 0.9999945295
## 719    1    1 1.255214e-05 0.9999874479
## 720    1    1 2.895790e-02 0.9710420971
## 721    1    1 2.434020e-02 0.9756598038
## 722    1    1 8.143374e-06 0.9999918566
## 723    1    1 6.006723e-06 0.9999939933
## 724    1    1 4.572509e-03 0.9954274913
## 725    1    1 1.268776e-02 0.9873122414
## 726    1    1 1.002537e-02 0.9899746262
## 727    1    1 9.824579e-02 0.9017542058
## 728    1    1 5.023061e-06 0.9999949769
## 729    1    1 1.956679e-02 0.9804332117
## 730    1    1 4.390053e-02 0.9560994725
## 731    1    1 1.865232e-02 0.9813476775
## 732    1    1 5.245660e-03 0.9947543397
## 733    1    1 1.174061e-01 0.8825939354
## 734    1    1 4.849856e-06 0.9999951501
## 735    1    1 1.688567e-01 0.8311433163
## 736    1    1 4.384169e-03 0.9956158314
## 737    1    1 2.894301e-06 0.9999971057
## 738    1    1 5.885201e-02 0.9411479851
## 739    1    1 2.429910e-06 0.9999975701
## 740    1    1 1.761952e-05 0.9999823805
## 741    1    1 6.871158e-02 0.9312884228
## 742    1    1 1.443727e-02 0.9855627266
## 743    1    1 2.414673e-02 0.9758532706
## 744    1    1 1.271464e-02 0.9872853633
## 745    1    1 1.543640e-02 0.9845636038
## 746    1    1 6.814593e-02 0.9318540707
## 747    1    1 9.776083e-06 0.9999902239
## 748    1    1 3.244112e-01 0.6755887843
## 749    1    1 9.625713e-06 0.9999903743
## 750    1    1 3.546919e-06 0.9999964531
## 751    1    1 3.894983e-01 0.6105017066
## 752    1    1 9.440987e-03 0.9905590131
## 753    1    1 4.379745e-03 0.9956202547
## 754    1    1 7.638623e-02 0.9236137703
## 755    1    1 2.460713e-06 0.9999975393
## 756    1    1 1.146020e-02 0.9885398001
## 757    1    1 5.091812e-06 0.9999949082
## 758    1    1 5.886564e-06 0.9999941134
```

```
## 759    1    1 1.142440e-01 0.8857560423
## 760    1    1 1.847253e-02 0.9815274698
## 761    1    1 3.782469e-03 0.9962175307
## 762    1    1 3.260927e-06 0.9999967391
## 763    1    1 5.490023e-06 0.9999945100
## 764    1    1 5.981689e-06 0.9999940183
## 765    1    1 1.487360e-01 0.8512639750
## 766    1    1 7.629686e-03 0.9923703140
## 767    1    1 7.255968e-06 0.9999927440
## 768    1    1 7.053113e-06 0.9999929469
## 769    1    0 6.682028e-01 0.3317971859
## 770    1    1 6.178175e-06 0.9999938218
## 771    1    1 2.892916e-03 0.9971070841
## 772    1    1 7.963736e-03 0.9920362639
## 773    1    1 2.309515e-06 0.9999976905
## 774    1    1 1.030083e-02 0.9896991739
## 775    1    1 7.417955e-03 0.9925820447
## 776    1    0 6.595621e-01 0.3404378788
## 777    1    1 3.788769e-02 0.9621123130
## 778    1    1 2.002727e-06 0.9999979973
## 779    1    1 7.643033e-02 0.9235696653
## 780    1    1 2.573715e-02 0.9742628507
## 781    1    1 5.322779e-03 0.9946772210
## 782    1    1 4.160785e-06 0.9999958392
## 783    1    1 2.154086e-02 0.9784591436
## 784    1    1 1.249958e-01 0.8750042450
## 785    1    1 2.485494e-06 0.9999975145
## 786    1    1 3.563239e-06 0.9999964368
## 787    1    1 1.550127e-02 0.9844987254
## 788    1    1 4.581237e-03 0.9954187632
## 789    1    1 7.705664e-03 0.9922943360
## 790    1    1 1.005759e-01 0.8994240845
## 791    1    1 4.502723e-02 0.9549727699
## 792    1    1 4.642078e-06 0.9999953579
## 793    1    1 5.173112e-03 0.9948268882
## 794    1    1 8.027650e-03 0.9919723501
## 795    1    1 6.800087e-02 0.9319991288
## 796    1    1 8.592306e-03 0.9914076942
## 797    1    1 2.393135e-06 0.9999976069
## 798    1    1 2.155055e-01 0.7844945079
## 799    1    1 7.762156e-02 0.9223784443
## 800    1    1 2.178241e-01 0.7821759037
## 801    1    1 1.629246e-01 0.8370753752
## 802    1    1 6.431248e-06 0.9999935688
## 803    1    1 1.992367e-02 0.9800763306
## 804    1    1 2.718772e-06 0.9999972812
## 805    1    1 3.362745e-05 0.9999663725
## 806    1    1 3.160809e-02 0.9683919088
## 807    1    1 7.706271e-06 0.9999922937
## 808    1    1 2.986969e-02 0.9701303111
## 809    1    1 9.559629e-02 0.9044037071
## 810    1    1 4.611237e-02 0.9538876324
## 811    1    1 3.029475e-02 0.9697052467
## 812    1    1 9.730132e-06 0.9999902699
```

```
## 813    1    1 7.349182e-02 0.9265081819
## 814    1    1 6.083180e-03 0.9939168203
## 815    1    1 1.958941e-01 0.8041058934
## 816    1    0 8.276627e-01 0.1723372912
## 817    1    1 8.900201e-02 0.9109979854
## 818    1    1 1.431040e-02 0.9856896034
## 819    1    1 4.156979e-06 0.9999958430
## 820    1    1 4.366882e-06 0.9999956331
## 821    1    1 7.869151e-06 0.9999921308
## 822    1    1 1.021073e-02 0.9897892665
## 823    1    1 6.846075e-03 0.9931539248
## 824    1    1 1.952803e-02 0.9804719719
## 825    1    1 9.872177e-02 0.9012782260
## 826    1    1 4.716499e-06 0.9999952835
## 827    1    1 1.854103e-02 0.9814589717
## 828    1    1 6.399399e-02 0.9360060057
## 829    1    1 5.971421e-03 0.9940285790
## 830    1    1 3.484403e-03 0.9965155974
## 831    1    1 2.307040e-06 0.9999976930
## 832    1    1 6.506660e-06 0.9999934933
## 833    1    1 8.256869e-03 0.9917431310
## 834    1    1 7.435895e-06 0.9999925641
## 835    1    1 5.954298e-02 0.9404570179
## 836    1    1 3.795956e-03 0.9962040440
## 837    1    1 2.784224e-02 0.9721577555
## 838    1    1 2.970026e-02 0.9702997384
## 839    1    0 7.139587e-01 0.2860412635
## 840    1    1 2.594793e-02 0.9740520672
## 841    1    1 6.055201e-03 0.9939447990
## 842    1    1 9.644383e-06 0.9999903556
## 843    1    1 7.855578e-06 0.9999921444
## 844    1    1 5.294596e-03 0.9947054038
## 845    1    1 4.985792e-03 0.9950142081
## 846    1    1 3.276924e-02 0.9672307603
## 847    1    1 1.212081e-01 0.8787918731
## 848    1    1 2.210013e-02 0.9778998672
## 849    1    1 9.095108e-02 0.9090489170
## 850    1    1 2.958585e-01 0.7041415450
## 851    1    1 7.848316e-06 0.9999921517
## 852    1    1 1.053690e-01 0.8946310256
## 853    1    1 1.558820e-02 0.9844118035
## 854    1    1 5.918091e-03 0.9940819088
## 855    1    1 3.965144e-06 0.9999960349
## 856    1    1 1.120651e-02 0.9887934866
## 857    1    1 1.171823e-02 0.9882817724
## 858    1    1 6.383843e-06 0.9999936162
## 859    1    1 7.020309e-02 0.9297969127
## 860    1    1 4.563751e-01 0.5436248577
## 861    1    1 1.303489e-01 0.8696510609
## 862    1    1 2.213703e-02 0.9778629701
## 863    1    1 1.208090e-02 0.9879191046
## 864    1    1 3.469321e-02 0.9653067919
## 865    1    1 6.143432e-06 0.9999938566
## 866    1    1 9.959654e-02 0.9004034584
```

