Forest fire

Seyedeh Shaghayegh Rabbanian

4/22/2021

Introduction

Forest fire is a disaster that causes economic and ecological damage and human life threat. Thus, predicting such critical environmental issue is essential to mitigate this threat. The fire prediction is based on the meteorological data corresponding to the critical weather elements that influence the forest fire occurrence, namely temperature, relative humidity and wind speed.

What is important to us in this study is that how much the classifier can predict the true cases (fire) correctly. Therefore our goal is to find the best method which minimizes misclassification rate as well as finding a logical relationship between the predictors in order to interpret their correlation or causality.

Data description and summary

The dataset includes **244 instances** that regroup data of two regions of Algeria, namely the Bejaia region located in the **northeast of Algeria** and the Sidi Belabbes region located in the **northwest of Algeria**. Each region consist of **122 instances**. The data set includes **11 predictors** and one **binary response** variable, which consists of two classes of **"fire" and "not fire"**. The goal is to classify the instances based on certain predictors correctly.

The predictors are as follow:

1. Date : (DD/MM/YYYY) Day, month ('june' to 'september'), year (2012) Weather data observations.
2. Temp : temperature noon (temperature max) in Celsius degrees: 22 to 42.
3. RH : Relative Humidity in %: 21 to 90.
4. Ws :Wind speed in km/h: 6 to 29.
5. Rain: total day in mm: 0 to 16.8 FWI Components.
6. Fine Fuel Moisture Code (FFMC) index from the FWI system: 28.6 to 92.5.
7. Duff Moisture Code (DMC) index from the FWI system: 1.1 to 65.9.
8. Drought Code (DC) index from the FWI system: 7 to 220.4.
9. Initial Spread Index (ISI) index from the FWI system: 0 to 18.5.
10. Buildup Index (BUI) index from the FWI system: 1.1 to 68.
11. Fire Weather Index (FWI) Index: 0 to 31.1.
12. Classes: two classes

First of all we read the dataset. When we check the structure of the data, we can see that there are **4 character** variables (date,DC, FWI and Classes). What I did to character predictors was to convert them to numeric to consider them in data analysis as they were numbers.I changed the date field with date format to day number. There was one typo in one of the observations which caused R to identify DC predictor as a character which was not correct. The problem was fixed before importing the data.

```
forestfire <- read.csv('C:/Users/srabba2/Desktop/forestfire - allnum.csv',header=TRUE)
str(forestfire)
```

```
## 'data.frame':    244 obs. of  12 variables:
##  $ date       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Temperature: int  29 29 26 25 27 31 33 30 25 28 ...
##  $ RH         : int  57 61 82 89 77 67 54 73 88 79 ...
##  $ Ws         : int  18 13 22 13 16 14 13 15 13 12 ...
##  $ Rain       : num  0 1.3 13.1 2.5 0 0 0 0 0.2 0 ...
##  $ FFMC       : num  65.7 64.4 47.1 28.6 64.8 82.6 88.2 86.6 52.9 73.2 ...
##  $ DMC        : num  3.4 4.1 2.5 1.3 3 5.8 9.9 12.1 7.9 9.5 ...
##  $ DC         : num  7.6 7.6 7.1 6.9 14.2 22.2 30.5 38.3 38.8 46.3 ...
##  $ ISI        : num  1.3 1 0.3 0 1.2 3.1 6.4 5.6 0.4 1.3 ...
##  $ BUI        : num  3.4 3.9 2.7 1.7 3.9 7 10.9 13.5 10.5 12.6 ...
##  $ FWI        : chr  "0.5" "0.4" "0.1" "0" ...
##  $ Classes    : chr  "not fire   " "not fire   " "not fire   " "not fire   " ...
```

```
summary(forestfire)
```

```
##       date         Temperature          RH              Ws       
##  Min.   :  1.00   Min.   :22.00   Min.   :21.00   Min.   : 6.0  
##  1st Qu.: 61.75   1st Qu.:30.00   1st Qu.:52.00   1st Qu.:14.0  
##  Median :122.50   Median :32.00   Median :63.00   Median :15.0  
##  Mean   :122.50   Mean   :32.17   Mean   :61.94   Mean   :15.5  
##  3rd Qu.:183.25   3rd Qu.:35.00   3rd Qu.:73.25   3rd Qu.:17.0  
##  Max.   :244.00   Max.   :42.00   Max.   :90.00   Max.   :29.0  
##       Rain              FFMC            DMC              DC        
##  Min.   : 0.0000   Min.   :28.60   Min.   : 0.70   Min.   :  6.90  
##  1st Qu.: 0.0000   1st Qu.:72.08   1st Qu.: 5.80   1st Qu.: 13.28  
##  Median : 0.0000   Median :83.50   Median :11.30   Median : 33.10  
##  Mean   : 0.7607   Mean   :77.89   Mean   :14.67   Mean   : 49.29  
##  3rd Qu.: 0.5000   3rd Qu.:88.30   3rd Qu.:20.75   3rd Qu.: 68.15  
##  Max.   :16.8000   Max.   :96.00   Max.   :65.90   Max.   :220.40  
##       ISI              BUI            FWI              Classes         
##  Min.   : 0.000   Min.   : 1.10   Length:244         Length:244        
##  1st Qu.: 1.400   1st Qu.: 6.00   Class :character   Class :character  
##  Median : 3.500   Median :12.25   Mode  :character   Mode  :character  
##  Mean   : 4.774   Mean   :16.66                                        
##  3rd Qu.: 7.300   3rd Qu.:22.52                                        
##  Max.   :19.000   Max.   :68.00
```

```
dim(forestfire)
```

```
## [1] 244  12
```

```
forestfire$FWI=as.numeric(forestfire$FWI)
```

```
## Warning: NAs introduced by coercion
```

Procedures for data cleaning and processing

#Missing values

Next important step in data analysis is data cleaning. First of all, we need to see if we have any missing values in our dataset.

