Credit Data Analysis with ML Algorithm

seunghun_Oh

2022-08-13

1) Artificial Neuralnet

1.1) Read Data

```
data_1 = read.csv("C:\\Credit\\train_Processed_8.csv", header = T, stringsAsFactors = T) # 파일 읽기
data_1$Credit_Score <- as.factor(data_1$Credit_Score) # 명목형변수
colnames(data_1) # 변수명 확인
```

```
##  [1] "Number"                   "Month"
##  [3] "Age"                      "Annual_Income"
##  [5] "Monthly_Inhand_Salary"    "Num_Bank_Accounts"
##  [7] "Num_Credit_Card"          "Interest_Rate"
##  [9] "Num_of_Loan"              "Delay_from_due_date"
## [11] "Num_of_Delayed_Payment"   "Changed_Credit_Limit"
## [13] "Num_Credit_Inquiries"     "Credit_Mix"
## [15] "Outstanding_Debt"         "Credit_Utilization_Ratio"
## [17] "Credit_History_Age"       "Total_EMI_per_month"
## [19] "Amount_invested_monthly"  "Monthly_Balance"
## [21] "Credit_Score"
```

```
summary(data_1) # 자료 요약
```

```
##     Number          Month        Age         Annual_Income
##  Min.   :    1   Min.   :1   Min.   :14.00   Min.   :    7022
##  1st Qu.:24999   1st Qu.:1   1st Qu.:24.00   1st Qu.:   19543
##  Median :49997   Median :1   Median :32.00   Median :   36964
##  Mean   :49997   Mean   :1   Mean   :32.98   Mean   :  162882
##  3rd Qu.:74995   3rd Qu.:1   3rd Qu.:41.00   3rd Qu.:   71879
##  Max.   :99993   Max.   :1   Max.   :55.00   Max.   :23658189
##  Monthly_Inhand_Salary Num_Bank_Accounts Num_Credit_Card  Interest_Rate
##  Min.   :  357.3       Min.   : 0.000    Min.   : 1.000   Min.   : 1.00
##  1st Qu.: 1630.7       1st Qu.: 4.000    1st Qu.: 4.000   1st Qu.: 7.00
##  Median : 3074.5       Median : 6.000    Median : 5.000   Median :13.00
##  Mean   : 4184.6       Mean   : 5.405    Mean   : 5.561   Mean   :14.53
##  3rd Qu.: 5866.9       3rd Qu.: 7.000    3rd Qu.: 7.000   3rd Qu.:20.00
##  Max.   :15136.7       Max.   :10.000    Max.   :10.000   Max.   :34.00
##   Num_of_Loan   Delay_from_due_date Num_of_Delayed_Payment Changed_Credit_Lim
it
##  Min.   :0.00   Min.   :-4.00       Min.   :   0.00        Min.   :-6.25
##  1st Qu.:2.00   1st Qu.:10.00       1st Qu.:   9.00        1st Qu.: 5.65
##  Median :3.00   Median :18.00       Median :  14.00        Median : 9.49
##  Mean   :3.55   Mean   :21.25       Mean   :  30.69        Mean   :10.46
##  3rd Qu.:5.00   3rd Qu.:28.00       3rd Qu.:  18.00        3rd Qu.:14.60
##  Max.   :9.00   Max.   :67.00       Max.   :4292.00        Max.   :36.49
##  Num_Credit_Inquiries    Credit_Mix   Outstanding_Debt
##  Min.   :   0.00              :   1   Min.   :   0.77
##  1st Qu.:   2.00      Bad     :3038   1st Qu.: 585.05
##  Median :   4.00      Good    :3760   Median :1177.54
##  Mean   :  18.71      Standard:5701   Mean   :1436.59
##  3rd Qu.:   8.00                      3rd Qu.:1959.01
##  Max.   :2397.00                      Max.   :4998.07
##  Credit_Utilization_Ratio Credit_History_Age Total_EMI_per_month
##  Min.   :21.03            Min.   :  1.0      Min.   :    0.00
##  1st Qu.:28.11            1st Qu.:136.0      1st Qu.:   28.48
##  Median :32.41            Median :215.0      Median :   65.72
##  Mean   :32.29            Mean   :216.2      Mean   : 1228.45
##  3rd Qu.:36.37            3rd Qu.:297.0      3rd Qu.:  144.67
##  Max.   :49.56            Max.   :397.0      Max.   :79880.00
##  Amount_invested_monthly Monthly_Balance      Credit_Score
##  Min.   :   0.00         Min.   :   0.0886   Good    :1928
##  1st Qu.:  72.48         1st Qu.: 275.1550   Poor    :3548
##  Median : 126.78         Median : 339.2150   Standard:7024
##  Mean   : 193.94         Mean   : 404.1646
##  3rd Qu.: 235.33         3rd Qu.: 467.0638
##  Max.   :1534.60         Max.   :1602.0405
```

