Bank Term Deposit Scheme Predictive Model

Anushree Tomar

7-10-2021

Import Required Libraries

Import Files

First read train and test data sets:-

```
train<-fread("G:\\Anushree G4\\hachthon\\Bank Term Deposit\\Complete-Data-Set\\Training_Dataset_Time_Dep
test<-fread("G:\\Anushree G4\\hachthon\\Bank Term Deposit\\Complete-Data-Set\\Testing_Dataset_Time_Depos

Sample_Submission<-fread("G:\\Anushree G4\\hachthon\\Bank Term Deposit\\Complete-Data-Set\\Sample Submis
```

Binning of Age

In this step we will create groups of the age attribute and label them as 0-4, 5-9, 10-14 and so on.

```
#Binning of Age
label <- c(paste(seq(0, 95, by = 5), seq(0 + 5 - 1, 100 - 1, by = 5),
                 sep = "-"), paste(100, "+", sep = ""))
train$age <- cut(train$age, breaks = c(seq(0, 100, by = 5), Inf), labels = label, right = FALSE)

test$age <- cut(test$age, breaks = c(seq(0, 100, by = 5), Inf), labels = label, right = FALSE)

# Drop duration

#train<-train[,-'duration']
#test<-test[,-'duration']
str(test)
```

```
## Classes 'data.table' and 'data.frame':   37018 obs. of  21 variables:
##  $ key           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age           : Factor w/ 21 levels "0-4","5-9","10-14",..: 12 12 8 9 12 10 12 9 5 6 ...
##  $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
##  $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
##  $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
##  $ default       : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
##  $ housing       : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
##  $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
##  $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
```

```
##  $ month          : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ day_of_week    : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ duration       : int  261 149 226 151 307 198 139 217 380 50 ...
##  $ campaign       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays          : int  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome       : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ emp.var.rate   : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons.price.idx: num  94 94 94 94 94 ...
##  $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m      : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr.employed    : num  5191 5191 5191 5191 5191 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Creating balanced data

```r
load("G:/Anushree G4/hachthon/Bank Term Deposit/Bank Term Deposit Scheme/bal_100.RData")
#balanced_data <-  SMOTE_NC(train[,-'key'], 'y', perc_maj=100)
```

```r
str(balanced_data)
```

```
## Classes 'data.table' and 'data.frame':   7970 obs. of  21 variables:
##  $ age            : Factor w/ 21 levels "0-4","5-9","10-14",..: 10 7 10 9 12 12 8 12 6 13 ...
##  $ job            : Factor w/ 12 levels "admin.","blue-collar",..: 5 1 2 10 10 10 2 10 9 5 ...
##  $ marital        : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 2 ...
##  $ education      : Factor w/ 7 levels "basic.4y","basic.6y",..: 6 3 7 5 1 1 1 1 3 6 ...
##  $ default        : Factor w/ 2 levels "no","unknown": 1 1 2 1 2 2 1 2 2 1 ...
##  $ housing        : Factor w/ 3 levels "no","unknown",..: 3 1 1 1 1 1 1 1 3 1 ...
##  $ loan           : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 3 1 1 1 ...
##  $ contact        : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
##  $ month          : Factor w/ 4 levels "aug","jun","may",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ day_of_week    : Factor w/ 5 levels "fri","mon","thu",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ duration       : num  140 175 136 1623 50 ...
##  $ campaign       : num  1 1 1 1 1 1 1 1 1 2 ...
##  $ pdays          : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome       : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ emp.var.rate   : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons.price.idx: num  94 94 94 94 94 ...
##  $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m      : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr.employed    : num  5191 5191 5191 5191 5191 ...
##  $ y              : Factor w/ 2 levels "no","yes": 1 1 1 2 1 1 1 1 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Re-Leveling of Attributes

We found new levels in the test data so that to solve our problem we releveled the train data as the test dataset.

```r
levels(balanced_data$education)<-levels(test$education)
levels(balanced_data$default)<-levels(test$default)
levels(balanced_data$month)<-levels(test$month)
```

Data Pre-processing for Catboost model

Encoding categorical variables

```r
cat_data<-balanced_data[, lapply(balanced_data, class) == 'factor', with = FALSE]
cont_data<-balanced_data[,lapply(balanced_data, class) != 'factor', with = FALSE]
cat_col <- colnames(cat_data)


encode <- sapply(cat_data, function(x) LabelEncoder.fit(x))
for (i in cat_col){
    cat_data[[i]] <- transform(encode[[i]], balanced_data[[i]])
}

cat_data <- cbind(cat_data, cont_data)
```

```r
str(cat_data)
```

```
## Classes 'data.table' and 'data.frame':   7970 obs. of  21 variables:
##  $ age           : int  6 3 6 5 8 8 4 8 2 9 ...
##  $ job           : int  5 1 2 10 10 10 2 10 9 5 ...
##  $ marital       : int  2 2 2 2 2 2 2 2 3 2 ...
##  $ education     : int  6 3 7 5 1 1 1 1 3 6 ...
##  $ default       : int  1 1 2 1 2 2 1 2 2 1 ...
##  $ housing       : int  3 1 1 1 1 1 1 1 3 1 ...
##  $ loan          : int  1 1 1 1 3 1 3 1 1 1 ...
##  $ contact       : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ month         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ day_of_week   : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ poutcome      : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ y             : int  1 1 1 2 1 1 1 1 1 1 ...
##  $ duration      : num  140 175 136 1623 50 ...
##  $ campaign      : num  1 1 1 1 1 1 1 1 1 2 ...
##  $ pdays         : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons.price.idx: num  94 94 94 94 94 ...
##  $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr.employed   : num  5191 5191 5191 5191 5191 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Data Partition

Let's partitioned the 70% of data into traindata and rest into the testdata.

