

# main\_\_proj5

Group 8

April 24, 2018

```
packages.used <- c("readxl", "ggplot2", "caret", "reshape2", "randomForest",  
                  "xgboost", "pROC", "e1071", "InformationValue", "devtools", "nnet")
```

```
# check packages that need to be installed.
```

```
packages.needed <- setdiff(packages.used,  
                           intersect(installed.packages()[,1],  
                                    packages.used))
```

```
# install additional packages
```

```
if(length(packages.needed) > 0) {  
  install.packages(packages.needed, dependencies = TRUE,  
                  repos = 'http://cran.us.r-project.org')  
}
```

```
library(readxl)  
library(ggplot2)  
library(caret)
```

```
## Loading required package: lattice
```

```
library(reshape2)  
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
```

```
## Attaching package: 'randomForest'
```

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

```
##
```

```
##      margin
```

```
library(xgboost)  
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
```

```
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##      cov, smooth, var
```

```
library(e1071)  
library(InformationValue)
```

```
##
```

```
## Attaching package: 'InformationValue'
```

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

```
##
##      confusionMatrix, precision, sensitivity, specificity
library(devtools)
library(nnet)
```

## Step 0: Specify directories.

Set the working directory to the data folder. Specify the training and the testing set. For data without an independent test/validation set, you need to create your own testing data by random subsampling. In order to obtain reproducible results, `set.seed()` whenever randomization is used.

```
#setwd("")
# here replace it with your own path or manually set it in RStudio to where this rmd file is located.
```

Provide directories for raw images. Training set and test set should be in different subfolders.

```
#data <- read_xls("../data/data.xls",sheet = "Data",range = "A1:Y30002")
#data <- apply(data,2,as.numeric)
#data <- as.data.frame(data)

#set.seed(04182018)
#test_idx <- sample(1:30000,6000)
#train_idx <- setdiff(1:30000, test_idx)
#train <- data[train_idx,]
#test <- data[test_idx,]

#write.csv(train,"train.csv")
##write.csv(test,"test.csv")

train <- read.csv("../data/train.csv")
test <- read.csv("../data/test.csv")
train <- train[,-1]
test <- test[,-1]
```

## Step 1: Set up controls for evaluation experiments.

In this chunk, we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set
- (number) K, the number of CV folds
- (T/F) run evaluation on an independent test set

```
run.cv=FALSE # do not run cross-validation on the training set
K <- 5 # number of CV folds
run.test=TRUE # run evaluation on an independent test set
```

Using cross-validation or independent test set evaluation, we compare the performance of different classifiers or classifiers with different specifications.

## Step 2: Process training and testing data.

```
train$X4[which(train$X4!=1)] <- 2 # other marital status merged to unmarried
train$Y <- as.factor(train$Y)
```

```
test$X4[which(test$X4!=1)] <- 2 # other marital status merged to unmarried
test$Y <- as.factor(test$Y)

#train_new <- train[,c("X1", "X5", "X24", "X25", "Y")]
#test_new <- test[,c("X1", "X5", "X24", "X25", "Y")]

#write.csv(train_new, file = "train_new.csv")
#write.csv(test_new, file = "test_new.csv")
```

### Step 3: Train classification models with training data.

Five different models are used to make prediction on whether a credit card client will default based on related information.