1.2) Stratified extraction(층화추출)

```
#Standard_7024, Poor_3548, Good_1928
#install.packages("sampling")
library(sampling)
stratified_sampling <- strata(data_1, stratanames = c("Credit_Score"), size =c(1
928, 1928, 1928),
                              method="srswor")

data_2 <- getdata(data_1, stratified_sampling)
table(data_2$Credit_Score)
```

```
##
##      Good      Poor Standard
##      1928      1928     1928
```

1.3) Scailing(표준화)

```
data_2$Num_Credit_Card = (data_2$Num_Credit_Card - min(data_2$Num_Credit_Card))/(max(data_2$Num_Credit_Card)-min(data_2$Num_Credit_Card))
data_2$Annual_Income = (data_2$Annual_Income - min(data_2$Annual_Income))/(max(data_2$Annual_Income)-min(data_2$Annual_Income))
data_2$Monthly_Inhand_Salary = (data_2$Monthly_Inhand_Salary - min(data_2$Monthly_Inhand_Salary))/(max(data_2$Monthly_Inhand_Salary)-min(data_2$Monthly_Inhand_Salary))
data_2$Num_Bank_Accounts = (data_2$Num_Bank_Accounts - min(data_2$Num_Bank_Accounts))/(max(data_2$Num_Bank_Accounts)-min(data_2$Num_Bank_Accounts))
data_2$Num_Credit_Card = (data_2$Num_Credit_Card - min(data_2$Num_Credit_Card))/(max(data_2$Num_Credit_Card)-min(data_2$Num_Credit_Card))
data_2$Interest_Rate = (data_2$Interest_Rate - min(data_2$Interest_Rate))/(max(data_2$Interest_Rate)-min(data_2$Interest_Rate))
data_2$Num_of_Loan = (data_2$Num_of_Loan - min(data_2$Num_of_Loan))/(max(data_2$Num_of_Loan)-min(data_2$Num_of_Loan))
data_2$Delay_from_due_date = (data_2$Delay_from_due_date - min(data_2$Delay_from_due_date))/(max(data_2$Delay_from_due_date)-min(data_2$Delay_from_due_date))
data_2$Num_of_Delayed_Payment = (data_2$Num_of_Delayed_Payment - min(data_2$Num_of_Delayed_Payment))/(max(data_2$Num_of_Delayed_Payment)-min(data_2$Num_of_Delayed_Payment))
data_2$Changed_Credit_Limit = (data_2$Changed_Credit_Limit - min(data_2$Changed_Credit_Limit))/(max(data_2$Changed_Credit_Limit)-min(data_2$Changed_Credit_Limit))
data_2$Num_Credit_Inquiries = (data_2$Num_Credit_Inquiries - min(data_2$Num_Credit_Inquiries))/(max(data_2$Num_Credit_Inquiries)-min(data_2$Num_Credit_Inquiries))
data_2$Outstanding_Debt = (data_2$Outstanding_Debt - min(data_2$Outstanding_Debt))/(max(data_2$Outstanding_Debt)-min(data_2$Outstanding_Debt))
data_2$Credit_Utilization_Ratio = (data_2$Credit_Utilization_Ratio - min(data_2$Credit_Utilization_Ratio))/(max(data_2$Credit_Utilization_Ratio)-min(data_2$Credit_Utilization_Ratio))
data_2$Credit_History_Age = (data_2$Credit_History_Age - min(data_2$Credit_History_Age))/(max(data_2$Credit_History_Age)-min(data_2$Credit_History_Age))
data_2$Total_EMI_per_month = (data_2$Total_EMI_per_month - min(data_2$Total_EMI_per_month))/(max(data_2$Total_EMI_per_month)-min(data_2$Total_EMI_per_month))
data_2$Amount_invested_monthly = (data_2$Amount_invested_monthly - min(data_2$Amount_invested_monthly))/(max(data_2$Amount_invested_monthly)-min(data_2$Amount_invested_monthly))
data_2$Monthly_Balance = (data_2$Monthly_Balance - min(data_2$Monthly_Balance))/(max(data_2$Monthly_Balance)-min(data_2$Monthly_Balance))
summary(data_2)
```

```