As we can see, there is one observation with two missing values in our dataset. We decided to remove the instance with the missing values.

```
sum(is.na(forestfire))
```

```
## [1] 2
```

```
forestfire=na.omit(forestfire)
dim(forestfire)
```

```
## [1] 243  12
```

Since one of the observations was removed from the dataset, the dimension has reduced to 243*12.

#Changing the structure of the response variable

The next step which needs to be done is to change the structure of the response variable.The current structure of response variable (Classes) is character which needs to be a factor. So, we convert it to a factor using factor function.

```
forestfire$Classes=factor(forestfire$Classes)
str(forestfire)
```

```
## 'data.frame':    243 obs. of  12 variables:
##  $ date       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Temperature: int  29 29 26 25 27 31 33 30 25 28 ...
##  $ RH         : int  57 61 82 89 77 67 54 73 88 79 ...
##  $ Ws         : int  18 13 22 13 16 14 13 15 13 12 ...
##  $ Rain       : num  0 1.3 13.1 2.5 0 0 0 0 0.2 0 ...
##  $ FFMC       : num  65.7 64.4 47.1 28.6 64.8 82.6 88.2 86.6 52.9 73.2 ...
##  $ DMC        : num  3.4 4.1 2.5 1.3 3 5.8 9.9 12.1 7.9 9.5 ...
##  $ DC         : num  7.6 7.6 7.1 6.9 14.2 22.2 30.5 38.3 38.8 46.3 ...
##  $ ISI        : num  1.3 1 0.3 0 1.2 3.1 6.4 5.6 0.4 1.3 ...
##  $ BUI        : num  3.4 3.9 2.7 1.7 3.9 7 10.9 13.5 10.5 12.6 ...
##  $ FWI        : num  0.5 0.4 0.1 0 0.5 2.5 7.2 7.1 0.3 0.9 ...
##  $ Classes    : Factor w/ 2 levels "fire   ","not fire   ": 2 2 2 2 2 1 1 1 2 2 ...
##  - attr(*, "na.action")= 'omit' Named int 166
##   ..- attr(*, "names")= chr "166"
```