```
## 867     1     1 2.425778e-06 0.9999975742
## 868     1     1 3.747476e-02 0.9625252379
## 869     1     1 7.668964e-03 0.9923310357
## 870     1     1 1.246298e-05 0.9999875370
## 871     1     1 2.068884e-01 0.7931116003
## 872     1     1 2.428253e-01 0.7571746638
## 873     1     1 8.740639e-06 0.9999912594
## 874     1     1 3.073294e-06 0.9999969267
## 875     1     1 3.137226e-06 0.9999968628
## 876     1     1 4.934265e-06 0.9999950657
## 877     1     1 3.299411e-01 0.6700588831
## 878     1     0 6.930735e-01 0.3069264630
## 879     1     1 1.392635e-02 0.9860736480
## 880     1     1 8.175596e-06 0.9999918244
## 881     1     1 2.456562e-06 0.9999975434
## 882     1     1 6.666303e-06 0.9999933337
## 883     1     1 6.854128e-02 0.9314587206
## 884     1     1 5.138040e-06 0.9999948620
## 885     1     1 5.123755e-06 0.9999948762
## 886     1     1 3.376130e-03 0.9966238699
## 887     1     1 7.173486e-06 0.9999928265
## 888     1     1 8.305355e-06 0.9999916946
## 889     1     1 1.157266e-01 0.8842733553
## 890     1     1 3.658485e-02 0.9634151467
## 891     1     1 1.190762e-01 0.8809238357
## 892     1     1 7.129757e-03 0.9928702430
## 893     1     1 1.919294e-02 0.9808070590
## 894     1     1 4.541747e-02 0.9545825277
## 895     1     1 1.569724e-02 0.9843027586
## 896     1     1 8.182245e-02 0.9181775530
## 897     1     1 9.957270e-06 0.9999900427
## 898     1     1 5.346651e-06 0.9999946533
## 899     1     1 2.415738e-01 0.7584262246
## 900     1     1 6.464336e-03 0.9935356639
## 901     1     1 2.313228e-03 0.9976867723
## 902     1     1 1.540434e-02 0.9845956641
## 903     1     1 1.115614e-02 0.9888438637
## 904     1     1 4.676514e-02 0.9532348584
## 905     1     1 9.775286e-03 0.9902247136
## 906     1     1 1.433733e-01 0.8566267209
## 907     1     1 2.914957e-02 0.9708504259
## 908     1     1 2.699150e-03 0.9973008498
## 909     1     1 2.124915e-01 0.7875084610
## 910     1     1 4.372466e-01 0.5627534299
## 911     1     1 4.944386e-03 0.9950556142
## 912     1     1 3.303731e-06 0.9999966963
## 913     1     1 4.506555e-02 0.9549344465
## 914     1     1 4.565286e-02 0.9543471428
## 915     1     1 4.357744e-06 0.9999956423
## 916     1     1 6.340765e-03 0.9936592353
## 917     1     1 5.924013e-03 0.9940759874
## 918     1     1 2.187154e-01 0.7812845606
## 919     1     1 7.186833e-02 0.9281316718
## 920     1     1 8.822517e-03 0.9911774827
```

```
## 921     1     1 4.102089e-01 0.5897910775
## 922     1     1 5.054401e-06 0.9999949456
## 923     1     1 3.187821e-06 0.9999968122
## 924     1     1 6.056746e-02 0.9394325394
## 925     1     1 1.373375e-02 0.9862662546
## 926     1     1 4.031269e-06 0.9999959687
## 927     1     1 3.744699e-06 0.9999962553
## 928     1     1 2.245416e-03 0.9977545842
## 929     1     1 4.442094e-06 0.9999955579
## 930     1     1 3.711608e-06 0.9999962884
## 931     1     1 2.891840e-02 0.9710815965
## 932     1     1 1.097296e-02 0.9890270427
## 933     1     1 4.539883e-06 0.9999954601
## 934     1     1 9.297316e-06 0.9999907027
## 935     1     1 1.106487e-02 0.9889351345
## 936     1     1 7.651381e-06 0.9999923486
## 937     1     1 2.380779e-06 0.9999976192
## 938     1     1 5.678707e-02 0.9432129294
## 939     1     1 7.319060e-02 0.9268094000
## 940     1     1 7.071015e-03 0.9929289853
## 941     1     1 1.084375e-01 0.8915624627
## 942     1     1 1.396421e-01 0.8603579300
## 943     1     1 4.312858e-06 0.9999956871
## 944     1     1 2.013037e-01 0.7986963205
## 945     1     1 3.667683e-01 0.6332316866
## 946     1     1 5.614223e-02 0.9438577696
## 947     1     1 5.421945e-06 0.9999945781
## 948     1     0 9.295232e-01 0.0704767892
## 949     1     1 6.186023e-06 0.9999938140
## 950     1     1 9.546309e-03 0.9904536915
## 951     1     1 6.821787e-02 0.9317821315
## 952     1     1 5.531184e-06 0.9999944688
## 953     1     1 9.830813e-03 0.9901691868
## 954     1     1 8.055550e-03 0.9919444496
## 955     1     1 4.012562e-02 0.9598743822
## 956     1     1 8.205944e-02 0.9179405640
## 957     1     1 1.251684e-02 0.9874831637
## 958     1     1 1.053608e-02 0.9894639175
## 959     1     1 1.158899e-02 0.9884110090
## 960     1     1 1.503276e-01 0.8496724390
## 961     1     1 9.610784e-06 0.9999903892
## 962     1     0 6.980314e-01 0.3019686159
## 963     1     1 4.363478e-02 0.9563652194
## 964     1     1 5.915441e-03 0.9940845588
## 965     1     1 2.507948e-02 0.9749205242
## 966     1     1 5.430216e-03 0.9945697843
## 967     1     1 1.283618e-02 0.9871638196
## 968     1     1 4.062638e-06 0.9999959374
## 969     1     1 9.983204e-06 0.9999900168
## 970     1     1 7.900190e-02 0.9209981025
## 971     1     1 3.892635e-02 0.9610736489
## 972     1     1 4.345627e-02 0.9565437300
## 973     1     1 2.362139e-02 0.9763786057
## 974     1     1 2.119807e-02 0.9788019318
```

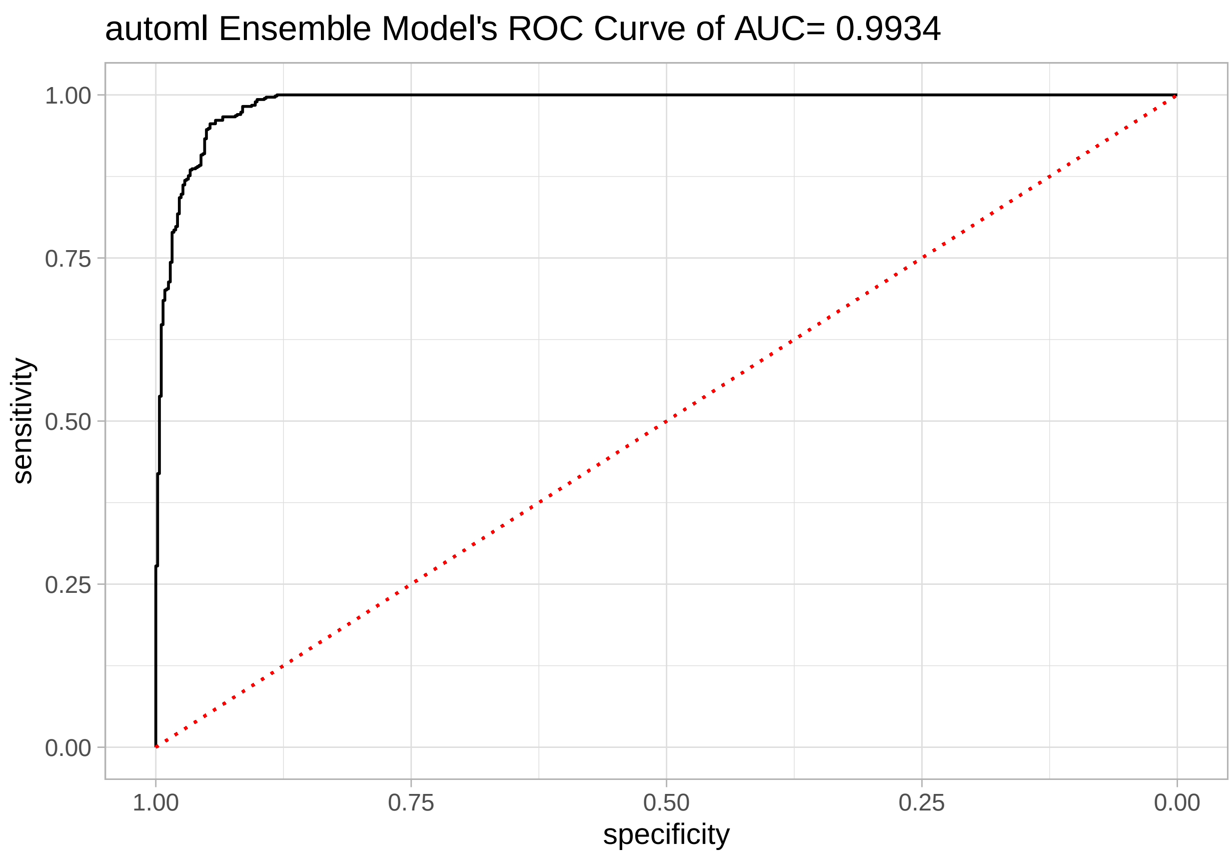
```
## 975     1     1 3.652615e-06 0.9999963474
## 976     1     1 1.782767e-02 0.9821723258
## 977     1     1 5.713089e-02 0.9428691103
## 978     1     1 6.502894e-06 0.9999934971
## 979     1     1 3.958497e-06 0.9999960415
## 980     1     1 1.464885e-02 0.9853511460
## 981     1     1 1.682882e-02 0.9831711816
## 982     1     1 4.254202e-01 0.5745797771
## 983     1     1 2.759892e-02 0.9724010755
## 984     1     1 8.052633e-06 0.9999919474
## 985     1     1 8.968882e-03 0.9910311183
## 986     1     1 7.434060e-02 0.9256594023
## 987     1     1 9.808429e-03 0.9901915712
## 988     1     1 4.453508e-06 0.9999955465
## 989     1     1 1.203592e-02 0.9879640815
## 990     1     1 2.595711e-06 0.9999974043
## 991     1     0 7.689119e-01 0.2310880816
## 992     1     1 7.616159e-06 0.9999923838
## 993     1     0 9.172507e-01 0.0827493208
## 994     1     1 8.813904e-03 0.9911860961
## 995     1     1 1.493951e-01 0.8506048959
## 996     1     1 1.106501e-01 0.8893498922
## 997     1     1 9.238829e-06 0.9999907612
## 998     1     1 9.637498e-06 0.9999903625
## 999     1     1 3.891940e-03 0.9961080601
## 1000    1     1 7.242686e-02 0.9275731370
## 1001    1     1 7.088037e-02 0.9291196325
## 1002    1     1 2.280161e-06 0.9999977198
## 1003    1     1 2.536206e-02 0.9746379435
## 1004    1     1 1.292113e-02 0.9870788658
## 1005    1     1 9.153254e-03 0.9908467457
## 1006    1     1 1.314393e-01 0.8685606595
## 1007    1     1 3.666508e-03 0.9963334918
## 1008    1     1 1.880980e-01 0.8119020035
## 1009    1     0 8.479348e-01 0.1520651776
## 1010    1     1 1.833088e-02 0.9816691244
## 1011    1     1 2.821072e-02 0.9717892766
## 1012    1     1 2.714078e-06 0.9999972859
## 1013    1     1 2.376748e-06 0.9999976233
## 1014    1     1 9.609869e-06 0.9999903901
## 1015    1     1 2.160968e-01 0.7839031688
## 1016    1     1 6.998975e-02 0.9300102489
## 1017    1     1 4.018741e-06 0.9999959813
## 1018    1     1 6.064156e-06 0.9999939358
## 1019    1     1 5.313296e-06 0.9999946867
## 1020    1     1 1.930176e-02 0.9806982448
## 1021    1     1 2.107916e-02 0.9789208393
## 1022    1     1 8.439490e-02 0.9156051002
## 1023    1     1 2.775060e-06 0.9999972249
## 1024    1     1 3.513156e-06 0.9999964868
## 1025    1     1 3.781496e-02 0.9621850364
## 1026    1     1 4.964102e-06 0.9999950359
## 1027    1     1 1.511383e-02 0.9848861726
## 1028    1     1 2.249919e-01 0.7750080857
```

