KNN Classification Model

KNN, Mice (PMM), Mean, Median, Piecewise, and RandomForest Imputed Tables

Lewis Rincon Castano

03-29-2023

Source: https://www.r-bloggers.com/2021/04/knn-algorithm-machine-learning/ SMOTE documentation: https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE

Model Summary: We obtained the following model results:

- Imputation tables accuracy: autoTune_VIM_KNN (76,73%), Mice-PMM method: (73,38%), median (74,94%), mean (74,83%), RandomForest (69,13%) and Piecewise (N/A).
- ROC values: autoTune_VIM_KNN (87,52%), Mice-PMM method: (85,87%), median (82,39%), mean (84,01%), RandomForest (82,62%) and Piecewise (N/A).
  - Piecewise had some missing values, and KNN classification model do not compute datasets with NaN values.
- Out **best model** came from the autoTune_VIM_KNN table due to its higher ROC and accuracy rate.

Notes:

- Other models were created by my group members, but this model was create by me.
- We got permission to publish this file from our project sponsor. However, we removed any identifiable information from it, such as company's name, employees, group members, and the original data set file.
- Previously, we used our sponsor's file and identified some missing values. Therefore, we did the following imputation tables: autoTune_VIM_KNN, Mice (PMM), Mean, Median, Piecewise, and RandomForest.
- Some of the comments on this file are extracted from the websites above. However, we edit some of comments according to this model.
- The list of variables to conduct our analysis came from the stepwise multiple linear regression. This process was completed on SPSS with the original sponsor file that is not imputed. Lewis tried to conduct this same analysis at RStudio, but it was impossible to obtain an outcome due to the missing variables. Each iteration of the variables and its accuracy was recorded on the autoTuneVIM KNN file section because it is the model with the higher accuracy rate.

```r
#Load Libraries
library(caret) # v6.0-93
library(pROC) # v1.18.0
library(mlbench) # v2.1-3
library(ggplot2) # v3.4.0
library(lattice) # v0.20-44
library(readxl) # v1.4.1
library(readr) # v1.4.1
library(dplyr) # v1.0.10
library(data.table) # v1.14.6
library(DMwR) # v0.4.1
library(devtools) # v2.4.4
library(themis) # v1.0.0
library(MLmetrics) # v1.1.1, to make the multiClassSummary metric work
```

Import Excel file: KNN imputed table

```r
data <- read_excel("C:/Users/lewis/Desktop/completeData_autotuneVIM_KNN_method_v2.xlsx")

# Create factors for the following columns
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# First, we need to replace the zeros and ones from this STATUS column. Previously
# we were not able to find the order of importance after creating a fit variable.
# Therefore, we replace the values of zero and one to make run the varImp(Fit) function.
setDT(data)[STATUS == 1, STATUS := 2]
setDT(data)[STATUS == 0, STATUS := 1]

# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(1,2),
                     labels = c("EMPLOYEE",
                                "TERMINATED"
                    ))

#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))
```

```r
# Data Partition: Let's create independent samples and create training and test
# dataset, 60% and 40% respectively, for prediction.

set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.6, 0.4))
training <- data[ind == 1,]
test <- data[ind == 2,]
str(training)
```

```
## Classes 'data.table' and 'data.frame':   531 obs. of  13 variables:
##  $ STATUS                                : Factor w/ 2 levels "EMPLOYEE","TERMINATED": 1 1 1 1 1
##  $ CAREER_RETURNS                        : num  206 671 985 950 962 409 826 7 671 769 ...
##  $ CAREER_HOURS_WORKED                   : num  0 12156 14153 11408 10476 ...
##  $ SIX_MONTH_DAILY_RETURN_AVERAGE        : num  6.6 10.5 12.7 12.1 12.4 ...
##  $ PEICES_DELIVERED_90_DAYS              : num  1054 22324 111226 28826 18658 ...
##  $ CAREER_6_MONTH_DELIVERED_AVERAGE      : num  152 500 1005 413 464 ...
##  $ WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE: num  0.667 0.667 0.667 0.667 0.667 ...
##  $ AGE                                   : num  55 41 54 60 43 31 30 27 56 60 ...
##  $ DAILY_DELIVERY_PAST_MONTH_AVERAGE     : num  533 5930 20872 7541 6778 ...
##  $ CAREER_DAILY_RETURN_AVERAGE           : num  14.7 11.6 13.8 14.1 12.8 ...
##  $ OVERNIGHT_SHIFTS                      : num  1 44 23 48 73 364 51 777 33 93 ...
##  $ JOB_DESCRIPTION                       : int  1 2 3 2 3 2 2 1 2 3 ...
##  $ SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE     : num  3.54 4.11 4.96 3.82 4.32 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# KNN Model
# Before making knn model we need to create train control. Let's create train
# control based on below code.
trControl <- trainControl(method = "repeatedcv", # resampling method
                          sampling = "smote", #  balance data
                          number = 10, # Either the number of folds or number of
                                       # resampling iterations
                          repeats = 5, # an indicator of how much of the hold-out predictions for each
                          # resample should be saved.
                          classProbs = TRUE, # a logical; should class probabilities
                          # be computed for classification models (along
                          # with predicted values) in each resample?
                          summaryFunction = twoClassSummary, #metrics that rely on
                          # class probabilities
                          savePredictions = TRUE, #an indicator of how much of
                          # the hold-out predictions for each resample should be saved.
                          allowParallel = FALSE # an indicator of how much of the
                          # hold-out predictions for each resample should be saved.

                            )
```

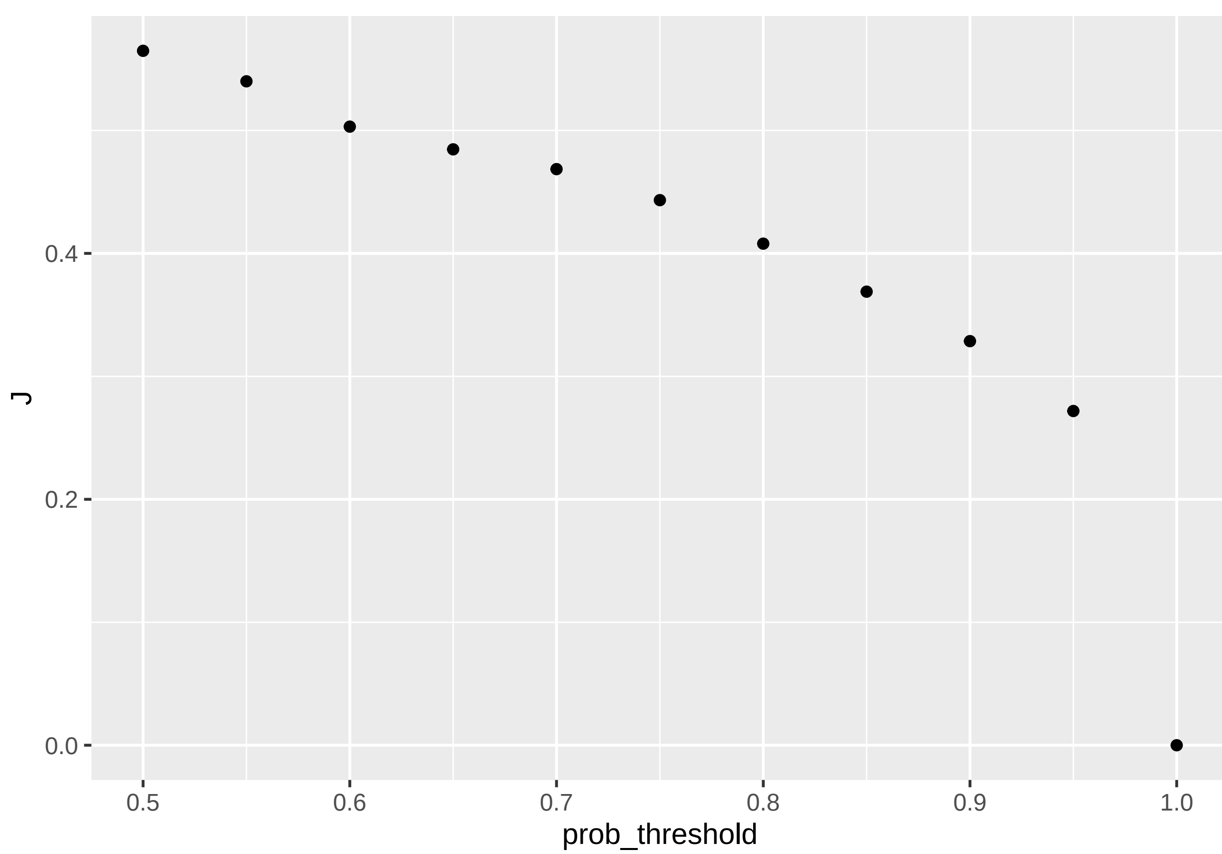
```r
# trainControl is from caret package, number of iteration is 10 times.
# and repeat the cross validation is 3 times.
set.seed(222)
fit <- train(STATUS ~ .,
             data = training,
             method = 'knn', # For classification and regression with tuning parameters
             tuneLength = 50, # # An integer denoting the amount of granularity in
```

```r
              # the tuning parameter grid
              trControl = trControl, # A list of values that define how this function acts.
              preProc = c("center", "scale"), #  string vector that defines a pre-processing of the
              # predictor data.
              metric = "ROC", # A list of values that define how this function acts.
              tuneGrid = expand.grid(k = 1:101) # A data frame with possible tuning values
              )
```

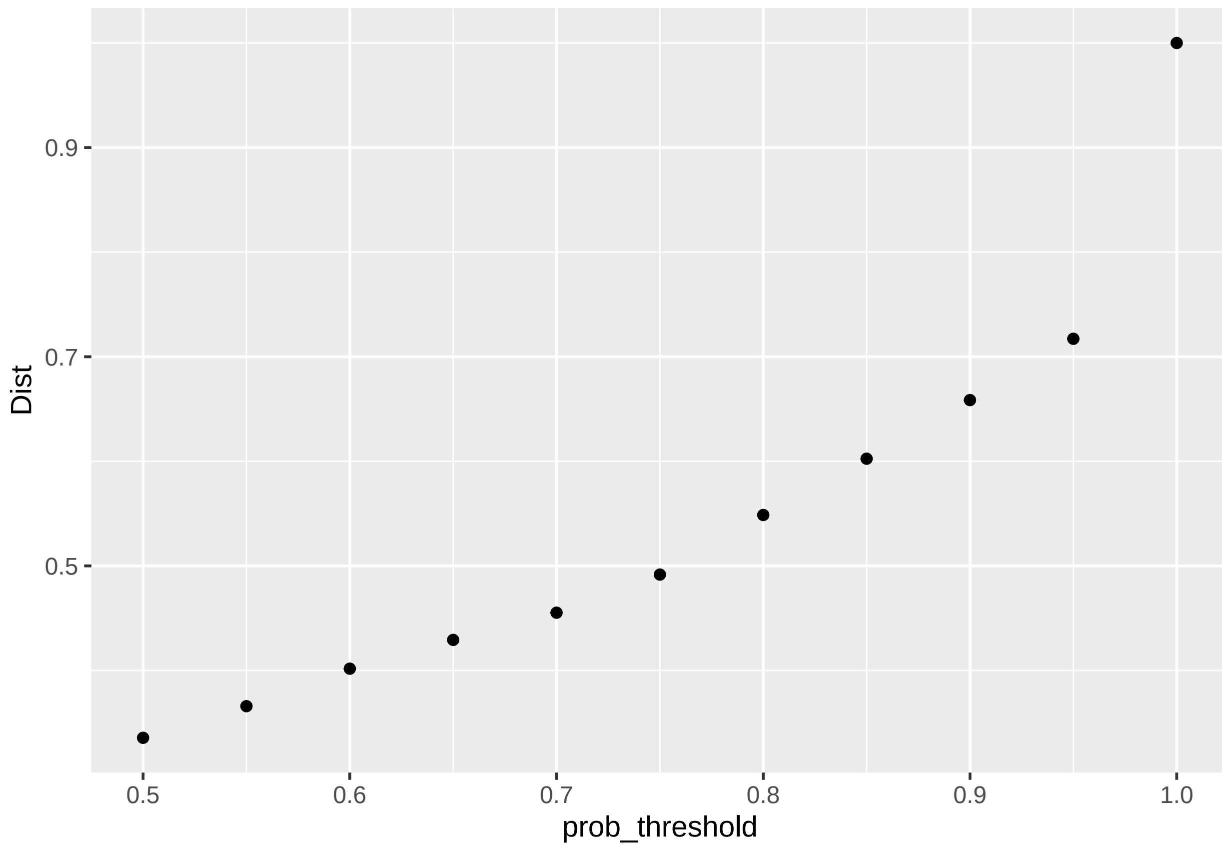
```r
# The following chunk of code was not part of the R-blogger template.
# Threshold definition:
#     A data frame with columns for each of the tuning parameters from the model
#     along with an additional column called prob_threshold for the probability
#     threshold. There are also columns for summary statistics averaged over
#     resamples with column names corresponding to the input argument statistics."

# This will generate a thresfold metrics table from 50% to 100% in increases of 5%.
resample_stats <- thresholder(fit,
                              threshold = seq(.5, 1, by = 0.05), #
                              final = TRUE)
```

```
## Warning in .fun(piece, ...): The following columns have missing values (NA), which have been removed
```

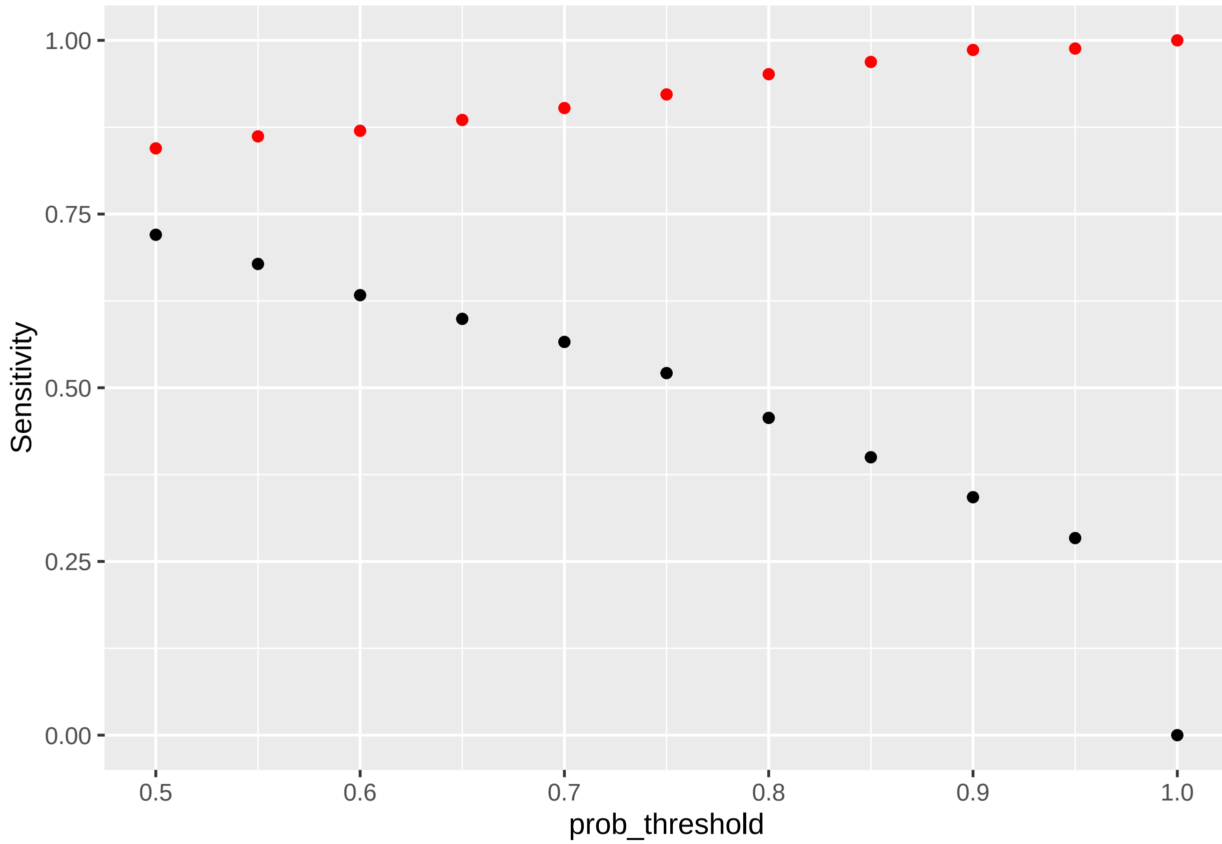
```r
# Plots related to the above table
ggplot(resample_stats, aes(x = prob_threshold, y = J)) +
geom_point()
```



```r
ggplot(resample_stats, aes(x = prob_threshold, y = Dist)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Sensitivity)) +
geom_point() + geom_point(aes(y = Specificity), col = "red")
```