#Correlation Matrix

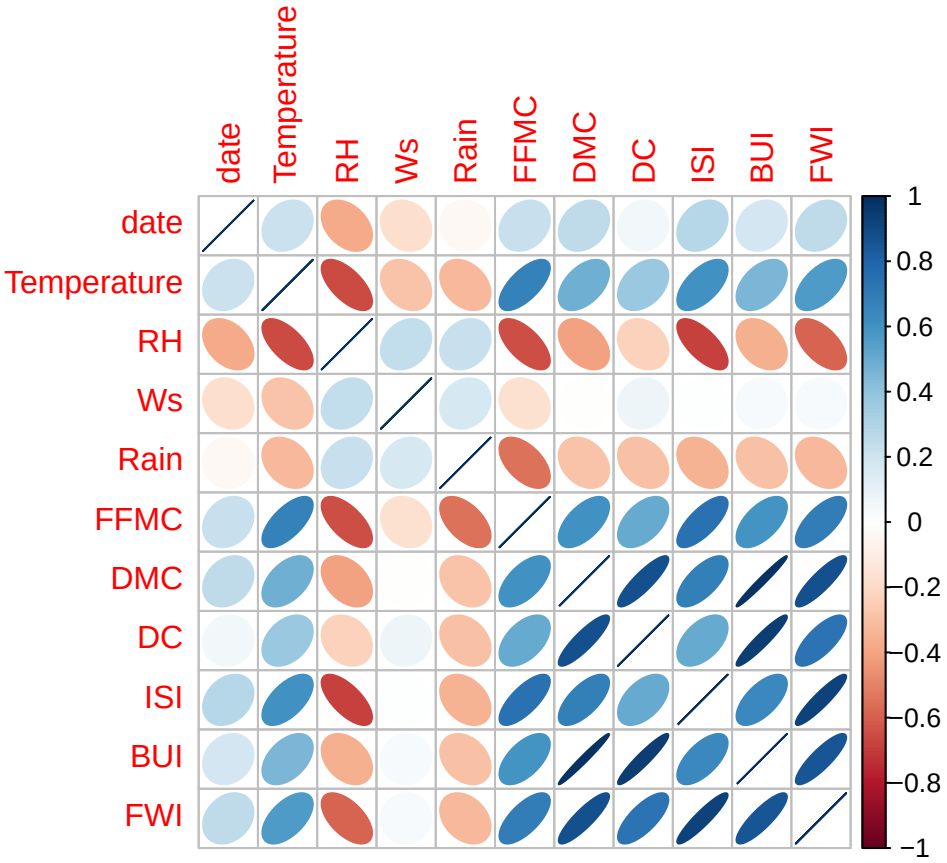
```
correlationmatrix <- cor(forestfire[,1:11])
correlationmatrix
```

```
##                    date Temperature         RH            Ws        Rain
## date         1.00000000   0.2162372 -0.3772094 -0.1704091900 -0.03128787
## Temperature  0.21623717   1.0000000 -0.6514003 -0.2845098897 -0.32649192
## RH          -0.37720936  -0.6514003  1.0000000  0.2440483822  0.22235608
## Ws          -0.17040919  -0.2845099  0.2440484  1.0000000000  0.17150618
## Rain        -0.03128787  -0.3264919  0.2223561  0.1715061807  1.00000000
## FFMC         0.22735847   0.6765681 -0.6448735 -0.1665482728 -0.54390619
## DMC          0.25934523   0.4856869 -0.4085192 -0.0007209737 -0.28877293
## DC           0.05819555   0.3762835 -0.2269411  0.0791345143 -0.29802308
## ISI          0.28107155   0.6038706 -0.6866670  0.0085316891 -0.34748393
## BUI          0.18205889   0.4597895 -0.3538405  0.0314384118 -0.29985152
## FWI          0.25335594   0.5666699 -0.5809567  0.0323677727 -0.32442156
##                   FFMC           DMC          DC           ISI         BUI
## date         0.2273585  0.2593452303  0.05819555  0.281071549  0.18205889
## Temperature  0.6765681  0.4856869230  0.37628353  0.603870559  0.45978947
## RH          -0.6448735 -0.4085191880 -0.22694112 -0.686667043 -0.35384055
## Ws          -0.1665483 -0.0007209737  0.07913451  0.008531689  0.03143841
## Rain        -0.5439062 -0.2887729260 -0.29802308 -0.347483929 -0.29985152
## FFMC         1.0000000  0.6036076410  0.50739666  0.740006828  0.59201101
## DMC          0.6036076  1.0000000000  0.87592466  0.680454326  0.98224849
## DC           0.5073967  0.8759246607  1.00000000  0.508643247  0.94198846
## ISI          0.7400068  0.6804543264  0.50864325  1.000000000  0.64409260
## BUI          0.5920110  0.9822484891  0.94198846  0.644092598  1.00000000
## FWI          0.6911320  0.8758641588  0.73952056  0.922894934  0.85797310
##                    FWI
## date         0.25335594
## Temperature  0.56666988
## RH          -0.58095675
## Ws           0.03236777
## Rain        -0.32442156
## FFMC         0.69113197
## DMC          0.87586416
## DC           0.73952056
## ISI          0.92289493
## BUI          0.85797310
## FWI          1.00000000
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
Cl=cor(forestfire[,1:11],use="complete.obs")
corrplot(Cl,method="ellipse")
```



By analyzing correlation matrix and plot, we can see that there is a high positive correlation between **BUI and DMC**, **DC and DMC**, **BUI and DC** and **FWI and ISI**.

There are strong negative correlations between **RH and Temperature**, **RH and FFMC** , and **RH and ISI**.

#Splitting dataset into train and test set

The next step that needs to be done is to split the dataset into training and test set. 70% of the data is randomly chosen for the training set and 30% is assigned to the test set. We will work with **train** and **test** to run the methods.

```
set.seed(1)
indx=sample (1: nrow(forestfire), size=0.7*nrow(forestfire))
train = forestfire[indx,]
test = forestfire[-indx,]
```

Methods

We applied different methods including Logistic regression, classification tree, bagging, random forest, MARS ,PRIM and SVM to our dataset. The performance of each method was evaluated using misclassification error rate.

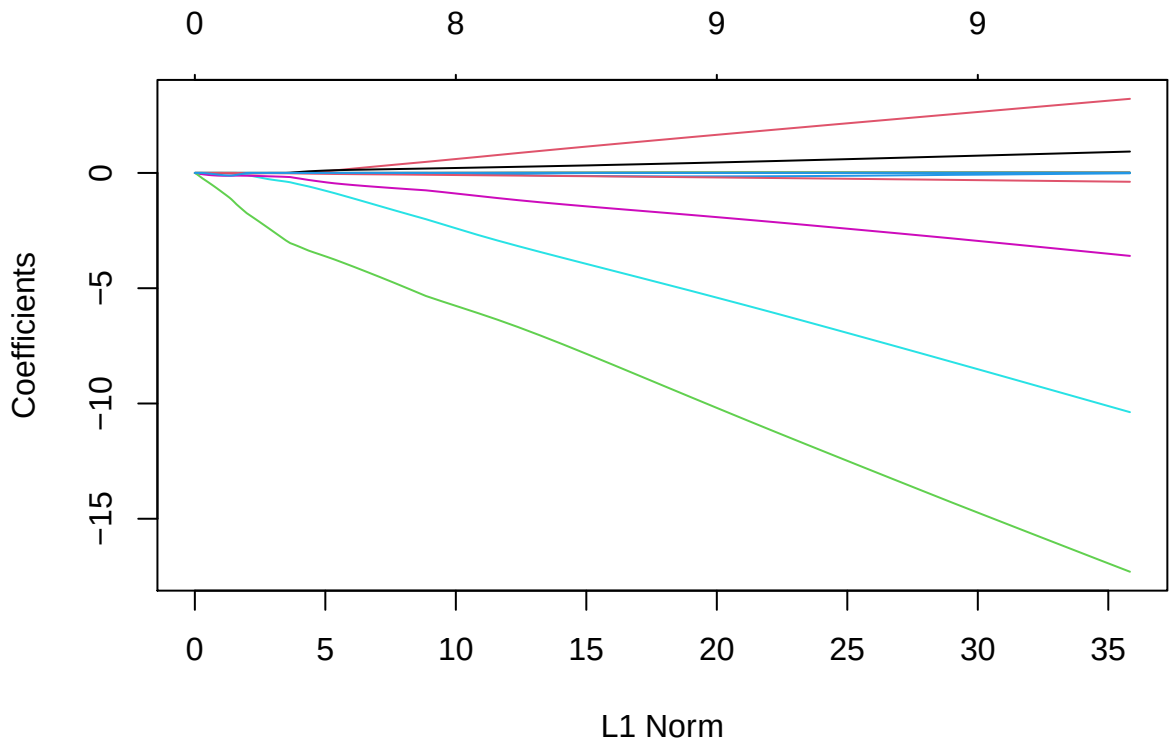
Logistic regression with L1 penalty

```
x=as.matrix(train[,1:11])
y=train[,12]
library(glmnet)
```

```
## Loading required package: Matrix
```