```
## 1029    1     1 3.007452e-01 0.6992548219
## 1030    1     1 1.590861e-02 0.9840913918
## 1031    1     1 2.013096e-02 0.9798690439
## 1032    1     1 1.002301e-01 0.8997698875
## 1033    1     1 7.178453e-06 0.9999928215
## 1034    1     1 2.498615e-01 0.7501384896
## 1035    1     1 1.350497e-02 0.9864950335
## 1036    1     1 5.389820e-06 0.9999946102
## 1037    1     1 1.772382e-02 0.9822761786
## 1038    1     1 3.100029e-06 0.9999969000
## 1039    1     1 7.939194e-06 0.9999920608
## 1040    1     1 2.926586e-02 0.9707341374
## 1041    1     1 4.690067e-01 0.5309933062
## 1042    1     1 7.780415e-02 0.9221958456
## 1043    1     0 7.042189e-01 0.2957811351
## 1044    1     1 3.429821e-02 0.9657017937
## 1045    1     0 5.662580e-01 0.4337420439
## 1046    1     1 9.225899e-06 0.9999907741
## 1047    1     1 3.359457e-02 0.9664054315
## 1048    1     1 9.143807e-02 0.9085619324
## 1049    1     1 3.203748e-03 0.9967962520
## 1050    1     1 1.181931e-01 0.8818069457
## 1051    1     1 3.836395e-06 0.9999961636
## 1052    1     1 7.790978e-03 0.9922090221
## 1053    1     1 1.226005e-02 0.9877399519
## 1054    1     1 1.716293e-01 0.8283707142
## 1055    1     1 3.941073e-02 0.9605892740
## 1056    1     1 6.494411e-06 0.9999935056
## 1057    1     1 1.912255e-02 0.9808774545
## 1058    1     1 1.351685e-01 0.8648315268
## 1059    1     1 1.255910e-02 0.9874408996
## 1060    1     1 4.272815e-03 0.9957271846
## 1061    1     1 4.689684e-06 0.9999953103
## 1062    1     1 3.177903e-01 0.6822096620
## 1063    1     1 1.013097e-05 0.9999898690
## 1064    1     1 2.887027e-01 0.7112972756
## 1065    1     1 8.014722e-03 0.9919852782
## 1066    1     0 6.348975e-01 0.3651024512
## 1067    1     1 5.157896e-06 0.9999948421
## 1068    1     1 2.911067e-01 0.7088933164
## 1069    1     1 1.816704e-01 0.8183295715
## 1070    1     1 1.129100e-02 0.9887090037
## 1071    1     1 2.886723e-03 0.9971132767
## 1072    1     1 8.457348e-03 0.9915426519
## 1073    1     1 1.390695e-02 0.9860930506
## 1074    1     1 1.132030e-02 0.9886797018
## 1075    1     1 3.840353e-02 0.9615964661
## 1076    1     1 3.784602e-06 0.9999962154
## 1077    1     1 9.902620e-03 0.9900973803
## 1078    1     0 9.477443e-01 0.0522557275
## 1079    1     1 6.805308e-02 0.9319469223
## 1080    1     1 1.530601e-02 0.9846939857
## 1081    1     1 5.861330e-02 0.9413866985
## 1082    1     1 8.809443e-03 0.9911905567
```

```
## 1083    1     1 7.275384e-03 0.9927246158
## 1084    1     1 5.500715e-03 0.9944992846
## 1085    1     1 5.961142e-02 0.9403885780
## 1086    1     1 7.085289e-06 0.9999929147
## 1087    1     1 5.407754e-02 0.9459224633
## 1088    1     1 2.216889e-01 0.7783111413
## 1089    1     1 9.780024e-06 0.9999902200
## 1090    1     1 1.132231e-01 0.8867768788
## 1091    1     1 3.318591e-02 0.9668140892
## 1092    1     1 6.476530e-06 0.9999935235
## 1093    1     0 5.585080e-01 0.4414919832
## 1094    1     1 1.050939e-05 0.9999894906
## 1095    1     1 1.276114e-02 0.9872388581
## 1096    1     1 9.366235e-02 0.9063376518
## 1097    1     1 9.984705e-03 0.9900152954
## 1098    1     1 3.958442e-02 0.9604155809
## 1099    1     1 9.971424e-03 0.9900285757
## 1100    1     1 2.660078e-02 0.9733992238
## 1101    1     1 4.028436e-06 0.9999959716
## 1102    1     1 2.223622e-06 0.9999977764
## 1103    1     1 4.155271e-02 0.9584472934
## 1104    1     1 7.529671e-06 0.9999924703
## 1105    1     1 1.158061e-05 0.9999884194
## 1106    1     1 6.980870e-02 0.9301913027
## 1107    1     1 9.978140e-02 0.9002185962
## 1108    1     0 6.721807e-01 0.3278193116
## 1109    1     1 3.913374e-06 0.9999960866
## 1110    1     1 2.613816e-02 0.9738618392
## 1111    1     1 2.721232e-02 0.9727876834
## 1112    1     1 9.880916e-02 0.9011908370
## 1113    1     1 9.374292e-03 0.9906257078
## 1114    1     0 8.438506e-01 0.1561494400
## 1115    1     1 2.108341e-01 0.7891659436
## 1116    1     1 2.268951e-06 0.9999977310
## 1117    1     1 3.890728e-02 0.9610927229
## 1118    1     1 3.589397e-03 0.9964106031
## 1119    1     1 4.466825e-06 0.9999955332
## 1120    1     1 8.437567e-03 0.9915624327
## 1121    1     1 3.529872e-02 0.9647012830
## 1122    1     1 1.371034e-02 0.9862896646
## 1123    1     1 1.083290e-01 0.8916709527
## 1124    1     1 1.190253e-02 0.9880974671
## 1125    1     1 1.557453e-01 0.8442546588
## 1126    1     1 2.903756e-01 0.7096243996
## 1127    1     1 5.245527e-06 0.9999947545
## 1128    1     1 4.549276e-06 0.9999954507
## 1129    1     1 1.117380e-02 0.9888262042
## 1130    1     1 1.319923e-02 0.9868007696
```

#plotting ROC curve of gradient boosted DT

```
ggroc(automl.auc)+ggtitle("automl Ensemble Model's ROC Curve of AUC= 0.9934")+geom_segment(aes(x=1,y=0,
```




```
temp<-temp%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.automl.summary.pred....",MSE=
,RMSE="t.RSME....RMSE.as.numeric.as.character.automl.summary.pred...."
,AUC="t.AUC....automl.auc.auc",classError="t.ClassError....mean.automl.summary.pred....automl.summary.o
```

#adding automled ensemble to the results data frame

