Project

Laukik Upadhye (01833608)

Data:

It is the data from a website where user click pattern is captured. It has 87 variables captured for 4107 users. It also includes a dependent variable, which says whether the user clicked on Buy button or not. Buy has data as 0's and 1's. 0 means used didn't click on buy and 1 means clicked.

This is a biased data. As we have 3859 as 0's and just 248 as 1's.

```
table(train$Buy)
```

```
   0    1
3859  248
```

Models used:

Random Forest

In random forest we create n number of trees (ntree) with considering specific number of variables (mtry) selected randomly for each decision tree. The classification result is based on max count of result given by all trees. In case of our data, simple random forest (without re sampling) providing ~50% accuracy.

To improve the accuracy, we need to perform resampling and increase the data for one's and decrease the records for zero's in the data set. Apart from that, we need to use the cross validation to divide the data in chunks and shuffle the selection.

```
1. trainCont <- trainControl(method = "repeatedcv",
2.                           number = 10,
3.                           repeats = 5,
4.                           verboseIter = FALSE,
5.                           sampling = "smote")
6. set.seed(608)
```

```
#options(warn=1)
rf <- train(Buy ~.,
            data = data,
             method = "rf",
             preProcess = c("scale", "center"),
             trControl = trainCont)
```

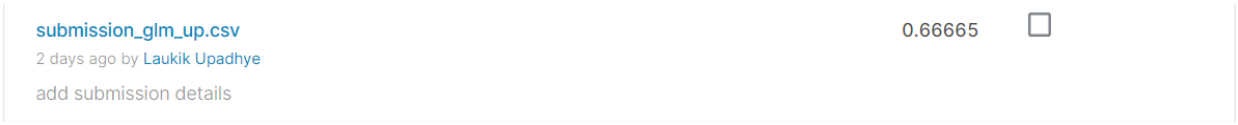
With this resampling and cross validation result got improved with almost ~8% to 58%. Here I have used scaling of data to get all the variables on same scale.

Logistic Regression:

This model is best when working with large number of independent variables. Even for this model we need to create resampling and this time I used sampling as up (increase the data for 1's).

```
trainCont <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 5,
                          verboseIter = FALSE,
                          sampling = "up")
set.seed(608)
#options(warn=1)
glm <- train(Buy ~.,
            data = data,
             method = "glm",
             preProcess = c("scale", "center"),
             trControl = trainCont)
```

With this model, the prediction improved by ~8% to 66%



XGBoost model:

Boosting is the concept of providing learning information from one model to other. The purpose for this is that the next model will learn from the misclassification of previous model. As it is a sequential process it is slow in building train model however, it provides comparatively better predictions. As at each iteration train error reduces.

XGBoost Tuning parameters:

It has many hyper parameters to control the behavior of model.

1. nrounds: It is like number of trees to create.
2. Eta: It sets the learning rate of the model. At every iteration what should be the minimum learning rate is decided by this parameter.
3. Gamma: This is used to avoid non-performing variables from creating decision model.
4. Max_depth: It decides what should be the max depth of the tree.
5. Min_child_weight: It is used to set the minimum number of instances at child nodes.
6. Subsample: It controls the number of samples provided to the tree

Cross validation in XGBoost:

Cross validation can be done with train function from caret library and xgboost has its own cv function. With cross validation, it provides best number of iteration as well as the min train error.

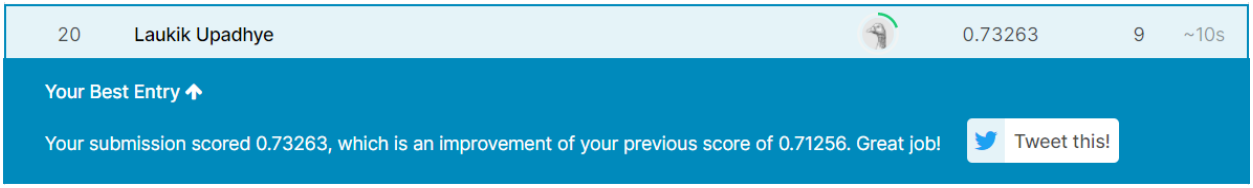
```
1. para <- list(booster = "gbtree",
2.                 objective = "binary:logistic",
3.                 eta=0.07,
4.                 gamma=0.01,
```

```
                max_depth=6,
                min_child_weight=1,
                subsample=1,
                colsample_bytree=1)
set.seed(608)
xgbtraincv <- xgb.cv( params = para, data = trainMat,
                  nrounds = 410,
                  nfold = 5,
                  showsd = T,
                  stratified = T,
                  early_stopping_rounds = 21,
                  maximize = F)
#xgbtraincv
set.seed(608)
xgb_train <- xgb.train (params = para, data = trainMat, nrounds
  = 15)
```

Result obtained

With this model I have achieved maximum of 73%. This is highest accuracy achieved so far.

Rank on Kaggle



Summary

In compare to all the models, xgboost gave the maximum accuracy. The parameters used to tune the model and the cross validation helped in the process. Additionally, the accuracy can be improved by adding the samplings to the train data set.

Conclusion

When using Random Forest or Logistic Regression it gave 58% and 66% respectively as the maximum accuracy. By using boosting method, it has increased the accuracy to 73% which

show the highest among all the model. Boosting has great impact on training the model. As at every iteration train error dropped and resulting in higher accuracy. Cross validation provided the best iteration number for building model.

Reflective statements:

1. Resampling is better than using default class ratio, to handle biased data
2. Cross validation is best way to tune any model
3. Boosting method increases model learning and give best version of train model