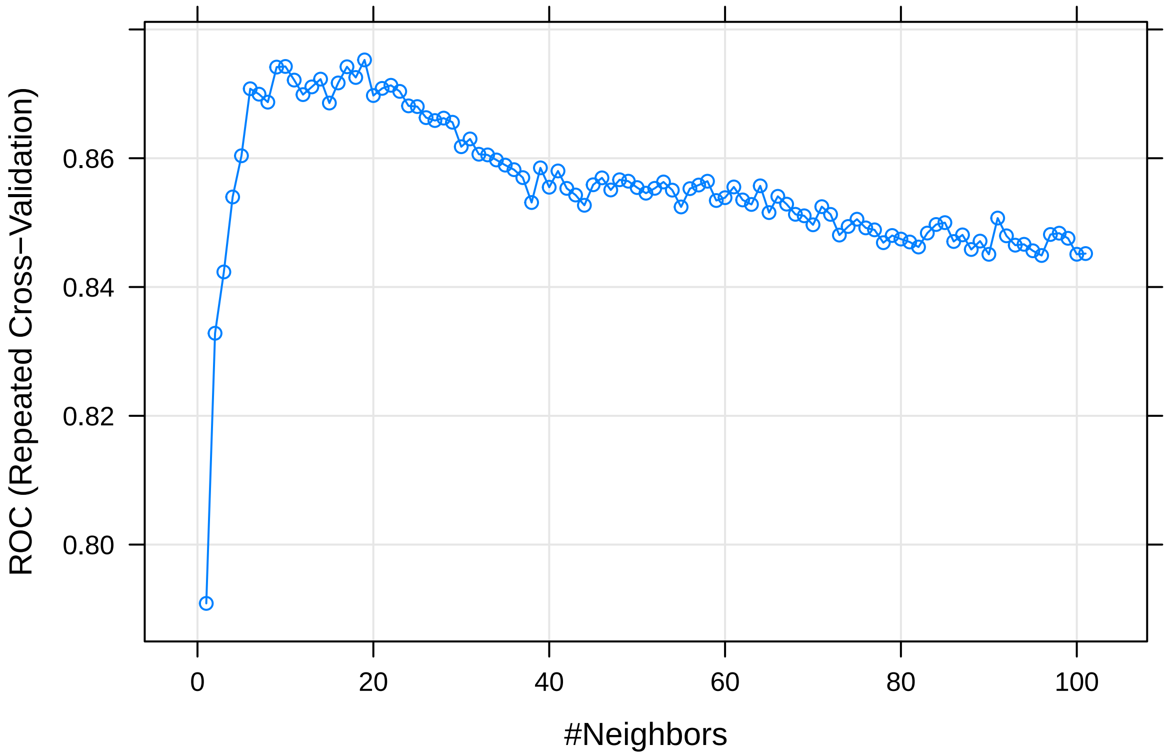

```r
# Model Performance
fit
```

```
## k-Nearest Neighbors
##
## 531 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 477, 478, 479, 478, 478, 478, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k    ROC        Sens       Spec
##     1  0.7908668  0.8962791  0.6854545
##     2  0.8328176  0.8640310  0.7400000
##     3  0.8423391  0.8528571  0.7585455
##     4  0.8539836  0.8295349  0.7623636
##     5  0.8603794  0.8295349  0.7707273
##     6  0.8707914  0.8084275  0.7838182
##     7  0.8699462  0.8024252  0.7789091
##     8  0.8687067  0.7939978  0.7860000
##     9  0.8741503  0.7943965  0.7941818
```

```
## 10 0.8742500 0.7916501 0.7940000
## 11 0.8721341 0.7729790 0.7878182
## 12 0.8698785 0.7739313 0.7916364
## 13 0.8710755 0.7584718 0.7978182
## 14 0.8722830 0.7552049 0.8074545
## 15 0.8685684 0.7430122 0.8154545
## 16 0.8716878 0.7332890 0.8234545
## 17 0.8742012 0.7286047 0.8407273
## 18 0.8725372 0.7309967 0.8229091
## 19 0.8752679 0.7206645 0.8427273
## 20 0.8697321 0.7183167 0.8367273
## 21 0.8708456 0.7108306 0.8421818
## 22 0.8713239 0.7066113 0.8501818
## 23 0.8703786 0.6972536 0.8501818
## 24 0.8681316 0.6972979 0.8501818
## 25 0.8680194 0.6878959 0.8481818
## 26 0.8663009 0.6902879 0.8543636
## 27 0.8658495 0.6856478 0.8521818
## 28 0.8662289 0.6860797 0.8501818
## 29 0.8655874 0.6758583 0.8618182
## 30 0.8617847 0.6790919 0.8580000
## 31 0.8629941 0.6772536 0.8658182
## 32 0.8606198 0.6786157 0.8578182
## 33 0.8605116 0.6767663 0.8561818
## 34 0.8597294 0.6725581 0.8660000
## 35 0.8589273 0.6720709 0.8660000
## 36 0.8582202 0.6683389 0.8621818
## 37 0.8569861 0.6702215 0.8640000
## 38 0.8531193 0.6683721 0.8601818
## 39 0.8585051 0.6660133 0.8700000
## 40 0.8554946 0.6604097 0.8738182
## 41 0.8580243 0.6660465 0.8621818
## 42 0.8553187 0.6627685 0.8758182
## 43 0.8542860 0.6594684 0.8678182
## 44 0.8526922 0.6567110 0.8658182
## 45 0.8558773 0.6562458 0.8680000
## 46 0.8569493 0.6590255 0.8700000
## 47 0.8550669 0.6562126 0.8756364
## 48 0.8566535 0.6520377 0.8794545
## 49 0.8564324 0.6557918 0.8816364
## 50 0.8554405 0.6483389 0.8816364
## 51 0.8545650 0.6464341 0.8814545
## 52 0.8553234 0.6436102 0.8858182
## 53 0.8563106 0.6455150 0.8876364
## 54 0.8550437 0.6417165 0.8856364
## 55 0.8524337 0.6385050 0.8816364
## 56 0.8552739 0.6361019 0.8892727
## 57 0.8558366 0.6356811 0.8854545
## 58 0.8564279 0.6404208 0.8874545
## 59 0.8534312 0.6329014 0.8872727
## 60 0.8538696 0.6249502 0.8874545
## 61 0.8555370 0.6268106 0.8852727
## 62 0.8535238 0.6225692 0.8912727
## 63 0.8528231 0.6253821 0.8912727
```

```
##    64  0.8557099  0.6235437  0.8914545
##    65  0.8515605  0.6249280  0.8894545
##    66  0.8540813  0.6188704  0.8912727
##    67  0.8528692  0.6240089  0.8912727
##    68  0.8512904  0.6211739  0.8872727
##    69  0.8510590  0.6127464  0.8892727
##    70  0.8496590  0.6203101  0.8894545
##    71  0.8524797  0.6141750  0.8894545
##    72  0.8512775  0.6155703  0.8892727
##    73  0.8480657  0.6146622  0.8912727
##    74  0.8494013  0.6099557  0.8892727
##    75  0.8505256  0.6066224  0.8894545
##    76  0.8491784  0.6028904  0.8912727
##    77  0.8488773  0.6048726  0.8912727
##    78  0.8468833  0.6108859  0.8894545
##    79  0.8479977  0.6076190  0.8912727
##    80  0.8474425  0.6057475  0.8854545
##    81  0.8469906  0.6006423  0.8912727
##    82  0.8462077  0.6001329  0.8894545
##    83  0.8483747  0.5959468  0.8894545
##    84  0.8497068  0.6001107  0.8876364
##    85  0.8499956  0.5921152  0.8930909
##    86  0.8470679  0.5922370  0.8874545
##    87  0.8480934  0.5856146  0.8892727
##    88  0.8458268  0.5894020  0.8892727
##    89  0.8471278  0.5912182  0.8854545
##    90  0.8450758  0.5893798  0.8854545
##    91  0.8507034  0.5833001  0.8912727
##    92  0.8479475  0.5883942  0.8874545
##    93  0.8464925  0.5833001  0.8872727
##    94  0.8466294  0.5865449  0.8892727
##    95  0.8456294  0.5795238  0.8832727
##    96  0.8448962  0.5823588  0.8874545
##    97  0.8481566  0.5846844  0.8872727
##    98  0.8483603  0.5837763  0.8874545
##    99  0.8475643  0.5739646  0.8930909
##   100  0.8450839  0.5734662  0.8930909
##   101  0.8451937  0.5720377  0.8892727
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 19.
```

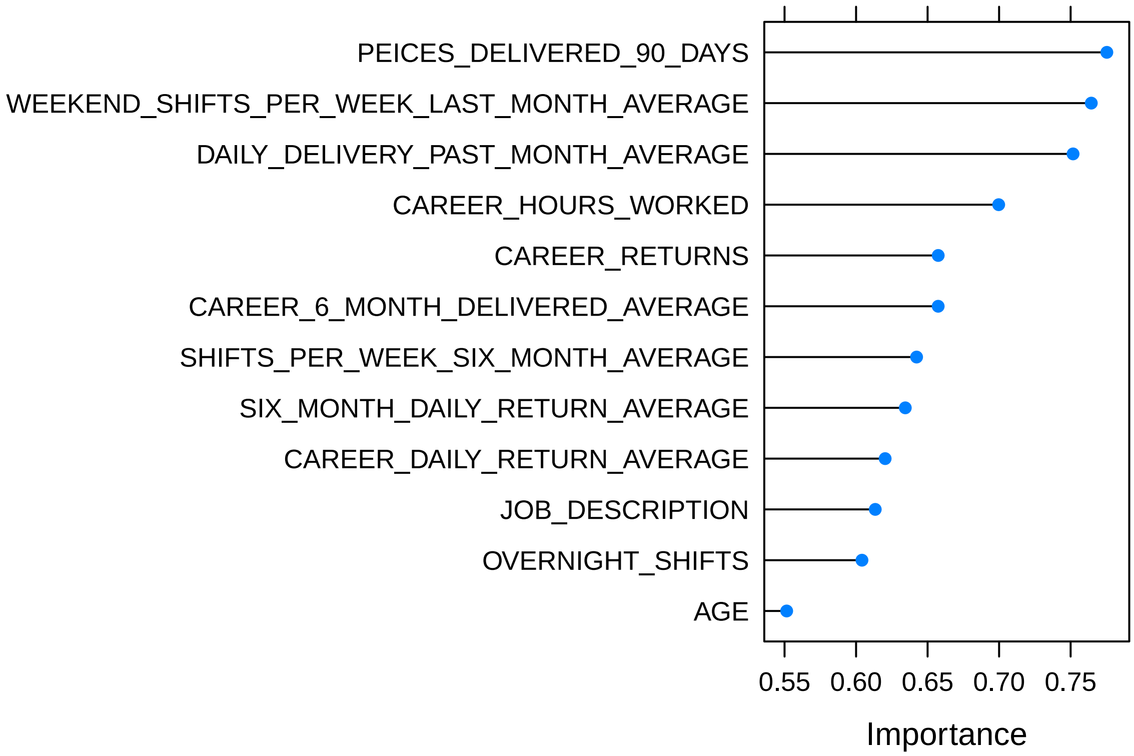
```
plot(fit)
```



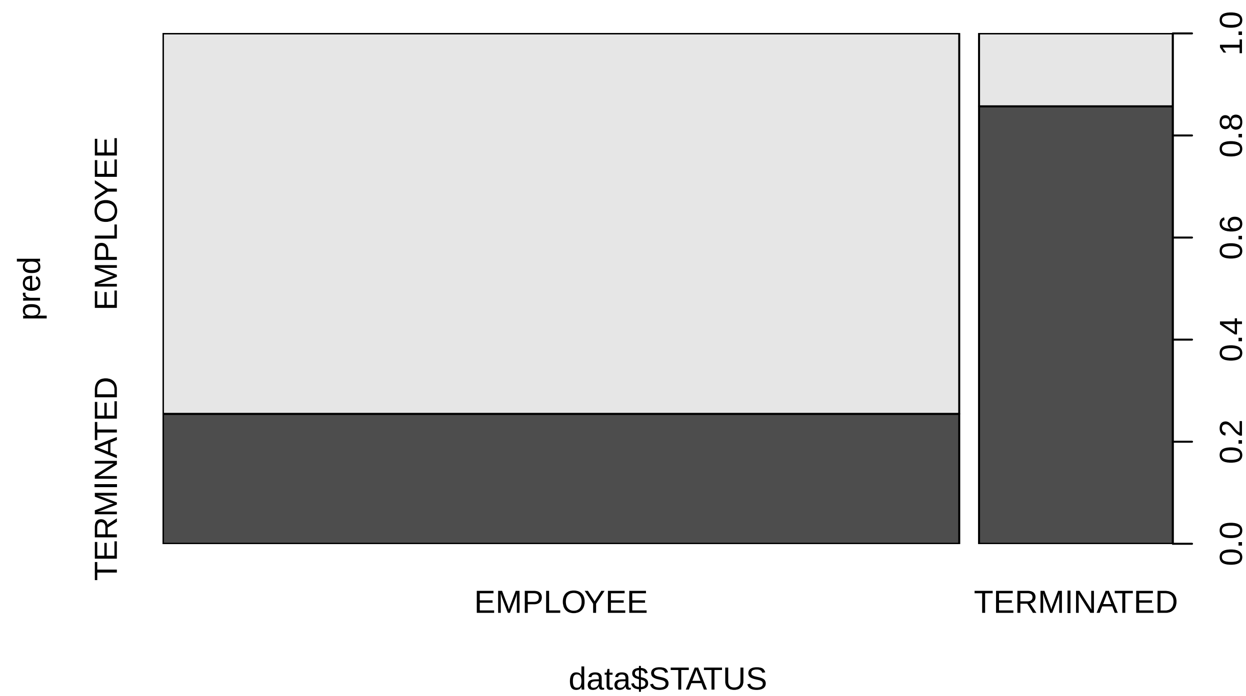

```r
# Loess r-squared variable importance
varImp(fit)
```

```
## ROC curve variable importance
##
##                                           Importance
## PEICES_DELIVERED_90_DAYS                      100.00
## WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE     95.14
## DAILY_DELIVERY_PAST_MONTH_AVERAGE              89.45
## CAREER_HOURS_WORKED                            66.22
## CAREER_RETURNS                                 47.33
## CAREER_6_MONTH_DELIVERED_AVERAGE               47.32
## SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE              40.58
## SIX_MONTH_DAILY_RETURN_AVERAGE                 37.05
## CAREER_DAILY_RETURN_AVERAGE                    30.75
## JOB_DESCRIPTION                                27.67
## OVERNIGHT_SHIFTS                               23.52
## AGE                                             0.00
```

```r
# variable importance visualization
importance <- varImp(fit, scale = FALSE)
plot(importance)
```




```r
# Create plot
pred <- predict(fit, newdata = data)
plot(pred ~ data$STATUS)
```




```
confusionMatrix(pred, data$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        536         25
##   TERMINATED      183        150
##
##                Accuracy : 0.7673
##                  95% CI : (0.7382, 0.7947)
##     No Information Rate : 0.8043
##     P-Value [Acc > NIR] : 0.9972
##
##                   Kappa : 0.4492
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7455
##             Specificity : 0.8571
##          Pos Pred Value : 0.9554
##          Neg Pred Value : 0.4505
##              Prevalence : 0.8043
##          Detection Rate : 0.5996
##    Detection Prevalence : 0.6275
```

```
##       Balanced Accuracy : 0.8013
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Mice (PMM) imputed table

```r
data <- read_excel("C:/Users/lewis/Downloads/completeData_mice_method.xlsx")
# Select and filter table with key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# First, we need to replace the zeros and ones from this STATUS column. Previously
# we were not able to find the order of importance after creating a fit variable.
# Therefore, we replace the values of zero and one to make run the varImp(Fit) function.
setDT(data)[STATUS == 1, STATUS := 2]
setDT(data)[STATUS == 0, STATUS := 1]

# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(1,2),
                     labels = c("EMPLOYEE",
                                "TERMINATED"
                    ))
#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))
```

```r
# Data Partition: Let's create independent samples and create training and test
# dataset, 60% and 40% respectively, for prediction.

set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.6, 0.4))
training <- data[ind == 1,]
test <- data[ind == 2,]
str(training)
```

```
## Classes 'data.table' and 'data.frame':   531 obs. of  13 variables:
##  $ STATUS                                    : Factor w/ 2 levels "EMPLOYEE","TERMINATED": 1 1 1 1 1
##  $ CAREER_RETURNS                            : num  206 671 985 950 962 409 826 7 671 769 ...
```

```
##  $ CAREER_HOURS_WORKED                      : num  0 12156 14153 11408 10476 ...
##  $ SIX_MONTH_DAILY_RETURN_AVERAGE           : num  6.6 10.5 12.7 12.1 12.4 ...
##  $ PEICES_DELIVERED_90_DAYS                 : num  1054 22324 111226 28826 18658 ...
##  $ CAREER_6_MONTH_DELIVERED_AVERAGE         : num  152 500 1005 413 464 ...
##  $ WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE: num  0 0.667 0.667 0.667 0.667 ...
##  $ AGE                                      : num  55 41 54 60 43 31 30 27 56 60 ...
##  $ DAILY_DELIVERY_PAST_MONTH_AVERAGE        : num  533 5930 20872 7541 6778 ...
##  $ CAREER_DAILY_RETURN_AVERAGE              : num  14.7 11.6 13.8 14.1 12.8 ...
##  $ OVERNIGHT_SHIFTS                         : num  1 44 23 48 73 364 51 777 33 93 ...
##  $ JOB_DESCRIPTION                          : int  1 2 3 2 3 2 2 1 2 3 ...
##  $ SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE        : num  4.26 4.11 4.96 3.82 4.32 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Before making  model we need to create train control. Let's create train
# control based on below code.
trControl <- trainControl(method = "repeatedcv", # resampling method
                         sampling = "smote", #  balance data
                         number = 10, # Either the number of folds or number of
                                     # resampling iterations
                         repeats = 5, # an indicator of how much of the hold-out predictions for each
                         # resample should be saved.
                         classProbs = TRUE, # a logical; should class probabilities
                         # be computed for classification models (along
                         # with predicted values) in each resample?
                         summaryFunction = twoClassSummary, #metrics that rely on
                         # class probabilities
                         savePredictions = TRUE, #an indicator of how much of
                         # the hold-out predictions for each resample should be saved.
                         allowParallel = FALSE # an indicator of how much of the
                         # hold-out predictions for each resample should be saved.