```
test.results[3,]<-c(R2=temp$R2,MSE=temp$MSE,RMSE=temp$RMSE,AUC=temp$AUC,classError=temp$classError)
rownames(test.results)<-c("Random Forest","Gradient Boosting","automled Ensemble")
```

#storing confusion matrix testing results

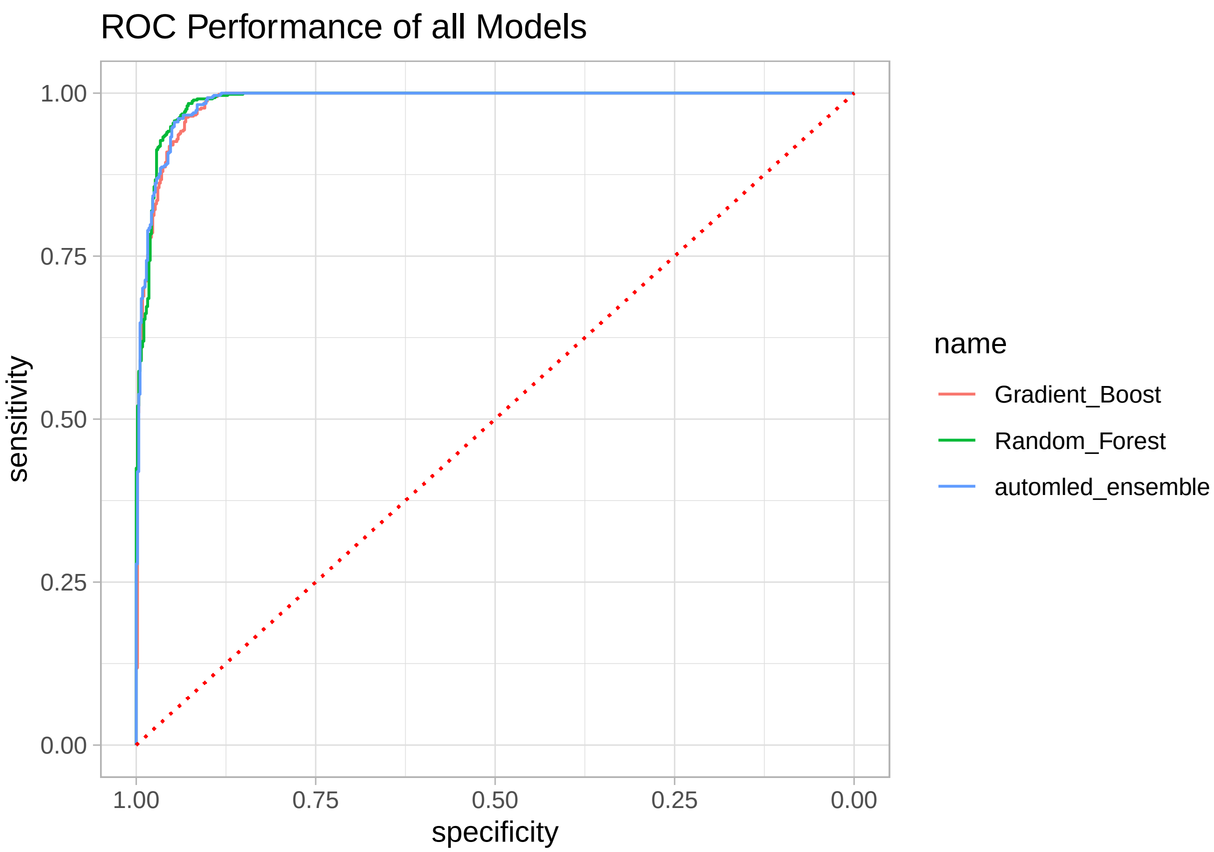
```
temp<-automl.con$byClass%>%as.data.frame()%>%t()
temp<-as.data.frame(temp)

confusionM<-confusionM%>%add_row(Sensitivity= temp$Sensitivity,Specificity=temp$Specificity, Pos_pred_v

rownames(confusionM)<-c("Random Forest","Gradient Boosting","automled Ensemble")
```

#See precision and recall for all three models

```
automl.perf<-h2o.performance(leader_model,test.h2o)%>%h2o.metric()%>%as.data.frame()%>%select(c(recall,
automl.perf$model<-"automled Ensemble"
combine_rpplots<-rbind(rf.perf,gb.perf,automl.perf)

ggplot(combine_rpplots,aes(recall,precision,group=model,color=model))+geom_line()+labs(title ="Precisio
```

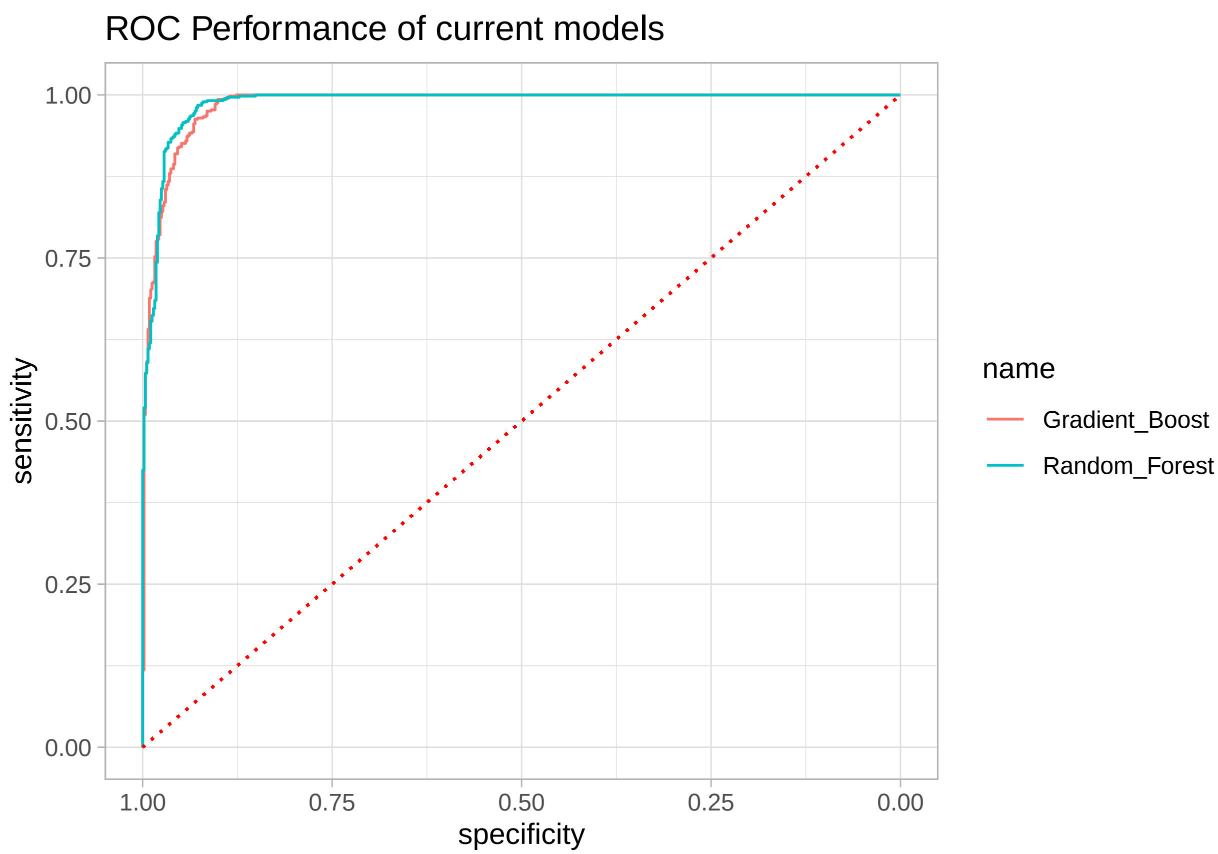
#see ROCs of all models

```
rocs<-list(Gradient_Boost=gb.auc,Random_Forest=rf.auc,automled_ensemble=automl.auc)
ggroc(rocs)+ggtitle("ROC Performance of all Models")+geom_segment(aes(x=1,y=0,xend=0,yend=1),linetype="d
```



```
library(pROC)
library(ggplot2)
```

```
rocs<-list(Gradient_Boost=gb.auc,Random_Forest=rf.auc)
ggroc(rocs)+ggtitle("ROC Performance of current models")+geom_segment(aes(x=1,y=0,xend=0,yend=1),linetyp
```



###Include NB, DPL, and SVM

```
#Including NB, DPL,SVM models in the roc graph

nb.pred<-h2o.predict(pros_nb,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                         |
```

```
nb.precsnprob<-h2o.predict(pros_nb,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                         |
```

```
nb.reclprob<-h2o.predict(pros_nb,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                         |
```

```
nb.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-nb.pred,
      N<-nb.reclprob,
      Y<-nb.precsnprob
  )

svm.pred<-h2o.predict(svm_model,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                              |
```

```r
svm.precsnprob<-h2o.predict(svm_model,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                              |
```

```r
svm.reclprob<-h2o.predict(svm_model,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                              |
```

```r
svm.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-svm.pred,
      N<-svm.reclprob,
      Y<-svm.precsnprob
  )


dl.pred<-h2o.predict(dl,test.h2o)%>%as.data.frame()%>%pull(predict)
```

```
##   |                                                                              |
```

```r
dl.precsnprob<-h2o.predict(dl,test.h2o)%>%as.data.frame()%>%pull(p1)
```

```
##   |                                                                              |
```

```r
dl.reclprob<-h2o.predict(dl,test.h2o)%>%as.data.frame()%>%pull(p0)
```

```
##   |                                                                              |
```

```r
dl.summary<-data.frame(
      obs<-test_data$WFRI_R,
      pred<-dl.pred,
      N<-dl.reclprob,
      Y<-dl.precsnprob
  )

nb.summary<-nb.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....nb.pred", N="N....nb.reclpr

nb.auc<-roc(nb.summary$obs,Y)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

