TFI_Restaurant_Project_Documentation

CONTENTS

ABSTRACT

Kaggle is an open-competition, open-community website for data scientists to congregate and compete in solving big data challenges. In this report, we explore one Kaggle competition hosted by Tab Food Investments (TFI) where participants are asked to predict annual revenue of Turkish restaurants. The data set itself contains many inherent problems which we present insight and offer potential solutions to. Using a simple ensemble method between Random Forest and Support Vector Machine, we were placed with a root mean squared error of 1744398.

INTRODUCTION

Kaggle is an online website designed to provide data scientists an open platform to learn, collaborate and compete on big data challlenges often hosted by corporations with large data sets. The primary focus of Kaggle is on predictive analytics where contestants are often given a training data set to fit their model and a test data set to submit their predictions to be evaluated. The evaluation process consists of collecting a subset of the test set prediction and calculating a score, usually on the errors of the predictions. Multiple submissions from each user are allowed and the best score from all submissions are displayed on the public leaderboards.

In this report, we ourselves become competitors and focus on the TFI Kaggle competition. The report will document our analysis and discovery of the training/test data sets given to us by TFI as well as the quantitative models we employ to predict the test set. Supporting R code will be provided to show detailed procedures and the report is presented in the same chronological order as we have approached the problem. The structure of this paper is as follows: we introduce and describe the TFI competition/data set ; problems with data set variables and solutions and details of the primary models within the study; ensemble methods are discussed within sections and presents our results of our actual test set performance along with the models and changes employed; final section concludes along with a brief discussion on limitations and future consideration.

Purpose

Finding a mathematical model to increase the effectiveness of investments in new restaurant sites would allow TFI to invest more in other important business areas, like sustainability, innovation, and training for new employees.
Using demographic, real estate, and commercial data, this competition challenges you to predict the annual restaurant sales of 100,000 regional locations. With over 1,200 quick service restaurants across the globe, TFI is thecompany behind some of the world's most well-known brands: Burger King, Sbarro, Popeyes, Usta Donerci, and Arby's. They employ over 20,000 people in Europe and Asia and make significant daily investments in developing new restaurant sites. New restaurant sites take large investments of time and capital to get up and running. When the wrong location for a restaurant brand is chosen, the site closes within 18 months and operating losses are incurred.

State-of-the-art

The problem encountered by TFI is deciding when and where to open new restaurants, which is largely a subjective process based on the personal judgment and experience of development teams. This subjective data is difficult to accurately extrapolate across geographies and cultures.
New restaurant sites take large investments of time and capital to get up and running. When the wrong location for a restaurant brand is chosen, the site closes within 18 months and operating losses are incurred. Finding a mathematical model to increase the effectiveness of investments in new restaurant sites would allow TFI to invest more in other important business areas, like sustainability, innovation, and training for new employees. Using demographic, real estate, and commercial data, this competition challenges you to predict the annual restaurant sales of 100,000 regional locations.

Method

The TFI Kaggle competition is a supervised learning regression problem where the objective is to develop a model and predict the revenue of the restaurants .Turkish restaurant revenues collected in a given year as data set , These restaurants are spread in different cities and offers different services.
In data set train data is much smaller than the test data and has 37 attributes which are obfuscated form wherein we have no information regarding their type(categorical or numeric) and the importance of the attributes is also unknown.
Opening date attributes tells the opening of the restaurant in a particular year and as opening date cannot be assumed as factor we will take the difference in years by subtracting the opening date with the current date.
The train data and test data have different Type levels as a result while predicting the model we will get an error while predicting. The disparity between the features for the training set and test set where the test set actually contains more information than the training set.

Evaluation

The score measure presented in the competition is the root mean squared error of the test set revenue.
RMSE =

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Where ^yi is the predicted revenue of the ith restaurant and y is the actual revenue of the ith restauarant

Method 1

Data Pre-Processing

Data pre-processing is the important step in any model building as it contributes in increasing the model accuracy

- Check for outliers
- Check for the missing values
- Revenue Distribution
- Parsing Open Date
- Feature selection

Model Building

RF+KFOLD+SVM

RF+KFOLD using important features derived from random forest

Ensemble model using average method of above models

Method 2

Data Pre-Processing

Data pre-processing is the important step in any model building as it contributes in increasing the model accuracy

- Check for outliers
- Check for the missing values
- Revenue Distribution
- Parsing Open Date
- Class imbalances
- Feature selection

Model Building

We can use PCA for choosing important features which explain 90% variance and
then build the below models .

Random Forest

KNN

Method 3

Data Pre-Processing

Data pre-processing is the important step in any model building as it contributes in increasing the model accuracy

- Check for outliers
- Check for the missing values
- Revenue Distribution
- Parsing Open Date
- Class imbalances
- Feature selection