                           )
```

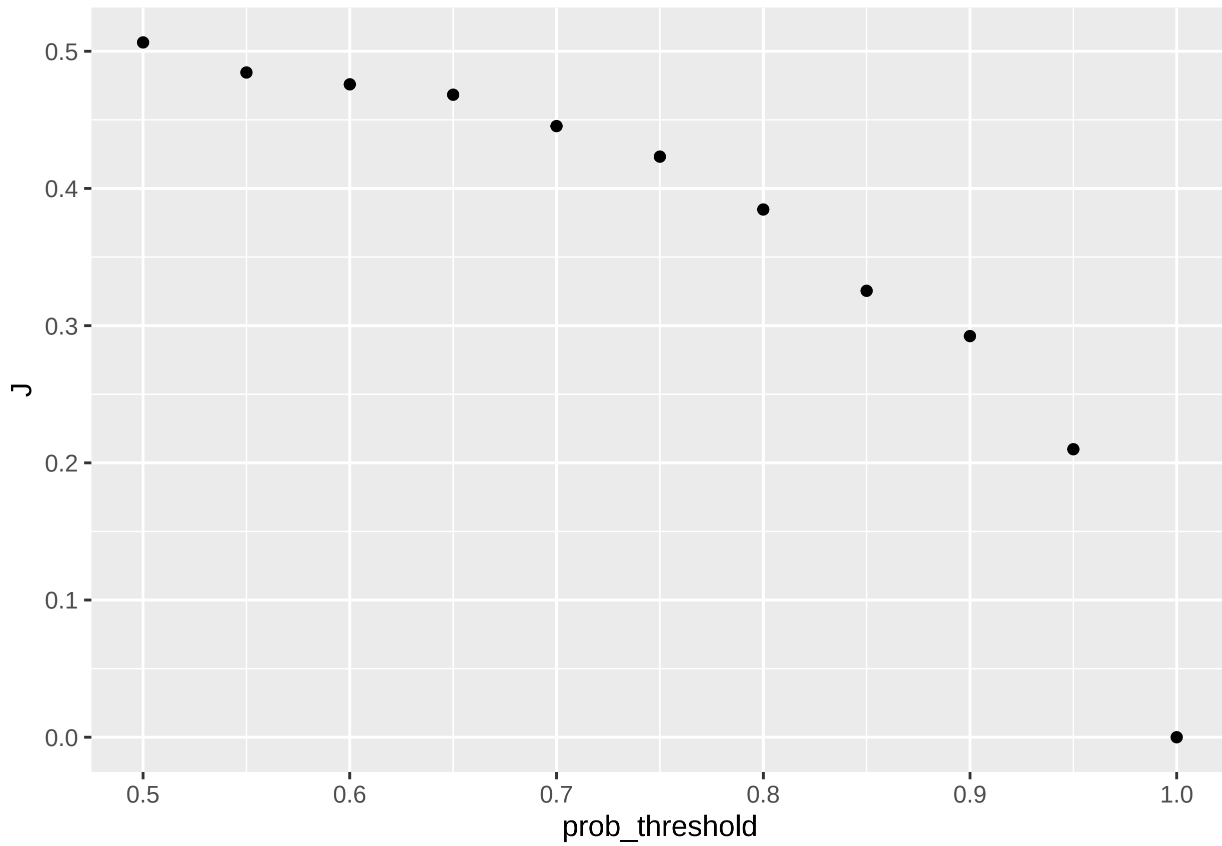
```r
# trainControl is from caret package, number of iteration is 10 times.
# and repeat the cross validation is 3 times.
set.seed(222)
fit <- train(STATUS ~ .,
             data = training,
             method = 'knn', # For classification and regression with tuning parameters
             tuneLength = 20, # # An integer denoting the amount of granularity in
             # the tuning parameter grid
             trControl = trControl, # A list of values that define how this function acts.
             preProc = c("center", "scale"), #  string vector that defines a pre-processing of the
             # predictor data.
             metric = "ROC", # A list of values that define how this function acts.
             tuneGrid = expand.grid(k = 1:60) # A data frame with possible tuning values
             )
```

```r
# The following chunk of code was not part of the R-blogger template.
# Threshold definition:
#     A data frame with columns for each of the tuning parameters from the model
#     along with an additional column called prob_threshold for the probability
#     threshold. There are also columns for summary statistics averaged over
#     resamples with column names corresponding to the input argument statistics."
```

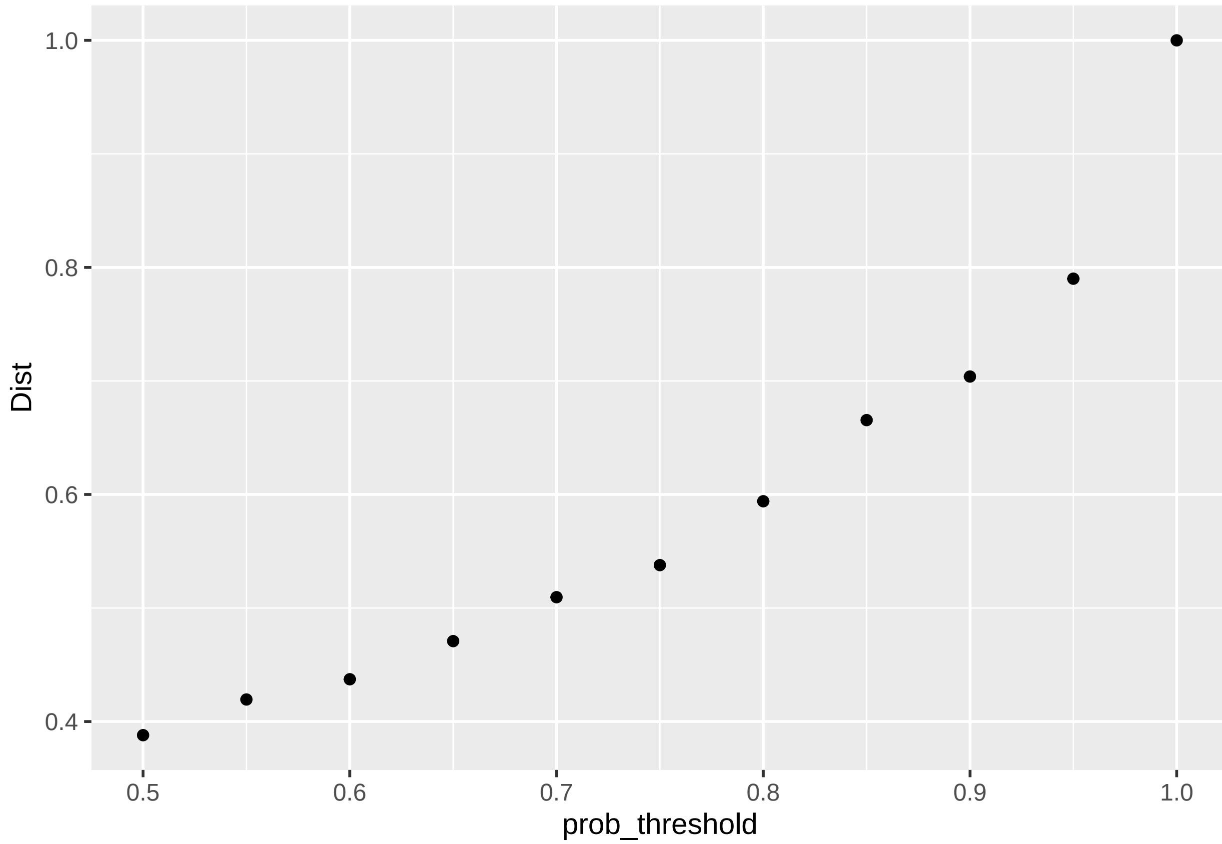
```r
# This will generate a thresfold metrics table from 50% to 100% in increases of 5%.
resample_stats <- thresholder(fit,
                              threshold = seq(.5, 1, by = 0.05), #
                              final = TRUE)
```

```
## Warning in .fun(piece, ...): The following columns have missing values (NA), which have been removed
```

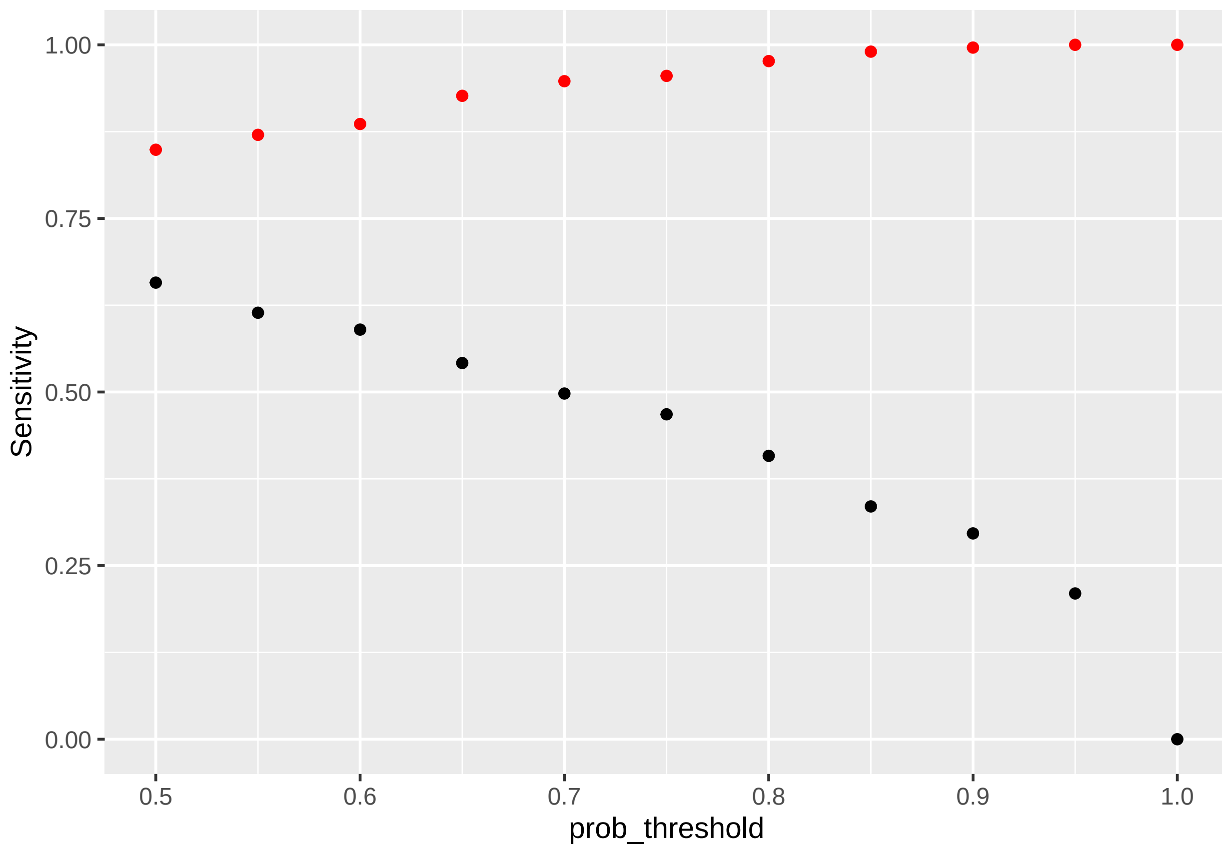
```r
# Plots related to the above table
ggplot(resample_stats, aes(x = prob_threshold, y = J)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Dist)) +
geom_point()
```




```
ggplot(resample_stats, aes(x = prob_threshold, y = Sensitivity)) +
geom_point() + geom_point(aes(y = Specificity), col = "red")
```

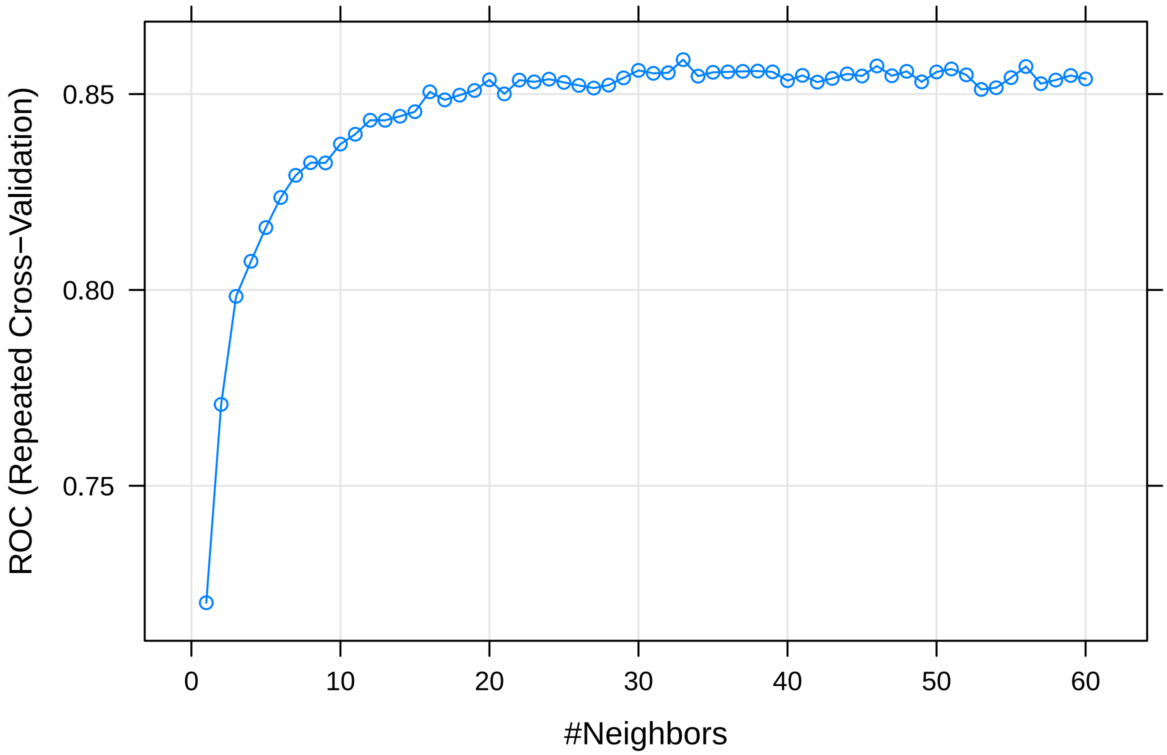



```r
# Model Performance
fit
```

```
## k-Nearest Neighbors
##
## 531 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 477, 478, 479, 478, 478, 478, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.7201393  0.8206423  0.6196364
##    2  0.7707792  0.7986489  0.6545455
##    3  0.7983545  0.7757697  0.7190909
##    4  0.8073128  0.7664009  0.7200000
##    5  0.8159165  0.7602990  0.7347273
##    6  0.8236047  0.7492027  0.7425455
##    7  0.8292478  0.7482060  0.7556364
##    8  0.8324844  0.7472757  0.7714545
##    9  0.8324266  0.7341528  0.7632727
```