```r
#cat_data <- cbind(cat_data, target)
trainindex<-createDataPartition(cat_data$y,p=0.7,list=F)
traindata<-cat_data[trainindex,]
testdata<-cat_data[-trainindex,]
```

Create train/test pools from train(balanced data) data

```r
y_train <- traindata[,"y"]
X_train <- traindata[,-'y']

y_test <- testdata[,"y"]
X_test <- testdata[,-"y"]

train_pool <- catboost.load_pool(data = X_train, label = y_train)
test_pool <- catboost.load_pool(data = X_test, label = y_test)
```

Build Catboost Model

Used overfitting detector for more faster training

```r
params_simple <- list(iterations = 500,
                      learning_rate=0.001,
                      depth=4,
                      loss_function = 'Logloss',
                      eval_metric='Logloss',
                      random_seed = 55,
                      od_type='Iter',
                      metric_period = 50,
                      od_wait=30,
                      use_best_model=TRUE,
                      logging_level = 'Silent')


model_simple <- catboost.train(train_pool, test_pool, params_simple)
```

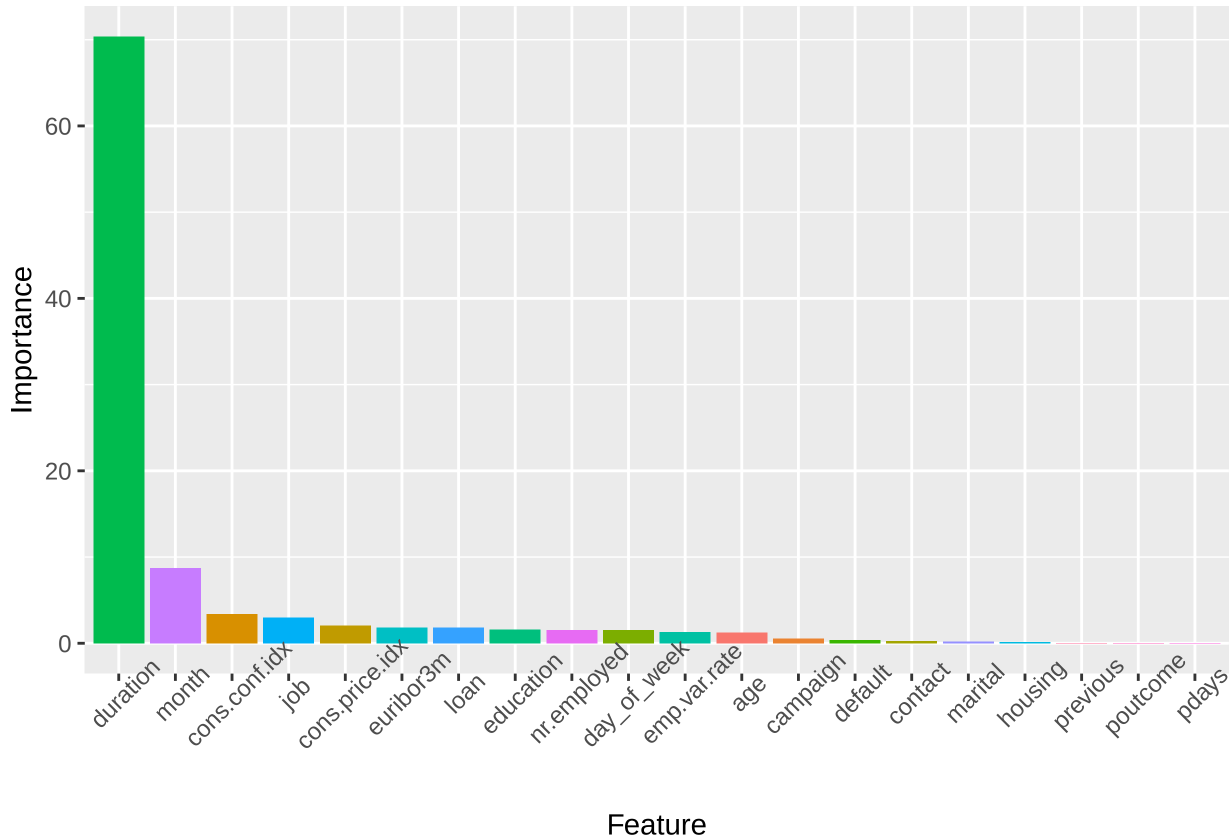
```
## Warning: Overfitting detector is active, thus evaluation metric is calculated on every iteration. 'me
```

```r
# Load Saveed model
#model_simple <- catboost::catboost.load_model("model")
cat('Model with Tuned Parameter tree count: ', model_simple$tree_count, '\n')
```

```
## Model with Tuned Parameter tree count:  500
```

Visualize important features

```
feat_imp<-catboost.get_feature_importance(model_simple)
feat_imp<-data.frame('Feature' = rownames(feat_imp), 'Importance' =feat_imp[,1])
feat_imp<-feat_imp[order(feat_imp$Importance,decreasing = T),]
ggplot(feat_imp, aes(x = Feature, y = Importance,fill=Feature)) +geom_bar(stat='identity') +
theme(axis.text.x= element_text(angle = 45)) +scale_x_discrete(limits = feat_imp$Feature)+theme(legend.p
```



Prediction on testdata

Confusion Matrix

```
preds <- catboost.predict(model_simple, test_pool,prediction_type = 'Class')
# In train and test data one hot code as 1,2
y_test<-ifelse(y_test==1,"0","1")
#table(preds, testdata[,y])
confusionMatrix(factor(y_test),factor(preds),mode="everything",positive = "1")
```

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