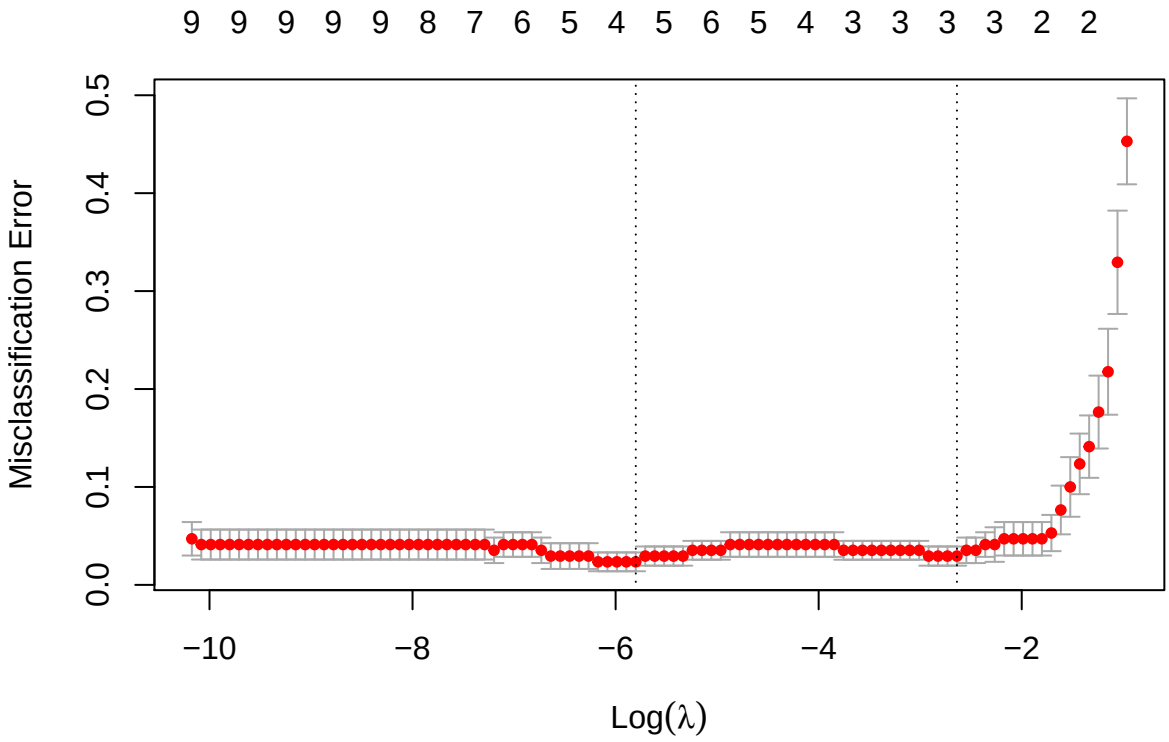
```
## Loaded glmnet 4.1
```

```
fit.glm = glmnet(x, y, family = "binomial", alpha=1) #LASSO type penalty
plot(fit.glm, xvar="norm") #Solution path
```



Here, we apply Cross Validation to find the optimal value of lambda.

```
set.seed(1) #able to reproduce CV results
cv.fit = cv.glmnet(x, y, family = "binomial", type.measure = "class")
plot(cv.fit)
```



As we can see below, the minimum value for Lambda is equal to 0.00302.

```
cv.fit
```

```
##
## Call:  cv.glmnet(x = x, y = y, type.measure = "class", family = "binomial")
##
## Measure: Misclassification Error
##
##      Lambda Index Measure       SE Nonzero
## min 0.00302    53 0.02353 0.009606       4
## 1se 0.07150    19 0.02941 0.009804       3
```

```
coef(cv.fit, s = "lambda.min")
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 20.88520238
## date         .
## Temperature  .
## RH           .
## Ws           .
## Rain        -0.31443773
## FFMC        -0.15439940
## DMC          .
```

```
## DC          -0.01320642
## ISI         -2.61021494
## BUI          .
## FWI          .
```

As we can see, using the minimum value of lambda, only the variables FFMC, ISI, DC and Rain are selected with negative coefficient values.