```
##   10  0.8372378  0.7393798  0.7825455
##   11  0.8397476  0.7313511  0.8060000
##   12  0.8433223  0.7285382  0.7938182
##   13  0.8432993  0.7281285  0.8056364
##   14  0.8443350  0.7117829  0.8041818
##   15  0.8454951  0.7140864  0.8134545
##   16  0.8505782  0.7164341  0.8076364
##   17  0.8485201  0.7089701  0.8230909
##   18  0.8497210  0.7018937  0.8230909
##   19  0.8509136  0.6986600  0.8232727
##   20  0.8536452  0.7052049  0.8249091
##   21  0.8500351  0.6986932  0.8292727
##   22  0.8535678  0.7024142  0.8270909
##   23  0.8531212  0.7033666  0.8329091
##   24  0.8538054  0.6976633  0.8274545
##   25  0.8529758  0.6963455  0.8370909
##   26  0.8522094  0.6888372  0.8369091
##   27  0.8515287  0.6911628  0.8390909
##   28  0.8522851  0.6827464  0.8412727
##   29  0.8541272  0.6818826  0.8429091
##   30  0.8560510  0.6771318  0.8450909
##   31  0.8552929  0.6719934  0.8390909
##   32  0.8554464  0.6682392  0.8450909
##   33  0.8587892  0.6575415  0.8489091
##   34  0.8545671  0.6631561  0.8429091
##   35  0.8555578  0.6631783  0.8450909
##   36  0.8556692  0.6575083  0.8469091
##   37  0.8557993  0.6529014  0.8414545
##   38  0.8558711  0.6561351  0.8370909
##   39  0.8556654  0.6510188  0.8543636
##   40  0.8534161  0.6570764  0.8527273
##   41  0.8547691  0.6463566  0.8487273
##   42  0.8530499  0.6500775  0.8527273
##   43  0.8539858  0.6430897  0.8449091
##   44  0.8551578  0.6454264  0.8547273
##   45  0.8546049  0.6412182  0.8410909
##   46  0.8571877  0.6454485  0.8509091
##   47  0.8546662  0.6421816  0.8565455
##   48  0.8558025  0.6426135  0.8507273
##   49  0.8531120  0.6407863  0.8567273
##   50  0.8556419  0.6347065  0.8549091
##   51  0.8564134  0.6365559  0.8523636
##   52  0.8549005  0.6426024  0.8581818
##   53  0.8511804  0.6365559  0.8467273
##   54  0.8516190  0.6384053  0.8607273
##   55  0.8541971  0.6351938  0.8603636
##   56  0.8570374  0.6337542  0.8589091
##   57  0.8526449  0.6356368  0.8545455
##   58  0.8535763  0.6369878  0.8641818
##   59  0.8547353  0.6304983  0.8685455
##   60  0.8538635  0.6319491  0.8778182
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 33.
```

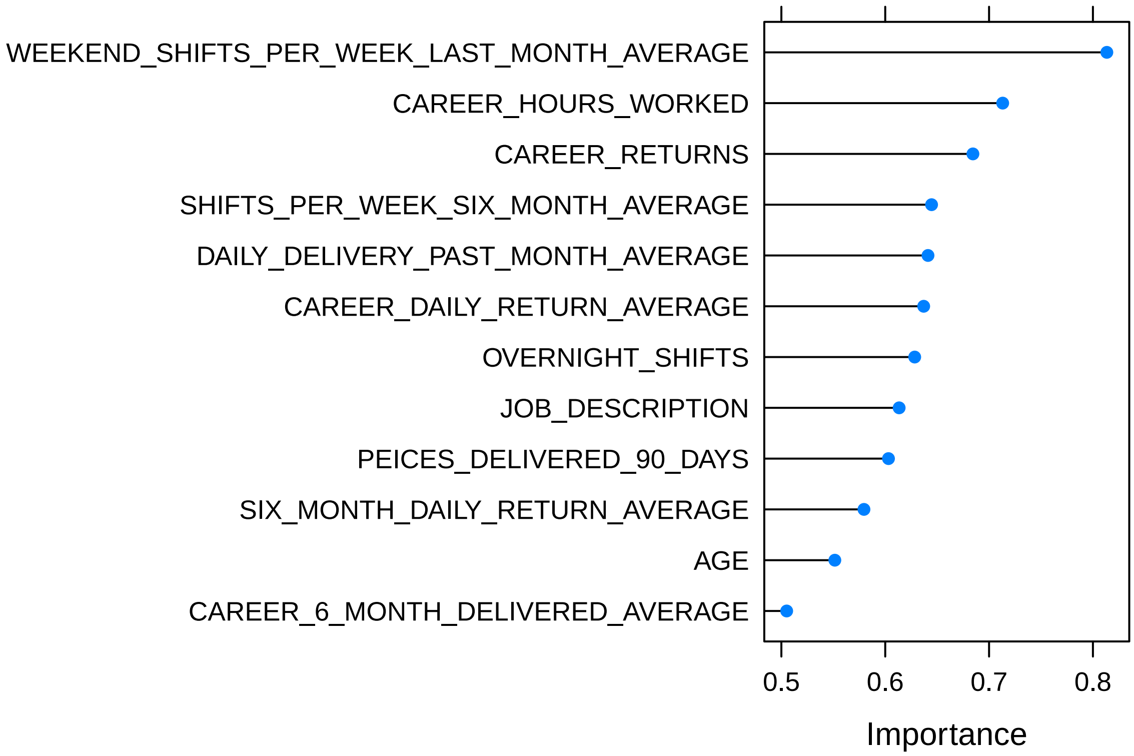
```r
plot(fit)
```



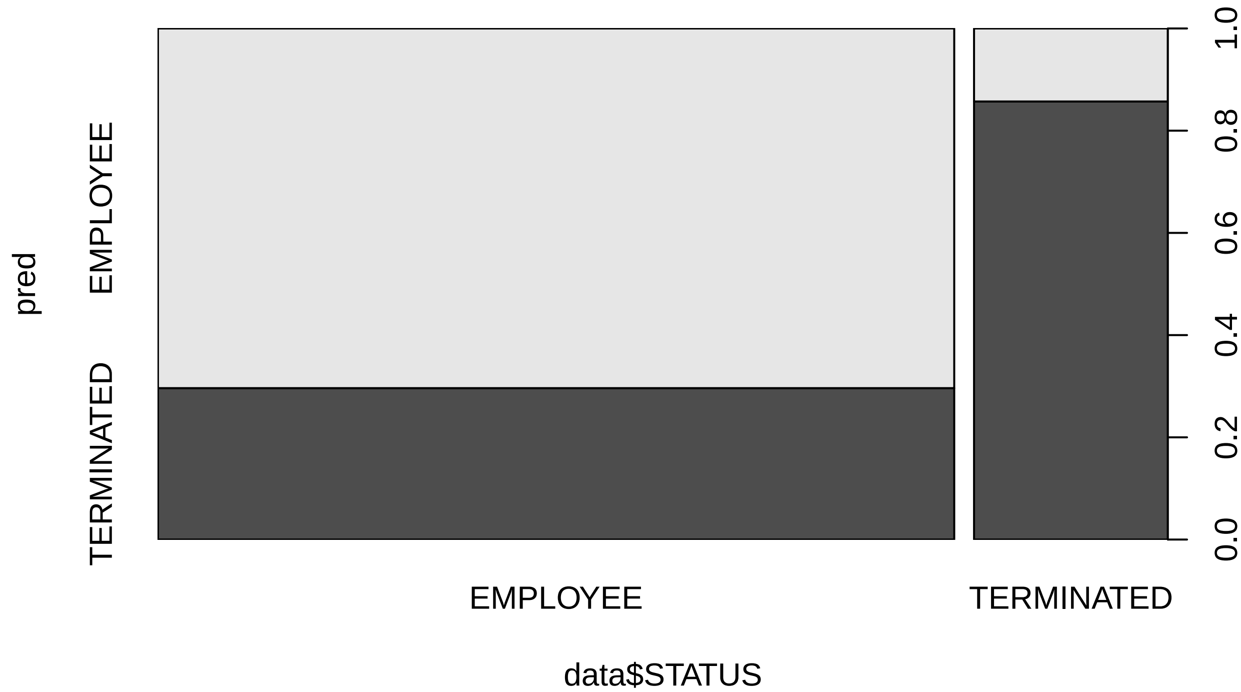

```r
# Loess r-squared variable importance
varImp(fit)
```

```
## ROC curve variable importance
##
##                                         Importance
## WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE     100.00
## CAREER_HOURS_WORKED                          67.47
## CAREER_RETURNS                               58.17
## SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE            45.25
## DAILY_DELIVERY_PAST_MONTH_AVERAGE            44.14
## CAREER_DAILY_RETURN_AVERAGE                  42.80
## OVERNIGHT_SHIFTS                             39.99
## JOB_DESCRIPTION                              35.13
## PEICES_DELIVERED_90_DAYS                     31.80
## SIX_MONTH_DAILY_RETURN_AVERAGE               24.14
## AGE                                          15.04
## CAREER_6_MONTH_DELIVERED_AVERAGE              0.00
```

```r
# variable importance visualization
importance <- varImp(fit, scale = FALSE)
plot(importance)
```




```
# Create plot
pred <- predict(fit, newdata = data)
plot(pred ~ data$STATUS)
```




```r
confusionMatrix(pred, data$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        506         25
##   TERMINATED      213        150
##
##                Accuracy : 0.7338
##                  95% CI : (0.7035, 0.7625)
##     No Information Rate : 0.8043
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3988
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7038
##             Specificity : 0.8571
##          Pos Pred Value : 0.9529
##          Neg Pred Value : 0.4132
##              Prevalence : 0.8043
##          Detection Rate : 0.5660
##    Detection Prevalence : 0.5940
```

```
##       Balanced Accuracy : 0.7804
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Median imputed table

```
data <- read_excel("C:/Users/lewis/Downloads/CompleteeData_median_imp.xlsx")
# Select and filter table with key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```
# First, we need to replace the zeros and ones from this STATUS column. Previously
# we were not able to find the order of importance after creating a fit variable.
# Therefore, we replace the values of zero and one to make run the varImp(Fit) function.
setDT(data)[STATUS == 1, STATUS := 2]
setDT(data)[STATUS == 0, STATUS := 1]

# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(1,2),
                     labels = c("EMPLOYEE",
                                "TERMINATED"
                    ))
#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))
```

```
# Data Partition: Let's create independent samples and create training and test
# dataset, 60% and 40% respectively, for prediction.

set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.6, 0.4))
training <- data[ind == 1,]
test <- data[ind == 2,]
str(training)
```

```
## Classes 'data.table' and 'data.frame':   531 obs. of  13 variables:
##  $ STATUS                                    : Factor w/ 2 levels "EMPLOYEE","TERMINATED": 1 1 1 1 1
##  $ CAREER_RETURNS                            : num  206 671 985 950 962 409 826 7 671 769 ...
```

```
##  $ CAREER_HOURS_WORKED                       : num  0 12156 14153 11408 10476 ...
##  $ SIX_MONTH_DAILY_RETURN_AVERAGE            : num  6.6 10.5 12.7 12.1 12.4 ...
##  $ PEICES_DELIVERED_90_DAYS                  : num  1054 22324 111226 28826 18658 ...
##  $ CAREER_6_MONTH_DELIVERED_AVERAGE          : num  152 500 1005 413 464 ...
##  $ WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE: num  0.667 0.667 0.667 0.667 0.667 ...
##  $ AGE                                       : num  55 41 54 60 43 31 30 27 56 60 ...
##  $ DAILY_DELIVERY_PAST_MONTH_AVERAGE         : num  533 5930 20872 7541 6778 ...
##  $ CAREER_DAILY_RETURN_AVERAGE               : num  14.7 11.6 13.8 14.1 12.8 ...
##  $ OVERNIGHT_SHIFTS                          : num  1 44 23 48 73 364 51 777 33 93 ...
##  $ JOB_DESCRIPTION                           : int  1 2 3 2 3 2 2 1 2 3 ...
##  $ SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE         : num  3.39 4.11 4.96 3.82 4.32 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Before making  model we need to create train control. Let's create train
# control based on below code.
trControl <- trainControl(method = "repeatedcv", # resampling method
                          sampling = "smote", #  balance data
                          number = 10, # Either the number of folds or number of
                                       # resampling iterations
                          repeats = 5, # an indicator of how much of the hold-out predictions for each
                          # resample should be saved.
                          classProbs = TRUE, # a logical; should class probabilities
                          # be computed for classification models (along
                          # with predicted values) in each resample?
                          summaryFunction = twoClassSummary, #metrics that rely on
                          # class probabilities
                          savePredictions = TRUE, #an indicator of how much of
                          # the hold-out predictions for each resample should be saved.
                          allowParallel = FALSE # an indicator of how much of the
                          # hold-out predictions for each resample should be saved.

                            )
```

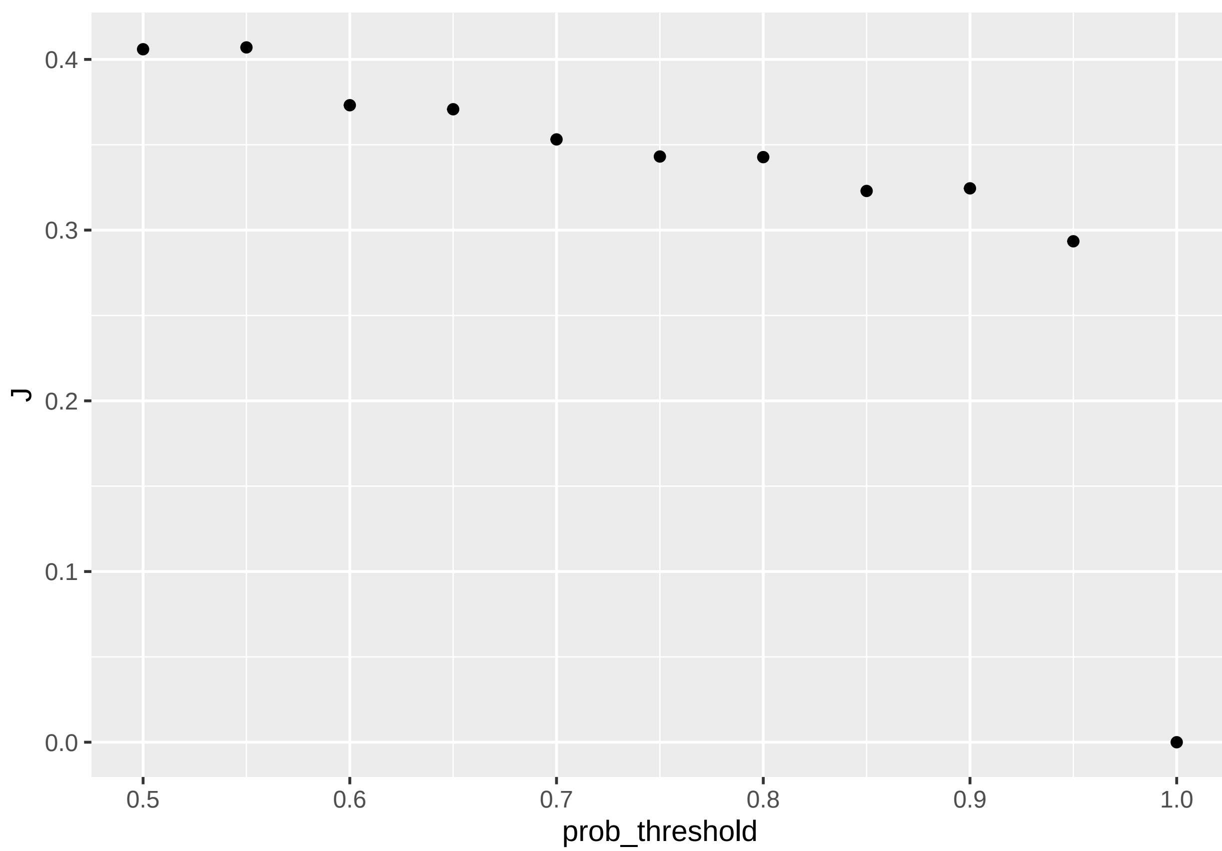
```
# trainControl is from caret package, number of iteration is 10 times.
# and repeat the cross validation is 3 times.
set.seed(222)
fit <- train(STATUS ~ .,
             data = training,
             method = 'knn', # For classification and regression with tuning parameters
             tuneLength = 20, # # An integer denoting the amount of granularity in
             # the tuning parameter grid
             trControl = trControl, # A list of values that define how this function acts.
             preProc = c("center", "scale"), #  string vector that defines a pre-processing of the
             # predictor data.
             metric = "ROC", # A list of values that define how this function acts.
             tuneGrid = expand.grid(k = 1:60) # A data frame with possible tuning values
             )
```

```
# The following chunk of code was not part of the R-blogger template.
# Threshold definition:
#     A data frame with columns for each of the tuning parameters from the model
#     along with an additional column called prob_threshold for the probability
#     threshold. There are also columns for summary statistics averaged over
#     resamples with column names corresponding to the input argument statistics."
```

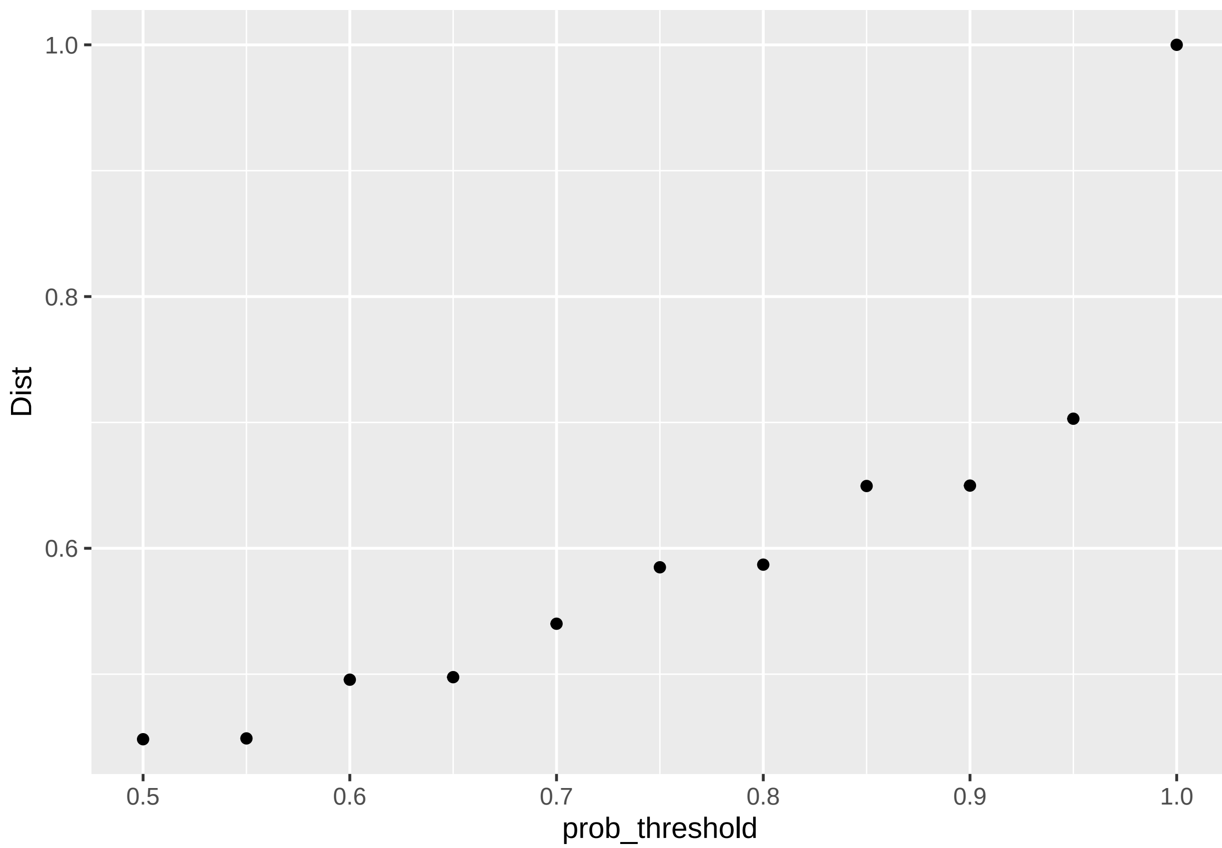
```r
# This will generate a thresfold metrics table from 50% to 100% in increases of 5%.
resample_stats <- thresholder(fit,
                              threshold = seq(.5, 1, by = 0.05), #
                              final = TRUE)
```

```
## Warning in .fun(piece, ...): The following columns have missing values (NA), which have been removed
```