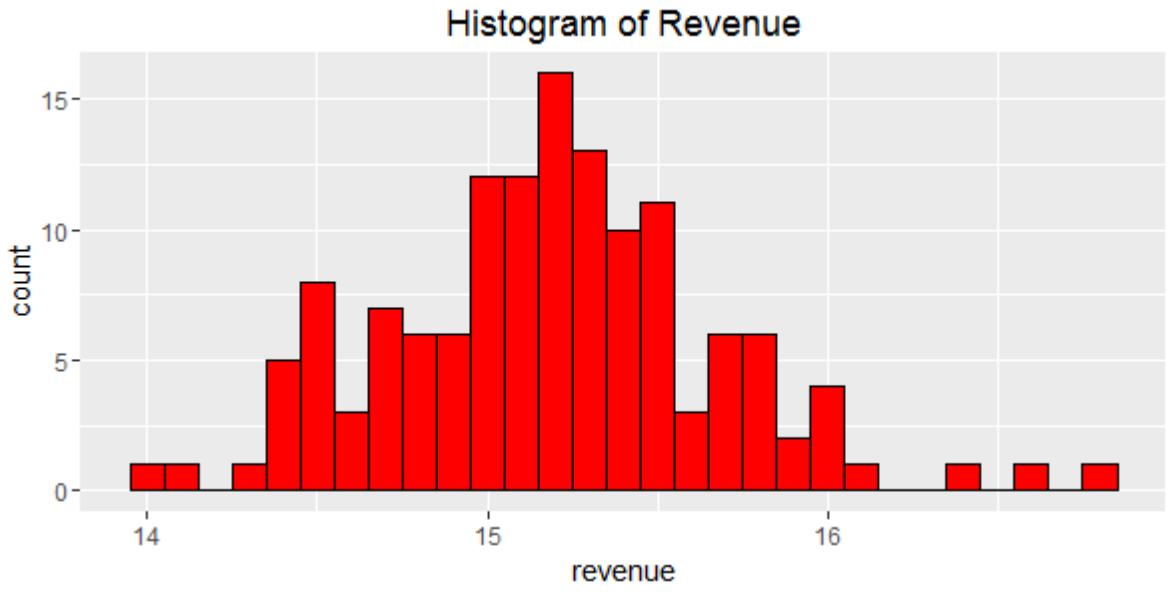
Model Building

Linear Model + Lasso and Ridge regression

DATA

File Description
train.csv : The training set. Use this dataset for training your model.
test.csv: The test set. To deter manual "guess" predictions, Kaggle has supplemented the test set with additional "ignored" data. These are not counted in the scoring.
sampleSubmission.csv : A sample submission file in the correct format
Field Description
Id : Restaurant id.
Open Date : opening date for a restaurant
City : City that the restaurant is in. Note that there are unicode in the names.
City Group : Type of the city. Big cities(class "1" in our processed data), or Other(class "0" in our processed data).
Type : Type of the restaurant. FC: Food Court(class "2" in our processed data), IL: Inline (class "1" in our processed data), DT: Drive Thru(class "3" in our processed data), MB: Mobile(class "4" in our processed data)
P1, P2 - P37 : There are three categories of these obfuscated data. Demographic data are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution,development scales. Real estate data mainly relate to the m2 of the location, front facade of the location, car parkavailability. Commercial data mainly include the existence of points of interest including schools, banks, other QSR operators.
Revenue : The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the target of predictive analysis. Please note that the values are transformed so they don't mean real dollar values.

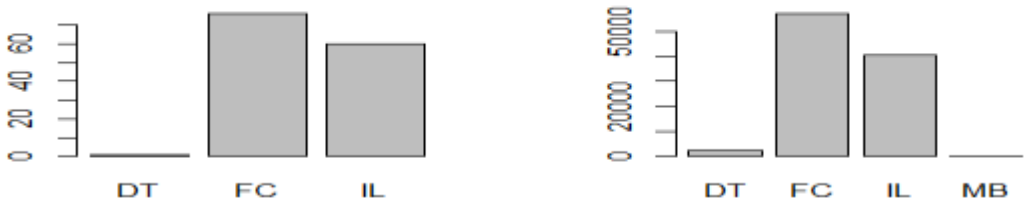
Data Engineering
Revenue Distribution :
Plotting a simple histogram of the revenue field show that it is not normally distributed but rather has a long tail toward the right. By taking log of the revenue field and obtain something that is more similar to a normal distribution.



If this distribution for revenue holds in the test set, transforming the variable before training models will improve
performance vastly, which holds in the training set

Checking for the missing values :
we checked if there are any missing values. And we did not find any, so no need of imputation here.

Class mismatch problem :
The unaccounted problem is simply the disparity between the features for the training set and test set where the test set actually contains more information than the training set.
This problem directly affects the categorical versus continuous problem which is discussed in the next section.
Within the training set, features that are treated as categorical can pose potential problems when being applied to the test set. These features are Type. Types pans FC ,IL and DT within the training set but is missing MB which is present within the test set. Most supervised learning models fitted through the training set will encounter errors due to missing coefficients for the additional classes. We propose two treatments for the Type and City issue below.



Parsing Date:

Calculating the no of days the restaurant was opened from the date of opening till current date.

Feature Selection:

PCA
Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize
However one needs to note that while computing PCA one cannot explain the attributes to the business user as while doing PCA the attributes are explained in term of variation as shown below.

K-fold cross validation
In k-fold cross-validation, the original sample is randomly partitioned into k equal sized sub samples. Of the k subsamples, a single sub sample is retained as the validation data for testing the model, and the remaining k – 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k sub samples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation.

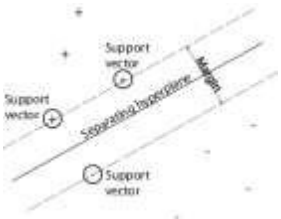
Results and Analysis

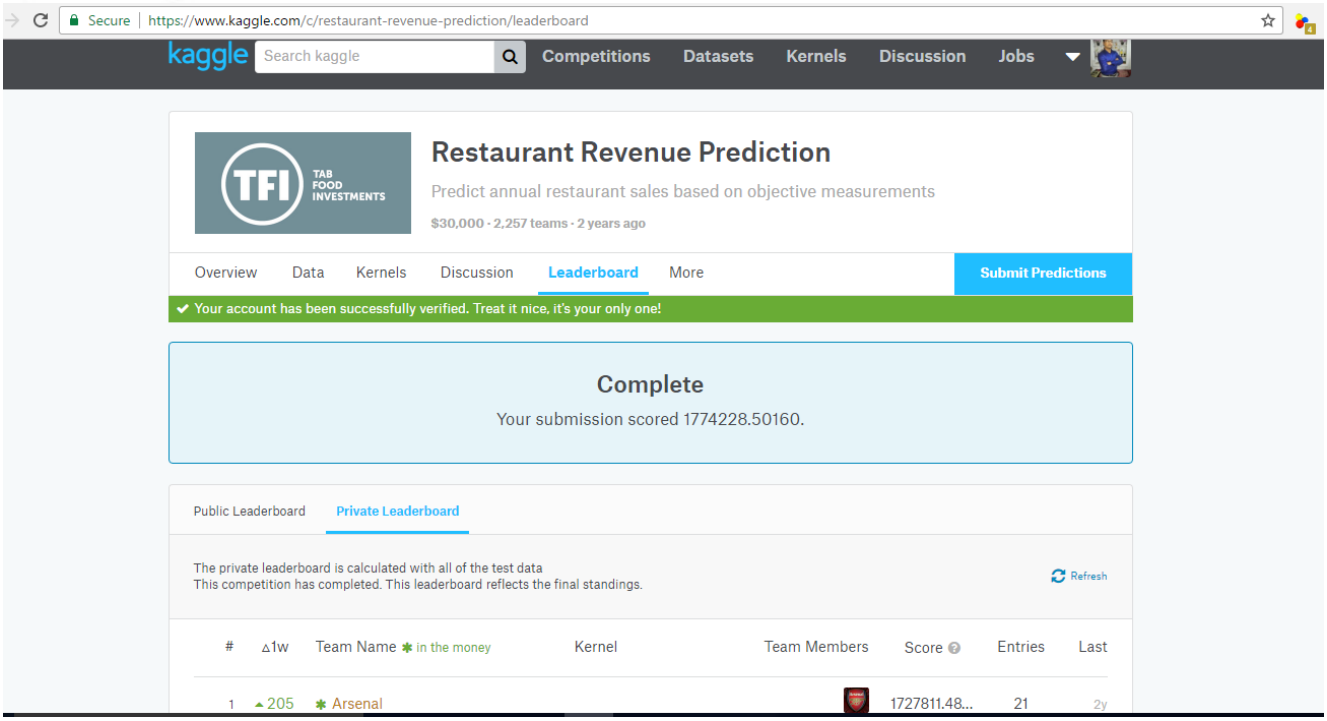
Method 1 = Combination of RF for selecting important features plus different Models