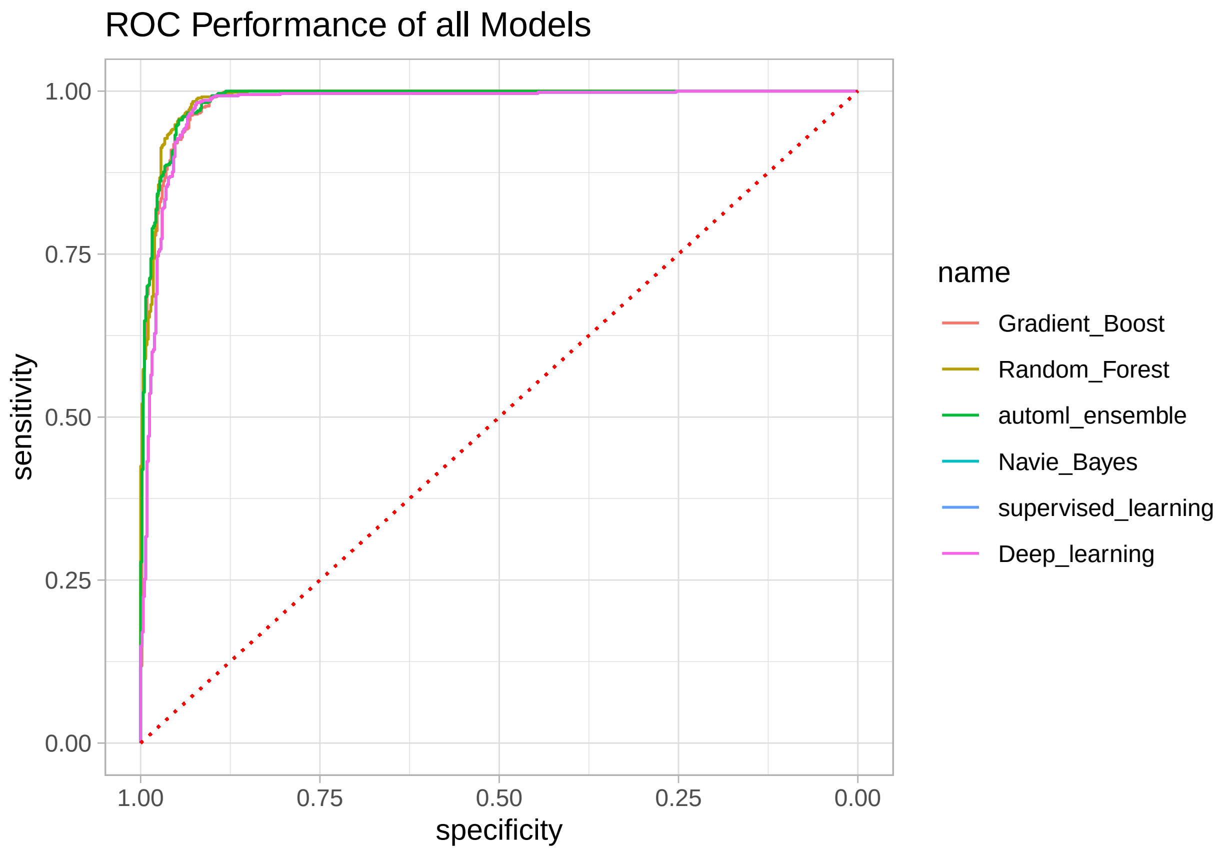
```r
svm.summary<-svm.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....svm.pred", N="N....svm.re

svm.auc<-roc(svm.summary$obs,Y)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
dl.summary<-dl.summary%>%rename(obs="obs....test_data.WFRI_R",pred="pred....dl.pred", N="N....dl.reclpre

dl.auc<-roc(dl.summary$obs,Y)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
rocs<-list(Gradient_Boost=gb.auc,Random_Forest=rf.auc,automl_ensemble=automl.auc,Navie_Bayes=nb.auc,supe
ggroc(rocs)+ggtitle("ROC Performance of all Models")+geom_segment(aes(x=1,y=0,xend=0,yend=1),linetype="
```



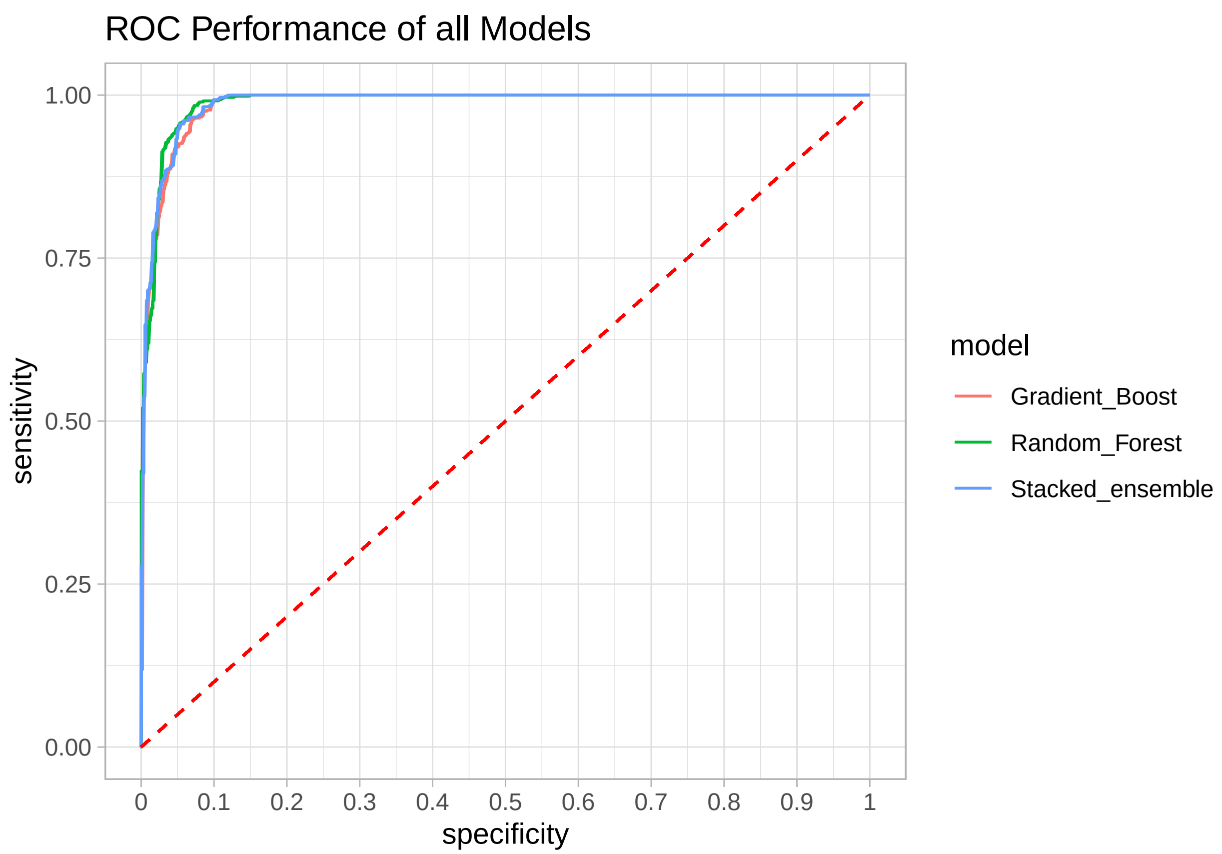

```r
#Updating test results for model performance for all remaining models
temp<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(nb.summary$pred)),y_true =as.numeric(as.character( nb
  t.mse<-MSE(as.numeric(as.character(nb.summary$pred)),as.numeric(as.character(nb.summary$obs))),
  t.RMSE<-RMSE(as.numeric(as.character(nb.summary$pred)),as.numeric(as.character(nb.summary$obs))),
  t.AUC<-nb.auc$auc,
  t.ClassError<-mean(nb.summary$pred!=nb.summary$obs)

)
temp<-temp%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.nb.summary.pred....",MSE="t.ms
,RMSE="t.RMSE....RMSE.as.numeric.as.character.nb.summary.pred....as.numeric.as.character.nb.summary.obs
,AUC="t.AUC....nb.auc.auc",classError="t.ClassError....mean.nb.summary.pred....nb.summary.obs.")

test.results[4,]<-c(R2=temp$R2,MSE=temp$MSE,RMSE=temp$RMSE,AUC=temp$AUC,classError=temp$classError)
```

```r
rownames(test.results)<-c("Random Forest","Gradient Boosting","automl- stack ensemble","Navie Bayes")

temp<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(svm.summary$pred)),y_true =as.numeric(as.character( sv
  t.mse<-MSE(as.numeric(as.character(svm.summary$pred)),as.numeric(as.character(svm.summary$obs))),
  t.RMSE<-RMSE(as.numeric(as.character(svm.summary$pred)),as.numeric(as.character(svm.summary$obs))),
  t.AUC<-svm.auc$auc,
  t.ClassError<-mean(svm.summary$pred!=svm.summary$obs)

)
temp<-temp%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.svm.summary.pred....",MSE="t.m
,RMSE="t.RMSE....RMSE.as.numeric.as.character.svm.summary.pred....as.numeric.as.character.svm.summary.ob
,AUC="t.AUC....svm.auc.auc",classError="t.ClassError....mean.svm.summary.pred....svm.summary.obs.")

test.results[5,]<-c(R2=temp$R2,MSE=temp$MSE,RMSE=temp$RMSE,AUC=temp$AUC,classError=temp$classError)
rownames(test.results)<-c("Random Forest","Gradient Boosting","automl- stack ensemble","Navie Bayes","Su

temp<-data.frame(
  t.R2<-R2_Score(y_pred = as.numeric(as.character(dl.summary$pred)),y_true =as.numeric(as.character( dl
  t.mse<-MSE(as.numeric(as.character(dl.summary$pred)),as.numeric(as.character(dl.summary$obs))),
  t.RMSE<-RMSE(as.numeric(as.character(dl.summary$pred)),as.numeric(as.character(dl.summary$obs))),
  t.AUC<-dl.auc$auc,
  t.ClassError<-mean(dl.summary$pred!=dl.summary$obs)

)
temp<-temp%>%rename(R2="t.R2....R2_Score.y_pred...as.numeric.as.character.dl.summary.pred....",MSE="t.ms
,RMSE="t.RMSE....RMSE.as.numeric.as.character.dl.summary.pred....as.numeric.as.character.dl.summary.obs
,AUC="t.AUC....dl.auc.auc",classError="t.ClassError....mean.dl.summary.pred....dl.summary.obs.")

test.results[6,]<-c(R2=temp$R2,MSE=temp$MSE,RMSE=temp$RMSE,AUC=temp$AUC,classError=temp$classError)
rownames(test.results)<-c("Random Forest","Gradient Boosting","automl- stack ensemble","Navie Bayes","Su
```

```r
# Assuming you have defined the rocs2 list with AUC values

rocs2 <- list(
  Gradient_Boost = roc(gb.summary$obs, gb.summary$Y),
  Random_Forest = roc(rf.summary$obs, rf.summary$Y),
  Stacked_ensemble = roc(automl.summary$obs, automl.summary$Y)

)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
# Convert the list of ROC objects to a data frame
roc_data <- do.call(rbind, lapply(names(rocs2), function(model) {
  data.frame(
    model = model,
    sensitivity = rocs2[[model]]$sensitivities,
    specificity = 1 - rocs2[[model]]$specificities
  )
}))

# Create a ggplot object with reversed x-axis layout
ggplot(roc_data, aes(x = specificity, y = sensitivity, color = model)) +
  geom_line() +  # Use default line weight
  ggtitle("ROC Performance of all Models") +
  geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1), linetype = "dashed", color = "red") +
  theme_light() +
  scale_x_continuous(
    breaks = seq(1, 0, by = -0.1),  # Reverse the breaks
    labels = seq(1, 0, by = -0.1),  # Reverse the labels
    limits = c(0, 1)  # Adjust the limits as needed
  )
```