### Model 1: Support Vector Machine

In parameter selecting part, for linear SVM, we select cost from 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000 and 10000. For RBF kernel, we select cost from 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10 and 100, and select gamma from 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000 and 10000.

```
train <- train[,c(1,2,4,5,12:17,18:24)]
test <- test[,c(1,2,4,5,12:17,18:24)]

transform_df <- function(df){
  colnames(df) <- c("X1", "X2", "X4", "X5", "X12", "X13", "X14", "X15", "X16", "X17", "X18", "X19", "X20", "X21", "X22", "X23", "X24", "X25", "X26", "X27", "X28", "X29", "X30", "X31", "X32", "X33", "X34", "X35", "X36", "X37", "X38", "X39", "X40", "X41", "X42", "X43", "X44", "X45", "X46", "X47", "X48", "X49", "X50", "X51", "X52", "X53", "X54", "X55", "X56", "X57", "X58", "X59", "X60", "X61", "X62", "X63", "X64", "X65", "X66", "X67", "X68", "X69", "X70", "X71", "X72", "X73", "X74", "X75", "X76", "X77", "X78", "X79", "X80", "X81", "X82", "X83", "X84", "X85", "X86", "X87", "X88", "X89", "X90", "X91", "X92", "X93", "X94", "X95", "X96", "X97", "X98", "X99", "X100")

  df$X2_1 <- ifelse(df$X2==1,1,0)
  df$X2_2 <- ifelse(df$X2==2,1,0)
  df$X4_1 <- ifelse(df$X4==1,1,0)
  df$X4_2 <- ifelse(df$X4==2,1,0)
  df <- df[, -c(2:3)]

  scaled_df <- scale(df[, -15])
  scaled_df <- cbind(scaled_df, df[, 15])
  colnames(scaled_df)[ncol(scaled_df)] <- "Y"
  return(scaled_df)
}

train_scaled <- transform_df(train)
test_scaled <- transform_df(test)

# Corss Validation
# linear SVM
if(run.cv){
  tuned_lin <- tune.svm(as.factor(Y)~., data = as.data.frame(train_scaled),
                        cost = 10^(-4:4), kernel = "linear",
                        tunecontrol = tune.control(cross = 5))
  best_par_lin <- tuned_lin$best.parameters
} else{
  best_par_lin <- 0.1
}
```

```

# RBF kernel SVM
if(run.cv){
  tuned_RBF <- tune.svm(as.factor(Y)~., data = as.data.frame(train_scaled),
    gamma = 10^(-4:4), cost = 10^(-6:2),
    kernel = "radial",
    tunecontrol = tune.control(cross = 5))
  best_par_RBF <- tuned_RBF$best.parameters
} else{
  best_par_RBF <- c(10,0.001)
}

load("../app/model_linear.rda")

#model_linear <- svm(as.factor(Y)~.,data = train_scaled,
#                    cost = best_par_lin,kernel = #"linear")

pred_linear <- predict(model_linear,test_scaled[, -ncol(test_scaled)])
err_lin <- mean(pred_linear!=as.numeric(test$Y))
#err_lin

#RBF kernel SVM with soft margin
load("../app/model_RBF.rda")

#model_RBF <- svm(as.factor(Y)~., data = train_scaled,
#                 cost = best_par_RBF[1], gamma = best_par_RBF[2], kernel = #"radial")

time1 <- system.time(pred_RBF <- predict(model_RBF, test_scaled[, -ncol(test_scaled)]))
err_RBFSVM <- mean(pred_RBF!=test$Y)
cat("error:",err_RBFSVM,"\n")

## error: 0.2216667
cat("time:",time1[3])

## time: 27.86

```

## Model 2: Random Forest

Data preprocessing

```

df <- read.csv('../data/data.csv', skip=1) #Read the data
df <- df[,-1]
df$default.payment.next.month <- as.factor(df$default.payment.next.month)
#Convert target variable to factor type
df$ID <- NULL #Remove extraneous variables

```

Model building

```

#Split data into training and test sets
set.seed(04182018)
test.i <- sample(1:nrow(df), .2*nrow(df), replace=FALSE)
test.data <- df[test.i,]
train.data <- df[-test.i,]

```

```

#Build random forest tuning grid
rf_tune <- expand.grid(mtry=2,
                      ntree = seq(100, 1000, by = 250))
rf_tune$accuracy <- numeric(nrow(rf_tune))

#Tune parameters to find best model
if(run.cv){
  for(i in 1:nrow(rf_tune)){
    our.rf <- randomForest(default.payment.next.month ~.,
                           data=train.data, na.action = na.omit,
                           mtry=rf_tune$mtry[i],
                           ntree=rf_tune$ntree[i])
    rf.preds <- predict(our.rf, test.data)
    rf_tune$accuracy[i] <- mean(rf.preds == test.data$default.payment.next.month, na.rm = TRUE)
  }
  best_rf_params <- rf_tune[which.max(rf_tune$accuracy),]
} else{
  best_rf_params <- c(2,600)
}

load("../app/model_rf.rda")
#final_rf <- randomForest(default.payment.next.month ~.,
#                           data=train.data, na.action = na.omit,
#                           mtry=best_rf_params[1],
#                           ntree=best_rf_params[2])

time2 <- system.time(rf.pred_final <- predict(final_rf, test.data))
err_rf <- mean(rf.pred_final!=test.data$default.payment.next.month)
cat("error:",err_rf,"\n")

## error: 0.1856667

cat("time:",time2)

## time: 2.87 0.03 2.92 NA NA