RF+KFOLD+SVM

a) **SVM**

Support vector machines are linear classification models that attempt to perform classification by separating the
classes with a hyperplane, such that the division between classes as wide as possible. Given data in the form
(xi,yi),xi ∈ Rp,yi ∈ −1,1 i.e. xi is the feature vector, and yi is the class the observation belongs to. In the case where
the data is linearly separable, the SVM wants to find two hyperplanes separating the data such that the distance
between them is maximized.



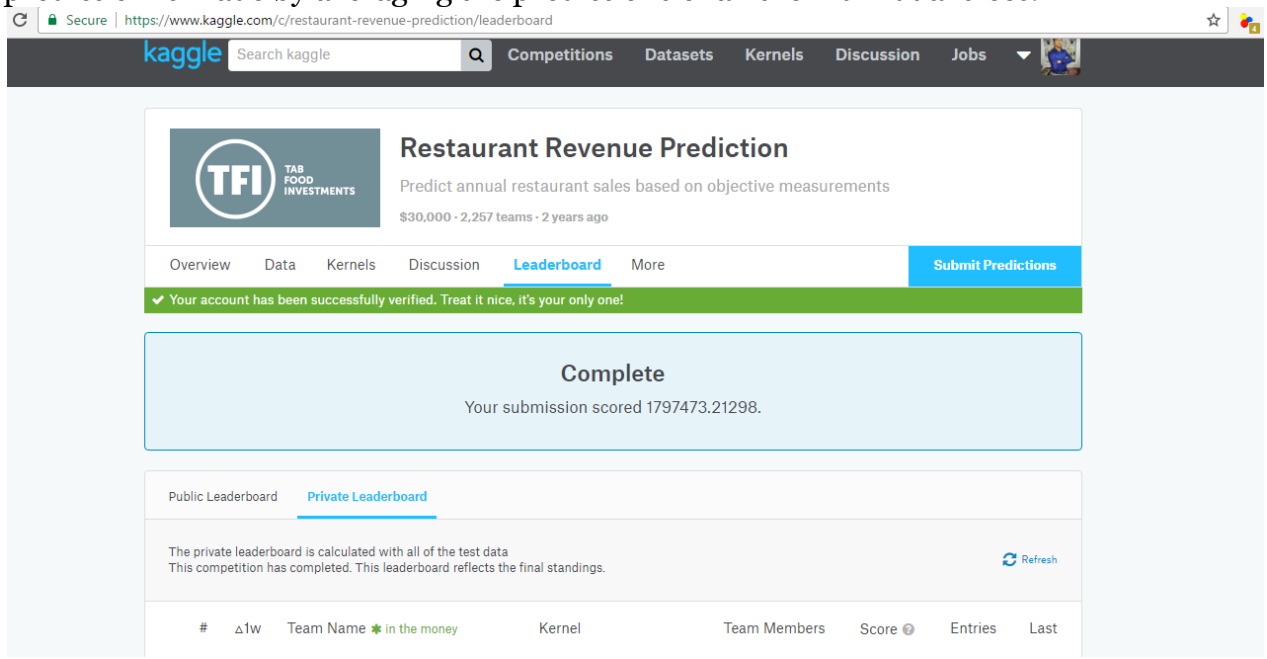


b) RF+KFOLD using important features derived from random forest

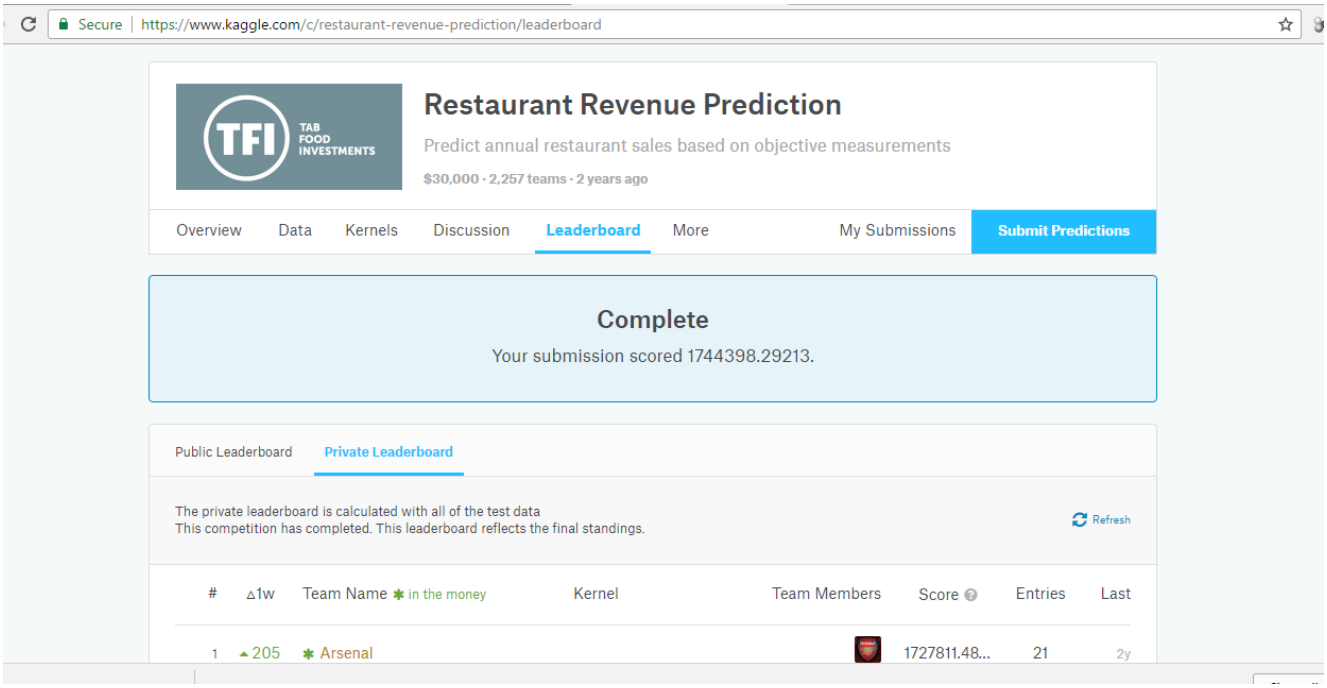
One of the models we are focusing is random forest. They are promising because they run efficiently on large datasets and flexibility.