```
## Prediction    0    1
##          0 1095  100
##          1   45 1150
##
##                Accuracy : 0.9393
##                  95% CI : (0.929, 0.9486)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8787
##
##  Mcnemar's Test P-Value : 7.31e-06
##
##             Sensitivity : 0.9200
##             Specificity : 0.9605
##          Pos Pred Value : 0.9623
##          Neg Pred Value : 0.9163
##               Precision : 0.9623
##                  Recall : 0.9200
##                      F1 : 0.9407
##              Prevalence : 0.5230
##          Detection Rate : 0.4812
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9403
##
##        'Positive' Class : 1
##
```

Encoding categorical variables of final test data

```
cat_data<-test[, lapply(test, class) == 'factor', with = FALSE]
cont_data<-test[,lapply(test, class) != 'factor', with = FALSE]
cont_data<-cont_data[,-'key']
cat_col <- colnames(cat_data)


encode <- sapply(cat_data, function(x) LabelEncoder.fit(x))
for (i in cat_col){
    cat_data[[i]] <- transform(encode[[i]], test[[i]])
}

cat_data <- cbind(cat_data, cont_data)

final_test<-catboost.load_pool(cat_data)
```

Prediction on final test

```
preds <- catboost.predict(model_simple, final_test,prediction_type = 'Class')
#preds<-ifelse(preds==0,"no","yes")
```

```r
submission1<-data.frame("key"=Sample_Submission$key,"y"=preds)
#write.csv(submission,"catboost4.9lrleveltuned_binage.csv",row.names = F,quote = F)

#output<-data.frame("Modelname"=c("catboost1","catboost2_no_smote","catboost3_binage","catboost4tuned_b

#write.csv(output,"models_score.csv",row.names = F,quote = F)
```

save model

```r
#catboost.save_model(model_simple, "model")


#model_simple <- catboost::catboost.load_model("model")
```

Test of loaded model

```r
#preds <- catboost.predict(model2, test_pool,prediction_type = 'Class')
#preds<-ifelse(preds==0,"1","2")
#confusionMatrix(factor(testdata[,y]),factor(preds))
```

Decision Tree Model

Data partitioning

```r
trainindex<-createDataPartition(balanced_data$y,p=0.7,list=F)
traindata1<-balanced_data[trainindex,]
testdata1<-balanced_data[-trainindex,]
```

```r
data_tree <- readRDS("./data_tree_relevel4.rds")
#data_tree <- rpart(y~., method = "class", data = traindata1)
summary(data_tree)
```

```
## Call:
## rpart(formula = y ~ ., data = traindata1, method = "class", control = rpart.control(minsplit = 3,
##     cp = 0.01))
##   n= 5580
##
##           CP nsplit rel error    xerror        xstd
## 1 0.86200717      0 1.0000000 1.0308244 0.013380628
## 2 0.01666667      1 0.1379928 0.1422939 0.006882791
## 3 0.01000000      3 0.1046595 0.1150538 0.006234230
##
## Variable importance
```

```
##     duration        month  day_of_week emp.var.rate  nr.employed    euribor3m
##           49           10            9            9            9            9
##    education          age          job      default
##            2            1            1            1
##
## Node number 1: 5580 observations,    complexity param=0.8620072
##   predicted class=no   expected loss=0.5  P(node) =1
##     class counts:  2790  2790
##    probabilities: 0.500 0.500
##   left son=2 (2567 obs) right son=3 (3013 obs)
##   Primary splits:
##       duration     < 412.0844 to the left,  improve=2086.4570, (0 missing)
##       month        splits as  RRLL------,   improve= 300.2673, (0 missing)
##       emp.var.rate < 1.100859 to the left,  improve= 254.4952, (0 missing)
##       nr.employed  < 5195.852 to the left,  improve= 248.9323, (0 missing)
##       euribor3m    < 4.859128 to the left,  improve= 248.0139, (0 missing)
##   Surrogate splits:
##       month        splits as  RRLL------,   agree=0.636, adj=0.209, (0 split)
##       emp.var.rate < 1.100859 to the left,  agree=0.628, adj=0.191, (0 split)
##       nr.employed  < 5195.852 to the left,  agree=0.627, adj=0.189, (0 split)
##       euribor3m    < 4.859128 to the left,  agree=0.627, adj=0.188, (0 split)
##       day_of_week  splits as  LLRRR,        agree=0.623, adj=0.180, (0 split)
##
## Node number 2: 2567 observations
##   predicted class=no   expected loss=0.03155434  P(node) =0.4600358
##     class counts:  2486    81
##    probabilities: 0.968 0.032
##
## Node number 3: 3013 observations,    complexity param=0.01666667
##   predicted class=yes  expected loss=0.1008961  P(node) =0.5399642
##     class counts:   304  2709
##    probabilities: 0.101 0.899
##   left son=6 (430 obs) right son=7 (2583 obs)
##   Primary splits:
##       duration     < 536.1943 to the left,  improve=51.69713, (0 missing)
##       month        splits as  RRLL------, improve=35.78471, (0 missing)
##       nr.employed  < 5191.048 to the left,  improve=33.20063, (0 missing)
##       age          splits as  ----LLRRRRLLL--------, improve=29.29166, (0 missing)
##       emp.var.rate < 1.100859 to the left,  improve=26.63044, (0 missing)
##   Surrogate splits:
##       marital splits as  RRRL, agree=0.858, adj=0.005, (0 split)
##
## Node number 6: 430 observations,    complexity param=0.01666667
##   predicted class=yes  expected loss=0.327907  P(node) =0.07706093
##     class counts:   141   289
##    probabilities: 0.328 0.672
##   left son=12 (99 obs) right son=13 (331 obs)
##   Primary splits:
##       education   splits as  LLLLRRL-, improve=105.94770, (0 missing)
##       age         splits as  ----LLRRRLLLL--------, improve= 77.83215, (0 missing)
##       month       splits as  RLLL------, improve= 60.43869, (0 missing)
##       day_of_week splits as  LRRRL, improve= 51.76906, (0 missing)
##       job         splits as  RLRRRLLL-RL-, improve= 50.93716, (0 missing)
##   Surrogate splits:
```