```

## Model 3: Xgboost

```

#Split data into predictors and target
#Note, features must be numeric, not integer, so convert them
df.features <- df[, -which(colnames(df) == 'default.payment.next.month')]
df.features <- apply(df.features, 2, function(x) as.numeric(x))

df.target <- df$default.payment.next.month

#Build XGboost tuning grid
xgb.tune <- expand.grid(eta=c(0.3, 0.5, 0.7),
                      gamma=c(0, 0.5, 1),
                      max_depth = c(2, 3, 4, 5, 10))
xgb.tune$accuracy <- numeric(nrow(xgb.tune))

# Tune parameters to find best model
#Each iteration performs 5-fold CV with 50 rounds of training XGBoost.

```

```

if(run.cv){
  set.seed(04182018)
  for(i in 1:nrow(xgb.tune)){
    t1 = Sys.time()
    print(paste('Starting iteration', i, 'of', nrow(xgb.tune), ':'))

    param_list <- list(max_depth=xgb.tune$max_depth[i],
                      eta=xgb.tune$eta[i],
                      gamma = xgb.tune$gamma[i],
                      silent=1,
                      nthread=2,
                      objective='multi:softmax')

    model <- xgb.cv(data = as.matrix(df.features),
                   nrounds = 50,
                   nfold = 5,
                   metrics = list("merror"),
                   label = df.target,
                   params = param_list,
                   num_class = 4)

    # Takes mean of 10 training rounds with highest classification rate
    xgb.tune$accuracy[i] <- 1-(mean(sort(model$evaluation_log$test_merror_mean)[1:10]))
    t2 = Sys.time()
    print(paste('Iteration', i, 'took : ', (t2-t1), 'seconds'))
  }

  # Show best parameters
  best_gb_params <- xgb.tune[which.max(xgb.tune$accuracy),]
  best_gb_params
} else{
  load("../app/model_xgb.rda")
}

#model <- xgboost(data = as.matrix(df.features), label = as.numeric(df.target)-1,
#               nrounds = 100, objective = "multi:softmax", num_class = 2)
test.data <- apply(test.data, 2, function(x) as.numeric(x))
test_Dmat <- xgb.DMatrix(as.matrix(test.data[, -ncol(test.data)]))
time3 <- system.time(pred_xgb <- predict(model, test_Dmat))
err_xgb <- mean(pred_xgb != test.data[, ncol(test.data)])
cat("error:", err_xgb, "\n")

## error: 0.1223333
cat("time:", time3)

## time: 0.31 0 0.06 NA NA