Then we predict our model using type "class" to evaluate the performance of the model and misclassification rate for glm method is 0.01369863.

```
x_test=as.matrix(test[,1:11])
pred1.glm=predict(cv.fit,newx=x_test,s="lambda.min",type="class") #class
table(pred1.glm,test[,12])
```

```
##
## pred1.glm     fire    not fire
##   fire             43           0
##   not fire          1          29
```

```
res.glm=table(pred1.glm,test$Classes)
mc.glm=1-sum(diag(res.glm))/sum(res.glm)
mc.glm
```

```
## [1] 0.01369863
```

Tree

Here, we applied a classification tree to our new dataset. Tree using default values for rpart.control is too shallow. The reason for that might be that the number of observations in our dataset are not too many. Therefore, I defined control parameters less than default values to have deeper tree.The misclassification error rate for the tree using predict function to evaluate our tree is 0.02739726.

```
set.seed(1)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.4
```

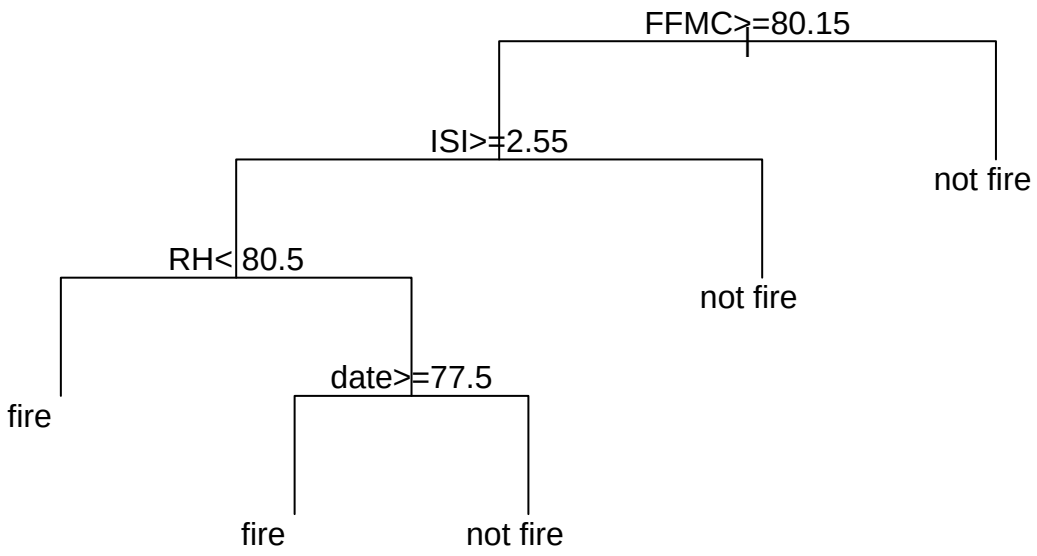
```
fit1.tree=rpart(Classes~.,data=train,control = rpart.control(cp=0.001,xval=0,maxsurrogate=0,maxcompete=0

printcp(fit1.tree)
```

```
##
## Classification tree:
## rpart(formula = Classes ~ ., data = train, control = rpart.control(cp = 0.001,
##     xval = 0, maxsurrogate = 0, maxcompete = 0, minsplit = 1))
##
## Variables actually used in tree construction:
## [1] date FFMC ISI  RH
##
## Root node error: 77/170 = 0.45294
##
```

```
## n= 170
##
##          CP nsplit rel error
## 1 0.9610390      0  1.000000
## 2 0.0259740      1  0.038961
## 3 0.0064935      2  0.012987
## 4 0.0010000      4  0.000000
```

```
par(xpd=NA)
plot(fit1.tree,uniform = T)
text(fit1.tree,use.n=F)
```




```
tree.misrate=rep(0,nrow(fit1.tree$cptable)-1)
for(i in 1:(nrow(fit1.tree$cptable)-1)){
  prune.fit1=prune(fit1.tree,cp=fit1.tree$cptable[(i+1),1])
  pred=predict(prune.fit1,newdata=test,type = "class")
  tab=table(pred,test$Classes)
  tree.misrate=1-sum(diag(tab))/sum(tab)
}

tree.misrate
```

```
## [1] 0.02739726
```

The single split tree (called *stump*) performs the best on our test set.