##      Number          Month        Age        Annual_Income     
##  Min.   :    1   Min.   :1   Min.   :14.00   Min.   :0.0000000  
##  1st Qu.:25281   1st Qu.:1   1st Qu.:24.00   1st Qu.:0.0005512  
##  Median :50589   Median :1   Median :33.00   Median :0.0013420  
##  Mean   :50299   Mean   :1   Mean   :33.45   Mean   :0.0075454  
##  3rd Qu.:75305   3rd Qu.:1   3rd Qu.:42.00   3rd Qu.:0.0029120  
##  Max.   :99985   Max.   :1   Max.   :55.00   Max.   :1.0000000  
##  Monthly_Inhand_Salary Num_Bank_Accounts Num_Credit_Card  Interest_Rate   
##  Min.   :0.00000       Min.   :0.0000    Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.08979       1st Qu.:0.3000    1st Qu.:0.3333   1st Qu.:0.2121  
##  Median :0.19218       Median :0.5000    Median :0.4444   Median :0.3636  
##  Mean   :0.27432       Mean   :0.5019    Mean   :0.4850   Mean   :0.4103  
##  3rd Qu.:0.39259       3rd Qu.:0.7000    3rd Qu.:0.6667   3rd Qu.:0.5758  
##  Max.   :1.00000       Max.   :1.0000    Max.   :1.0000   Max.   :1.0000  
##   Num_of_Loan     Delay_from_due_date Num_of_Delayed_Payment
##  Min.   :0.0000   Min.   :0.0000      Min.   :0.000000      
##  1st Qu.:0.2222   1st Qu.:0.1831      1st Qu.:0.001864      
##  Median :0.3333   Median :0.2817      Median :0.002796      
##  Mean   :0.3800   Mean   :0.3397      Mean   :0.006903      
##  3rd Qu.:0.5556   3rd Qu.:0.4366      3rd Qu.:0.004194      
##  Max.   :1.0000   Max.   :1.0000      Max.   :1.000000      
##  Changed_Credit_Limit Num_Credit_Inquiries    Credit_Mix   Outstanding_Debt
##  Min.   :0.0000       Min.   :0.0000000            :   1   Min.   :0.0000  
##  1st Qu.:0.2613       1st Qu.:0.0008344    Bad     :1393   1st Qu.:0.1186  
##  Median :0.3479       Median :0.0016688    Good    :1757   Median :0.2358  
##  Mean   :0.3705       Mean   :0.0080880    Standard:2633   Mean   :0.2827  
##  3rd Qu.:0.4378       3rd Qu.:0.0033375                    3rd Qu.:0.3802  
##  Max.   :1.0000       Max.   :1.0000000                    Max.   :1.0000  
##  Credit_Utilization_Ratio Credit_History_Age Total_EMI_per_month
##  Min.   :0.0000           Min.   :0.0000     Min.   :0.0000000  
##  1st Qu.:0.2579           1st Qu.:0.3788     1st Qu.:0.0003554  
##  Median :0.4143           Median :0.5707     Median :0.0008337  
##  Mean   :0.4124           Mean   :0.5676     Mean   :0.0159763  
##  3rd Qu.:0.5614           3rd Qu.:0.7727     3rd Qu.:0.0017936  
##  Max.   :1.0000           Max.   :1.0000     Max.   :1.0000000  
##  Amount_invested_monthly Monthly_Balance    Credit_Score     ID_unit     
##  Min.   :0.00000         Min.   :0.0000   Good    :1928   Min.   :    1  
##  1st Qu.:0.04812         1st Qu.:0.1742   Poor    :1928   1st Qu.: 3161  
##  Median :0.08478         Median :0.2165   Standard:1928   Median : 6324  
##  Mean   :0.13214         Mean   :0.2611                   Mean   : 6288  
##  3rd Qu.:0.15712         3rd Qu.:0.3037                   3rd Qu.: 9414  
##  Max.   :1.00000         Max.   :1.0000                   Max.   :12499  
##       Prob          Stratum 
##  Min.   :0.2745   Min.   :1  
##  1st Qu.:0.2745   1st Qu.:1  
##  Median :0.5434   Median :2  
##  Mean   :0.6060   Mean   :2  
##  3rd Qu.:1.0000   3rd Qu.:3  
##  Max.   :1.0000   Max.   :3  
```