A random forest is an ensemble of decision trees created using random variable selection and bagging.For each individual tree, a random sample with replacement of training
data is used for training.At each node,split is created by only looking at random subset of variables.

Commonly used number for each split is square root of the number of variables.The prediction is made by averaging the predictions of all the individual trees.



c) Ensemble using average method

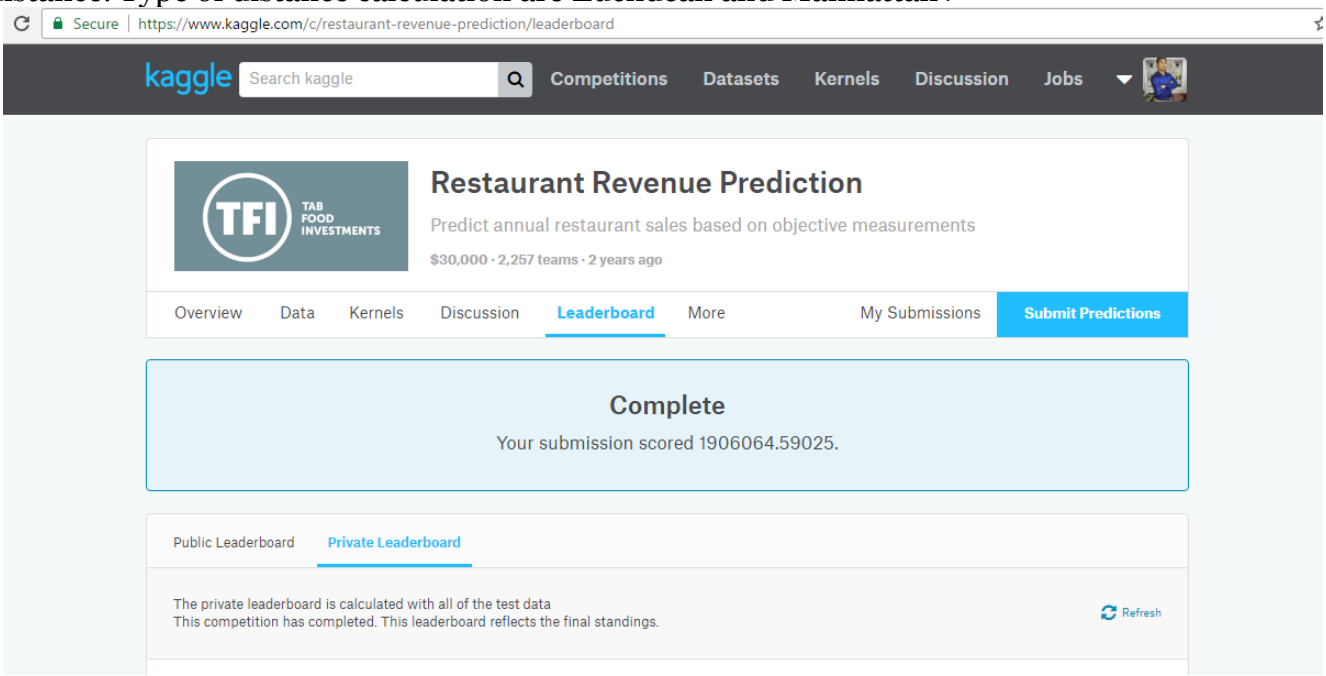


Method 2 = Combination of PCA for selecting attributes with 90% variation + Different Models

Combination

a) PCA+RF b) PCA+KNN

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbours. KNN regression uses these two ways of calculating the distance. Type of distance calculation are Euclidean and Manhattan .



sampleSubmission_final_pca_rf.csv 1961282.51116 1962895.81898

2 days ago by K Kishore Kumar

TFI Kaggle competition can be framed as a supervised learning problem where the objective is to develop a model and a set of preprocess procedures to accurately predict a cross-sectional sample of Turkish restaurant revenues collected in a given year. Method used is implemented PCA for important features and used those for building model like RF.

Method 3= Linear model

Combination

a) Linear Model + Lasso and Ridge regression

Linear regression is a statistical procedure for predicting the value of a dependent variable (Revenue) from an independent variable when the relationship between the variables can be described with a linear model.
A linear regression equation can be written as Yp= mX + b,
Where Yp is the predicted value of the dependent variable,
m is the slope of the regression line and b is the Y-intercept of the regression line.

```
226
227 library(glmnet)
228 # fit model
229 fit2=glmnet(trainL,y,lambda=cv$lambda.min,alpha=0)
230 predict(fit2,trainL)
231 library(DMwR)
232 RIDGEtrain = regr.eval(y, predict(fit2,trainL))
233 RIDGEtest = regr.eval(ytest, predict(fit2,testL))
234 RIDGEtrain
235 RIDGEtest
236 #Model Selection
237 coef(fit2)
238 cv.ridge=cv.glmnet(trainL,y,alpha=0)
239 plot(cv.ridge)
240 coef(cv.ridge)
241
242 ###################################################################
243
244
245 finalerros <- data.frame(rbind(LASSOtrain,LASSOtest,
246                                RIDGEtrain,RIDGEtest))
247 finalerros
```

228:9 (Untitled)

Console G:/Project/test.csv/

```
P374                 2.813808e-03
P375                -4.183791e-04
P376                -2.945470e-03
P378                 .
revenue              1.766566e-02
> finalerros <- data.frame(rbind(LASSOtrain,LASSOtest,
+ RIDGEtrain,RIDGEtest))
> finalerros
                  mae          mse       rmse         mape
LASSOtrain 0.01241992 0.0004832456 0.02198285 0.0008209368
LASSOtest  0.08824402 0.0118138379 0.10869148 0.0057887049
RIDGEtrain 0.01241992 0.0004832456 0.02198285 0.0008209368
RIDGEtest  0.08824402 0.0118138379 0.10869148 0.0057887049
>
```

Appendice

Method 1 = Combination of RF for selecting important features plus different Models