Bagging

Then we applied bagging method with 201 bootstrap samples to our dataset.

```
library(nnet)
B=201
n=nrow(train)
set.seed(1)
bootsamples=rmultinom(B,n,rep(1,n)/n)
trees=vector(mode="list",length=B)
pred_boot=prob_boot=matrix(0,nrow(test),B)
fit2_bagging=rpart(Classes~.,data=train,control=rpart.control(cp=0.001,xval=0,
                                                              maxsurrogate=0,maxcompete=0,minsplit = 1))
for(i in 1:B){
  trees[[i]]=update(fit2_bagging,weight=bootsamples[,i])
  pred_boot[,i]=predict(trees[[i]],test,type="class")
  prob_boot[,i]=predict(trees[[i]],test,type="prob")[,2]
}
bag.vote=apply(pred_boot,1,median)
bag.prob=apply(prob_boot,1,mean)
bag.prob2=as.numeric(bag.prob>=0.5)
tab.bag.vote=table(bag.vote,test$Classes)
tab.bag.prob=table(bag.prob2,test$Classes)
tab.bag.vote; mc.bag=1-sum(diag(tab.bag.vote))/sum(tab.bag.vote) ;mc.bag
```

```
##
## bag.vote fire    not fire
##        1      44           1
##        2       0          28
```

```
## [1] 0.01369863
```

```
tab.bag.prob; mc.bag2=1-sum(diag(tab.bag.prob))/sum(tab.bag.prob);mc.bag2
```

```
##
## bag.prob2 fire    not fire
##         0      44           1
##         1       0          28
```

```
## [1] 0.01369863
```

From above, we can see the misclassification rate of 201 trees on test set is 0.01369863 (using averaged probability) is slightly better than single split tree on the original data.

We can also calculate the area under the ROC curve (AUC) using AUC package. The AUC score for bagging is slightly better than one split classification tree.

```
library(AUC)
```

```
## AUC 0.3.0
```

```
## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```
pred.prune.tree=predict(prune(fit1.tree,cp=0.0259740),test,type="prob")[,2]
auc.score1=auc(roc(pred.prune.tree,test$Classes))
auc.score1
```

```
## [1] 0.9827586
```

```
auc.score2=auc(roc(bag.prob,test$Classes))
auc.score2
```

```
## [1] 0.9992163
```

Multivariate adaptive regression splines (MARS)

```
#set.seed(1)


train_num=train
test_num=test
train_num$Classes=as.numeric(train$Classes)
test_num$Classes=as.numeric(test$Classes)
train.data.num=as.data.frame(train_num)
test.data.num=as.data.frame(test_num)

library(mda)
```

```
## Warning: package 'mda' was built under R version 4.0.5
```

```
## Loading required package: class
```

```
## Loaded mda 0.5-2
```

```
set.seed(1)
fit1_mars=mars(train.data.num[,1:11],train.data.num[,12])
pred1.mars=predict(fit1_mars,test.data.num[,1:11])
temp1=as.numeric(pred1.mars>=1.5)
res1=table(temp1,test.data.num$Classes)
mc.mars1=1-sum(diag(res1))/sum(res1)
mc.mars1
```

```
## [1] 0.01369863
```

```
fit2_mars=mars(train.data.num[,1:11],train.data.num[,12],degree=2)
pred2.mars=predict(fit2_mars,test.data.num[,1:11])
temp2=as.numeric(pred2.mars>=1.5)
res2=table(temp2,test.data.num$Classes)
mc.mars2=1-sum(diag(res2))/sum(res2)
mc.mars2
```

```
## [1] 0.01369863
```

The misclassification error rate for both mars functions using default degree and degree=2 was the same.

PRIM

```
library(prim)
```

```
## Warning: package 'prim' was built under R version 4.0.5
```

```
set.seed(1)
thr=1.43
fireforest.prim <- prim.box(x=train.data.num[,1:11],
y=train.data.num[,12],
peel.alpha=0.05, paste.alpha=0.01,threshold=thr,threshold.type = 1)
pred1.earth=predict(fireforest.prim,newdata=test.data.num[,1:11])
temp1.prim=as.numeric(pred1.earth<=1)
res1.prim=table(temp1.prim,test.data.num$Classes)
mc.prim=1-sum(diag(res1.prim))/sum(res1.prim)
mc.prim
```

```
## [1] 0.02739726
```

Although PRIM is designed for regression, it can handle two class outcome by simply coding it as 0 and 1. Therefore, in binary classification, we need to find a box which maximizes the mean response. I used prim package in order to run prim. The misclassification error rate for prim model is 0.02739726.

Generally PRIM is more patient algorithm in comparison to CART. So, it has more stable results. But here, the misclassification error rate is same as CART and it did not give us better results. It may be relevant to below reason:

High response data points cannot be **grouped (clustered)** by the simple rules in PRIM. Maybe it can be grouped by some **linear combinations** of the variables but not the original variables using the simple rules. Therefore, PRIM gave us same results as CART with no improvements.

Random forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

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

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

```
set.seed(1)
fireforest.rf=randomForest(Classes~.,data=train,mtry=2,importance=TRUE)
print(fireforest.rf)
```

```
##
## Call:
##  randomForest(formula = Classes ~ ., data = train, mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.94%
## Confusion matrix:
##             fire    not fire    class.error
## fire             90           3  0.03225806
## not fire          2          75  0.02597403
```

```
library(ipred)
```

```
## Warning: package 'ipred' was built under R version 4.0.5
```

```
set.seed(1)
error.RF=numeric(20)
for(i in 1:20){
  error.RF[i]=errorest(Classes~.,data=train, model=randomForest,mtry=2)$error
}
summary(error.RF)
```

