# Captial One Coding Challenge

## Prediction Problem

Yilin Zhao

*Handed In: March 27, 2016*

1. **Extreme Gradient Boosting**

### 1.1 Introduction to XGBoost

I use package XGBoost in R to apply the extreme gradient boosting algorithm, which is an efficient implementation of gradient boosting framework. The advantage of XGBoost package is that it includes efficient linear model solver and tree learning algorithms, and the package can automatically do parallel computation on a single machine which could be more than 10 times faster than existing gradient boosting packages. It also introduces a regularization term to get rid of overfitting as well as control the model complexity.

### 1.2 Data Discription

- There are 254 variables. 250 numeric variables and 4 categorical variables.
- The numeric variables follow normal distribution with mean = 0.
- There are missing values in the numeric variables. f_71 has the largest number of missing values: 132.

### 1.3 Data Preparation for XGBoost

- Build a design matrix as an input of XGBoost because XGBoost can only handle numeric variables.
- Replace all the missing values by the mean of that column.

```
DataProcessing <- function(input){
  for(i in 1:ncol(input)){
    for(j in 1:nrow(input)){
      if(is.na(input[j,i])){
        input[j,i] = round(mean(input[,i], na.rm=T),3)
      }
    }
  }
  input <- model.matrix(~., data = input)
  input <- input[,-1]
  return(input)
}
```

Function of Data Processing

2. **Parameter Tuning and Model Implementation**

### 2.1 Parameter Tuning

Use five different sets of model parameters to train five different models using the same train data set and test data set.

- *eta* controls the learning rate: {0.03, 0.05, 0.2, 0.08, 0.3}

- *Max_depth* is the maximum depth of a tree: {2, 4, 6, 8}

```r
param.change=function(paramlist, new.eta, new.max_depth){
  num=length(paramlist) + 1
  paramlist[[num]]=paramlist[[1]]
  paramlist[[num]]$eta=new.eta
  paramlist[[num]]$max_depth=new.max_depth
  paramlist
}
xg.params=list(
  "eval_metric" = "rmse",
  'lambda' = 0.05,
  "eta" = 0.03,
  "max_depth" = 6
)
param.list=list(xg.params)
param.list=param.change(param.list, 0.05, 8)
param.list=param.change(param.list, 0.2, 4)
param.list=param.change(param.list, 0.08, 6)
param.list=param.change(param.list, 0.3,2)
```

Function of Parameter Change

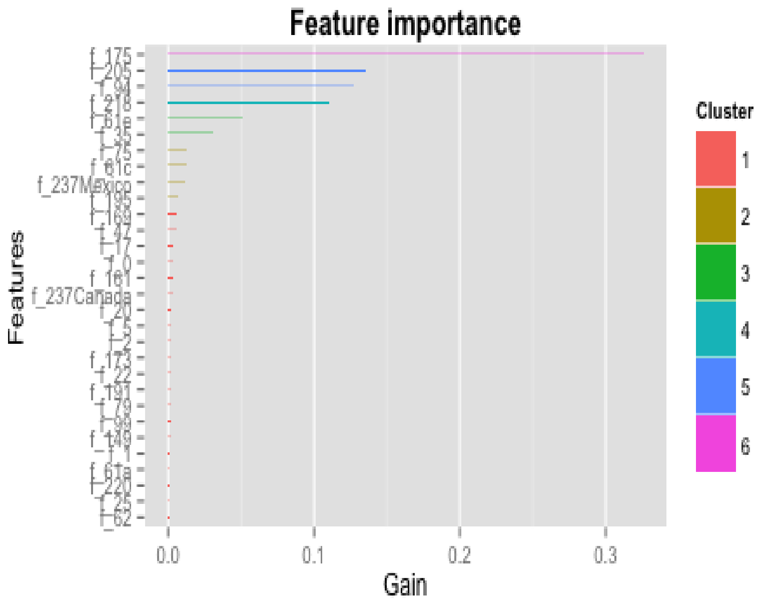
### 2.2 Model Implementation

- Use 5-folds cross validation to get rid of overfitting.
- Record the MSE in each fold using different parameter list.
- If MSE $< 7$ then use the corresponding fitting model to predict the result of test data.
- Use the mean of the predictions as my final prediction.

```r
XGBoost <- function(mytrain_df, i, dtrain, dtest, param){
  newtrain <- as.matrix(subset(mytrain_df, id in list[-i]))
  newtest <- as.matrix(subset(mytrain_df, id in c(i)) )
  dmytrain <- xgb.DMatrix(newtrain[,-1], label=newtrain[,1])
  dmytest <- xgb.DMatrix(newtest[,-1], label=newtest[,1])
  cv.nround <- 250
  bst.cv = xgb.train(param=param, data = dmytrain,
                     nrounds = cv.nround,watchlist=list(validation=dmytest),
                     early.stop.round=10)
  pred_train <- predict(bst.cv, dtrain)
  pred_test <- predict(bst.cv, dtest)
  mse_tmp <- mean((predict(bst.cv,dmytest)-newtest[,1])^2)
  names <- dimnames(newtrain[,-1])[[2]]
  return(list(pred_train,pred_test,mse_tmp,names,bst.cv))
}
```

## 3. Conclusions

- I get lowest cv-test MSE by using $eta = 0.2$ and $Max\_depth = 4$.
- The final MSE for the train data set is 1.175894.
- I use information gain as a measurement of variable importance. Variable f_175 has the most important effect on target.
- I also get high information gain on variables f_94, f_205, f_35, and f_218. As for categorical variables.
- f_61 with level 'e' and 'c' is more important than other levels. f_237 with level 'Mexico' and 'Canada' also have important effect on the target.



Variable Importance Plot