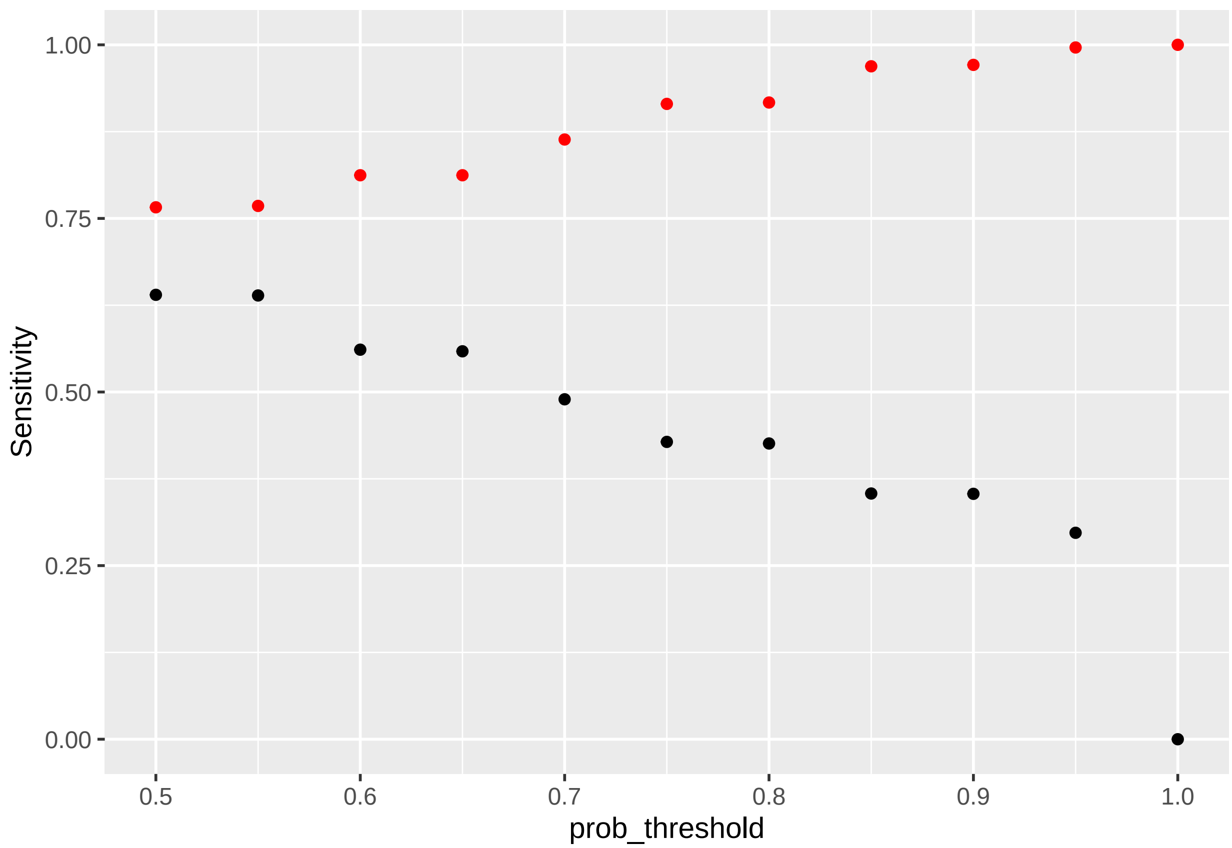
```r
# Plots related to the above table
ggplot(resample_stats, aes(x = prob_threshold, y = J)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Dist)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Sensitivity)) +
geom_point() +
geom_point(aes(y = Specificity), col = "red")
```

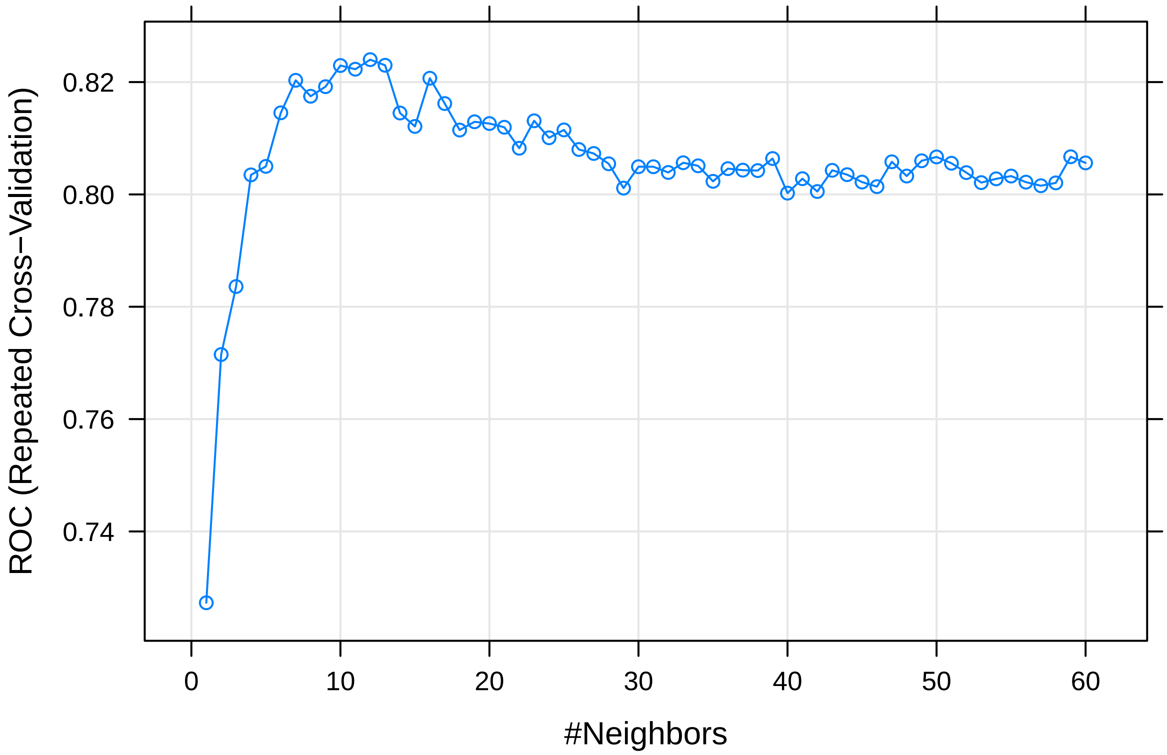



```
# Model Performance
fit
```

```
## k-Nearest Neighbors
##
## 531 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 477, 478, 479, 478, 478, 478, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.7273002  0.8444408  0.6067273
##    2  0.7714939  0.8079734  0.6240000
##    3  0.7835932  0.7878848  0.6809091
##    4  0.8034762  0.7753156  0.7176364
##    5  0.8049901  0.7570875  0.6976364
##    6  0.8145227  0.7459136  0.6912727
##    7  0.8203135  0.7333223  0.7116364
##    8  0.8174738  0.7268882  0.7072727
##    9  0.8191813  0.7141528  0.7210909
```

```
##   10  0.8229484  0.7034219  0.7305455
##   11  0.8222854  0.6879402  0.7270909
##   12  0.8239937  0.6861351  0.7470909
##   13  0.8229916  0.6772979  0.7405455
##   14  0.8144929  0.6739313  0.7478182
##   15  0.8121036  0.6731118  0.7421818
##   16  0.8206813  0.6570986  0.7480000
##   17  0.8161764  0.6595681  0.7485455
##   18  0.8114600  0.6563123  0.7540000
##   19  0.8129318  0.6478959  0.7538182
##   20  0.8126155  0.6436102  0.7538182
##   21  0.8119547  0.6451274  0.7429091
##   22  0.8082219  0.6375748  0.7527273
##   23  0.8131053  0.6451052  0.7758182
##   24  0.8100818  0.6352270  0.7658182
##   25  0.8114824  0.6394352  0.7676364
##   26  0.8079990  0.6269214  0.7778182
##   27  0.8072871  0.6220709  0.7694545
##   28  0.8054528  0.6160687  0.7794545
##   29  0.8011196  0.6193245  0.7660000
##   30  0.8049282  0.6183721  0.7870909
##   31  0.8049259  0.6113732  0.7856364
##   32  0.8039058  0.5977852  0.7809091
##   33  0.8056362  0.6099889  0.7923636
##   34  0.8050909  0.6047730  0.7856364
##   35  0.8023302  0.6052049  0.7810909
##   36  0.8045863  0.6053156  0.7794545
##   37  0.8043177  0.6001661  0.7892727
##   38  0.8042400  0.6030565  0.7812727
##   39  0.8063875  0.6038649  0.7943636
##   40  0.8002274  0.6035327  0.7814545
##   41  0.8028004  0.6071872  0.7836364
##   42  0.8005131  0.5969657  0.7890909
##   43  0.8042888  0.5968660  0.7849091
##   44  0.8035143  0.5950609  0.7796364
##   45  0.8021970  0.5964673  0.7787273
##   46  0.8013844  0.5978295  0.7830909
##   47  0.8058108  0.5876080  0.8060000
##   48  0.8032682  0.5884828  0.7980000
##   49  0.8059876  0.5866334  0.7809091
##   50  0.8066735  0.5871207  0.8025455
##   51  0.8055468  0.5852159  0.8118182
##   52  0.8038793  0.5903544  0.8081818
##   53  0.8020994  0.5702436  0.8047273
##   54  0.8027757  0.5753821  0.8065455
##   55  0.8032688  0.5740199  0.7925455
##   56  0.8021800  0.5786489  0.8161818
##   57  0.8015456  0.5716501  0.8120000
##   58  0.8020557  0.5636877  0.7976364
##   59  0.8067006  0.5688594  0.8103636
##   60  0.8056089  0.5614286  0.8130909
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 12.
```

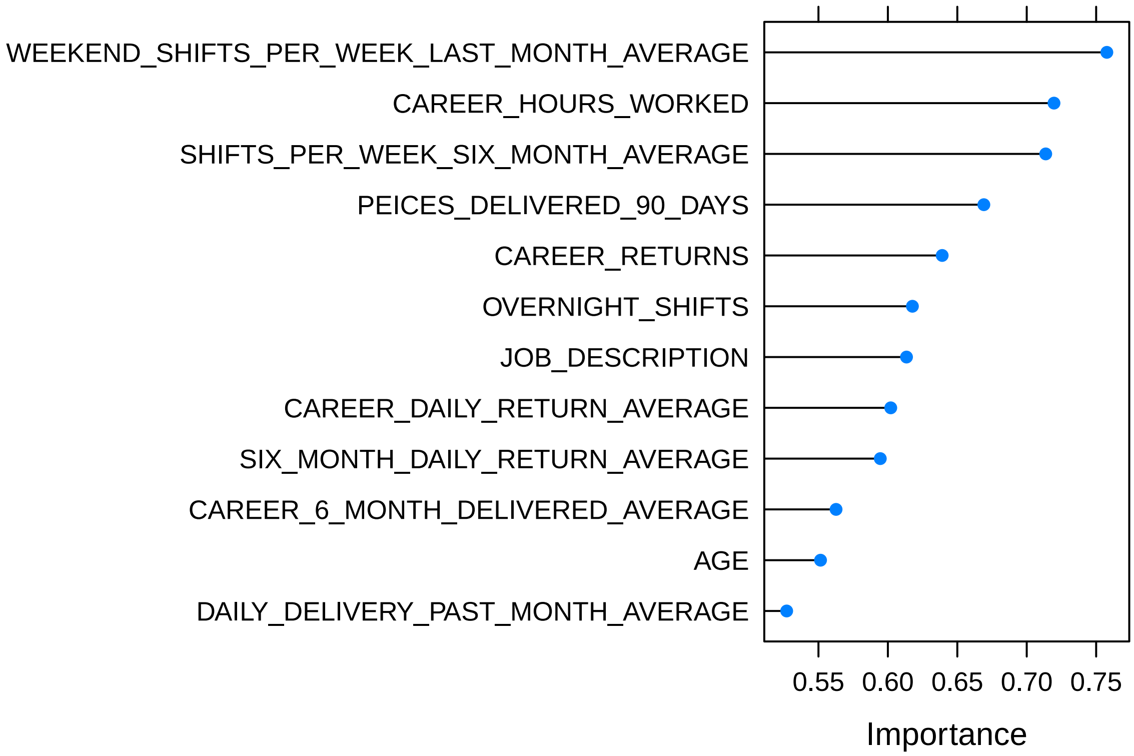
```r
plot(fit)
```



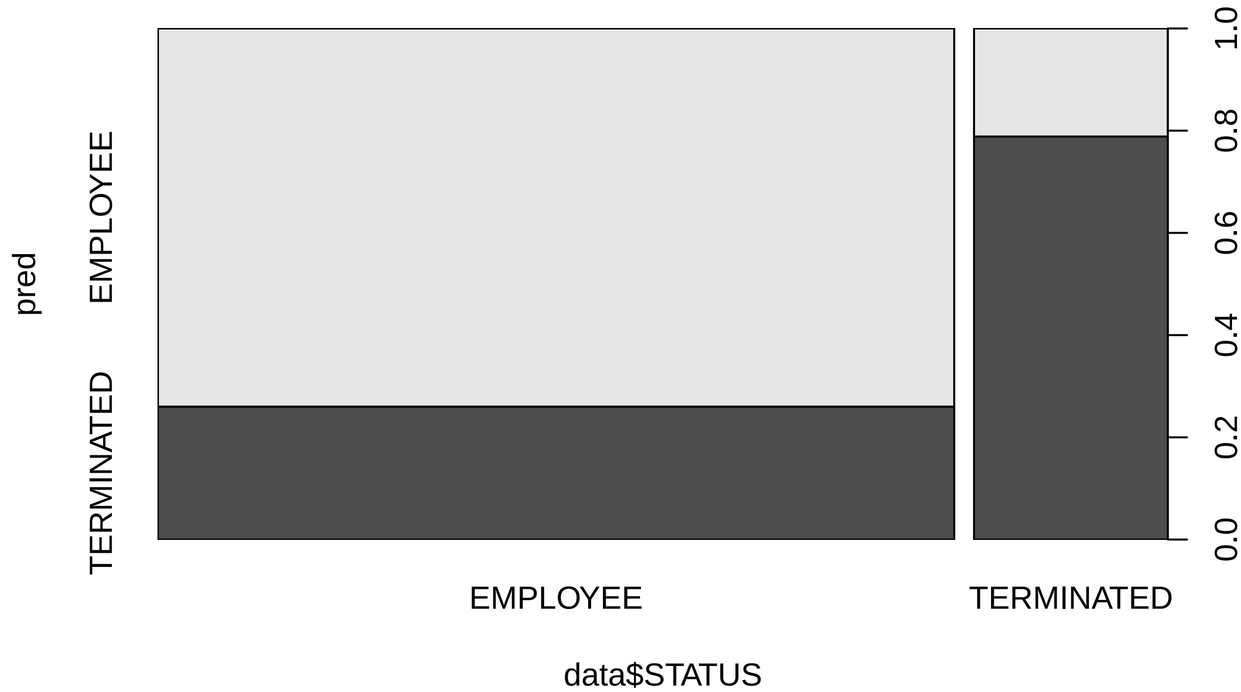

```r
# Loess r-squared variable importance
varImp(fit)
```

```
## ROC curve variable importance
##
##                                            Importance
## WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE     100.00
## CAREER_HOURS_WORKED                             83.49
## SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE               80.91
## PEICES_DELIVERED_90_DAYS                        61.58
## CAREER_RETURNS                                  48.57
## OVERNIGHT_SHIFTS                                39.27
## JOB_DESCRIPTION                                 37.43
## CAREER_DAILY_RETURN_AVERAGE                     32.51
## SIX_MONTH_DAILY_RETURN_AVERAGE                  29.24
## CAREER_6_MONTH_DELIVERED_AVERAGE                15.43
## AGE                                             10.57
## DAILY_DELIVERY_PAST_MONTH_AVERAGE                0.00
```

```r
# variable importance visualization
importance <- varImp(fit, scale = FALSE)
plot(importance)
```




```
# Create plot
pred <- predict(fit, newdata = data)
plot(pred ~ data$STATUS)
```




```
confusionMatrix(pred, data$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        532         37
##   TERMINATED      187        138
##
##                Accuracy : 0.7494
##                  95% CI : (0.7197, 0.7775)
##     No Information Rate : 0.8043
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3991
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7399
##             Specificity : 0.7886
##          Pos Pred Value : 0.9350
##          Neg Pred Value : 0.4246
##              Prevalence : 0.8043
##          Detection Rate : 0.5951
##    Detection Prevalence : 0.6365
```

```
##       Balanced Accuracy : 0.7642
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Mean imputed table

```r
# Import Excel file: Mean imputed table
data <- read_excel("C:/Users/lewis/Downloads/completeData_mean_imp.xlsx")
# Select and filter table with key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE

                )
```

```r
# First, we need to replace the zeros and ones from this STATUS column. Previously
# we were not able to find the order of importance after creating a fit variable.
# Therefore, we replace the values of zero and one to make run the varImp(Fit) function.
setDT(data)[STATUS == 1, STATUS := 2]
setDT(data)[STATUS == 0, STATUS := 1]

# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(1,2),
                     labels = c("EMPLOYEE",
                                "TERMINATED"
                    ))
#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))
```

```r
# Data Partition: Let's create independent samples and create training and test
# dataset, 60% and 40% respectively, for prediction.

set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.6, 0.4))
training <- data[ind == 1,]
test <- data[ind == 2,]
str(training)
```

```
## Classes 'data.table' and 'data.frame':   531 obs. of  13 variables:
##  $ STATUS                                    : Factor w/ 2 levels "EMPLOYEE","TERMINATED": 1 1 1 1 1
```

```
##  $ CAREER_RETURNS                          : num  206 671 985 950 962 409 826 7 671 769 ...
##  $ CAREER_HOURS_WORKED                     : num  0 12156 14153 11408 10476 ...
##  $ SIX_MONTH_DAILY_RETURN_AVERAGE          : num  6.6 10.5 12.7 12.1 12.4 ...
##  $ PEICES_DELIVERED_90_DAYS                : num  1054 22324 111226 28826 18658 ...
##  $ CAREER_6_MONTH_DELIVERED_AVERAGE        : num  152 500 1005 413 464 ...
##  $ WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE: num  0.449 0.667 0.667 0.667 0.667 ...
##  $ AGE                                     : num  55 41 54 60 43 31 30 27 56 60 ...
##  $ DAILY_DELIVERY_PAST_MONTH_AVERAGE       : num  533 5930 20872 7541 6778 ...
##  $ CAREER_DAILY_RETURN_AVERAGE             : num  14.7 11.6 13.8 14.1 12.8 ...
##  $ OVERNIGHT_SHIFTS                        : num  1 44 23 48 73 364 51 777 33 93 ...
##  $ JOB_DESCRIPTION                         : int  1 2 3 2 3 2 2 1 2 3 ...
##  $ SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE       : num  2.76 4.11 4.96 3.82 4.32 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Before making model we need to create train control. Let's create train
# control based on below code.
trControl <- trainControl(method = "repeatedcv", # resampling method
                          sampling = "smote", #  balance data
                          number = 10, # Either the number of folds or number of
                                       # resampling iterations
                          repeats = 5, # an indicator of how much of the hold-out predictions for each
                          # resample should be saved.
                          classProbs = TRUE, # a logical; should class probabilities
                          # be computed for classification models (along
                          # with predicted values) in each resample?
                          summaryFunction = twoClassSummary, #metrics that rely on
                          # class probabilities
                          savePredictions = TRUE, #an indicator of how much of
                          # the hold-out predictions for each resample should be saved.
                          allowParallel = FALSE # an indicator of how much of the
                          # hold-out predictions for each resample should be saved.