```

## Model 4: Logistic Regression

```

# Establish common downstream variables
classColumn <- "Y"

```

```

badIndicator <- 1
goodIndicator <- 0
montonicConstraint <- "No"

# Training data
dataTrainingSample <- read.csv("../data/train.csv")
dataTrainingSample <- dataTrainingSample[1:5999,]
dataTrainingSample <- as.data.frame(dataTrainingSample)

# Testing data
dataTestingSample <- read.csv("../data/test.csv")
dataTestingSample <- as.data.frame(dataTestingSample)

# Get totals for downstream
numberOfObservations <- nrow(dataTrainingSample)
numberOfBads <- nrow(dataTrainingSample[dataTrainingSample[,classColumn]==badIndicator,])
numberOfGoods <- numberOfObservations-numberOfBads

# Create tons of model combinations
Variables <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X11","X12",
              "X13","X14","X15","X16","X17","X18","X19","X20","X21","X22","X23")
logisticSummary <- matrix(0, nrow = 1, ncol = 14)
colnames(logisticSummary) <- c("Variables", "Formula", "Var1", "Var2", "Var3", "Var4", "Var5", "Var6",
                                "Var7", "Var8", "Var9", "Var10", "Var11", "Var12")
logisticSummary <- as.data.frame(logisticSummary)

logisticModels <- NULL

for (i in (1:length(Variables)))
{
  Vars <- 1
  temp <- logisticSummary

  FORMULA <- paste0(classColumn, " ~ ", Variables[i])
  MODEL <- glm(as.formula(FORMULA), data=dataTrainingSample, family=binomial(link="logit"))
  PREDICTED <- plogis(predict(MODEL, dataTestingSample))

  temp[1,"AUC"] <- auc(dataTestingSample[,classColumn],PREDICTED)
  temp[1,"P < 0.05"] <- length(which(as.numeric(coef(summary(MODEL))[,4]) < 0.05))-(Vars+1)

  if (temp[1,"P < 0.05"]==0)
  {
    temp[1,"Variables"] <- 1
    temp[1,"Var1"] <- Variables[i]
    temp[1, "Formula"] <- FORMULA

    logisticModels <- rbind(logisticModels, temp)

    variablesSublist1 <- Variables[which(Variables%in%c(Variables[i])==FALSE)]

    for (j in (1:length(variablesSublist1)))
    {
      temp <- logisticSummary
      Vars <- 2
    }
  }
}

```

```

FORMULA <- paste0(classColumn, " ~ ", Variables[i], " + ", variablesSublist1[j])
MODEL <- glm(as.formula(FORMULA), data=dataTrainingSample, family=binomial(link="logit"))
PREDICTED <- plogis(predict(MODEL, dataTestingSample))

temp[1,"AUC"] <- auc(dataTestingSample[,classColumn],PREDICTED)
temp[1,"P < 0.05"] <- length(which(as.numeric(coef(summary(MODEL))[,4]) < 0.05))-(Vars+1)

if (temp[1,"P < 0.05"]==0)
{
  temp[1,"Variables"] <- 2
  temp[1,"Var1"] <- Variables[i]
  temp[1,"Var2"] <- variablesSublist1[j]
  temp[1, "Formula"] <- FORMULA

  logisticModels <- rbind(logisticModels, temp)

  variablesSublist2 <- Variables[which(Variables%in%c(Variables[i], variablesSublist1[j])==FALSE)]

  for (k in (1:length(variablesSublist2)))
  {
    temp <- logisticSummary
    Vars <- 3

    FORMULA <- paste0(classColumn, " ~ ", Variables[i], " + ", variablesSublist1[j], " + ", variablesSublist2[k])
    MODEL <- glm(as.formula(FORMULA), data=dataTrainingSample, family=binomial(link="logit"))
    PREDICTED <- plogis(predict(MODEL, dataTestingSample))

    temp[1,"AUC"] <- auc(dataTestingSample[,classColumn],PREDICTED)
    temp[1,"P < 0.05"] <- length(which(as.numeric(coef(summary(MODEL))[,4]) < 0.05))-(Vars+1)

    if (temp[1,"P < 0.05"]==0)
    {
      temp[1,"Variables"] <- 3
      temp[1,"Var1"] <- Variables[i]
      temp[1,"Var2"] <- variablesSublist1[j]
      temp[1,"Var3"] <- variablesSublist2[k]
      temp[1, "Formula"] <- FORMULA

      logisticModels <- rbind(logisticModels, temp)

    }

  }

}

}

}

}

}

}

}

# Order from least to most predictive
logisticModels <- logisticModels[order(logisticModels$AUC),]

```



```
tail(logisticModels)
```

```
##      Variables      Formula Var1 Var2 Var3 Var4 Var5 Var6 Var7 Var8
## 25      3 Y ~ X1 + X6 + X9   X1  X6  X9    0    0    0    0    0
## 69      3 Y ~ X1 + X9 + X6   X1  X9  X6    0    0    0    0    0
## 241     3 Y ~ X6 + X1 + X9   X6  X1  X9    0    0    0    0    0
## 305     3 Y ~ X6 + X9 + X1   X6  X9  X1    0    0    0    0    0
## 1081    3 Y ~ X9 + X1 + X6   X9  X1  X6    0    0    0    0    0
## 1109    3 Y ~ X9 + X6 + X1   X9  X6  X1    0    0    0    0    0
##      Var9 Var10      AUC P < 0.05
## 25      0      0 0.7289305      0
## 69      0      0 0.7289305      0
## 241     0      0 0.7289305      0
## 305     0      0 0.7289305      0
## 1081    0      0 0.7289305      0
## 1109    0      0 0.7289305      0
```

```
fit <- glm(default.payment.next.month~LIMIT_BAL+PAY_0+PAY_5,data = train.data,family = binomial("logit"))
time4 <- system.time(pred_log <- predict(fit,newdata = as.data.frame(test.data[, -ncol(test.data)]),type = "probs"))
pred_log <- ifelse(pred_log>0.5,1,0)
err_log <- mean(pred_log!=test.data[,ncol(test.data)])
cat("error:",err_log,"\n")
```

```
## error: 0.1918333
```

```
cat("time:",time4)
```

```
## time: 0.03 0 0.04 NA NA
```