RF+KFOLD , RF+KFOLD+SVM and Ensemble using average method

```
##############################################################################
########################
## Set the current Working Directory

rm(list = ls())
setwd("G:/Project/test.csv")

###############################################################################


## Data Preprocessing
##############################################################################
#################
install.packages("car")
library(car)

library(MASS)
library(caret)
## reading file

##############################################################################
########################
### Understanding the train data ###
dataTrain=read.csv("train.csv",header=T,sep=",")
summary(dataTrain)

str(dataTrain)
dim(dataTrain)
```

```
names(dataTrain)
sum(is.na(dataTrain)) ## '0' No missing values
#############################################################################
########################

#############################################################################
########################
### Understanding the test data ###
dataTest=read.csv("test.csv",header=T,sep=",")
summary(dataTest)
str(dataTest)
dim(dataTest)
names(dataTest)
sum(is.na(dataTest)) ## '0' No missing values


#############################################################################
########################
## Combine train and test dataset for analysis without target variable "revenue"
## As the data in train dataset is small , we need to combine the both
## to ensure there will not be any class imbalance
revenue <- dataTrain$revenue
dataTrain <- dataTrain[,-c(1,3,43)]

Id <- dataTest$Id
dataTest <- dataTest[,-c(1,3)]
lgrevenue <- log10(revenue)
dataFinal=rbind(dataTrain,dataTest)

names(dataFinal)
str(dataFinal)
dim(dataFinal)
colnames(dataFinal)
```

```
## Converting the Open date to Number of years from current day
dataFinal$Open.Date=Sys.Date()-as.Date(as.character(dataFinal$Open.Date),'%m/%d/%Y')
dataFinal$Open.Date=as.numeric(dataFinal$Open.Date)/365
names(dataFinal)[names(dataFinal) == 'Open.Date'] <- 'NoOfYrs'

#Renaming DT to FC and MB to IL
plot(dataFinal$Type)
dataFinal$Type<- recode(dataFinal$Type,"'DT'='FC'")
dataFinal$Type<- recode(dataFinal$Type,"'MB'='IL'")
plot(dataFinal$Type)
str(dataFinal$Type)
levels(dataFinal$Type)
sum(is.na(dataFinal))
str(dataFinal)
##Seperate Numeric and Categorical attributes
dataNum=dataFinal[,c(1,5,6,7,16,29,30,31,32)]
dataCat=dataFinal[,c(2,3,4,8,9,10,11,12,13,14,15,17,18,19,20,21,22,23,24,25,26,27,28,

                     33,34,35,36,37,38,39,40)]

dataCat=data.frame(apply(dataCat,2,function(x){as.factor(x)}))
str(dataCat) ## All Categorical variables converted to Factor

## Standardize the Numercial attributes
library(vegan)
dataNumStd <- decostand(dataNum,"standardize")
summary(dataNumStd)

## Create dummies for the Categorical variables
library(dummies)
function(x){
  dataCatDummy=data.frame(dummy(dataCat$x))
```

```
  dataCat=subset(dataCat,select=-c(x))
  dataCat=data.frame(dataCat,dataCatDummy)}
dataCatDummy=dummy.data.frame(dataCat) ## Create dummies for Categorrical variables

## Combine standardized Numerical and Categorical data frames
dataFinal=data.frame(dataNumStd,dataCatDummy)
str(dataFinal) ## 257 Variables
dim(dataFinal)
nrow(dataFinal)
names(dataFinal)

## Seperate train and test datasets
Train <- dataFinal[1:137, ]
Test <- dataFinal[138:nrow(dataFinal), ]
Train = data.frame(Train,revenue) ## Binding the revenue column from train datase
names(Train)[names(Train) == 'dataTrain.revenue'] <- 'revenue'
Train$revenue=log(Train$revenue)  ## Converting revenue to log as the value looks large
dim(Train)
dim(Test)
names(Train)
names(Test)
nrow(Train)
nrow(Test)
sum(is.na(Train))
sum(is.na(Test))
Train[1,]
## Identify the outliers in Revenue and remove them for analysis
## After removing outliers Adjusted R square was positive
boxplot(Train$revenue)
abline(h = mean(Train$revenue), lty=2)
IQ = IQR(Train$revenue)
```

```
Q1 = quantile(Train$revenue,0.25)

Q3 = quantile(Train$revenue,0.75)
lowerinnerfence =  Q1 - 1.5*IQ
upperinnerfence = Q3 + 1.5*IQ
lowerouterfence = Q1 - 3*IQ
upperouterfence = Q3 + 3*IQ
outliers = boxplot(Train$revenue, plot=FALSE)$out

#Extract the outliers from the original data frame
Train=Train[!(Train$revenue %in% outliers),]

#### Split the data into Train and Test sets
smp_size <- floor(0.7 * nrow(Train))
set.seed(123)
tr_data <- sample(seq_len(nrow(Train)), size = smp_size)
train <- Train[tr_data, ]

test <- Train[-tr_data, ]

##############################################################################
#############


## Model building

##############################################################################
#############