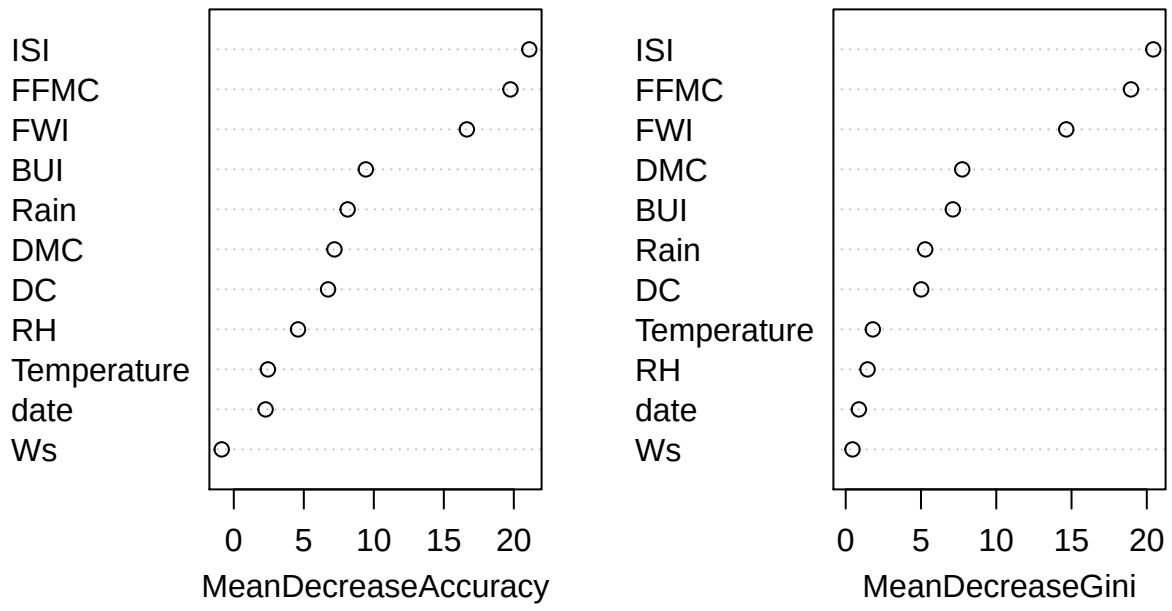
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02353 0.02353 0.02941 0.02824 0.02941 0.03529
```

When I compared the median of 10-fold CV with the OOB estimate of error rate, I concluded that they are the same(2.94%).

```
pred.rf=predict(fireforest.rf,test[,1:11])
res.rf=table(pred.rf,test$Classes)
mc.rf=1-sum(diag(res.rf))/sum(res.rf)
mc.rf
```

```
## [1] 0.01369863
```

```
varImpPlot(fireforest.rf)
```




```
importance(fireforest.rf)
```

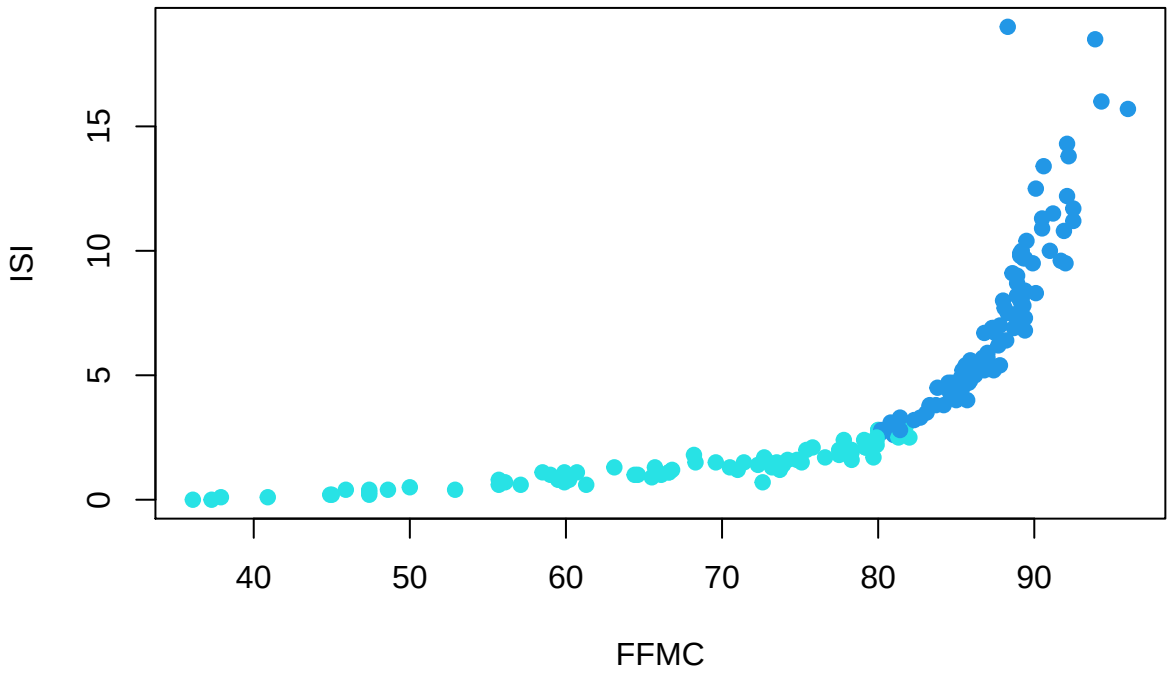
```
##                 fire    not fire    MeanDecreaseAccuracy MeanDecreaseGini
## date         3.5399321   -1.531719            2.2702957        0.8740191
## Temperature  1.6998229    1.917277            2.4344719        1.8016138
## RH           3.9191854    2.528311            4.5881955        1.4517614
## Ws           0.1080504   -1.617017           -0.8632997        0.4511614
## Rain         7.9861182    2.514016            8.1297998        5.2795322
## FFMC        17.2804392   15.032802           19.7606042       18.9505350
## DMC          6.7727512    2.754360            7.1885694        7.7398218
## DC           5.9223268    3.164798            6.7330243        5.0156692
## ISI         18.1300964   15.685664           21.1058208       20.4318410
## BUI          7.9059033    5.171060            9.4295073        7.1178142
## FWI         13.1781791   12.089869           16.6465847       14.6513837
```

Variable importance help us to recognize the most important variables. As we can see, FFMC,ISI and FWI are the three most important variables. The misclassification error rate for random forest method is equal to 0.01369863.

SVM for classification with the most important predictors (ISI & FFMC)

From the previous analyses, we know that FMCC and ISI are the most important predictors in fire forest classification. Now, a linear Support Vector Machine classifier is used with these two parameters in order to predict the outcome. Below, the 2D feature space is shown.