Data Split(80:20)(훈련 및 평가데이터 분할)

```
library(caret)
```

```
## 필요한 패키지를 로딩중입니다: ggplot2
```

```
## 필요한 패키지를 로딩중입니다: lattice
```

```
##
## 다음의 패키지를 부착합니다: 'caret'
```

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

```
training <- createDataPartition(data_2$Number, p=0.8, list=FALSE)
td <- data_2[training,]
vd <- data_2[-training,]
rm(data_2, training)

colnames(td)
```

```
##  [1] "Number"                     "Month"
##  [3] "Age"                        "Annual_Income"
##  [5] "Monthly_Inhand_Salary"     "Num_Bank_Accounts"
##  [7] "Num_Credit_Card"            "Interest_Rate"
##  [9] "Num_of_Loan"                "Delay_from_due_date"
## [11] "Num_of_Delayed_Payment"    "Changed_Credit_Limit"
## [13] "Num_Credit_Inquiries"      "Credit_Mix"
## [15] "Outstanding_Debt"           "Credit_Utilization_Ratio"
## [17] "Credit_History_Age"         "Total_EMI_per_month"
## [19] "Amount_invested_monthly"   "Monthly_Balance"
## [21] "Credit_Score"               "ID_unit"
## [23] "Prob"                       "Stratum"
```

```
td <- td[, -c(1,2,22,23,24)]
vd <- vd[, -c(1,2,22,23,24)]
```

Neural net Model

```
#install.packages('RMySQL', repos='http://cran.us.r-project.org')

#install.packages("neuralnet")
library(neuralnet)

set.seed(2)
td_x = model.matrix(Credit_Score ~ ., td)
Credit_Score = ifelse(td$Credit_Score == "Poor", 0,
                  ifelse(td$Credit_Score == "Standard", 0.5, 1))

td1 = data.frame(cbind(td_x, Credit_Score))
NN = neuralnet(Credit_Score ~ . ,td1, hidden = 4, linear.output = F, err.fct =
'sse', likelihood = T)
```

Visualization (모델 시각화)

```
plot(NN)
```

Validation (모델 예측정확도 평가 with Confusion_Matrix)

```
vd_x = model.matrix(Credit_Score ~ ., vd)

Credit_Score = ifelse(vd$Credit_Score == "Poor", 0,
                 ifelse(vd$Credit_Score == "Standard", 0.5, 1))

vd1 = data.frame(cbind(vd_x, Credit_Score))

nn.results <- compute(NN, vd1)
predict_y = ifelse(nn.results$net.result <= 0.25, 0,
                   ifelse(nn.results$net.result > 0.25 & nn.results$net.result <
0.75, 0.5, 1))

cfm <-confusionMatrix(as.factor(predict_y), as.factor(vd1$Credit_Score))
cfm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0 0.5   1
##        0   327  95   1
##        0.5  50 233  83
##        1    22  68 277
##
## Overall Statistics
##
##                Accuracy : 0.724
##                  95% CI : (0.6973, 0.7497)
##     No Information Rate : 0.3452
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5857
##
##  Mcnemar's Test P-Value : 1.459e-07
##
## Statistics by Class:
##
##                      Class: 0 Class: 0.5 Class: 1
## Sensitivity            0.8195     0.5884   0.7673
## Specificity            0.8732     0.8250   0.8868
## Pos Pred Value         0.7730     0.6366   0.7548
## Neg Pred Value         0.9018     0.7937   0.8935
## Prevalence             0.3452     0.3426   0.3123
## Detection Rate         0.2829     0.2016   0.2396
## Detection Prevalence   0.3659     0.3166   0.3175
## Balanced Accuracy      0.8464     0.7067   0.8271
```

2) Random Forest

2.1) Modeling

```
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
##
## 다음의 패키지를 부착합니다: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(2345)
rf_model <- randomForest(Credit_Score~ ., data=td)
pred <- predict(rf_model,newdata=vd)
pred2 <- ifelse(Credit_Score == "Poor", 0,
           ifelse(Credit_Score == "Standard", 0.5, 1))
```