Tables - Confusion Matrix

```r
names(nb_cm$byClass)
```

```r
names(nb_cm$overall)
```

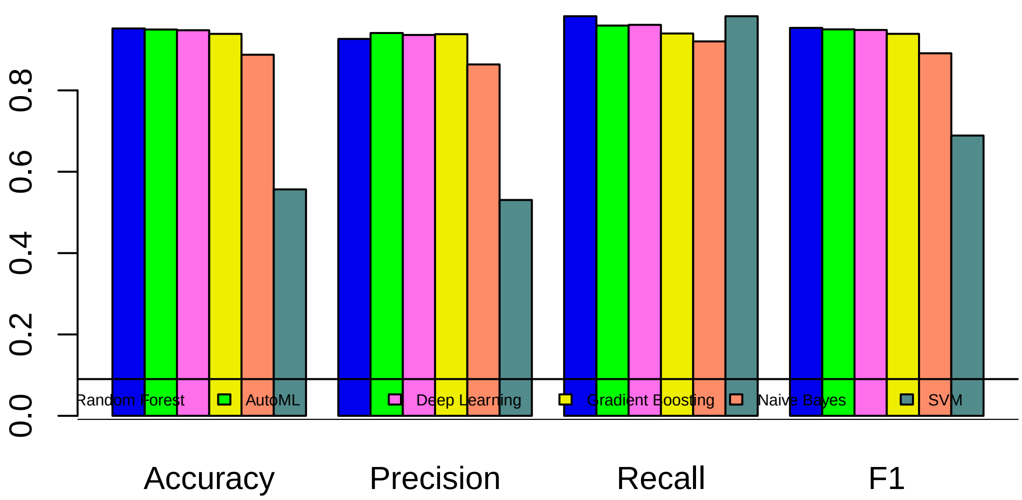
```
## [1] "Accuracy"       "Kappa"          "AccuracyLower"  "AccuracyUpper"
## [5] "AccuracyNull"   "AccuracyPValue" "McnemarPValue"
```

```r
metrics_drf <- c(rf.con$overall[1], rf.con$byClass[5], rf.con$byClass[6], rf.con$byClass[7])
metrics_aml <- c(aml_cm$overall[1], aml_cm$byClass[5], aml_cm$byClass[6], aml_cm$byClass[7])
metrics_dl <- c(dl_cm$overall[1], dl_cm$byClass[5], dl_cm$byClass[6], dl_cm$byClass[7])
metrics_gbm <- c(gb.con$overall[1], gb.con$byClass[5], gb.con$byClass[6], gb.con$byClass[7])
metrics_nb <- c(nb_cm$overall[1], nb_cm$byClass[5], nb_cm$byClass[6], nb_cm$byClass[7])
metrics_svm <- c(svm_cm$overall[1], svm_cm$byClass[5], svm_cm$byClass[6], svm_cm$byClass[7])

# Round the values to four decimals
metrics_drf <- round(metrics_drf, 4)
metrics_aml <- round(metrics_aml, 4)
metrics_dl <- round(metrics_dl, 4)
metrics_gbm <- round(metrics_gbm, 4)
metrics_nb <- round(metrics_nb, 4)
metrics_svm <- round(metrics_svm, 4)

tab_pref <- rbind(metrics_drf, metrics_aml, metrics_dl, metrics_gbm, metrics_nb, metrics_svm)
rownames(tab_pref) <- c("Distributed Random Forest",
                        "AutoML", "Deep Learning",
                        "Gradient Boosting Machine",
                        "Naive Bayes", "Support Vector Machine")


# Identify the column index of the accuracy values in your data frame
accuracy_column_index <- 1

# Order the rows of the data frame based on accuracy in descending order
tab_pref <- tab_pref[order(-tab_pref[, accuracy_column_index]), ]

# Create the data frame with the "Model" index name
(tab.pref <- data.frame( tab_pref))
```

```
##                           Accuracy Precision Recall     F1
## Deep Learning               0.9522    0.9265 0.9823 0.9536
## Distributed Random Forest   0.9496    0.9410 0.9593 0.9500
## AutoML                      0.9478    0.9362 0.9611 0.9485
## Gradient Boosting Machine   0.9389    0.9382 0.9398 0.9390
## Naive Bayes                 0.8876    0.8638 0.9204 0.8912
## Support Vector Machine      0.5566    0.5306 0.9823 0.6890
```

Visualization

```r
# Increase bottom margin
par(mar = c(6, 4, 6, 4))

# Your existing barplot code
bp <- barplot(tab_pref,
              col = c("#0000EE", "#00FF00", "#FF6EEB", "#EEEE00", "#FF8C69", "#528B8B"),
              beside = TRUE)

# Legend at the bottom
legend(x = "bottom",
       y = -0.4,
       legend = c("Random Forest", "AutoML", "Deep Learning", "Gradient Boosting", "Naive Bayes", "SVM"
       fill = c("#0000EE", "#00FF00", "#FF6EEB", "#EEEE00", "#FF8C69", "#528B8B"),
       cex = 0.53,
       ncol = 6)
```



Model Accuracy and Unique Hyperparameters

```r
# Create a data frame with the information
model_data <- data.frame(
  Model = c("Distributed Random Forest (DRF)",
            "AutoML (Automatic Machine Learning)",
            "Deep Learning",
```

```r
            "Gradient Boosting Machine (GBM)",
            "Naïve-Bayes (NB)",
            "Support Vector Machine (SVM)"
  ),
  Accuracy = round(c(rf.con$overall[1], aml_cm$overall[1], dl_cm$overall[1],
             gb.con$overall[1], nb_cm$overall[1], svm_cm$overall[1]), 4),
  Unique_Hyperparameters = c("balance_classes = FALSE",
                             "max_models = 5",
                             "Hidden = c(1), activation = 'Tanh', epochs = 1000",
                             "learn_rate = 0.1, ntrees = 1000",
                             "",
                             "Gamma = 0.01, rank_ratio = 0.1"
  )
)

# Order the values in descending order based on the "Model" column
model_df2 <- model_data[order(model_data$Model, decreasing = FALSE), ]

# Print the table
(model.df2 <- data.frame(model_df2))
```

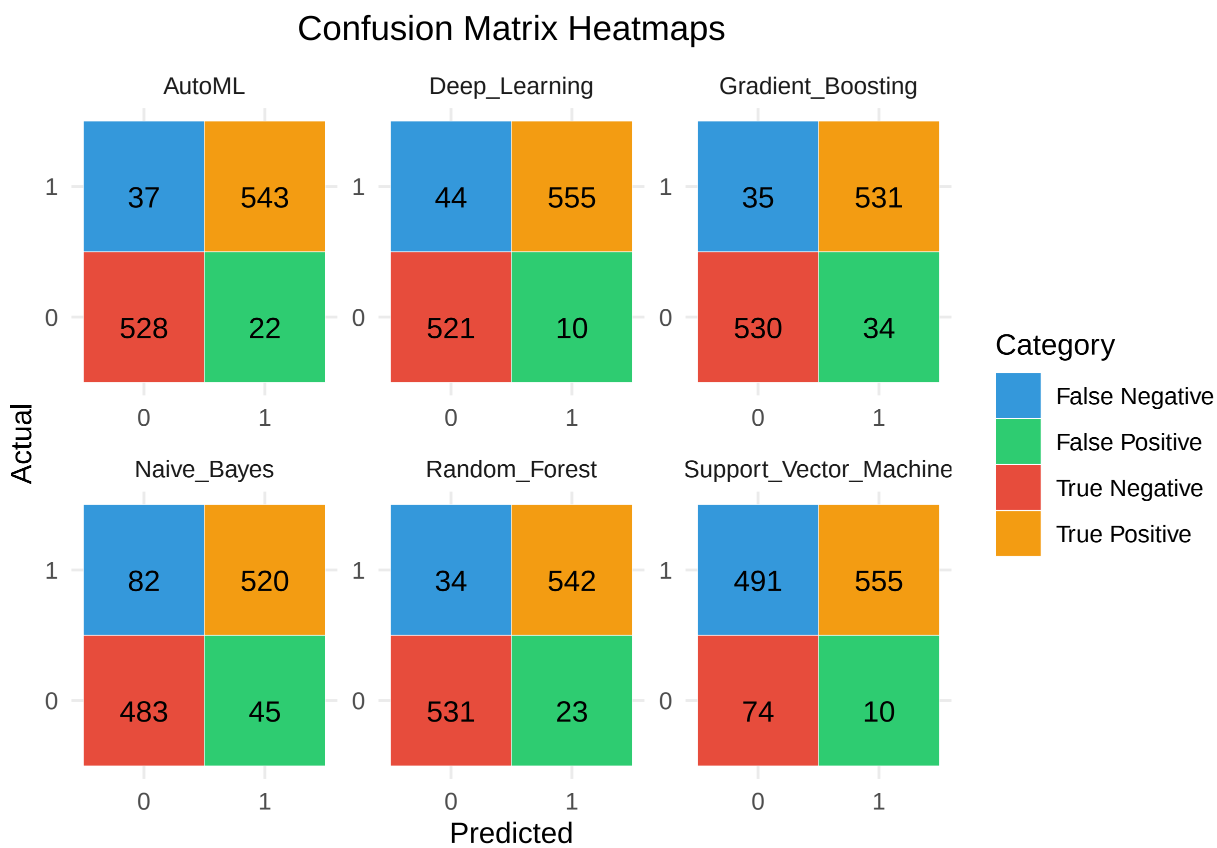
```
##                                  Model Accuracy
## 2 AutoML (Automatic Machine Learning)   0.9478
## 3                       Deep Learning   0.9522
## 1     Distributed Random Forest (DRF)   0.9496
## 4     Gradient Boosting Machine (GBM)   0.9389
## 5                    Naïve-Bayes (NB)   0.8876
## 6        Support Vector Machine (SVM)   0.5566
##                               Unique_Hyperparameters
## 2                                     max_models = 5
## 3 Hidden = c(1), activation = 'Tanh', epochs = 1000
## 1                            balance_classes = FALSE
## 4                    learn_rate = 0.1, ntrees = 1000
## 5
## 6                     Gamma = 0.01, rank_ratio = 0.1
```