```
##       age         splits as  ----LLRRRLLLR--------, agree=0.853, adj=0.364, (0 split)
##       job         splits as  RLRRRLLL-RR-, agree=0.849, adj=0.343, (0 split)
##       default     splits as  RL-, agree=0.840, adj=0.303, (0 split)
##       day_of_week splits as  LRRRL, agree=0.821, adj=0.222, (0 split)
##       month       splits as  RRLL------, agree=0.812, adj=0.182, (0 split)
##
## Node number 7: 2583 observations
##   predicted class=yes  expected loss=0.06310492  P(node) =0.4629032
##     class counts:   163  2420
##    probabilities: 0.063 0.937
##
## Node number 12: 99 observations
##   predicted class=no   expected loss=0.03030303  P(node) =0.01774194
##     class counts:    96     3
##    probabilities: 0.970 0.030
##
## Node number 13: 331 observations
##   predicted class=yes  expected loss=0.1359517  P(node) =0.059319
##     class counts:    45   286
##    probabilities: 0.136 0.864
```

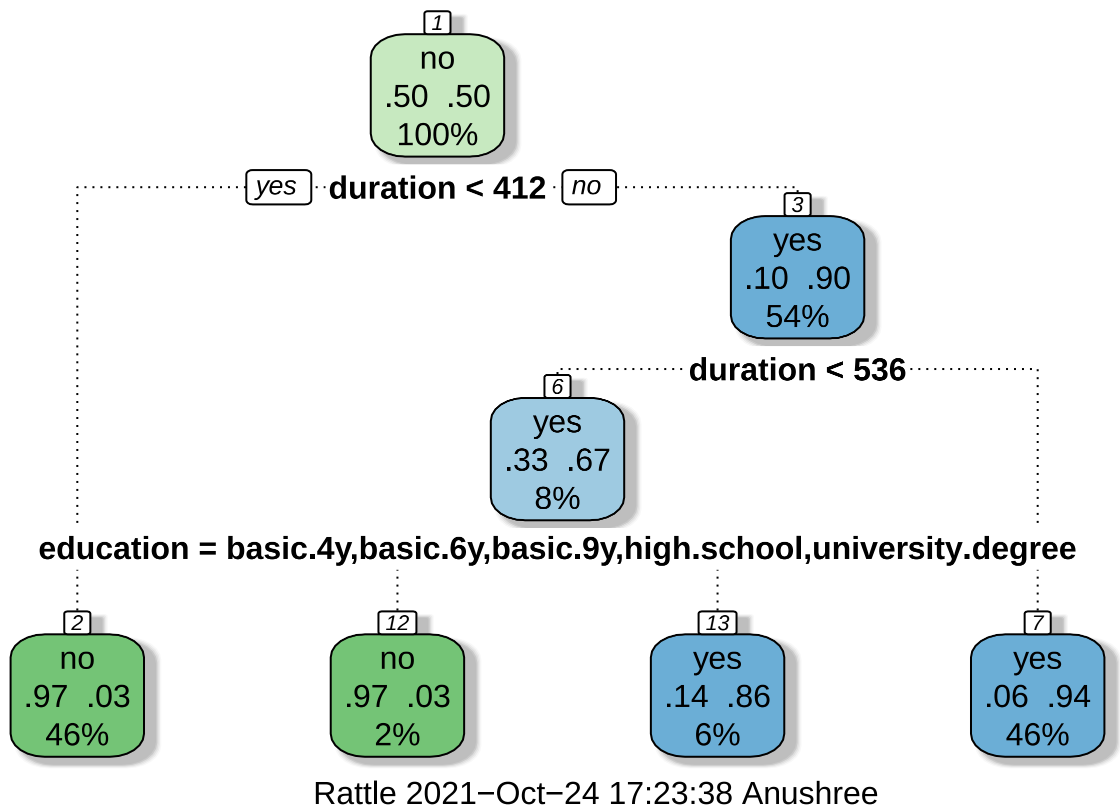
```r
# Find cp value of min xerror
#data_tree <- rpart(y~., method = "class", data = traindata1,control = rpart.control(minsplit=3,cp=.01))

summary(data_tree)
```

```
## Call:
## rpart(formula = y ~ ., data = traindata1, method = "class", control = rpart.control(minsplit = 3,
##     cp = 0.01))
##   n= 5580
##
##           CP nsplit rel error    xerror        xstd
## 1 0.86200717      0 1.0000000 1.0308244 0.013380628
## 2 0.01666667      1 0.1379928 0.1422939 0.006882791
## 3 0.01000000      3 0.1046595 0.1150538 0.006234230
##
## Variable importance
##     duration        month  day_of_week emp.var.rate  nr.employed    euribor3m
##           49           10            9            9            9            9
##    education          age          job      default
##            2            1            1            1
##
## Node number 1: 5580 observations,    complexity param=0.8620072
##   predicted class=no   expected loss=0.5  P(node) =1
##     class counts:  2790  2790
##    probabilities: 0.500 0.500
##   left son=2 (2567 obs) right son=3 (3013 obs)
##   Primary splits:
##       duration     < 412.0844 to the left,  improve=2086.4570, (0 missing)
##       month        splits as  RRLL------,   improve= 300.2673, (0 missing)
##       emp.var.rate < 1.100859 to the left,  improve= 254.4952, (0 missing)
##       nr.employed  < 5195.852 to the left,  improve= 248.9323, (0 missing)
##       euribor3m    < 4.859128 to the left,  improve= 248.0139, (0 missing)
##   Surrogate splits:
```