2.2) Validation (모델 예측정확도 평가 with Confusion_Matrix)

```
cfm_rf <-confusionMatrix(as.factor(pred), as.factor(vd$Credit_Score))
cfm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Good Poor Standard
##   Good      361    8       33
##   Poor        0  382       40
##   Standard    0    9      323
##
## Overall Statistics
##
##                Accuracy : 0.9221
##                  95% CI : (0.9052, 0.9369)
##     No Information Rate : 0.3452
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8833
##
##  Mcnemar's Test P-Value : 4.349e-13
##
## Statistics by Class:
##
##                      Class: Good Class: Poor Class: Standard
## Sensitivity               1.0000      0.9574          0.8157
## Specificity               0.9484      0.9472          0.9882
## Pos Pred Value            0.8980      0.9052          0.9729
## Neg Pred Value            1.0000      0.9768          0.9114
## Prevalence                0.3123      0.3452          0.3426
## Detection Rate            0.3123      0.3304          0.2794
## Detection Prevalence      0.3478      0.3651          0.2872
## Balanced Accuracy         0.9742      0.9523          0.9019
```

3) Gradient Boosting Tree by XGB

3.1) Load Data_set

```
dt_xgboost <- data_1
dt_xgboost<- dt_xgboost[, -c(1,2)]
dt_xgboost$Credit_Score = ifelse(dt_xgboost$Credit_Score == "Poor", 0,
                            ifelse(dt_xgboost$Credit_Score == "Standard",1, 2))
```

3.2) Data_Preprocessing

```
#install.packages("xgboost")
library(xgboost)

#install.packages("Matrix")
library(Matrix)

dt_xgb_sparse_matrix <- sparse.model.matrix(Credit_Score ~., data = dt_xgboost)
train_index <- sample(1:nrow(dt_xgb_sparse_matrix), 2500)
```

3.3) train_Data & Test_data Labeling(훈련 및 평가데이터 생성)

```
train_x <- dt_xgb_sparse_matrix[train_index,]
test_x <- dt_xgb_sparse_matrix[-train_index,]
train_y <- dt_xgboost[train_index,'Credit_Score']
test_y <- dt_xgboost[-train_index,'Credit_Score']

dtrain <- xgb.DMatrix(data=train_x, label=as.matrix(train_y))
dtest <- xgb.DMatrix(data=test_x, label=as.matrix(test_y))
```

3.4) set the parameter

```
param <- list(max_depth =3,
              eta=0.1,
              verbose = 0,
              nthread = 2,
              objective = "multi:softmax",
              eval_metric = "mlogloss",
              verbose = F,
              prediction =T
              )
```

3.5) XGBoost Modeling

```
xgb <- xgb.train(params = param,
                 data = dtrain,
                 nrounds = 10,
                 subsample = 0.5,
                 colsample_bytree = 0.5,
                 num_class = 15
                 )
```

```
## Warning in check.booster.params(params, ...): The following parameters were p
rovided multiple times:
##  verbose
##   Only the last value for each of them will be used.
```

```
## [13:53:52] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "prediction", "verbose" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language
bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actual
ly being used
##   but getting flagged wrongly here. Please open an issue if you find any such
cases.
```

3.5) predict train_set & test_set

```
train_y_pred <- predict(xgb, dtrain)
test_y_pred <- predict(xgb, dtest)
```

3.6) KS statistics for train_set & test_set

```
#install.packages("MLmetrics")
library(MLmetrics)
```

```
##
## 다음의 패키지를 부착합니다: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

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

