Model and Ensemble - Click Predictions - Machine Learning

Importing in Packages and reading in the data

```
#importing Packages
library("nnet")
library("data.table")
```

```
## Warning: package 'data.table' was built under R version 3.4.2
```

```
library("glmnet")
```

```
## Warning: package 'glmnet' was built under R version 3.4.3

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

```
library("caret")
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.2
```

```
library("rpart")



# logloss function
LogLoss<-function(actual, predicted)
{
  result<- -1/length(actual)*(sum((actual*log(predicted)+(1-actual)*log(1-predicted))))
  return(result)
}


df <- fread('ProjectTrainingData.csv')
```

```
##
Read 0.0% of 31991090 rows
Read 1.1% of 31991090 rows
Read 2.3% of 31991090 rows
Read 3.3% of 31991090 rows
Read 4.4% of 31991090 rows
Read 5.7% of 31991090 rows
Read 6.8% of 31991090 rows
Read 7.9% of 31991090 rows
Read 8.8% of 31991090 rows
Read 10.1% of 31991090 rows
Read 11.3% of 31991090 rows
Read 12.3% of 31991090 rows
Read 13.3% of 31991090 rows
```

```
Read 14.5% of 31991090 rows
Read 15.4% of 31991090 rows
Read 16.6% of 31991090 rows
Read 17.8% of 31991090 rows
Read 18.7% of 31991090 rows
Read 20.0% of 31991090 rows
Read 21.2% of 31991090 rows
Read 22.4% of 31991090 rows
Read 23.6% of 31991090 rows
Read 24.8% of 31991090 rows
Read 25.8% of 31991090 rows
Read 27.1% of 31991090 rows
Read 28.3% of 31991090 rows
Read 29.2% of 31991090 rows
Read 30.4% of 31991090 rows
Read 31.4% of 31991090 rows
Read 32.5% of 31991090 rows
Read 33.7% of 31991090 rows
Read 34.9% of 31991090 rows
Read 36.0% of 31991090 rows
Read 37.2% of 31991090 rows
Read 37.8% of 31991090 rows
Read 39.0% of 31991090 rows
Read 40.2% of 31991090 rows
Read 41.3% of 31991090 rows
Read 42.5% of 31991090 rows
Read 43.5% of 31991090 rows
Read 44.8% of 31991090 rows
Read 46.1% of 31991090 rows
Read 47.4% of 31991090 rows
Read 48.6% of 31991090 rows
Read 49.9% of 31991090 rows
Read 51.2% of 31991090 rows
Read 52.4% of 31991090 rows
Read 53.6% of 31991090 rows
Read 53.9% of 31991090 rows
Read 55.1% of 31991090 rows
Read 56.2% of 31991090 rows
Read 57.3% of 31991090 rows
Read 58.5% of 31991090 rows
Read 59.7% of 31991090 rows
Read 60.9% of 31991090 rows
Read 62.1% of 31991090 rows
Read 63.3% of 31991090 rows
Read 64.5% of 31991090 rows
Read 65.7% of 31991090 rows
Read 66.8% of 31991090 rows
Read 68.0% of 31991090 rows
Read 69.1% of 31991090 rows
Read 70.3% of 31991090 rows
Read 71.3% of 31991090 rows
Read 72.3% of 31991090 rows
Read 72.7% of 31991090 rows
Read 73.8% of 31991090 rows
```

```
Read 75.0% of 31991090 rows
Read 76.2% of 31991090 rows
Read 77.3% of 31991090 rows
Read 78.5% of 31991090 rows
Read 79.6% of 31991090 rows
Read 80.8% of 31991090 rows
Read 82.0% of 31991090 rows
Read 83.2% of 31991090 rows
Read 84.3% of 31991090 rows
Read 85.4% of 31991090 rows
Read 86.6% of 31991090 rows
Read 87.7% of 31991090 rows
Read 88.9% of 31991090 rows
Read 90.0% of 31991090 rows
Read 91.2% of 31991090 rows
Read 92.3% of 31991090 rows
Read 93.5% of 31991090 rows
Read 94.6% of 31991090 rows
Read 95.7% of 31991090 rows
Read 96.8% of 31991090 rows
Read 97.2% of 31991090 rows
Read 98.2% of 31991090 rows
Read 99.5% of 31991090 rows
Read 31991090 rows and 24 (of 24) columns from 4.651 GB file in 00:02:28
```

cleaning the training dataset

```
train_data = df

train_data = subset(train_data, select = -c(hour))