Confusion Matrix Model Heatmaps

```r
#install.packages("dplyr")
library(dplyr)
```

```r
# Function to create a custom confusion matrix plot
create_confusion_matrix_plot <- function(cm, title) {
  # Convert confusion matrix to a data frame
  cm_data <- as.data.frame(cm$table)

  # Create a custom confusion matrix plot with different colors for each category
  ggplot(cm_data, aes(x = Reference, y = Prediction, fill = as.factor(Category), label = Freq)) +
    geom_tile(color = "white") +
    geom_text(vjust = 1, hjust = 0.5) +  # Center text
    scale_fill_manual(values = c("#3498db", "#2ecc71", "#e74c3c", "#f39c12"), name = "Category") +  # A
```

```r
    labs(title = title,
         x = "Predicted",
         y = "Actual") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))  # Center the title
}

# List of confusion matrices with categories
confusion_matrices <- list(
  Random_Forest = rf.con,
  AutoML = aml_cm,
  Deep_Learning = dl_cm,
  Gradient_Boosting = gb.con,
  Naive_Bayes = nb_cm,
  Support_Vector_Machine = svm_cm
)

# Add a 'Category' column to the data frames representing the four categories
all_cm_data <- do.call(rbind, lapply(names(confusion_matrices), function(name) {
  cm <- confusion_matrices[[name]]
  cm_data <- as.data.frame(cm$table)
  cm_data$Category <- factor(rep(c("True Negative", "False Negative", "False Positive", "True Positive"
  cm_data$Model <- name
  return(cm_data)
}))

# Create and display a facet_wrap confusion matrix plot with different colors for each category
ggplot(all_cm_data, aes(x = Reference, y = Prediction, fill = as.factor(Category), label = Freq)) +
  geom_tile(color = "white") +
  geom_text(vjust = 1, hjust = 0.5) +  # Center text
  scale_fill_manual(values = c("#3498db", "#2ecc71", "#e74c3c", "#f39c12"), name = "Category") +  # Ass
  labs(title = "Confusion Matrix Heatmaps",
       x = "Predicted",
       y = "Actual") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5)) +  # Center the title
  facet_wrap(~Model, scales = "free")  # Create a separate panel for each model
```



Confusion Matrix Tables for Models

```r
# Create an empty list to store dataframes
confusion_matrix_dataframes <- list()

for (name in names(confusion_matrices)) {
  cm <- confusion_matrices[[name]]

  # Accessing individual values from the confusion matrix
  TP <- cm$table[2, 2]  # second row and second column
  TN <- cm$table[1, 1]  # first row and first column
  FP <- cm$table[1, 2]  # first row and second column
  FN <- cm$table[2, 1]  # second row and first column

  # Create a dataframe for each model
  model_dataframe <- data.frame(
    Model = name,
    True_Positive = TP,
    True_Negative = TN,
    False_Positive = FP,
    False_Negative = FN
  )

  # Append the dataframe to the list
```

```
  confusion_matrix_dataframes[[name]] <- model_dataframe
}

# Combine all dataframes into a single dataframe without row names
all_confusion_matrices_df <- bind_rows(confusion_matrix_dataframes)

# Combine all dataframes into a single dataframe without row names
all_confusion_matrices_df <- bind_rows(confusion_matrix_dataframes)

# Arrange the dataframe in descending order based on the "True_Positive" column
all_confusion_matrices_df <- arrange(all_confusion_matrices_df, desc(True_Positive))

# Print the resulting dataframe
print(all_confusion_matrices_df)
```

```
##                    Model True_Positive True_Negative False_Positive
## 1          Deep_Learning           555           521             10
## 2 Support_Vector_Machine           555            74             10
## 3                 AutoML           543           528             22
## 4          Random_Forest           542           531             23
## 5      Gradient_Boosting           531           530             34
## 6            Naive_Bayes           520           483             45
##   False_Negative
## 1             44
## 2            491
## 3             37
## 4             34
## 5             35
## 6             82
```

True Positive Rate vs False Positive Rate

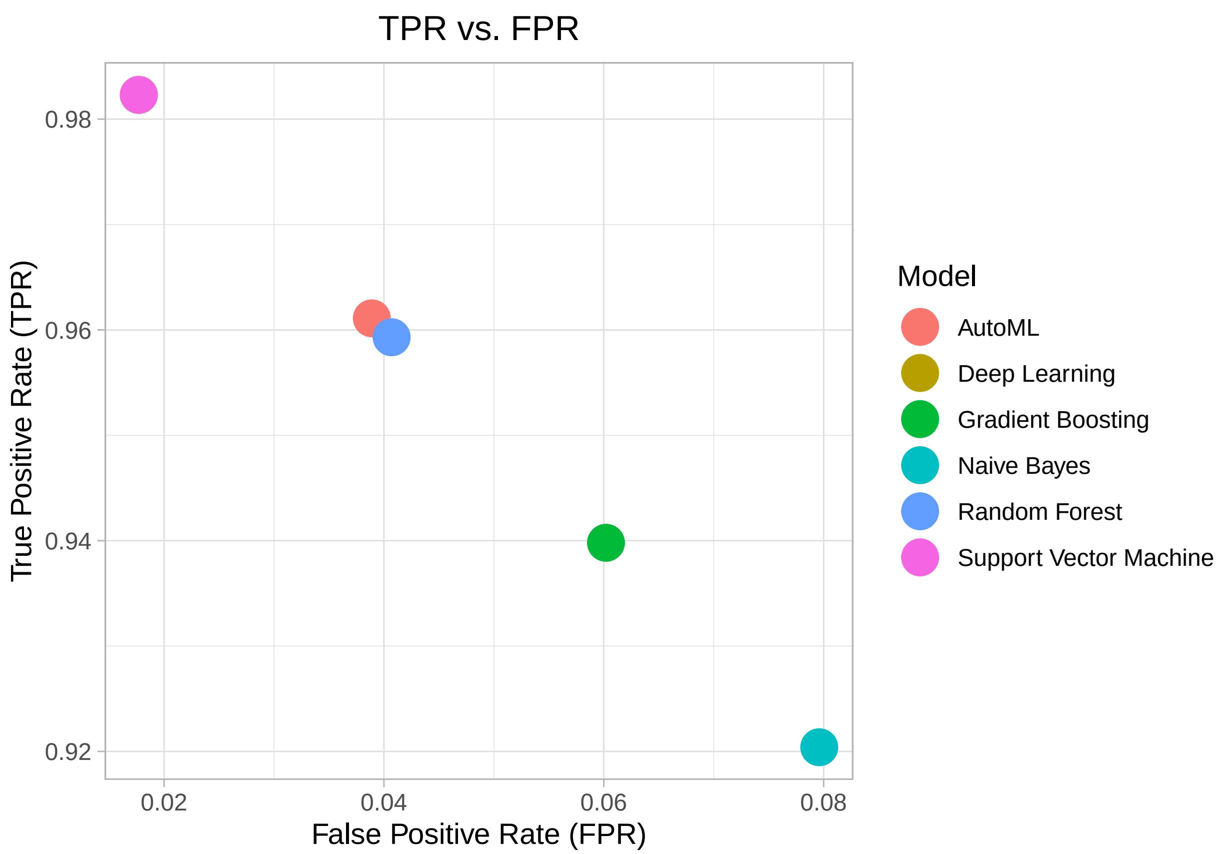
```
# Extract True Positive Rate or Sensitivity values
sens_drf <- round(rf.con$byClass[1], 4)
sens_aml <- round(aml_cm$byClass[1], 4)
sens_dl <- round(dl_cm$byClass[1], 4)
sens_gbm <- round(gb.con$byClass[1], 4)
sens_nb <- round(nb_cm$byClass[1], 4)
sens_svm <- round(svm_cm$byClass[1], 4)

# Calculate False Positive Rate (FPR)
fpr_drf <- round(1 - sens_drf, 4)
fpr_aml <- round(1 - sens_aml, 4)
fpr_dl <- round(1 - sens_dl, 4)
fpr_gbm <- round(1 - sens_gbm, 4)
fpr_nb <- round(1 - sens_nb, 4)
fpr_svm <- round(1 - sens_svm, 4)

# Create a data frame to store the results
results <- data.frame(
  Model = c("Random Forest", "AutoML", "Deep Learning", "Gradient Boosting", "Naive Bayes", "Support Ve
```

```r
  TPR = c(sens_drf, sens_aml, sens_dl, sens_gbm, sens_nb, sens_svm),
  FPR = c(fpr_drf, fpr_aml, fpr_dl, fpr_gbm, fpr_nb, fpr_svm)
)

# Arrange in descending order
results <- results[order(-results$TPR), ]

# Multiply values by 100
#results$TPR <- results$TPR * 100
#results$FPR <- results$FPR * 100

# Print the results
print(results)
```

```
##                    Model    TPR    FPR
## 3          Deep Learning 0.9823 0.0177
## 6 Support Vector Machine 0.9823 0.0177
## 2                 AutoML 0.9611 0.0389
## 1          Random Forest 0.9593 0.0407
## 4      Gradient Boosting 0.9398 0.0602
## 5            Naive Bayes 0.9204 0.0796
```