```
##       month        splits as  RRLL------,   agree=0.636, adj=0.209, (0 split)
##       emp.var.rate < 1.100859 to the left,  agree=0.628, adj=0.191, (0 split)
##       nr.employed  < 5195.852 to the left,  agree=0.627, adj=0.189, (0 split)
##       euribor3m    < 4.859128 to the left,  agree=0.627, adj=0.188, (0 split)
##       day_of_week  splits as  LLRRR,        agree=0.623, adj=0.180, (0 split)
##
## Node number 2: 2567 observations
##   predicted class=no   expected loss=0.03155434  P(node) =0.4600358
##     class counts:  2486    81
##    probabilities: 0.968 0.032
##
## Node number 3: 3013 observations,    complexity param=0.01666667
##   predicted class=yes  expected loss=0.1008961  P(node) =0.5399642
##     class counts:   304  2709
##    probabilities: 0.101 0.899
##   left son=6 (430 obs) right son=7 (2583 obs)
##   Primary splits:
##       duration     < 536.1943 to the left,  improve=51.69713, (0 missing)
##       month        splits as  RRLL------, improve=35.78471, (0 missing)
##       nr.employed  < 5191.048 to the left,  improve=33.20063, (0 missing)
##       age          splits as  ----LLRRRRLLL--------, improve=29.29166, (0 missing)
##       emp.var.rate < 1.100859 to the left,  improve=26.63044, (0 missing)
##   Surrogate splits:
##       marital splits as  RRRL, agree=0.858, adj=0.005, (0 split)
##
## Node number 6: 430 observations,    complexity param=0.01666667
##   predicted class=yes  expected loss=0.327907  P(node) =0.07706093
##     class counts:   141   289
##    probabilities: 0.328 0.672
##   left son=12 (99 obs) right son=13 (331 obs)
##   Primary splits:
##       education   splits as  LLLLRRL-, improve=105.94770, (0 missing)
##       age         splits as  ----LLRRRLLLL--------, improve= 77.83215, (0 missing)
##       month       splits as  RLLL------, improve= 60.43869, (0 missing)
##       day_of_week splits as  LRRRL, improve= 51.76906, (0 missing)
##       job         splits as  RLRRRLLL-RL-, improve= 50.93716, (0 missing)
##   Surrogate splits:
##       age         splits as  ----LLRRRLLLR--------, agree=0.853, adj=0.364, (0 split)
##       job         splits as  RLRRRLLL-RR-, agree=0.849, adj=0.343, (0 split)
##       default     splits as  RL-, agree=0.840, adj=0.303, (0 split)
##       day_of_week splits as  LRRRL, agree=0.821, adj=0.222, (0 split)
##       month       splits as  RRLL------, agree=0.812, adj=0.182, (0 split)
##
## Node number 7: 2583 observations
##   predicted class=yes  expected loss=0.06310492  P(node) =0.4629032
##     class counts:   163  2420
##    probabilities: 0.063 0.937
##
## Node number 12: 99 observations
##   predicted class=no   expected loss=0.03030303  P(node) =0.01774194
##     class counts:    96     3
##    probabilities: 0.970 0.030
##
## Node number 13: 331 observations
```

```
##    predicted class=yes  expected loss=0.1359517  P(node) =0.059319
##      class counts:    45   286
##     probabilities: 0.136 0.864
```

Decision Tree



Cp results

```
##
## Classification tree:
## rpart(formula = y ~ ., data = traindata1, method = "class", control = rpart.control(minsplit = 3,
##     cp = 0.01))
##
## Variables actually used in tree construction:
## [1] duration  education
##
## Root node error: 2790/5580 = 0.5
##
## n= 5580
##
##          CP nsplit rel error  xerror      xstd
## 1 0.862007      0   1.00000 1.03082 0.0133806
## 2 0.016667      1   0.13799 0.14229 0.0068828
## 3 0.010000      3   0.10466 0.11505 0.0062342
```

Making Prediction on test data

```
## class_final
##   no  yes
## 1144 1246
```

Confusion Matrix

```
## Confusion Matrix and Statistics
##
##
##         no  yes
##   no  1107   37
##   yes   88 1158
##
##                Accuracy : 0.9477
##                  95% CI : (0.938, 0.9563)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8954
##
##  Mcnemar's Test P-Value : 7.744e-06
##
##             Sensitivity : 0.9690
##             Specificity : 0.9264
##          Pos Pred Value : 0.9294
##          Neg Pred Value : 0.9677
##               Precision : 0.9294
##                  Recall : 0.9690
##                      F1 : 0.9488
##              Prevalence : 0.5000
##          Detection Rate : 0.4845
##    Detection Prevalence : 0.5213
##       Balanced Accuracy : 0.9477
##
##        'Positive' Class : yes
##
```

Making Prediction on final test data

Save and Load model

```
# save the model to disk
#saveRDS(data_tree, "./data_tree_relevel4.rds")




# load the model
```

```r
#super_model <- readRDS("./data_tree_relevel4.rds")
#prob_final <- predict(super_model,test[,-"key"])
#class_final <- data.frame("y"=predict(super_model,test[,-"key"],type="class"))
#table(class_final)
```

XgBoost Model