                            )
```

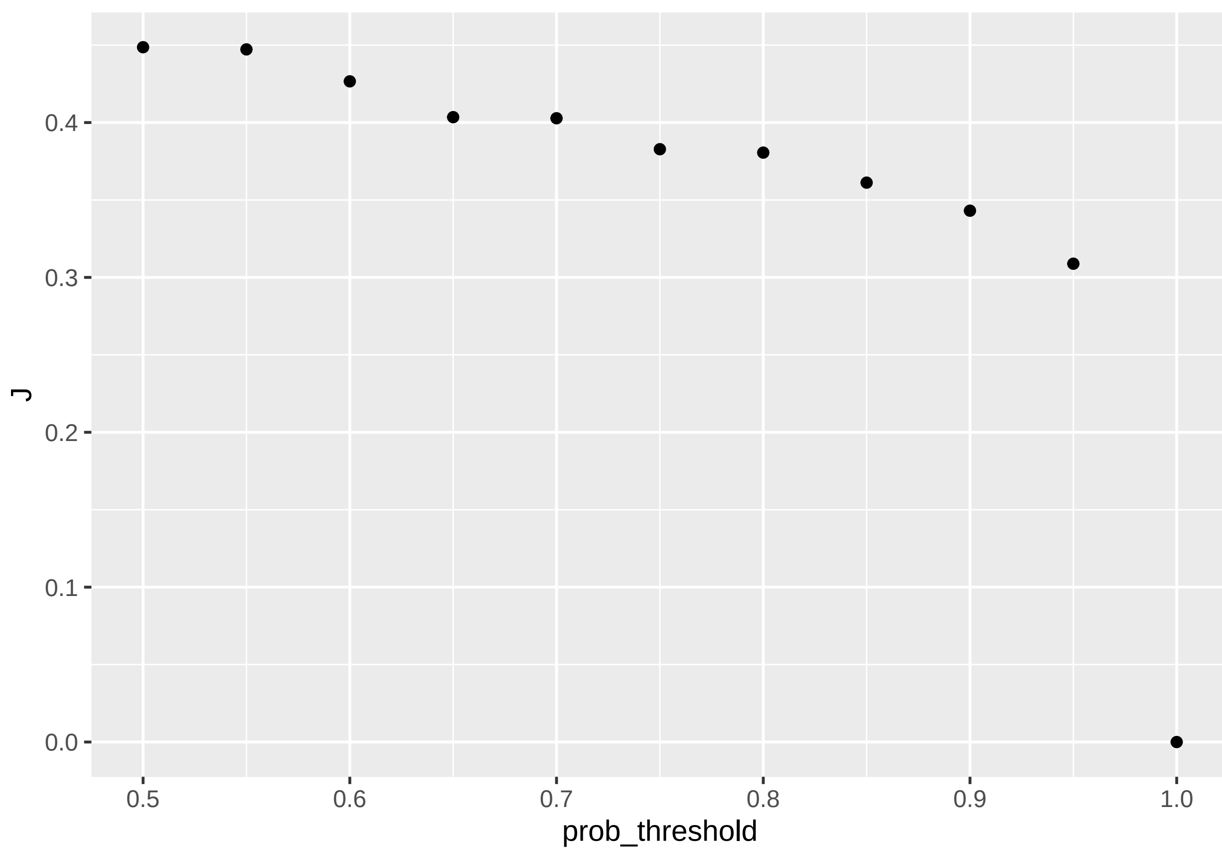
```r
# trainControl is from caret package, number of iteration is 10 times.
# and repeat the cross validation is 3 times.
set.seed(222)
fit <- train(STATUS ~ .,
             data = training,
             method = 'knn', # For classification and regression with tuning parameters
             tuneLength = 20, # # An integer denoting the amount of granularity in
             # the tuning parameter grid
             trControl = trControl, # A list of values that define how this function acts.
             preProc = c("center", "scale"), #  string vector that defines a pre-processing of the
             # predictor data.
             metric = "ROC", # A list of values that define how this function acts.
             tuneGrid = expand.grid(k = 1:60) # A data frame with possible tuning values
             )
```

```r
# The following chunk of code was not part of the R-blogger template.
# Threshold definition:
#     A data frame with columns for each of the tuning parameters from the model
#     along with an additional column called prob_threshold for the probability
#     threshold. There are also columns for summary statistics averaged over
```

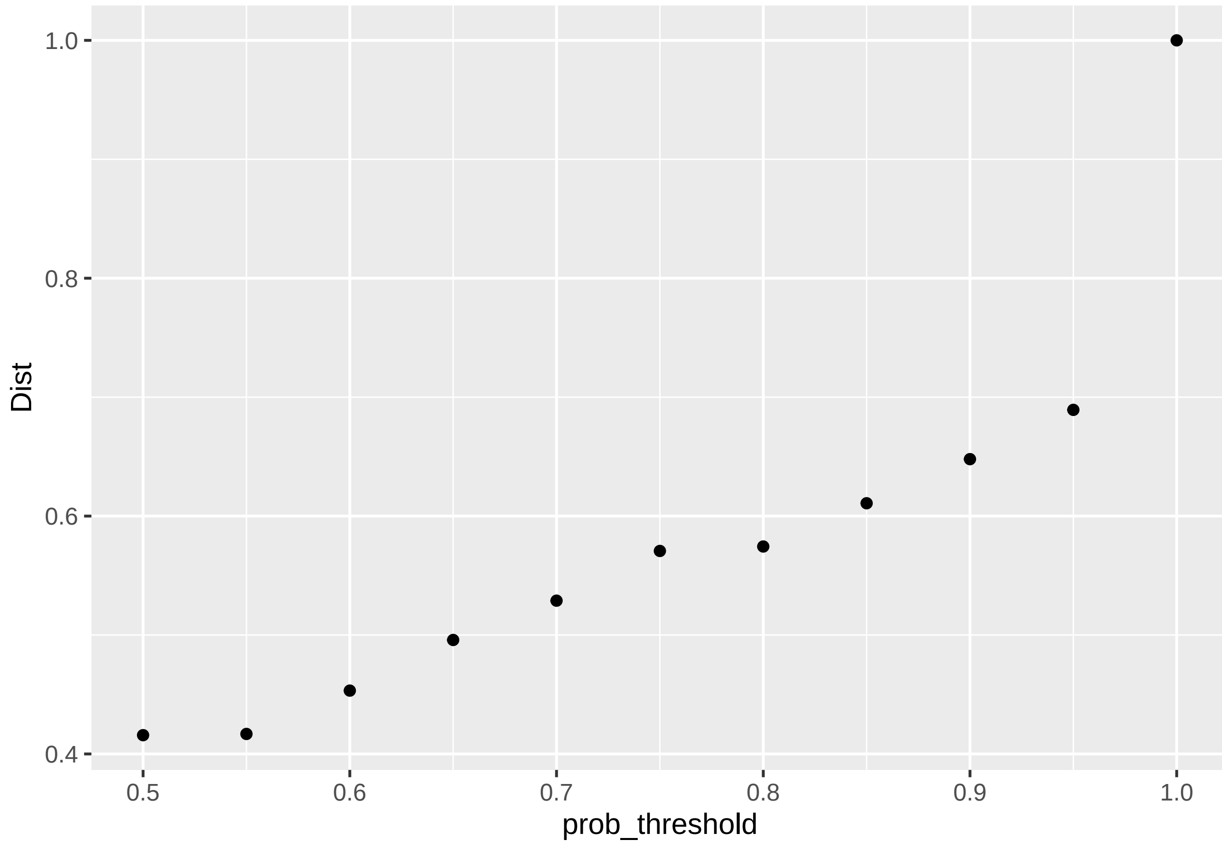
```r
#     resamples with column names corresponding to the input argument statistics."

# This will generate a thresfold metrics table from 50% to 100% in increases of 5%.
resample_stats <- thresholder(fit,
                              threshold = seq(.5, 1, by = 0.05), #
                              final = TRUE)
```

```
## Warning in .fun(piece, ...): The following columns have missing values (NA), which have been removed
```

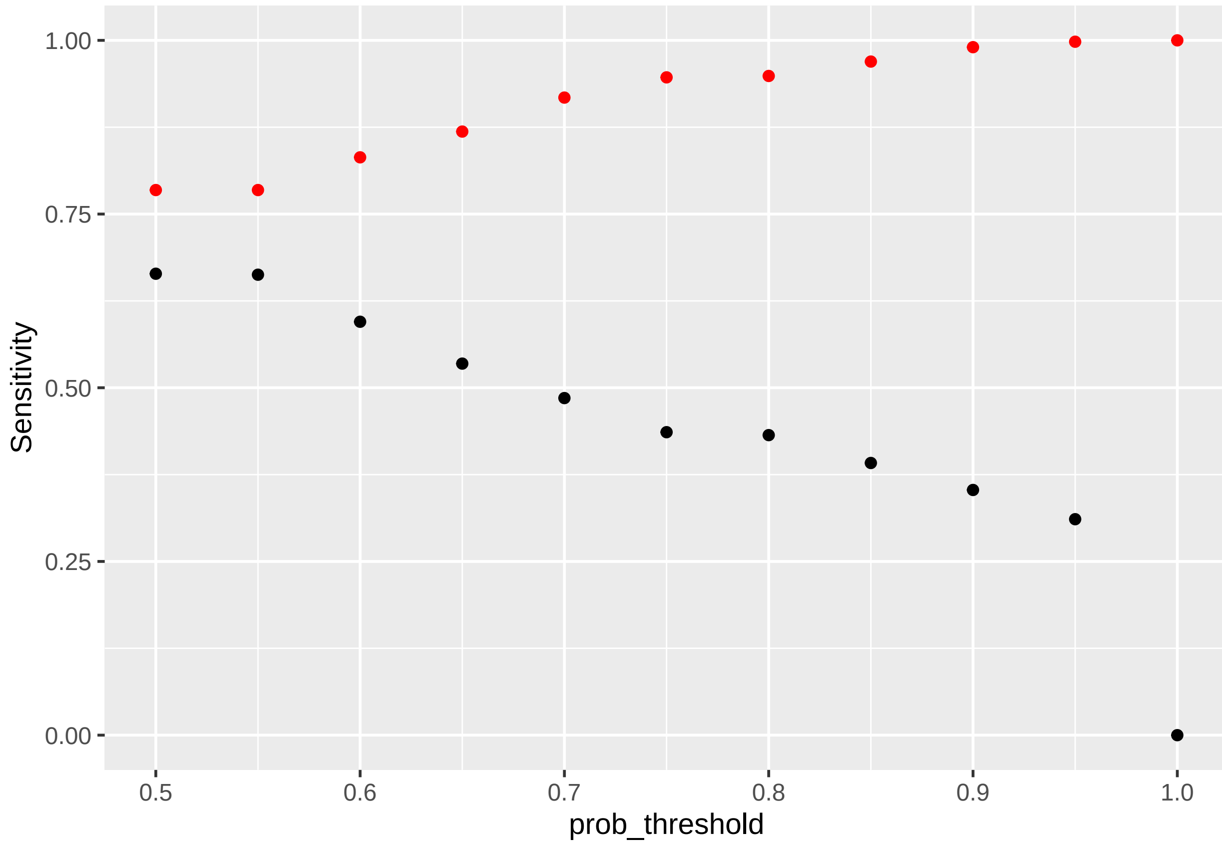
```r
# Plots related to the above table
ggplot(resample_stats, aes(x = prob_threshold, y = J)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Dist)) +
geom_point()
```




```
ggplot(resample_stats, aes(x = prob_threshold, y = Sensitivity)) +
geom_point() + geom_point(aes(y = Specificity), col = "red")
```

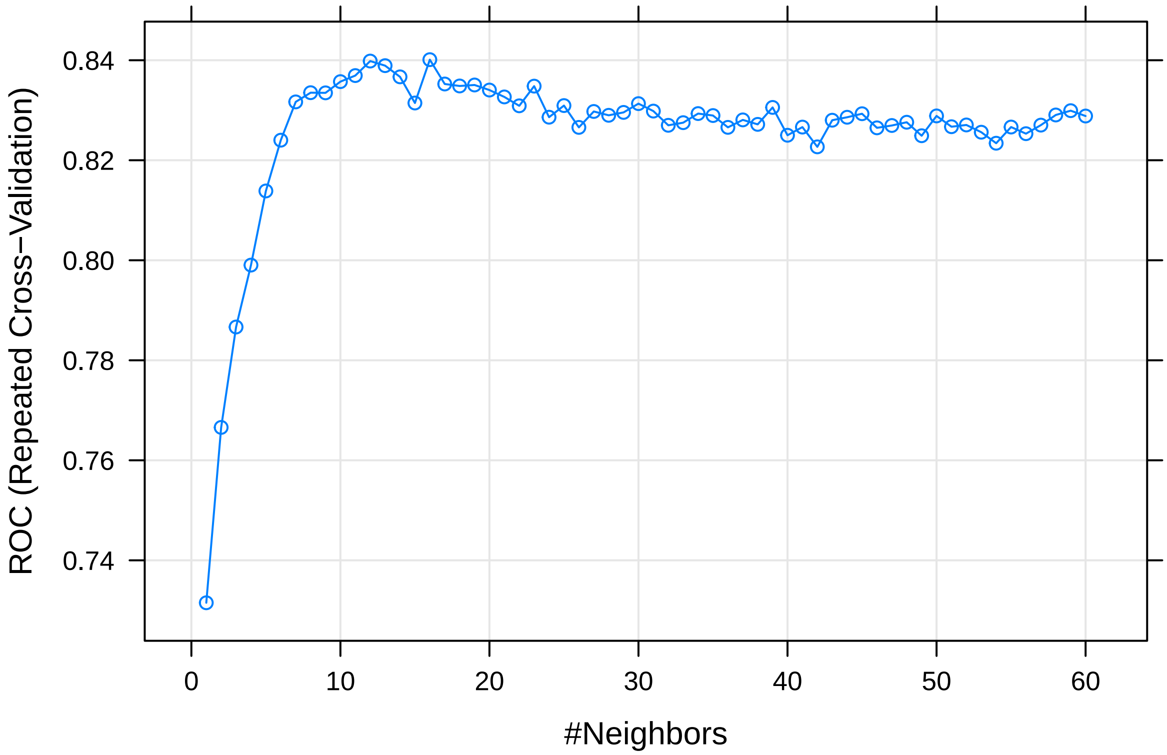



```
# Model Performance
fit
```

```
## k-Nearest Neighbors
##
## 531 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 477, 478, 479, 478, 478, 478, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.7315101  0.8613178  0.5985455
##    2  0.7665752  0.8295460  0.6120000
##    3  0.7866520  0.8005316  0.6729091
##    4  0.7990341  0.7865338  0.6938182
##    5  0.8138616  0.7781395  0.7096364
##    6  0.8240122  0.7814618  0.7185455
##    7  0.8316724  0.7599225  0.7072727
##    8  0.8335070  0.7603987  0.6952727
##    9  0.8334774  0.7500775  0.7134545
```