# train_data$date=floor(train_data$hour/100)-141020
# train_data$hour=train_data$hour-14102000-train_data$date*100

train_data[!(C1 %in% c(1005,1002,1010)),C1:=0]
train_data[!(banner_pos %in% c(0,1)),banner_pos:=999]
train_data[!(site_id %in% c("85f751fd","1fbe01fe","e151e245")),site_id:="0"]
train_data[!(site_domain %in% c("c4e18dd6","f3845767","7e091613","7687a86e")),site_domain:="0"]
train_data[!(site_category %in% c("50e219e0","f028772b","28905ebd","3e814130")),site_category:="0"]
train_data[!(app_id %in% c("ecad2386")),app_id:="0"]
train_data[!(app_domain %in% c("7801e8d9","2347f47a","ae637522")),app_domain:="0"]
train_data[!(app_category %in% c("07d7df22","0f2161f8","cef3e649","8ded1f7a")),app_category:="0"]
train_data[!(device_id %in% c("a99f214a")),device_id:="0"]
train_data[!(device_model %in% c("8a4875bd","d787e91b","1f0bc64f")),device_model:="0"]
train_data[!(C15 %in% c(320,300,216,728)),C15:=0]
train_data[!(C16 %in% c(50,250,36)),C16:=0]
train_data[!(C17 %in% c(1722)),C17:=0]
train_data[!(C19 %in% c(35,39,167,47)),C19:=0]
train_data[!(C20 %in% c(-1,100084)),C20:=0]
train_data[!(C21 %in% c(23,79,221)),C21:=0]
train_data=train_data[,-c("device_ip","C14")]
```

Creating a random sample

```
#random sample function
randomSample = function(df,n) {
```

```
  return (df[sample(nrow(df), n),])
}

#creating a random sample
set.seed(99)
trainsample = randomSample(train_data, 1000000)
# View(head(trainsample))
trainsample = subset(trainsample, select = -c(id))
```

removing redundant features (based on iterative running of models) and converting variables to factors

```
# removing redundant columns
trainsample[(site_domain %in% c("f3845767")),site_domain:="0"]
trainsample[(app_id %in% c("f3845767")),app_id:="1"]


#converting variables into factors
# trainsample$date = as.factor(trainsample$date)
# trainsample$hour = as.factor(trainsample$hour)

trainsample$C1 = as.factor(trainsample$C1)
trainsample$banner_pos = as.factor(trainsample$banner_pos)
trainsample$site_id = as.factor(trainsample$site_id)
trainsample$site_domain = as.factor(trainsample$site_domain)
trainsample$site_category = as.factor(trainsample$site_category)
trainsample$app_domain = as.factor(trainsample$app_domain)
trainsample$site_category = as.factor(trainsample$site_category)

# trainsample$app_id = as.factor(trainsample$app_id)

trainsample$app_domain = as.factor(trainsample$app_domain)
trainsample$app_category = as.factor(trainsample$app_category)
trainsample$device_id = as.factor(trainsample$device_id)
trainsample$device_model = as.factor(trainsample$device_model)
trainsample$device_type = as.factor(trainsample$device_type)
trainsample$device_conn_type = as.factor(trainsample$device_conn_type)
trainsample$C15 = as.factor(trainsample$C15)
trainsample$C16 = as.factor(trainsample$C16)
trainsample$C17 = as.factor(trainsample$C17)
trainsample$C18 = as.factor(trainsample$C18)
trainsample$C19 = as.factor(trainsample$C19)
trainsample$C20 = as.factor(trainsample$C20)
trainsample$C21 = as.factor(trainsample$C21)
```

Creating Training and validation datasets

```
#creating training and validation sets
TrainInd <- ceiling(nrow(trainsample)*0.6)
TrainData <- trainsample[1:TrainInd,]
ValData <- trainsample[(TrainInd+1):nrow(trainsample),]
```

Model 1 : Logistic Regression and predicting the results

```
#running logistic Regression
logit = glm(click ~ . , family=binomial(link='logit'), data = TrainData)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
# summary(model)
# logLik(model)
# AIC(model)

#predicting the results
pred_val_logit = predict(logit , newdata = ValData, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
pred_train_logit = predict(logit , newdata = TrainData, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

Computing the Log Loss for Logistic Regression using the Validation datasets

```
print("Log Loss for Logistic Regression")
```

```
## [1] "Log Loss for Logistic Regression"
```

```
val_loss_logit = LogLoss(actual=ValData$click, predicted=pred_val_logit)

print(val_loss_logit)
```

```
## [1] 0.426411
```

Ensembling

Including the predicted results for logistic Regression into the training dataset and Training a decision tree on it. This will enable us to capitalize on both models.

Model 2 : Training a decision Tree Model with the logistic Regression results included

```
#including the logistic regression results

TrainData2 = TrainData
pred_train_logit = as.data.frame(pred_train_logit)
TrainData2$pred_logit = pred_train_logit$pred_train_logit


ValData2 = ValData
pred_val_logit = as.data.frame(pred_val_logit)
ValData2$pred_logit = pred_val_logit$pred_val_logit
```

Using the training set logistic regression probabilities in the decision tree training

```
TrainData2$click = as.factor(TrainData2$click)


# library("rpart")
rpc <- rpart.control(minsplit=20,maxdepth=20,cp=0)
ens_dtree <- rpart(click ~ .,data=TrainData2,control=rpc)

# Find the cp parameter for the best model and prune the tree.
bestcp <- ens_dtree$cptable[which.min(ens_dtree$cptable[,"xerror"]),"CP"]
```

```
# Store the "best" model in out1
ens_dtree2 <- prune(ens_dtree,cp=bestcp)
```