```
KS_Stat(train_y_pred, train_y)
```

```
## [1] 61.52912
```

```
KS_Stat(test_y_pred, test_y)
```

```
## [1] 56.09002
```

3.7) Caculate the feature importance Matrix

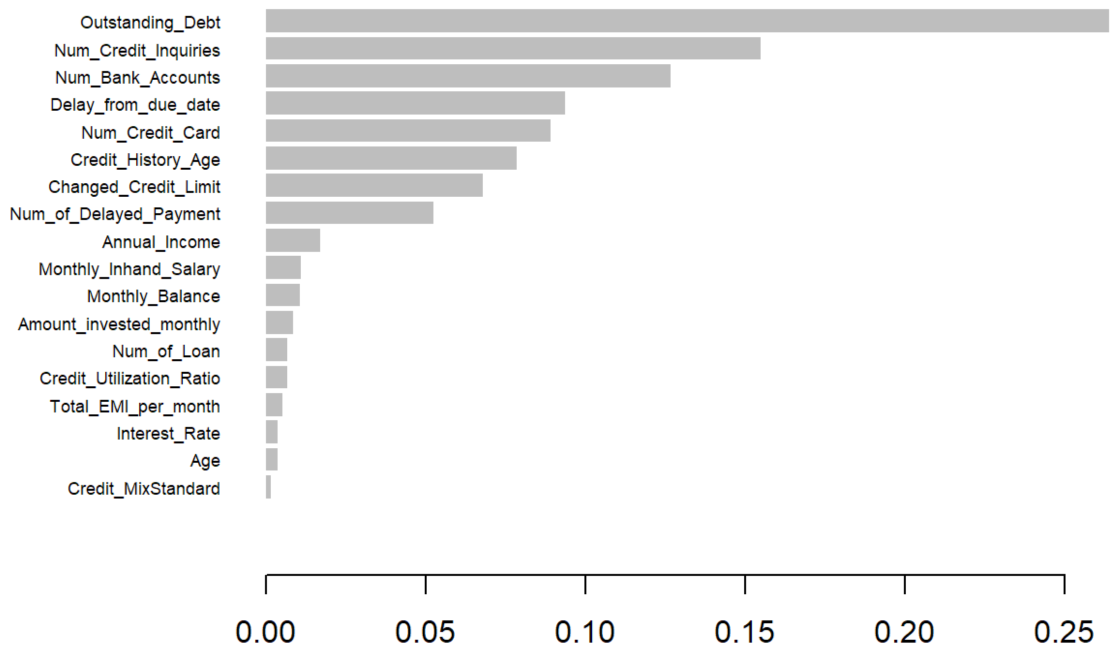
```
names <-dimnames(dtrain)[[2]]
names
```

```
##  [1] "(Intercept)"              "Age"
##  [3] "Annual_Income"            "Monthly_Inhand_Salary"
##  [5] "Num_Bank_Accounts"        "Num_Credit_Card"
##  [7] "Interest_Rate"            "Num_of_Loan"
##  [9] "Delay_from_due_date"      "Num_of_Delayed_Payment"
## [11] "Changed_Credit_Limit"     "Num_Credit_Inquiries"
## [13] "Credit_MixBad"            "Credit_MixGood"
## [15] "Credit_MixStandard"       "Outstanding_Debt"
## [17] "Credit_Utilization_Ratio" "Credit_History_Age"
## [19] "Total_EMI_per_month"      "Amount_invested_monthly"
## [21] "Monthly_Balance"
```

```
importance_martix <- xgb.importance(names, model =xgb)
importance_martix
```

```
##                      Feature        Gain       Cover  Frequency
##  1:         Outstanding_Debt 0.263835403 0.109850597 0.08900524
##  2:     Num_Credit_Inquiries 0.154934532 0.067771552 0.07329843
##  3:        Num_Bank_Accounts 0.126516295 0.117734350 0.07853403
##  4:      Delay_from_due_date 0.093598432 0.098718346 0.10471204
##  5:          Num_Credit_Card 0.088932431 0.165306532 0.07853403
##  6:       Credit_History_Age 0.078326084 0.103358359 0.08900524
##  7:     Changed_Credit_Limit 0.067799099 0.092714540 0.08900524
##  8:   Num_of_Delayed_Payment 0.052334546 0.078249157 0.09947644
##  9:            Annual_Income 0.016949591 0.018106288 0.03664921
## 10:    Monthly_Inhand_Salary 0.010885397 0.029128162 0.03141361
## 11:          Monthly_Balance 0.010450613 0.021675316 0.05235602
## 12:  Amount_invested_monthly 0.008426516 0.031585178 0.03664921
## 13:              Num_of_Loan 0.006673443 0.009140378 0.02094241
## 14: Credit_Utilization_Ratio 0.006527960 0.009204665 0.02094241
## 15:      Total_EMI_per_month 0.005120735 0.009774207 0.04712042
## 16:            Interest_Rate 0.003704934 0.014629229 0.01570681
## 17:                      Age 0.003674491 0.020038292 0.02617801
## 18:       Credit_MixStandard 0.001309497 0.003014849 0.01047120
```

```
xgb.plot.importance(importance_martix[1:20])
```


Outstanding_Debt
Num_Credit_Inquiries
Num_Bank_Accounts
Delay_from_due_date
Num_Credit_Card
Credit_History_Age
Changed_Credit_Limit
Num_of_Delayed_Payment
Annual_Income
Monthly_Inhand_Salary
Monthly_Balance
Amount_invested_monthly
Num_of_Loan
Credit_Utilization_Ratio
Total_EMI_per_month
Interest_Rate
Age
Credit_MixStandard
0.00
0.05
0.10
0.15
0.20
0.25