## Model 5: Neural Network

```
options(warn = -1)
# This is required for the nnet package
dataTestingSample$Y <- as.factor(dataTestingSample$Y)

# Plug in everything to a neural network
FORMULA <- logisticModels[nrow(logisticModels), "Formula"]
MODEL <- nnet(as.formula(FORMULA), data=dataTrainingSample, size = 2, rang = 0.1,decay = 5e-4, maxit = 1000)

## # weights: 11
## initial value 1388.647330
## iter 10 value 1046.756065
## final value 1046.750403
## converged

time5 <- system.time(dataTestingSample$PREDICTION <- predict(MODEL, dataTestingSample[, -which(colnames(dataTestingSample) == "Y")]))

dataTestingSample$PREDICTION[dataTestingSample$PREDICTION>0.5]<-1
dataTestingSample$PREDICTION[dataTestingSample$PREDICTION<=0.5]<-0
#dataTestingSample$PREDICTION == dataTestingSample$Y
err_nn <- sum(dataTestingSample$PREDICTION != dataTestingSample$Y)/nrow(dataTestingSample)
cat("error:",err_nn,"\n")
```

```
## error: 0.2216667
```

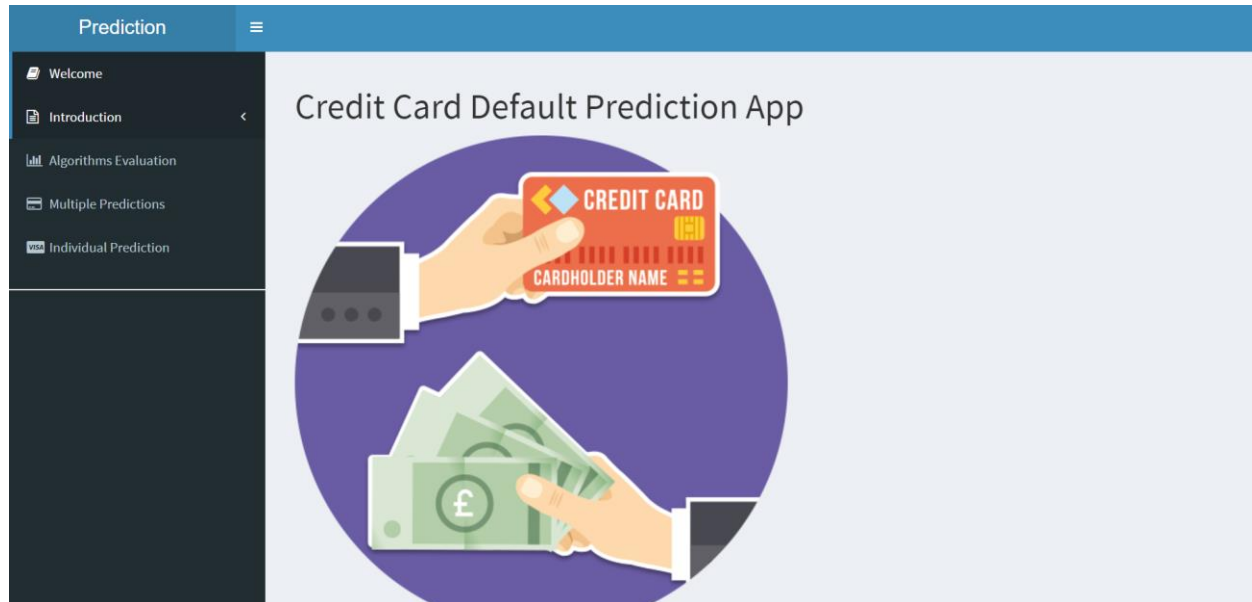
```
cat("time:",time5)
```

```
## time: 0.04 0 0.04 NA NA
```