## Normal multivariate regression
## Was unsatisfactory - not working at all
MultLM <- lm(train$revenue~., data=train,na.action=NULL)
summary(MultLM)
```

```
## Model evaluation
library(DMwR)
Lmpred <- fitted(MultLM)

regr.eval(train$revenue,fitted(MultLM))  #error on train data
pred <- predict.lm(MultLM, test)
regr.eval(test$revenue, pred)#error on test data



##############################################################
###################
##Let us try Non Linear models
## Random Forests
##############################################################
###################
library(randomForest)


revenue_rf <- randomForest(revenue ~ .,
data=train,trControl=trainControl(method="repeatedcv",number=10,repeats = 5),
                keep.forest=TRUE, strata=T,ntree=50,replace=T)
## Adding sample,replace and ntree=50,

# View results and understand important attributes
print(revenue_rf)
revenue_rf$predicted
revenue_rf$importance # gives 1st column (accuracy will reduce if imp var are removed)
round(importance(revenue_rf), 2)

# Extract and store important variables obtained from the random forest model
Imp_revenue_rf <- data.frame(revenue_rf$importance)
Imp_revenue_rf <- data.frame(row.names(Imp_revenue_rf),Imp_revenue_rf[,1])
colnames(Imp_revenue_rf) = c('Attributes','Importance')
Imp_revenue_rf <- Imp_revenue_rf[order(Imp_revenue_rf$Importance , decreasing =
```

```
TRUE),]
Imp_revenue_rf <- Imp_revenue_rf[1:6,]



# plot (directly prints the important attributes)
varImpPlot(revenue_rf)

# Predict on Train data
pred_model_train <-predict(revenue_rf,train[,-258], norm.votes=TRUE)
rf_pred <- pred_model_train
result_train <- data.frame(train$revenue,pred_model_train);

result_train
RMSETrain=mean((result_train[1] - result_train[2])^2)
RMSETrain
## 0.03663608 - without replace
## 0.02706133 - with replace

# Predicton Test Data
pred_model_test <-predict(revenue_rf,test[,-258], norm.votes=TRUE)
result_test <- data.frame("actual _values"= test$revenue,pred_model_test);
result_test
RMSETest=mean((result_test[1] - result_test[2])^2)
RMSETest
## 0.1369601 - without replace
## 0.1262862 - with replace

##############################################################
##################
## Create sampleSubmission with Random Forest Model
dataTest<-cbind(Id,dataTest)
names(dataTest)
pred_model_Test <-predict(revenue_rf,Test, norm.votes=TRUE)
```

```
sampleSubmission <- data.frame(dataTest[1],exp(pred_model_Test));## As the revenue was
## converted to log in the beginning the prediction will be exponential
names(sampleSubmission)[names(sampleSubmission) == 'exp.pred_model_Test.'] <-
'revenue'
write.csv(sampleSubmission,"sampleSubmission_RF1.csv",row.names = F)

##################################################################
##################

##################################################################
###################
## Let us try SVM
##################################################################
###################
#### Split the data into Train and Test sets
library(tm)
library(e1071)
revenue_svm = svm(revenue~.,data=train,scale=F,epsilon=0.4,cost=3)
summary(revenue_svm)
pred_svm_train=predict(revenue_svm,train)
svm_pred <- pred_svm_train
res_svm_train=data.frame(train$revenue,pred_svm_train)
RMSETrainSvm=mean((res_svm_train[1] - res_svm_train[2])^2)
RMSETrainSvm

pred_svm_test=predict(revenue_svm,test)
res_svm_test=data.frame(test$revenue,pred_svm_test)
RMSETestSvm=mean((res_svm_test[1] - res_svm_test[2])^2)
RMSETestSvm

#### Trying to tune the SVM for optimum parameters

tuneResult <- tune(svm, revenue~.,data = Train,scale=F,
                   ranges = list(epsilon = seq(0,0.5,0.1), cost = 1:3))
```

```
print(tuneResult)

plot(tuneResult) ## Darker the region, lower the error
```

```
## Create sampleSubmission with SVM Model
pred_svm_Test <-predict(revenue_svm,Test, norm.votes=TRUE)
sampleSubmission <- data.frame(dataTest[1],exp(pred_svm_Test));## As the revenue was
## converted to log in the beginning the prediction will be exponential
names(sampleSubmission)[names(sampleSubmission) == 'exp.pred_svm_Test.'] <-
'revenue'
write.csv(sampleSubmission,"sampleSubmission_SVM.csv",row.names = F)
############################################################################
##################

##### Ensemble learning ########################

#################### average
rf <- exp(pred_model_Test)
svm <- exp(pred_svm_Test)
finalensemble <- rf + svm
finalensemble <- finalensemble/2


 id=0:99999
 id<-as.data.frame(id)
 sample_submission<-cbind(id,finalensemble)
 head(sample_submission)
 write.csv(sample_submission,"sampleSubmission_final_ensemble.csv",row.names = F)
###========================================================
==================================================
```