```
##   10  0.8357231  0.7365891  0.7320000
##   11  0.8369347  0.7407752  0.7267273
##   12  0.8398347  0.7286379  0.7341818
##   13  0.8389141  0.7113843  0.7383636
##   14  0.8366777  0.7160687  0.7529091
##   15  0.8314470  0.7090144  0.7514545
##   16  0.8401191  0.7029125  0.7650909
##   17  0.8352481  0.6954817  0.7365455
##   18  0.8348508  0.6875415  0.7467273
##   19  0.8350527  0.6912957  0.7540000
##   20  0.8340374  0.6922038  0.7600000
##   21  0.8326624  0.6833998  0.7500000
##   22  0.8308841  0.6819601  0.7521818
##   23  0.8348218  0.6764230  0.7560000
##   24  0.8286103  0.6646069  0.7603636
##   25  0.8309315  0.6693245  0.7716364
##   26  0.8265717  0.6609192  0.7605455
##   27  0.8297569  0.6609081  0.7758182
##   28  0.8289889  0.6552713  0.7872727
##   29  0.8295683  0.6506645  0.7896364
##   30  0.8313231  0.6622813  0.8065455
##   31  0.8298319  0.6506312  0.8007273
##   32  0.8269865  0.6394574  0.8005455
##   33  0.8275135  0.6455260  0.8043636
##   34  0.8293350  0.6478516  0.8101818
##   35  0.8289380  0.6431451  0.7985455
##   36  0.8265796  0.6399225  0.8009091
##   37  0.8280686  0.6441307  0.8121818
##   38  0.8271756  0.6319934  0.8183636
##   39  0.8305659  0.6371096  0.8240000
##   40  0.8250014  0.6343632  0.8147273
##   41  0.8266313  0.6328350  0.8149091
##   42  0.8226794  0.6250055  0.8218182
##   43  0.8280066  0.6230786  0.8372727
##   44  0.8285988  0.6155592  0.8294545
##   45  0.8293027  0.6286600  0.8298182
##   46  0.8264834  0.6160687  0.8243636
##   47  0.8269527  0.6165670  0.8274545
##   48  0.8276002  0.6030343  0.8363636
##   49  0.8249040  0.6151827  0.8274545
##   50  0.8288664  0.6067110  0.8376364
##   51  0.8267052  0.6095127  0.8407273
##   52  0.8270589  0.6063455  0.8501818
##   53  0.8256044  0.5978516  0.8465455
##   54  0.8234069  0.5893798  0.8449091
##   55  0.8266396  0.5955592  0.8523636
##   56  0.8253161  0.5936102  0.8449091
##   57  0.8270233  0.5851827  0.8529091
##   58  0.8290510  0.5838206  0.8590909
##   59  0.8299153  0.5815061  0.8643636
##   60  0.8288430  0.5651052  0.8507273
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 16.
```

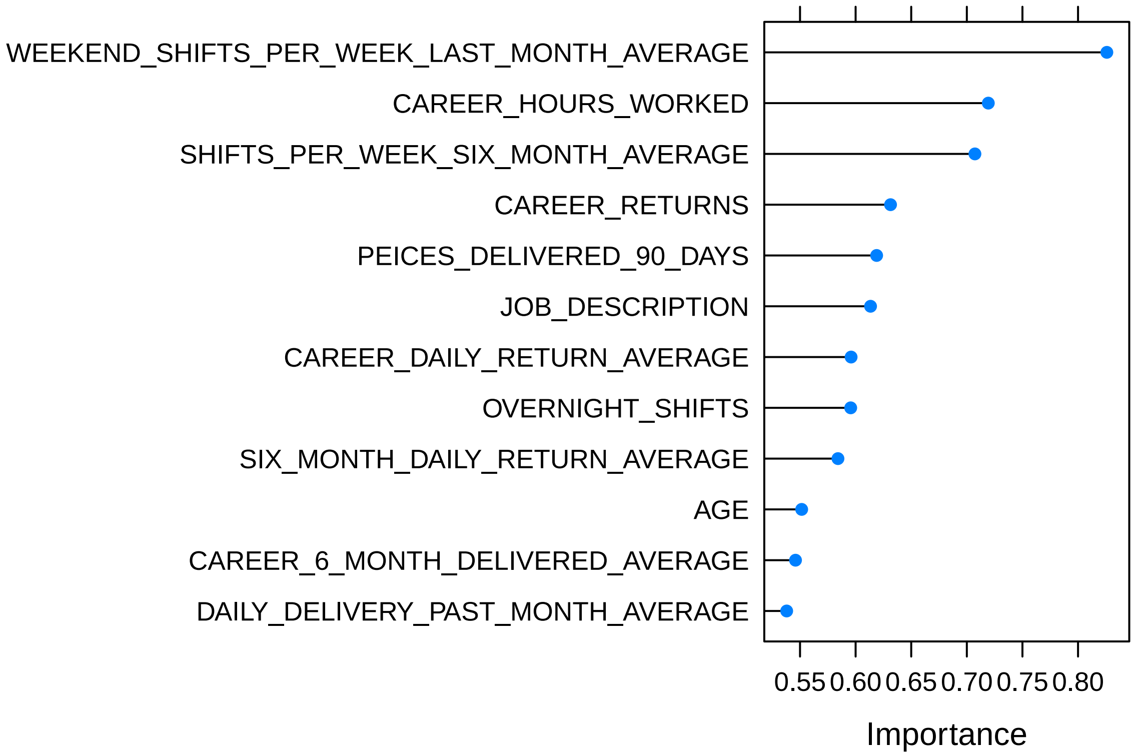
```r
plot(fit)
```



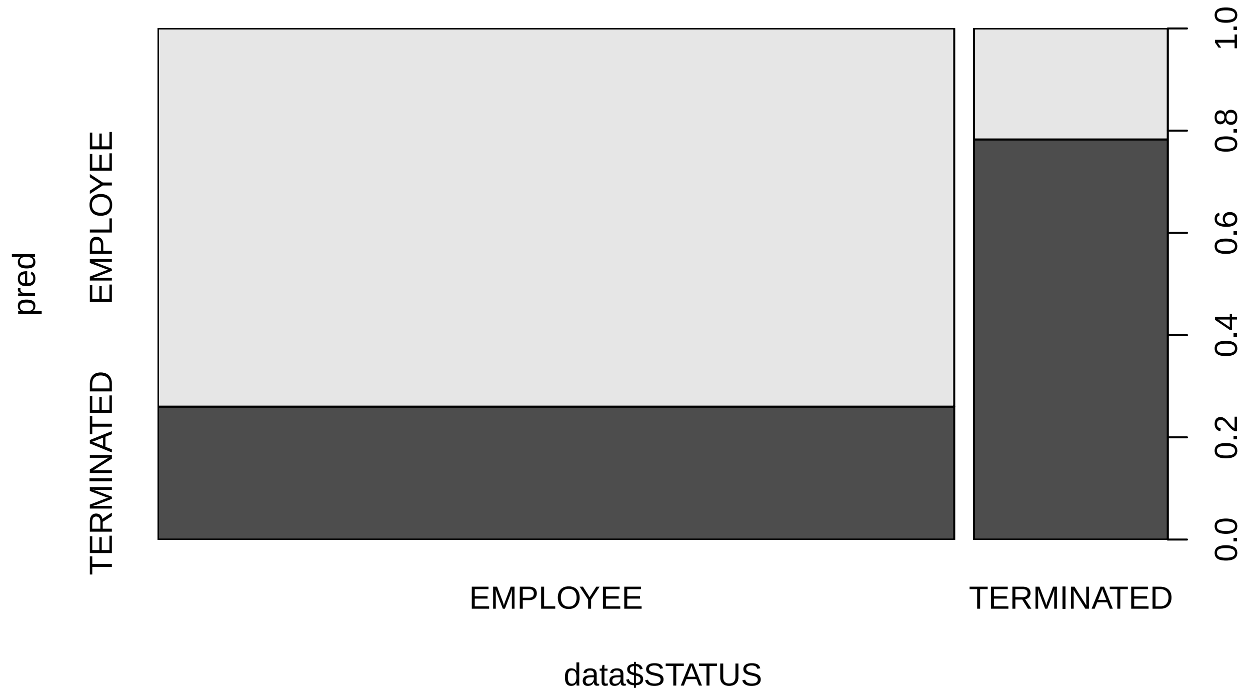

```r
# Loess r-squared variable importance
varImp(fit)
```

```
## ROC curve variable importance
##
##                                            Importance
## WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE    100.000
## CAREER_HOURS_WORKED                            62.979
## SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE              58.791
## CAREER_RETURNS                                 32.439
## PEICES_DELIVERED_90_DAYS                       28.098
## JOB_DESCRIPTION                                26.195
## CAREER_DAILY_RETURN_AVERAGE                    20.128
## OVERNIGHT_SHIFTS                               20.002
## SIX_MONTH_DAILY_RETURN_AVERAGE                 16.019
## AGE                                             4.684
## CAREER_6_MONTH_DELIVERED_AVERAGE                2.758
## DAILY_DELIVERY_PAST_MONTH_AVERAGE               0.000
```

```r
# variable importance visualization
importance <- varImp(fit, scale = FALSE)
plot(importance)
```




```
# Create plot
pred <- predict(fit, newdata = data)
plot(pred ~ data$STATUS)
```




```
confusionMatrix(pred, data$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        532         38
##   TERMINATED      187        137
##
##                Accuracy : 0.7483
##                  95% CI : (0.7185, 0.7765)
##     No Information Rate : 0.8043
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3954
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7399
##             Specificity : 0.7829
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.4228
##              Prevalence : 0.8043
##          Detection Rate : 0.5951
##    Detection Prevalence : 0.6376
```

```
##       Balanced Accuracy : 0.7614
##
##        'Positive' Class : EMPLOYEE
##
```

Import Excel file: Random Forest method

```r
# Import Excel file: Mean imputed table
data <- read_excel("C:/Users/lewis/Downloads/randomforest output.xlsx")
# Select and filter table with key variables
data <- select(data, STATUS,
               CAREER_RETURNS,
               CAREER_HOURS_WORKED,
               SIX_MONTH_DAILY_RETURN_AVERAGE,
               PEICES_DELIVERED_90_DAYS,
               CAREER_6_MONTH_DELIVERED_AVERAGE,
               WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE,
               AGE,
               DAILY_DELIVERY_PAST_MONTH_AVERAGE,
               CAREER_DAILY_RETURN_AVERAGE,
               OVERNIGHT_SHIFTS,
               JOB_DESCRIPTION,
               SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE
                )
```

```r
# First, we need to replace the zeros and ones from this STATUS column. Previously
# we were not able to find the order of importance after creating a fit variable.
# Therefore, we replace the values of zero and one to make run the varImp(Fit) function.
setDT(data)[STATUS == 1, STATUS := 2]
setDT(data)[STATUS == 0, STATUS := 1]

# Create factors for the following columns
data$STATUS <- factor(data$STATUS, level = c(1,2),
                     labels = c("EMPLOYEE",
                                "TERMINATED"
                    ))
#Change job description type from char > factor > integer
data$JOB_DESCRIPTION=as.integer(as.factor(data$JOB_DESCRIPTION))
```

```r
# Data Partition: Let's create independent samples and create training and test
# dataset, 60% and 40% respectively, for prediction.

set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.6, 0.4))
training <- data[ind == 1,]
test <- data[ind == 2,]
str(training)
```

```
## Classes 'data.table' and 'data.frame':   531 obs. of  13 variables:
##  $ STATUS                                    : Factor w/ 2 levels "EMPLOYEE","TERMINATED": 1 1 1 1 1
##  $ CAREER_RETURNS                            : num  206 671 985 950 962 409 826 7 671 769 ...
```

```
##  $ CAREER_HOURS_WORKED                      : num  0 12156 14153 11408 10476 ...
##  $ SIX_MONTH_DAILY_RETURN_AVERAGE           : num  6.6 10.5 12.7 12.1 12.4 ...
##  $ PEICES_DELIVERED_90_DAYS                 : num  1054 22324 111226 28826 18658 ...
##  $ CAREER_6_MONTH_DELIVERED_AVERAGE         : num  152 500 1005 413 464 ...
##  $ WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE: num  0 0.667 0.667 0.667 0.667 ...
##  $ AGE                                      : num  55 41 54 60 43 31 30 27 56 60 ...
##  $ DAILY_DELIVERY_PAST_MONTH_AVERAGE        : num  533 5930 20872 7541 6778 ...
##  $ CAREER_DAILY_RETURN_AVERAGE              : num  14.7 11.6 13.8 14.1 12.8 ...
##  $ OVERNIGHT_SHIFTS                         : num  1 44 23 48 73 364 51 777 33 93 ...
##  $ JOB_DESCRIPTION                          : int  1 2 3 2 3 2 2 1 2 3 ...
##  $ SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE        : num  0.037 4.107 4.964 3.821 4.321 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Before making model we need to create train control. Let's create train
# control based on below code.
trControl <- trainControl(method = "repeatedcv", # resampling method
                          sampling = "smote", #  balance data
                          number = 10, # Either the number of folds or number of
                                      # resampling iterations
                          repeats = 5, # an indicator of how much of the hold-out predictions for each
                          # resample should be saved.
                          classProbs = TRUE, # a logical; should class probabilities
                          # be computed for classification models (along
                          # with predicted values) in each resample?
                          summaryFunction = twoClassSummary, #metrics that rely on
                          # class probabilities
                          savePredictions = TRUE, #an indicator of how much of
                          # the hold-out predictions for each resample should be saved.
                          allowParallel = FALSE # an indicator of how much of the
                          # hold-out predictions for each resample should be saved.

                            )
```

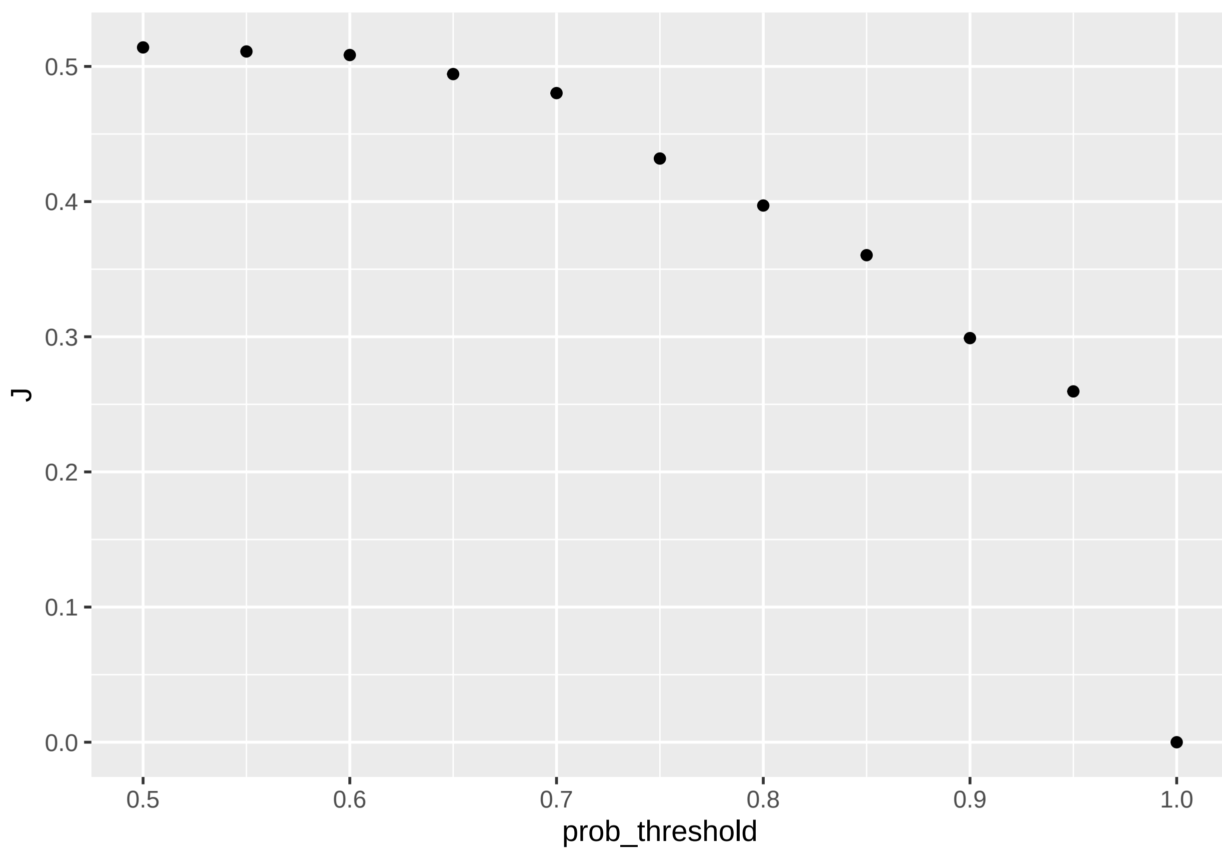
```
# trainControl is from caret package, number of iteration is 10 times.
# and repeat the cross validation is 3 times.
set.seed(222)
fit <- train(STATUS ~ .,
             data = training,
             method = 'knn', # For classification and regression with tuning parameters
             tuneLength = 20, # # An integer denoting the amount of granularity in
             # the tuning parameter grid
             trControl = trControl, # A list of values that define how this function acts.
             preProc = c("center", "scale"), #  string vector that defines a pre-processing of the
             # predictor data.
             metric = "ROC", # A list of values that define how this function acts.
             tuneGrid = expand.grid(k = 1:60) # A data frame with possible tuning values
             )
```

```
# The following chunk of code was not part of the R-blogger template.
# Threshold definition:
#     A data frame with columns for each of the tuning parameters from the model
#     along with an additional column called prob_threshold for the probability
#     threshold. There are also columns for summary statistics averaged over
#     resamples with column names corresponding to the input argument statistics."
```

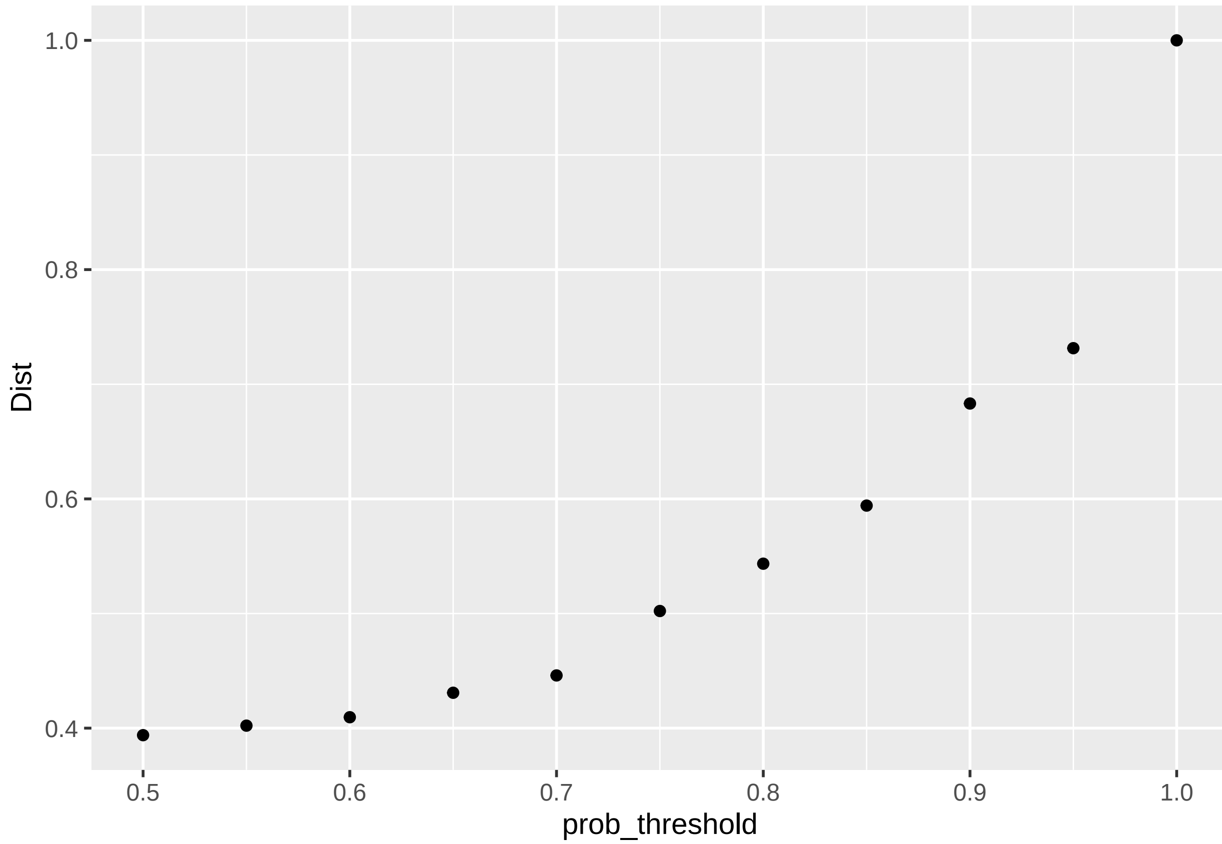
```r
# This will generate a thresfold metrics table from 50% to 100% in increases of 5%.
resample_stats <- thresholder(fit,
                              threshold = seq(.5, 1, by = 0.05), #
                              final = TRUE)
```