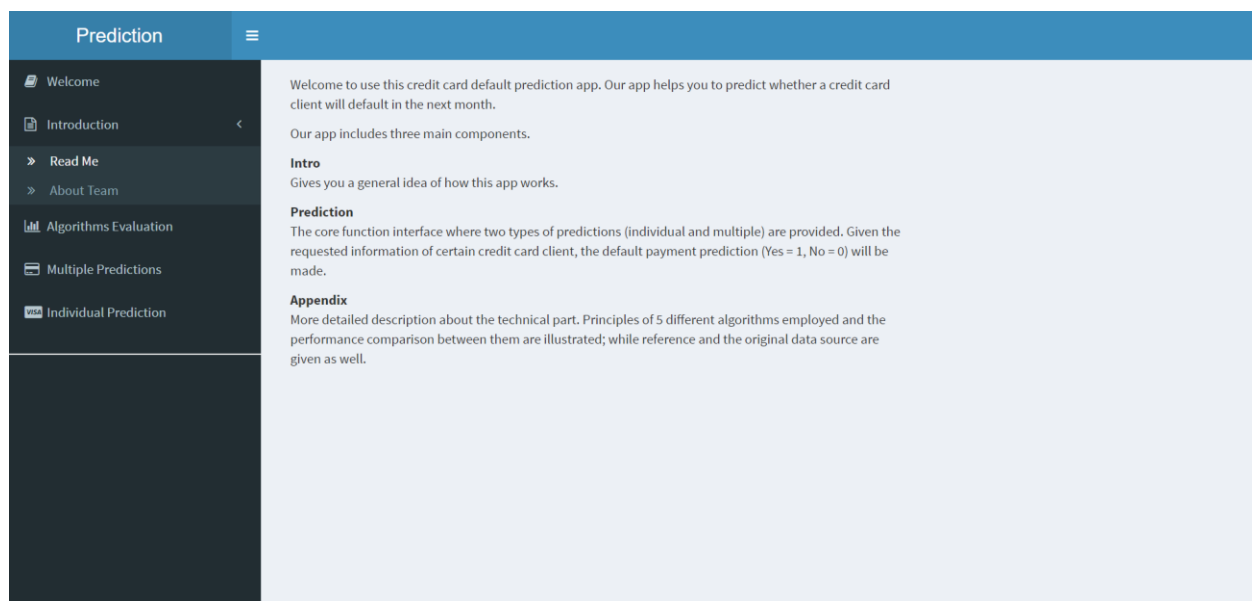
Reference: [https://gist.githubusercontent.com/Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f1454864427e3a189c2fe33f3/nnet\\_plot\\_update.r](https://gist.githubusercontent.com/Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f1454864427e3a189c2fe33f3/nnet_plot_update.r)

# Shiny App

## Homepage



## Introduction – Read Me



# Algorithms Evaluation – Algorithms Introductions

Prediction

Welcome

Introduction

» Read Me

» About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Algorithms Introductions

Algorithms Evaluation

Reference

**Logistics Regression (LR)**  
A special case of linear regression model where the dependent variable (DV) is categorical.

**Support Vector Machine (SVM)**  
Supervised learning models which constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space.

**Random Forest (RF)**  
An ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

**XGBoost**  
An advanced implementation of gradient boosting algorithm by adding new models sequentially until no further improvement is achieved.

**Artificial Neural Network (ANN)**  
A nonlinear statistical model based on the structure and functions of biological neural networks when relationships between inputs and outputs is complex.

## Algorithms Evaluation

Prediction

Welcome

Introduction

» Read Me

» About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Algorithms Introductions

Algorithms Evaluation

Reference

Algorithm	Test Accuracy	Prediction Time (6000 observations)
SVM	0.779	28.75s
Random Forest	0.815	3.07s
Xgboost	0.878	0.05s
Logistic Regression	0.808	0.10s
Neural Network	0.779	4.41s

# Multiple Predictions

Note: User can upload a csv data with a certain format (based on the tutorial given in the Shiny App), we can provide the prediction results.