```
## [17:23:40] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evalu
## [17:23:41] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evalu
## [17:23:41] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evalu
## [17:23:41] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evalu
## [17:23:41] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evalu
## [1]  train-logloss:0.418079+0.018458 test-logloss:0.422120+0.019449
## Multiple eval metrics are present. Will use test_logloss for early stopping.
## Will train until test_logloss hasn't improved in 20 rounds.
##
## [11] train-logloss:0.056472+0.006527 test-logloss:0.070557+0.004554
## [21] train-logloss:0.038167+0.001534 test-logloss:0.056398+0.005190
## [31] train-logloss:0.036333+0.001503 test-logloss:0.055755+0.005637
## [41] train-logloss:0.035760+0.001048 test-logloss:0.055741+0.005873
## [51] train-logloss:0.035378+0.001016 test-logloss:0.055712+0.005888
## [61] train-logloss:0.034698+0.001870 test-logloss:0.055569+0.005749
## [71] train-logloss:0.034371+0.001843 test-logloss:0.055346+0.005945
## [75] train-logloss:0.034370+0.001843 test-logloss:0.055340+0.005952
```

```
## [1] 75
```

```
## ##### xgb.cv 5-folds
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
##     1          0.4180788       0.018457512         0.4221200      0.019448988
##     2          0.2916370       0.021070474         0.2985952      0.022684681
##     3          0.2143626       0.026835703         0.2224412      0.025062821
##     4          0.1626732       0.013509762         0.1722442      0.012489204
##     5          0.1343722       0.007387992         0.1450620      0.013268746
##     6          0.1086284       0.010382068         0.1188212      0.009598144
##     7          0.0855150       0.005633857         0.0964198      0.005815448
##     8          0.0760960       0.005814306         0.0881712      0.007515437
##     9          0.0665510       0.004857613         0.0786544      0.005473432
##    10          0.0623734       0.006723171         0.0751198      0.004349430
##    11          0.0564720       0.006527203         0.0705570      0.004553935
##    12          0.0534416       0.004809190         0.0677660      0.005075447
##    13          0.0502830       0.004847588         0.0655400      0.005145742
##    14          0.0459556       0.003189333         0.0615736      0.005196891
##    15          0.0446756       0.003726848         0.0608396      0.005163835
##    16          0.0424450       0.002576060         0.0594428      0.005426075
##    17          0.0415548       0.002818960         0.0588390      0.005784232
##    18          0.0404828       0.002117312         0.0582436      0.005644077
##    19          0.0394028       0.001566536         0.0571430      0.005253231
##    20          0.0387124       0.001222343         0.0569520      0.005214819
##    21          0.0381674       0.001534349         0.0563978      0.005190462
##    22          0.0377178       0.002029868         0.0563312      0.005375615
```

```
##    23        0.0373722       0.001898726       0.0563822      0.005347247
##    24        0.0370504       0.001640447       0.0562890      0.005395766
##    25        0.0369106       0.001632074       0.0562672      0.005411844
##    26        0.0369106       0.001632074       0.0562676      0.005413336
##    27        0.0368210       0.001506866       0.0562946      0.005441494
##    28        0.0364784       0.001602297       0.0559482      0.005537295
##    29        0.0364784       0.001602297       0.0559488      0.005538337
##    30        0.0364784       0.001602297       0.0559494      0.005539075
##    31        0.0363330       0.001503052       0.0557554      0.005637273
##    32        0.0357596       0.001047767       0.0557492      0.005874420
##    33        0.0357596       0.001047767       0.0557460      0.005874271
##    34        0.0357596       0.001047767       0.0557440      0.005873898
##    35        0.0357596       0.001047767       0.0557430      0.005873541
##    36        0.0357596       0.001047767       0.0557424      0.005873601
##    37        0.0357596       0.001047767       0.0557420      0.005873527
##    38        0.0357596       0.001047767       0.0557416      0.005873453
##    39        0.0357596       0.001047767       0.0557416      0.005873453
##    40        0.0357596       0.001047767       0.0557414      0.005873244
##    41        0.0357596       0.001047767       0.0557414      0.005873244
##    42        0.0357596       0.001047767       0.0557414      0.005873244
##    43        0.0357596       0.001047767       0.0557414      0.005873244
##    44        0.0355580       0.001150271       0.0559216      0.005762107
##    45        0.0355578       0.001150438       0.0559164      0.005765025
##    46        0.0355578       0.001150438       0.0559136      0.005766609
##    47        0.0355578       0.001150438       0.0559116      0.005767736
##    48        0.0355578       0.001150438       0.0559106      0.005768303
##    49        0.0355578       0.001150438       0.0559102      0.005768535
##    50        0.0355578       0.001150438       0.0559098      0.005768762
##    51        0.0353776       0.001015527       0.0557124      0.005888206
##    52        0.0353776       0.001015527       0.0557112      0.005888977
##    53        0.0352670       0.001083172       0.0556922      0.005911216
##    54        0.0352670       0.001083172       0.0556920      0.005911105
##    55        0.0352670       0.001083172       0.0556920      0.005910998
##    56        0.0352670       0.001083172       0.0556922      0.005910759
##    57        0.0350256       0.001357857       0.0555556      0.005732425
##    58        0.0350256       0.001357857       0.0555528      0.005728850
##    59        0.0350256       0.001357857       0.0555510      0.005726553
##    60        0.0346984       0.001869970       0.0555678      0.005748039
##    61        0.0346984       0.001869970       0.0555686      0.005749065
##    62        0.0346984       0.001869970       0.0555692      0.005749835
##    63        0.0346984       0.001869970       0.0555694      0.005750092
##    64        0.0346984       0.001869970       0.0555696      0.005750348
##    65        0.0346984       0.001869970       0.0555698      0.005750605
##    66        0.0346984       0.001869970       0.0555698      0.005750605
##    67        0.0346984       0.001869970       0.0555698      0.005750605
##    68        0.0346984       0.001869970       0.0555698      0.005750605
##    69        0.0345388       0.001799475       0.0554272      0.005850830
##    70        0.0343708       0.001842905       0.0553510      0.005939781
##    71        0.0343706       0.001842992       0.0553460      0.005945192
##    72        0.0343702       0.001843009       0.0553432      0.005948164
##    73        0.0343702       0.001843009       0.0553414      0.005950129
##    74        0.0343702       0.001843009       0.0553404      0.005951231
##    75        0.0343702       0.001843009       0.0553398      0.005951855
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
```