Visualization of TPR vs FPR

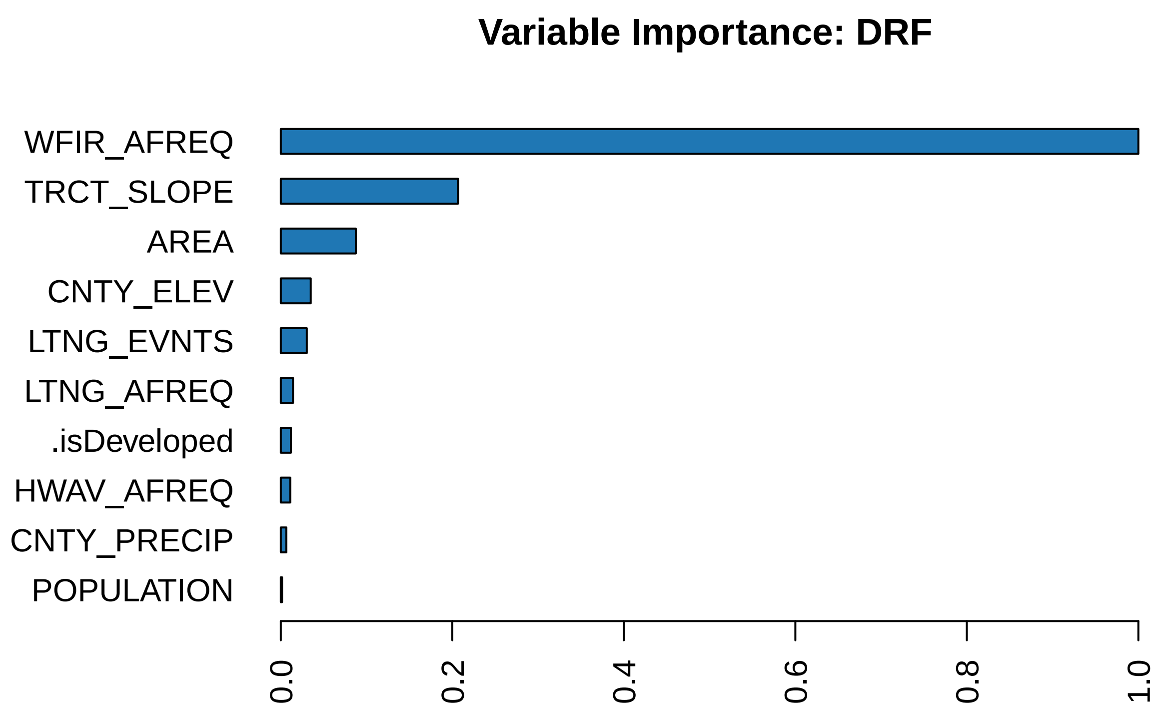
```r
# Arrange in descending order by TPR
results <- results[order(-results$TPR), ]

# Create a scatter plot for each model
ggplot(results, aes(x = FPR, y = TPR, group = Model, color = Model)) +
  geom_point(size = 6) +
  labs(title = "TPR vs. FPR",
       x = "False Positive Rate (FPR)",
       y = "True Positive Rate (TPR)") +
  theme_light() +
  theme(plot.title = element_text(hjust = 0.5))
```



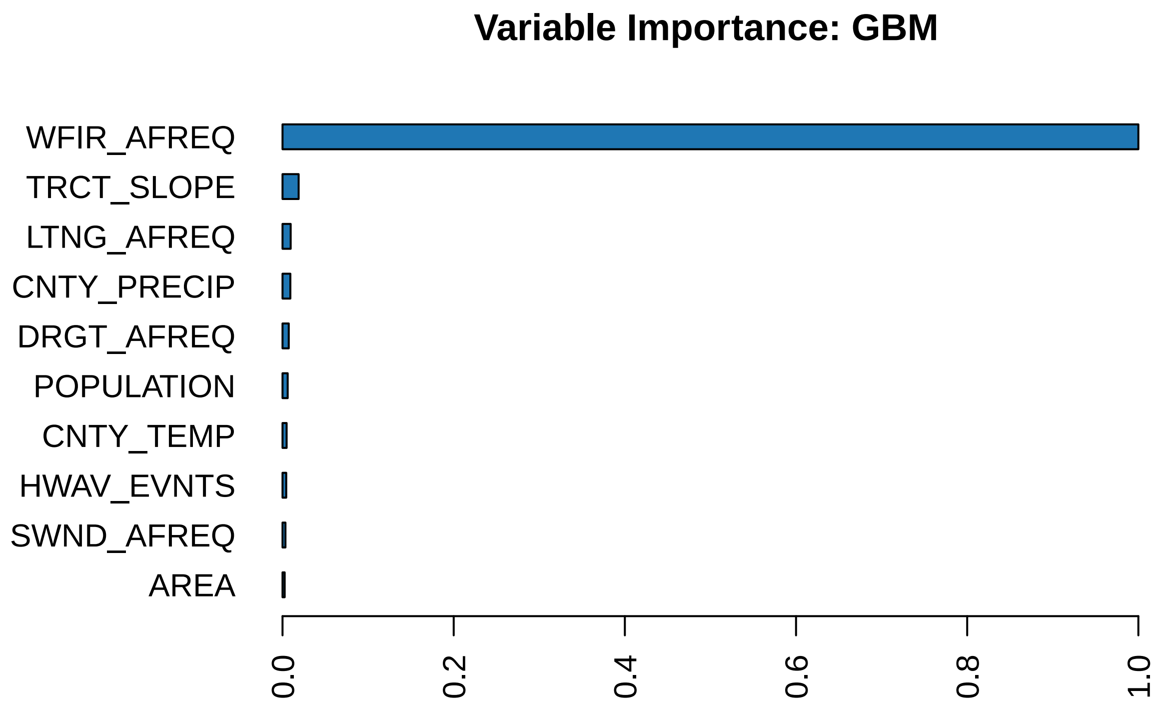
##Variance Importance Plot

These model doesn't have variable importances: `aml`, `pros_nb`, and `svm_model`

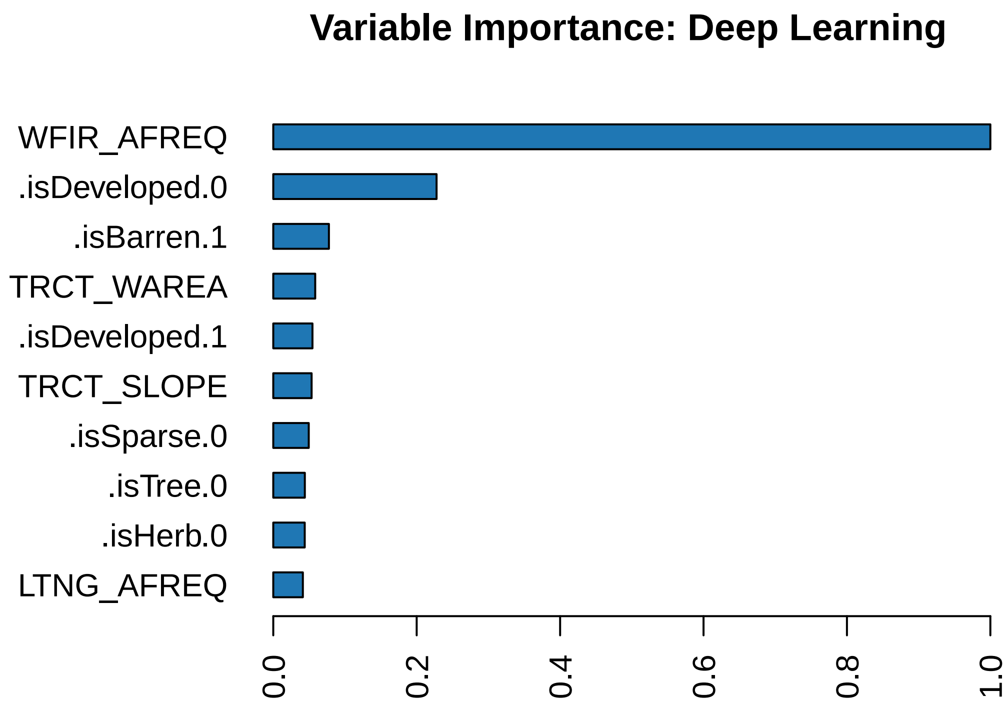
```
h2o.varimp_plot(rf.v2)
```




```r
h2o.varimp_plot(gb_v2)
```




```
h2o.varimp_plot(dl)
```




```
h2o.shutdown()
```