Predicting results

```
pred_val_ens_dtree <- predict(ens_dtree2,newdata=ValData2,type="prob")
pred_train_ens_dtree <- predict(ens_dtree2,newdata=TrainData2,type="prob")
```

Getting the log loss for the ensemble on the validation dataset

```
pred_val_ens_dtree = as.data.frame(pred_val_ens_dtree)


# ValData$click = as.numeric(ValData$click)
val_loss_ens_dtree = LogLoss(actual= ValData$click, predicted = pred_val_ens_dtree$`1`)

print("Log Loss for Ensemble Decision Tree")
```

```
## [1] "Log Loss for Ensemble Decision Tree"
```

```
print(val_loss_ens_dtree)
```

```
## [1] 0.423086
```

Evaluating the test dataset

Creating a prediction function for the test dataset

Reading in the test dataset and reducing features

```
# TestData <- fread('ProjectTestData.csv')
# # dt_trainsample <- fread('trainsample.csv')
# # TestData = 0
#
# test_data = TestData
# TestData = 0
# test_data = subset(test_data, select = -c(hour))
#
#
# # TestData$date=floor(TestData$hour/100)-141020
# # test_data$hour=test_data$hour-14102000-test_data$date*100
#
# test_data[!(C1 %in% c(1005,1002,1010)),C1:=0]
# test_data[!(banner_pos %in% c(0,1)),banner_pos:=999]
# test_data[!(site_id %in% c("85f751fd","1fbe01fe","e151e245")),site_id:="0"]
# test_data[!(site_domain %in% c("c4e18dd6","f3845767","7e091613","7687a86e")),site_domain:="0"]
# test_data[!(site_category %in% c("50e219e0","f028772b","28905ebd","3e814130")),site_category:="0"]
# test_data[!(app_id %in% c("ecad2386")),app_id:="0"]
# test_data[!(app_domain %in% c("7801e8d9","2347f47a","ae637522")),app_domain:="0"]
# test_data[!(app_category %in% c("07d7df22","0f2161f8","cef3e649","8ded1f7a")),app_category:="0"]
# test_data[!(device_id %in% c("a99f214a")),device_id:="0"]
# test_data[!(device_model %in% c("8a4875bd","d787e91b","1f0bc64f")),device_model:="0"]
# test_data[!(C15 %in% c(320,300,216,728)),C15:=0]
# test_data[!(C16 %in% c(50,250,36)),C16:=0]
# test_data[!(C17 %in% c(1722)),C17:=0]
# test_data[!(C19 %in% c(35,39,167,47)),C19:=0]
# test_data[!(C20 %in% c(-1,100084)),C20:=0]
# test_data[!(C21 %in% c(23,79,221)),C21:=0]
```

```
# test_data=test_data[,-c("device_ip","C14")]
#
#
# # removing redundant columns
# test_data[(site_domain %in% c("f3845767")),site_domain:="0"]
# test_data[(app_id %in% c("f3845767")),app_id:="1"]
#
#
# # fwrite(test_data, 'test_short.csv')
```

Factorizing the test dataset and removing further columns

```
#converting variables into factors
# test_data$date = as.factor(test_data$date)
# test_data$hour = as.factor(test_data$hour)

# test_data$C1 = as.factor(test_data$C1)
# test_data$banner_pos = as.factor(test_data$banner_pos)
# test_data$site_id = as.factor(test_data$site_id)
# test_data$site_domain = as.factor(test_data$site_domain)
# test_data$site_category = as.factor(test_data$site_category)
# test_data$app_domain = as.factor(test_data$app_domain)
# test_data$site_category = as.factor(test_data$site_category)

# test_data$app_id = as.factor(test_data$app_id)
#
# test_data$app_domain = as.factor(test_data$app_domain)
# test_data$app_category = as.factor(test_data$app_category)
# test_data$device_id = as.factor(test_data$device_id)
# test_data$device_model = as.factor(test_data$device_model)
# test_data$device_type = as.factor(test_data$device_type)
# test_data$device_conn_type = as.factor(test_data$device_conn_type)
# test_data$C15 = as.factor(test_data$C15)
# test_data$C16 = as.factor(test_data$C16)
# test_data$C17 = as.factor(test_data$C17)
# test_data$C18 = as.factor(test_data$C18)
# test_data$C19 = as.factor(test_data$C19)
# test_data$C20 = as.factor(test_data$C20)
# test_data$C21 = as.factor(test_data$C21)
```

predicting results for logistic regression and appending them to a test dataset copy

```
#
# test_data2 = as.data.table(test_data)
#
# test_data2$pred = predict(logit , newdata = test_data, type = 'response')
#
#
# pred_ens = as.dataframe(pred_test_logit)
#
# test_df =
```