Method 2 = Combination of PCA for selecting attributes with 90% variation + Different Models

PCA+RF , PCA+KNN

```
#####################################################################
########################
rm(list = ls())
setwd("G:/Project/test.csv")
#setwd("G:/Project/train.csv")
#####################################################################
########################

## Data Preprocessing

#####################################################################
########################
install.packages("car")
library(car)
library(MASS)
library(caret)
## reading file
#####################################################################
####################
### Understanding the train data ###
train_data<-read.csv("train.csv",header = T,sep = ",")
dim(train_data)
names(train_data)
summary(train_data)
str(train_data)
sum(is.na(train_data)) ## 'o' No missing values
```

```
#############################################################################
####################
### Understanding the test data ###

test_data<- read.csv("test.csv", header = T, sep = ",")
dim(test_data)
names(test_data)
summary(test_data)
str(test_data)
sum(is.na(train_data)) ## '0' No missing values
#############################################################################
####################
## Plotting Type data from Test and Train data ###

par(mfrow=c(1,2))
plot(train_data$Type)
plot(test_data$Type)
#############################################################################
####################

## Removing revenue,Id and City from Train data #####

revenue <- train_data$revenue
train_data <- train_data[,-c(1,3,43)]

## Removing Id and City from Test data #####

test_data <- test_data[,-c(1,3)]
lgrevenue <- log10(revenue)
#############################################################################
###################
## Combine train and test dataset for analysis without target variable "revenue"
##to ensure there will not be any class imbalance
```

```
newdata <- rbind(train_data,test_data)
names(newdata)
str(newdata)
dim(newdata)
colnames(newdata)
grep("P35",colnames(newdata))

###### calculating number of years restaurant is open##########################
today=Sys.Date()
newdata=cbind(newdata,today)
class(today)
head(newdata$Open.Date)
newdata$Open.Date=as.Date(newdata$Open.Date, format = "%m/%d/%Y")


library(lubridate)

newdata$open_year=as.numeric(year(newdata$Open.Date))
newdata$current_year=as.numeric(year(today))
class(newdata$open_year)

newdata$diffyears=(newdata$current_year)-(newdata$open_year)
head(newdata$diffyears)
str(newdata)
names(newdata)
newdata$today
newdata$open_year

#newdata=newdata[,-c(1,2,3,45,44,43)]
str(newdata$Type)
dim(newdata)
names(newdata)
```

```
par(mfrow=c(1,2))
grep("today",colnames(newdata))
newdata=newdata[,-c(1,41,42,43)]



#################################################################
##################
#Renaming DT to FC and MB to IL
plot(newdata$Type)
newdata$Type<- recode(newdata$Type,"'DT'='FC'")
newdata$Type<- recode(newdata$Type,"'MB'='IL'")

plot(newdata$Type)

str(newdata$Type)

levels(newdata$Type)
# do NA for MB ,DT and do conversion using knn..
## whole data



sum(is.na(newdata))
##Seperate Numeric and Categorical attributes
dataNum=newdata[,c(4,5,6,15,28,29,30,31,40)]
dataCat=newdata[,c(1,2,3,7,8,9,10,11,12,13,14,16,17,18,19,20,21,22,23,24,25,26,27,32,33,34,3
5,36,37,38,39)]
dataCat=data.frame(apply(dataCat,2,function(x){as.factor(x)}))
str(dataCat)
#str(dataNumStd)
str(newdata$Type)
#sum(is.na(dataNumStd))
```

```
#Performing Normalization for Numeric data
library(vegan)
dataNumStd <- decostand(dataNum,"standardize")
summary(dataNumStd)



## Create dummies for the Categorical variables
library(dummies)
function(x){
  dataCatDummy=data.frame(dummy(dataCat$x))
  dataCat=subset(dataCat,select=-c(x))
  dataCat=data.frame(dataCat,dataCatDummy)}
dataCatDummy=dummy.data.frame(dataCat) ## Create dummies for Categorical variables
#############################################################
###################

## Combine standardized Numerical and Categorical data frames

dataFinal=data.frame(dataNumStd,dataCatDummy)
str(dataFinal) ## 257 Variables
dim(dataFinal)
nrow(dataFinal)
names(dataFinal)
#############################################################
#################
## Seperate train and test datasets
Train <- dataFinal[1:137, ]
Test <- dataFinal[138:nrow(dataFinal), ]
class(lgrevenue)

str(Train)
str(Test)
# Train = data.frame(Train,lgrevenue) ## Bind the revenue column after pca
```

```
#str(Train)
Train[1,]
nrow(Train)
sum(is.na(Train))



names(Train)
dim(Train)
dim(Test)
#names(Train)[names(Train) == 'lgrevenue'] <- 'revenue'
nrow(Train)
nrow(Test)

## using pca

str(Train)
names(Train)
train_merged<-Train[,-
c(22,43,55,93,102,114,118,123,124,150,181,186,215,222,227,231,241,246)]   ##   all   are   0
removing
#grep("P36.8",colnames(Train))
dim(train_merged)
#View(Train)
names(train_merged)
pca_model<-prcomp(train_merged,scale. = T)
summary(pca_model)
str(pca_model)
final_train<-data.frame(pca_model$x)
names(final_train)
dim(final_train)
final_train_data<-final_train[,1:60]
dim(final_train_data)
names(final_train_data)
str(final_train)
```

```
final_test<-predict(pca_model,Test)
str(final_test)
class(final_test)
final_test<-data.frame(final_test)
summary(final_test)
names(final_test)
summary(final_test)

final_test<-final_test[,1:60]
str(final_test)

str(final_train_data)



#Train$revenue=log(Train$revenue)  ## Converting revenue to log as the value looks large

#### Split the data into Train and Test sets###do in another model
#smp_size <- floor(0.7 * nrow(Train))
#set.seed(123)
#tr_data <- sample(seq_len(nrow(Train)), size = smp_size)
#train <- Train[tr_data, ]
#test <- Train[-tr_data, ]

########################## Random forest#################################

final_train_data<-cbind(revenue,final_train_data)
names(final_train_data)
library(randomForest)
rf_model <- randomForest(final_train_data$revenue ~ ., data= final_train_data,
keep.forest=TRUE, ntree=30)
dim(final_train_data)
dim(final_test)
##  prediction to final model
fitted(rf_model)
```

```
library(DMwR)
#error on train data
regr.eval(final_train_data$revenue, predict(rf_model,final_train_data[,-1])) # 0
prediction_train<-predict(rf_model,final_train_data[,-1])
prediction<- predict(rf_model,final_test)
prediction<-as.data.frame(prediction)
head(prediction)

######Submission file#####################

id=0:99999
id<-as.data.frame(id)
sample_submission<-cbind(id,prediction)
head(sample_submission)
write.csv(sample_submission,"sampleSubmission_final_pca_rf.csv",row.names = F)



##############################################
##knnn


#rf_knn

model_knn=train(revenue~ .,
            data=final_train_data,
            preProcess = c("center","scale"), tuneLength = 20,
            method="knn",
            trControl=trainControl(method="repeatedcv",number=10,repeats = 5),
            prox=TRUE,allowParallel=TRUE)


#error on train data
regr.eval(final_train_data$revenue, fitted(model_knn))
```

```
prediction_train<-predict(model_knn,final_train_data[,-1])
prediction<-predict(model_knn,final_test)
prediction_test_knn<-as.data.frame(prediction)

###### Submission file#########################

Id=0:99999
Id=as.data.frame(Id)
sampleSubmission=cbind(Id,prediction_test_knn)
head(sampleSubmission)
write.csv(sampleSubmission,"sampleSubmissionfinal_pca_knn.csv",row.names = F)
```