```
## Best iteration:
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
##    75          0.0343702       0.001843009         0.0553398      0.005951855

## ##### xgb.cv 5-folds
## call:
##   xgb.cv(params = params, data = dtrain, nrounds = 75, nfold = 5,
##     showsd = T, stratified = T, print_every_n = 10, early_stopping_rounds = 20,
##     maximize = F, gamma = 4)
## params (as set within xgb.cv):
##   booster = "gbtree", objective = "binary:logistic", eta = "0.4", max_depth = "15", min_child_weight
## callbacks:
##   cb.print.evaluation(period = print_every_n, showsd = showsd)
##   cb.evaluation.log()
##   cb.early.stop(stopping_rounds = early_stopping_rounds, maximize = maximize,
##     verbose = verbose)
## niter: 75
## best_iteration: 75
## best_ntreelimit: 75
## evaluation_log:
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
##     1          0.4180788       0.018457512         0.4221200      0.019448988
##     2          0.2916370       0.021070474         0.2985952      0.022684681
##     3          0.2143626       0.026835703         0.2224412      0.025062821
##     4          0.1626732       0.013509762         0.1722442      0.012489204
##     5          0.1343722       0.007387992         0.1450620      0.013268746
##     6          0.1086284       0.010382068         0.1188212      0.009598144
##     7          0.0855150       0.005633857         0.0964198      0.005815448
##     8          0.0760960       0.005814306         0.0881712      0.007515437
##     9          0.0665510       0.004857613         0.0786544      0.005473432
##    10          0.0623734       0.006723171         0.0751198      0.004349430
##    11          0.0564720       0.006527203         0.0705570      0.004553935
##    12          0.0534416       0.004809190         0.0677660      0.005075447
##    13          0.0502830       0.004847588         0.0655400      0.005145742
##    14          0.0459556       0.003189333         0.0615736      0.005196891
##    15          0.0446756       0.003726848         0.0608396      0.005163835
##    16          0.0424450       0.002576060         0.0594428      0.005426075
##    17          0.0415548       0.002818960         0.0588390      0.005784232
##    18          0.0404828       0.002117312         0.0582436      0.005644077
##    19          0.0394028       0.001566536         0.0571430      0.005253231
##    20          0.0387124       0.001222343         0.0569520      0.005214819
##    21          0.0381674       0.001534349         0.0563978      0.005190462
##    22          0.0377178       0.002029868         0.0563312      0.005375615
##    23          0.0373722       0.001898726         0.0563822      0.005347247
##    24          0.0370504       0.001640447         0.0562890      0.005395766
##    25          0.0369106       0.001632074         0.0562672      0.005411844
##    26          0.0369106       0.001632074         0.0562676      0.005413336
##    27          0.0368210       0.001506866         0.0562946      0.005441494
##    28          0.0364784       0.001602297         0.0559482      0.005537295
##    29          0.0364784       0.001602297         0.0559488      0.005538337
##    30          0.0364784       0.001602297         0.0559494      0.005539075
##    31          0.0363330       0.001503052         0.0557554      0.005637273
##    32          0.0357596       0.001047767         0.0557492      0.005874420
##    33          0.0357596       0.001047767         0.0557460      0.005874271
```

```
##    34           0.0357596       0.001047767          0.0557440       0.005873898
##    35           0.0357596       0.001047767          0.0557430       0.005873541
##    36           0.0357596       0.001047767          0.0557424       0.005873601
##    37           0.0357596       0.001047767          0.0557420       0.005873527
##    38           0.0357596       0.001047767          0.0557416       0.005873453
##    39           0.0357596       0.001047767          0.0557416       0.005873453
##    40           0.0357596       0.001047767          0.0557414       0.005873244
##    41           0.0357596       0.001047767          0.0557414       0.005873244
##    42           0.0357596       0.001047767          0.0557414       0.005873244
##    43           0.0357596       0.001047767          0.0557414       0.005873244
##    44           0.0355580       0.001150271          0.0559216       0.005762107
##    45           0.0355578       0.001150438          0.0559164       0.005765025
##    46           0.0355578       0.001150438          0.0559136       0.005766609
##    47           0.0355578       0.001150438          0.0559116       0.005767736
##    48           0.0355578       0.001150438          0.0559106       0.005768303
##    49           0.0355578       0.001150438          0.0559102       0.005768535
##    50           0.0355578       0.001150438          0.0559098       0.005768762
##    51           0.0353776       0.001015527          0.0557124       0.005888206
##    52           0.0353776       0.001015527          0.0557112       0.005888977
##    53           0.0352670       0.001083172          0.0556922       0.005911216
##    54           0.0352670       0.001083172          0.0556920       0.005911105
##    55           0.0352670       0.001083172          0.0556920       0.005910998
##    56           0.0352670       0.001083172          0.0556922       0.005910759
##    57           0.0350256       0.001357857          0.0555556       0.005732425
##    58           0.0350256       0.001357857          0.0555528       0.005728850
##    59           0.0350256       0.001357857          0.0555510       0.005726553
##    60           0.0346984       0.001869970          0.0555678       0.005748039
##    61           0.0346984       0.001869970          0.0555686       0.005749065
##    62           0.0346984       0.001869970          0.0555692       0.005749835
##    63           0.0346984       0.001869970          0.0555694       0.005750092
##    64           0.0346984       0.001869970          0.0555696       0.005750348
##    65           0.0346984       0.001869970          0.0555698       0.005750605
##    66           0.0346984       0.001869970          0.0555698       0.005750605
##    67           0.0346984       0.001869970          0.0555698       0.005750605
##    68           0.0346984       0.001869970          0.0555698       0.005750605
##    69           0.0345388       0.001799475          0.0554272       0.005850830
##    70           0.0343708       0.001842905          0.0553510       0.005939781
##    71           0.0343706       0.001842992          0.0553460       0.005945192
##    72           0.0343702       0.001843009          0.0553432       0.005948164
##    73           0.0343702       0.001843009          0.0553414       0.005950129
##    74           0.0343702       0.001843009          0.0553404       0.005951231
##    75           0.0343702       0.001843009          0.0553398       0.005951855
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
## Best iteration:
##  iter train_logloss_mean train_logloss_std test_logloss_mean test_logloss_std
##    75           0.0343702       0.001843009          0.0553398       0.005951855
```

Prediction on test data