## Data uploading page and tutorial

Prediction

Welcome

Introduction

Read Me

About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Choose Your Testing Data Table (CSV file)

Browse... testforshow.csv

Upload complete

Submit

Tutorial

Test Data

Prediction

To make multiple prediction, a data frame consisting the relevant information of credit card clients you want to predict is required. The data frame should include 16 explanatory variables as follows.  
X1: Amount of the given credit (NT dollar); it includes both the individual consumer credit and his/her family (supplementary) credit.  
X2: Gender (1 = male; 2 = female).  
X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).  
X4: Marital status (1 = married; 2 = single).  
X5: Age (year).  
X6-X11: History of past 6 months payment. (-1 = pay duly; 1-8 = payment delay for 1-8 months; 9 = payment delay for 9 months or above).  
X12-X17: Amount of bill statement (NT dollar) of the past six months.  
X18-X23: Amount of payment (NT dollar) of the past six months.  
Example:  

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23
1	230000	1	1	2	35	2	2	2	2	2	2	117277	119823	120986	121864	123749	120937	582	4500	3900	5000	0	4500
2	130000	1	3	1	56	-1	-1	-1	0	-1	-1	582	0	582	291	441	441	0	582	0	441	441	291
3	50000	2	2	1	31	2	0	0	2	0	0	43979	44970	46015	46993	47940	50000	1744	3955	0	3954	3713	2900
4	30000	2	2	2	25	0	0	0	0	0	-2	15493	16427	17379	18100	0	0	1500	1379	10000	0	0	0
5	220000	2	1	2	35	-1	-1	-1	-1	-1	-1	5564	7443	5572	5572	5572	5774	7479	5000	5000	5000	5802	3379

## Show the table uploaded by user

Prediction

Welcome

Introduction

Read Me

About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Choose Your Testing Data Table (CSV file)

Browse... testforshow.csv

Upload complete

Submit

Tutorial

Test Data

Prediction

Show 5 entries

Search:

X	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2
1	230000	1	1	2	35	2	2	2	2	2	2	117277	119823
2	130000	1	3	1	56	-1	-1	-1	0	-1	-1	582	0
3	50000	2	2	1	31	2	0	0	2	0	0	43979	44970
4	30000	2	2	2	25	0	0	0	0	0	-2	15493	16427
5	220000	2	1	2	35	-1	-1	-1	-1	-1	-1	5564	7443

X LIMIT\_BAL SEX EDUCATION MARRIAGE AGE PAY\_0 PAY\_2 PAY\_3 PAY\_4 PAY\_5 PAY\_6 BILL\_AMT1 BILL\_AMT2

Showing 1 to 5 of 6,000 entries

Previous 1 2 3 4 5 ... 1200 Next

## Show the prediction results

Prediction

Welcome

Introduction

Read Me

About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Choose Your Testing Data Table (CSV file)

Browse...

testforshow.csv

Upload complete

Submit

Tutorial

Test Data

Prediction

Note: Xgboost prediction result is recommended, use random forest result as a reference.

Show 25 entries

Search:

ID	Xgboost	Random_Forest
1	Yes	No
2	Yes	Yes
3	Yes	No
4	No	No
5	No	No
6	No	No
7	No	No
8	No	No

## Individual prediction

Note: User can fulfill the information of a customer, we will give out the prediction result for this customer based on the information given.

Prediction

Welcome

Introduction

Read Me

About Team

Algorithms Evaluation

Multiple Predictions

Individual Prediction

Gender

☒ Male ☐ Female

Marriage

☐ Married ☒ Unmarried

Education

☒ Graduate or above ☐ University ☐ High School ☐ Others

Amount of given credit

0

Age

30

History of past 6 months payment

Pay Duly

Pay Duly

Pay Duly

Pay Duly

Pay Duly

Pay Duly

Amount of bill statement last 6 months

0

0

0

0

0

0

Amount of payment last 6 months

0

0

0

0

0

0

Predict

Tutorial

Prediction

Will this customer make default payment next month?

No