Method 3= Linear model

Linear Model + Lasso and Ridge regression

```
#################################################################
#######################
## Set the current Working Directory

rm(list = ls())
setwd("G:/Project/test.csv")

#################################################################
#######################


## Data Preprocessing
#################################################################
################
install.packages("car")
library(car)
```

```
library(MASS)
library(caret)
## reading file

#########################################################################
########################
### Understanding the train data ###
dataTrain=read.csv("train.csv",header=T,sep=",")
summary(dataTrain)

str(dataTrain)
dim(dataTrain)
names(dataTrain)
sum(is.na(dataTrain)) ## 'o' No missing values
#########################################################################
########################



#########################################################################
########################
### Understanding the test data ###
dataTest=read.csv("test.csv",header=T,sep=",")
summary(dataTest)
str(dataTest)
dim(dataTest)
names(dataTest)
sum(is.na(dataTest)) ## 'o' No missing values



#########################################################################
########################
## Combine train and test dataset for analysis without target variable "revenue"
## As the data in train dataset is small , we need to combine the both
## to ensure there will not be any class imbalance
```

```
revenue <- dataTrain$revenue
dataTrain <- dataTrain[,-c(1,3,43)]

Id <- dataTest$Id
dataTest <- dataTest[,-c(1,3)]
lgrevenue <- log10(revenue)
dataFinal=rbind(dataTrain,dataTest)

names(dataFinal)
str(dataFinal)
dim(dataFinal)
colnames(dataFinal)

## Converting the Open date to Number of years from current day
dataFinal$Open.Date=Sys.Date()-as.Date(as.character(dataFinal$Open.Date),'%m/%d/%Y')
dataFinal$Open.Date=as.numeric(dataFinal$Open.Date)/365
names(dataFinal)[names(dataFinal) == 'Open.Date'] <- 'NoOfYrs'

#Renaming DT to FC and MB to IL
plot(dataFinal$Type)
dataFinal$Type<- recode(dataFinal$Type,"'DT'='FC'")
dataFinal$Type<- recode(dataFinal$Type,"'MB'='IL'")
plot(dataFinal$Type)
str(dataFinal$Type)
levels(dataFinal$Type)
sum(is.na(dataFinal))
str(dataFinal)
##Seperate Numeric and Categorical attributes
dataNum=dataFinal[,c(1,5,6,7,16,29,30,31,32)]
dataCat=dataFinal[,c(2,3,4,8,9,10,11,12,13,14,15,17,18,19,20,21,22,23,24,25,26,27,28,

                     33,34,35,36,37,38,39,40)]

dataCat=data.frame(apply(dataCat,2,function(x){as.factor(x)}))
```

```
str(dataCat) ## All Categorical variables converted to Factor

## Standardize the Numercial attributes
library(vegan)
dataNumStd <- decostand(dataNum,"standardize")
summary(dataNumStd)

## Create dummies for the Categorical variables
library(dummies)
function(x){
  dataCatDummy=data.frame(dummy(dataCat$x))



  dataCat=subset(dataCat,select=-c(x))
  dataCat=data.frame(dataCat,dataCatDummy)}
dataCatDummy=dummy.data.frame(dataCat) ## Create dummies for Categorrical variables

## Combine standardized Numerical and Categorical data frames
dataFinal=data.frame(dataNumStd,dataCatDummy)
str(dataFinal) ## 257 Variables
dim(dataFinal)
nrow(dataFinal)
names(dataFinal)



## Seperate train and test datasets
Train <- dataFinal[1:137, ]
Test <- dataFinal[138:nrow(dataFinal), ]
Train = data.frame(Train,revenue) ## Binding the revenue column from train datase
names(Train)[names(Train) == 'dataTrain.revenue'] <- 'revenue'

Train$revenue=log(Train$revenue)  ## Converting revenue to log as the value looks large
```

```
dim(Train)
dim(Test)
names(Train)
names(Test)
nrow(Train)
nrow(Test)
sum(is.na(Train))
sum(is.na(Test))
Train[1,]
## Identify the outliers in Revenue and remove them for analysis
## After removing outliers Adjusted R square was positive
boxplot(Train$revenue)
abline(h = mean(Train$revenue), lty=2)
IQ = IQR(Train$revenue)
Q1 = quantile(Train$revenue,0.25)
Q3 = quantile(Train$revenue,0.75)
lowerinnerfence =  Q1 - 1.5*IQ
upperinnerfence = Q3 + 1.5*IQ
lowerouterfence = Q1 - 3*IQ
upperouterfence = Q3 + 3*IQ
outliers = boxplot(Train$revenue, plot=FALSE)$out

#Extract the outliers from the original data frame
Train=Train[!(Train$revenue %in% outliers),]

#### Split the data into Train and Test sets
smp_size <- floor(0.7 * nrow(Train))
set.seed(123)
tr_data <- sample(seq_len(nrow(Train)), size = smp_size)
train <- Train[tr_data, ]

test <- Train[-tr_data, ]
```

```
#################################################################
#############

## Model building

#################################################################
#############
## Normal multivariate regression
## Was unsatisfactory - not working at all
MultLM <- lm(train$revenue~., data=train,na.action=NULL)
summary(MultLM)

## Model evaluation
library(DMwR)
Lmpred <- fitted(MultLM)
regr.eval(train$revenue,fitted(MultLM))  #error on train data

pred <- predict.lm(MultLM, test)
regr.eval(test$revenue, pred)#error on test data



##################linear regression predictions#####################

Linearpredictions <- predict.lm(MultLM, Test)
#regr.eval(final_test, prediction) # 1.47
#mae        mse        rmse        mape
#4.520635e+11 2.113490e+18 1.453785e+09 5.257039e+12

Linearpredictions <-as.data.frame(Linearpredictions)
Linearpredictions <- exp(Linearpredictions)
head(Linearpredictions)
```

```
id=0:99999
id<-as.data.frame(id)
sample_submission<-cbind(id,Linearpredictions)
head(sample_submission)
write.csv(sample_submission,"sampleSubmission_final_linearreg.csv",row.names = F)




############################################################################
#############
## Lasso Reggression
#Converted the data into matrix form to input into glm model
trainL = as.matrix(train)
testL = as.matrix(test)

#Target Varaible
y=train$revenue
ytest = test$revenue
#install.packages("glmnet")
library(glmnet)

#########################################################
#cv.glmnet will help you choose lambda
cv <- cv.glmnet(trainL,y)

#lambda.min - value of lambda that gives minimum cvm - mean cross-validated error
#################################################
# Lasso Regression  using glmnet - L1 norm

fit1=glmnet(trainL,y,lambda=cv$lambda.min,alpha=0)
predict(fit1,trainL)
library(DMwR)
LASSOtrain = regr.eval(y, predict(fit1,trainL))
```

```
LASSOtest = regr.eval(ytest, predict(fit1,testL))
LASSOtrain
LASSOtest

#Model Selection
coef(fit1)
cv.lasso=cv.glmnet(trainL,y)
plot(cv.lasso)
coef(cv.lasso)




##############################################################################
#############
# Ridge Regression  using glmnet  - L2 norm
## Using Ridge regression also, the evaluation metrics on train and test
## are same as Lasso regression

library(glmnet)
# fit model
fit2=glmnet(trainL,y,lambda=cv$lambda.min,alpha=0)
predict(fit2,trainL)
library(DMwR)
RIDGEtrain = regr.eval(y, predict(fit2,trainL))
RIDGEtest = regr.eval(ytest, predict(fit2,testL))
RIDGEtrain
RIDGEtest
#Model Selection
coef(fit2)
cv.ridge=cv.glmnet(trainL,y,alpha=0)
plot(cv.ridge)
coef(cv.ridge)
```

```
#################################################################


finalerros <- data.frame(rbind(LASSOtrain,LASSOtest,
                               RIDGEtrain,RIDGEtest))
finalerros
###mae        mse      rmse       mape
###LASSOtrain 0.01241992 0.0004832456 0.02198285 0.0008209368
###LASSOtest  0.08824402 0.0118138379 0.10869148 0.0057887049
###RIDGEtrain 0.01241992 0.0004832456 0.02198285 0.0008209368
###RIDGEtest  0.08824402 0.0118138379 0.10869148 0.0057887049

#################################################################
###################
## If we observe, Multivariate Linear reggression was not working
## Of Lasso,Ridge both were providing same RMSE values in train and test
#################################################################
###################
```