```r
#model prediction
 xgbpred <- predict (xgb1,dtest)
```

```r
 xgbpred <- ifelse (xgbpred > 0.5,"1","0")
xgbpred<-data.frame("y"= xgbpred)

  xgbpredfinal <- predict(xgb1,dtestfinal)
  xgbpredfinal <- ifelse(xgbpredfinal > 0.5,"1","0")
  xgbpredfinal<-data.frame("y"= xgbpredfinal)
  Xgboostpred<-cbind(test[,"key"], xgbpredfinal)
  submission3<-Xgboostpred
# write.csv(Xgboostpred,"XGboost_new2.csv",row.names = FALSE,quote = F)

 #saveRDS(xgb1, "XGboost_new2_mod.rds")
```

Confusion Matrix

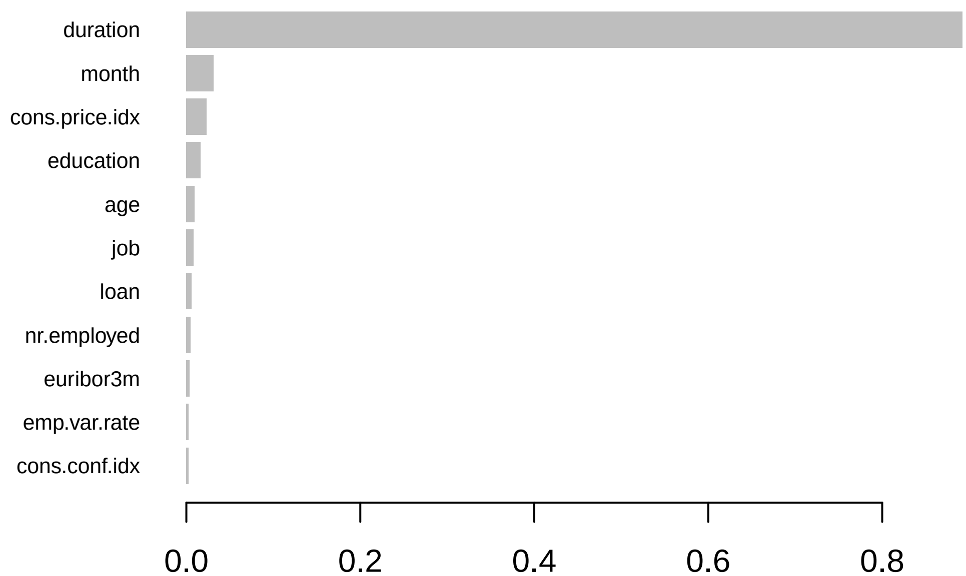
```r
confusionMatrix(factor(xgbpred$y),factor(testdata$y),positive = "1",mode="everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1136    9
##          1   59 1186
##
##                Accuracy : 0.9715
##                  95% CI : (0.9641, 0.9778)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9431
##
##  Mcnemar's Test P-Value : 2.814e-09
##
##             Sensitivity : 0.9925
##             Specificity : 0.9506
##          Pos Pred Value : 0.9526
##          Neg Pred Value : 0.9921
##               Precision : 0.9526
##                  Recall : 0.9925
##                      F1 : 0.9721
##              Prevalence : 0.5000
##          Detection Rate : 0.4962
##    Detection Prevalence : 0.5209
##       Balanced Accuracy : 0.9715
##
##        'Positive' Class : 1
##
```

```r
importance <- xgb.importance(feature_names = colnames(dtrain), model = xgb1)
head(importance)
```

```
##            Feature        Gain       Cover  Frequency
## 1:        duration 0.892141707 0.671911644 0.42307692
## 2:           month 0.031306892 0.134298142 0.19230769
## 3: cons.price.idx 0.023098346 0.044538876 0.09615385
## 4:       education 0.016275224 0.004399767 0.01923077
## 5:             age 0.009422156 0.010373028 0.03846154
## 6:             job 0.008071452 0.030988795 0.05769231
```

```r
xgb.plot.importance(importance_matrix = importance)
```



Final Model

We selected Decision Tree Model as our Final model based on the leaderboard score.