```
## Warning in .fun(piece, ...): The following columns have missing values (NA), which have been removed
```

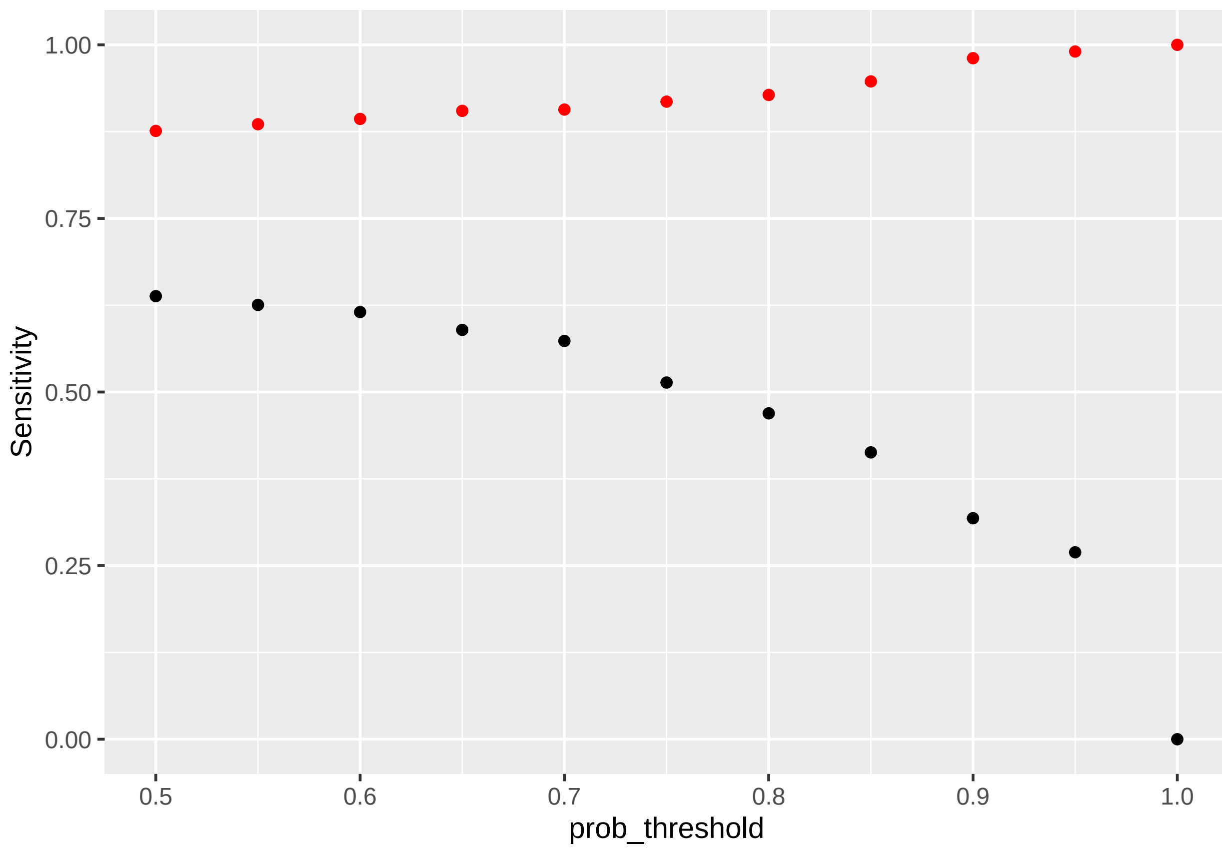
```r
# Plots related to the above table
ggplot(resample_stats, aes(x = prob_threshold, y = J)) +
geom_point()
```




```r
ggplot(resample_stats, aes(x = prob_threshold, y = Dist)) +
geom_point()
```




```
ggplot(resample_stats, aes(x = prob_threshold, y = Sensitivity)) +
geom_point() + geom_point(aes(y = Specificity), col = "red")
```

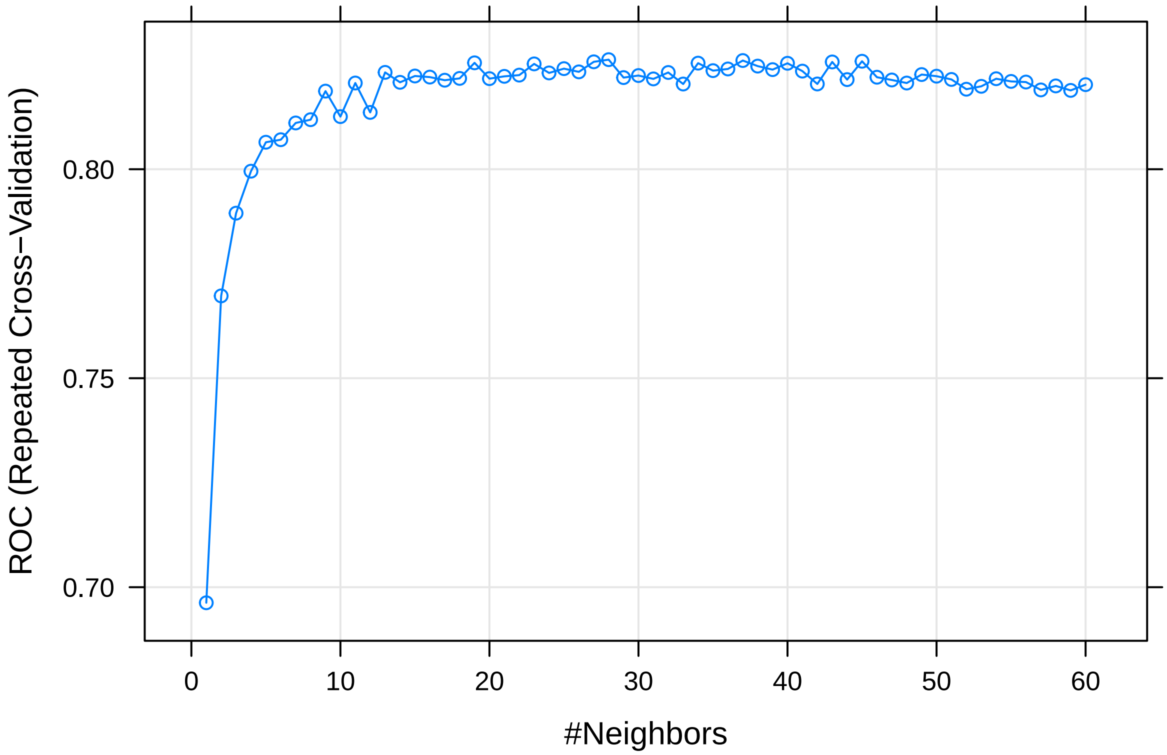



```r
# Model Performance
fit
```

```
## k-Nearest Neighbors
##
## 531 samples
##  12 predictor
##   2 classes: 'EMPLOYEE', 'TERMINATED'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 477, 478, 479, 478, 478, 478, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    1  0.6962722  0.8109081  0.5816364
##    2  0.7696925  0.7922924  0.6514545
##    3  0.7894838  0.7759579  0.7401818
##    4  0.7995220  0.7553821  0.7181818
##    5  0.8064515  0.7465559  0.7572727
##    6  0.8070589  0.7320155  0.7656364
##    7  0.8110641  0.7268660  0.7876364
##    8  0.8118497  0.7216722  0.7909091
##    9  0.8186869  0.7152049  0.8020000
```

```
##    10  0.8125860  0.7049612  0.8120000
##    11  0.8206209  0.7002436  0.8221818
##    12  0.8135946  0.6969657  0.8027273
##    13  0.8231710  0.6894906  0.8400000
##    14  0.8207868  0.6801218  0.8381818
##    15  0.8222992  0.6726910  0.8429091
##    16  0.8220418  0.6731783  0.8394545
##    17  0.8212914  0.6671096  0.8505455
##    18  0.8217265  0.6666334  0.8390909
##    19  0.8254527  0.6619158  0.8336364
##    20  0.8216730  0.6591251  0.8610909
##    21  0.8222209  0.6558250  0.8650909
##    22  0.8225108  0.6600332  0.8629091
##    23  0.8251989  0.6549391  0.8645455
##    24  0.8230382  0.6506866  0.8665455
##    25  0.8240641  0.6469878  0.8625455
##    26  0.8232931  0.6484053  0.8589091
##    27  0.8256952  0.6441417  0.8603636
##    28  0.8262157  0.6474529  0.8681818
##    29  0.8219205  0.6390033  0.8629091
##    30  0.8224132  0.6408859  0.8783636
##    31  0.8216064  0.6394795  0.8603636
##    32  0.8231450  0.6380842  0.8685455
##    33  0.8203795  0.6357032  0.8667273
##    34  0.8253745  0.6371429  0.8645455
##    35  0.8235853  0.6357254  0.8781818
##    36  0.8239846  0.6361905  0.8667273
##    37  0.8259991  0.6338427  0.8629091
##    38  0.8246698  0.6376190  0.8589091
##    39  0.8238287  0.6310631  0.8680000
##    40  0.8253477  0.6334109  0.8720000
##    41  0.8234704  0.6348394  0.8623636
##    42  0.8203921  0.6334109  0.8605455
##    43  0.8256668  0.6310631  0.8663636
##    44  0.8214383  0.6319934  0.8680000
##    45  0.8258321  0.6296456  0.8740000
##    46  0.8220098  0.6305980  0.8703636
##    47  0.8213537  0.6325028  0.8740000
##    48  0.8206156  0.6301218  0.8618182
##    49  0.8226456  0.6277962  0.8701818
##    50  0.8222551  0.6306202  0.8547273
##    51  0.8214779  0.6296678  0.8701818
##    52  0.8191383  0.6301661  0.8641818
##    53  0.8198349  0.6287708  0.8645455
##    54  0.8216513  0.6329457  0.8661818
##    55  0.8210091  0.6282614  0.8678182
##    56  0.8208513  0.6272757  0.8583636
##    57  0.8189684  0.6296678  0.8640000
##    58  0.8199264  0.6329457  0.8680000
##    59  0.8188570  0.6254596  0.8698182
##    60  0.8202357  0.6297231  0.8658182
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 28.
```

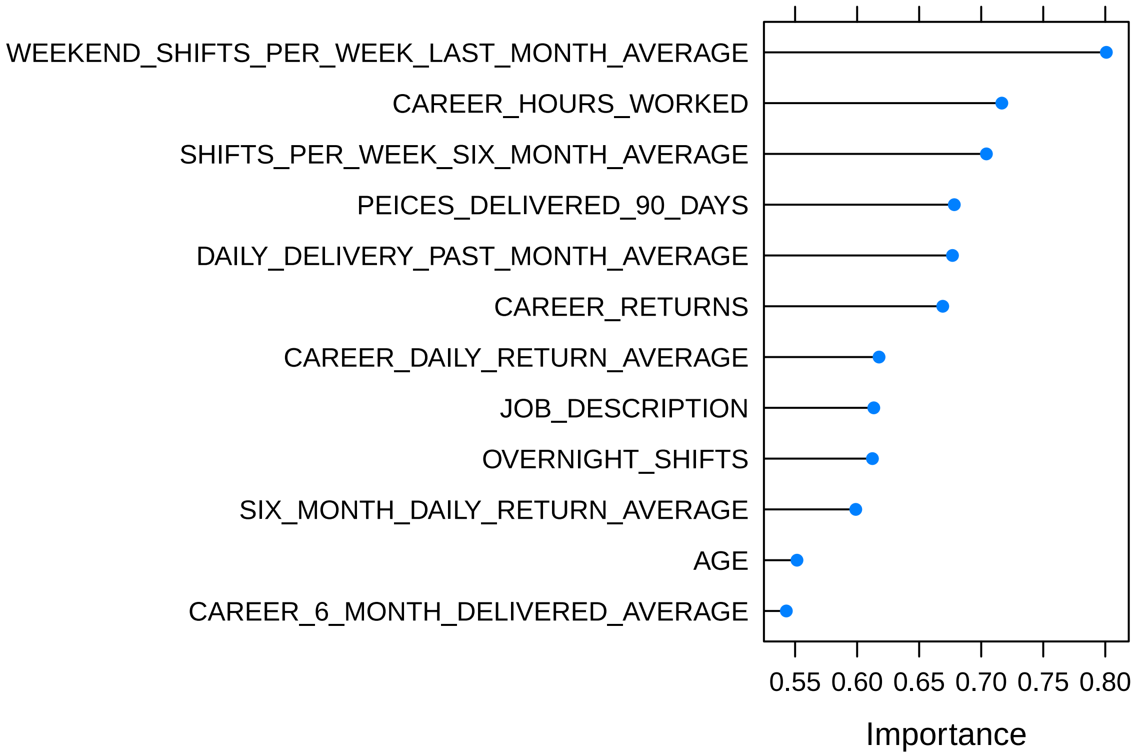
```r
plot(fit)
```



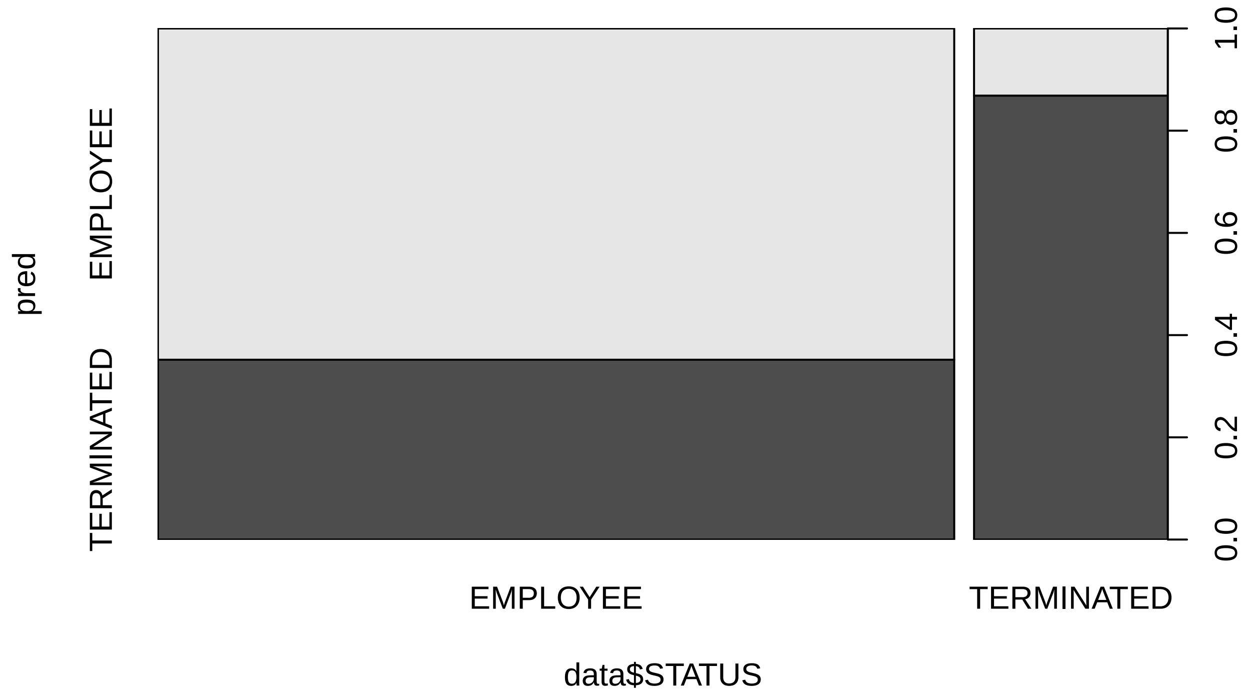

```r
# Loess r-squared variable importance
varImp(fit)
```

```
## ROC curve variable importance
##
##                                             Importance
## WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_AVERAGE     100.000
## CAREER_HOURS_WORKED                             67.333
## SHIFTS_PER_WEEK_SIX_MONTH_AVERAGE               62.519
## PEICES_DELIVERED_90_DAYS                        52.491
## DAILY_DELIVERY_PAST_MONTH_AVERAGE               51.923
## CAREER_RETURNS                                  48.877
## CAREER_DAILY_RETURN_AVERAGE                     28.973
## JOB_DESCRIPTION                                 27.329
## OVERNIGHT_SHIFTS                                26.903
## SIX_MONTH_DAILY_RETURN_AVERAGE                  21.697
## AGE                                              3.324
## CAREER_6_MONTH_DELIVERED_AVERAGE                 0.000
```

```r
# variable importance visualization
importance <- varImp(fit, scale = FALSE)
plot(importance)
```




```r
# Create plot
pred <- predict(fit, newdata = data)
plot(pred ~ data$STATUS)
```




```r
confusionMatrix(pred, data$STATUS)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   EMPLOYEE TERMINATED
##   EMPLOYEE        466         23
##   TERMINATED      253        152
##
##                Accuracy : 0.6913
##                  95% CI : (0.6598, 0.7214)
##     No Information Rate : 0.8043
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3451
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6481
##             Specificity : 0.8686
##          Pos Pred Value : 0.9530
##          Neg Pred Value : 0.3753
##              Prevalence : 0.8043
##          Detection Rate : 0.5213
##    Detection Prevalence : 0.5470
```

```
##       Balanced Accuracy : 0.7583
##
##        'Positive' Class : EMPLOYEE
##
```

Import CSV File: Piecewise Imputed Table

```r
piecewise_df <- read_csv("C:/Users/lewis/Downloads/interpolated_ops_features_piecewise.csv")
```

```
## Rows: 894 Columns: 30
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): DELETE_CODE, JOB_DESCRIPTION
## dbl (28): Unnamed: 0, TECH, AGE, STATUS, WEEKEND_SHIFTS_PER_WEEK_LAST_MONTH_...
##
## 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.
```

```r
data<-piecewise_df
which(is.na(data)) # No need to create a KNN model because this df has NAN values
```

```
##  [1]  5365  7153  8047  8941 10729 12517 19669 19670 19671 19672 19673 20560
## [13] 20561 20562 24139 25033 25927
```