```
x=as.matrix(train_num[,c(6,9)])
y=as.vector(train_num[,12])
xt=as.matrix(test_num[,c(6,9)])
yt=as.vector(test_num[,12])
plot(x,col=y+3,pch=19)
```



It can be seen from the figure that the feature space is almost linearly separable and a simple linear SVM should be able to divide the feature space into two regions correspning to “fire” or “not fire”. The SVM is implemented in the chunk of code below:

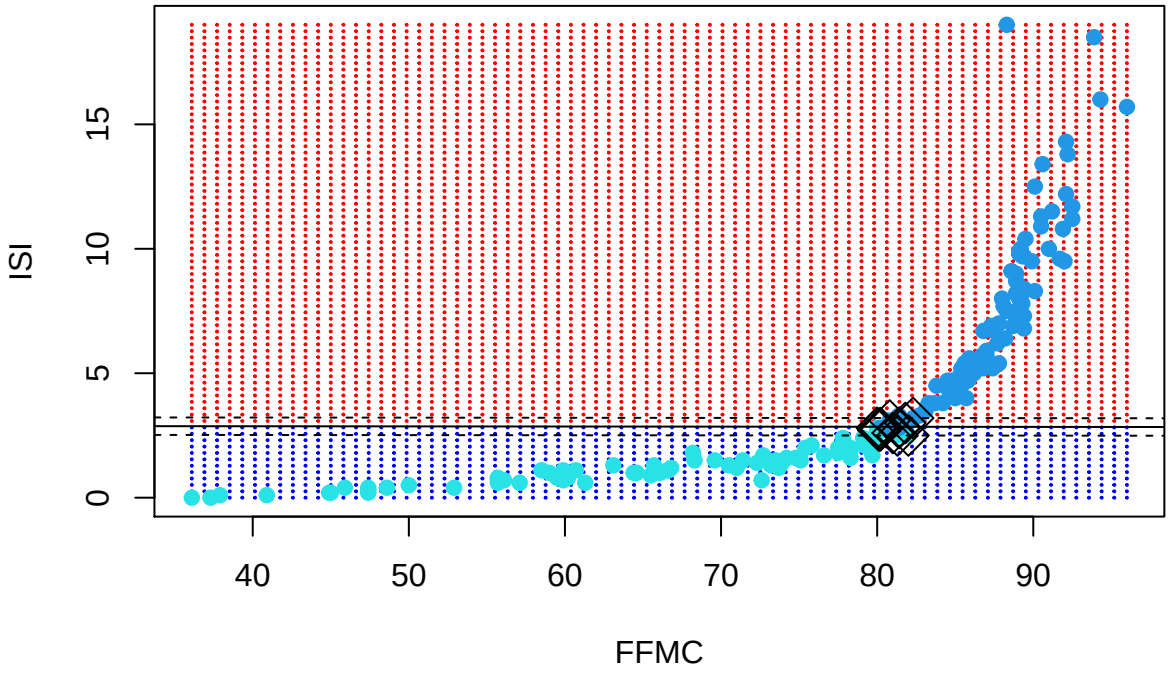
```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
dat=data.frame(x,y=as.factor(y))
svmfit=svm(y~.,data=dat,kernel="linear",cost=10,scale=FALSE)


make.grid=function(x,n=75){
    grange=apply(x,2,range)
    x1=seq(from=grange[1,1],to=grange[2,1],length=n)
    x2=seq(from=grange[1,2],to=grange[2,2],length=n)
    expand.grid(FFMC=x1,ISI=x2)
    }
xgrid=make.grid(x)
```

```
ygrid=predict(svmfit,xgrid)
beta=drop(t(svmfit$coefs)%*%x[svmfit$index,])
beta0=svmfit$rho
plot(xgrid,col=c("red","blue")[as.numeric(ygrid)],pch=20,cex=0.2)
points(x,col=y+3,pch=19)
points(x[svmfit$index,],pch=5,cex=2)
abline(beta0/beta[2],-beta[1]/beta[2])
abline((beta0-1)/beta[2],-beta[1]/beta[2],lty=2)
abline((beta0+1)/beta[2],-beta[1]/beta[2],lty=2)
```



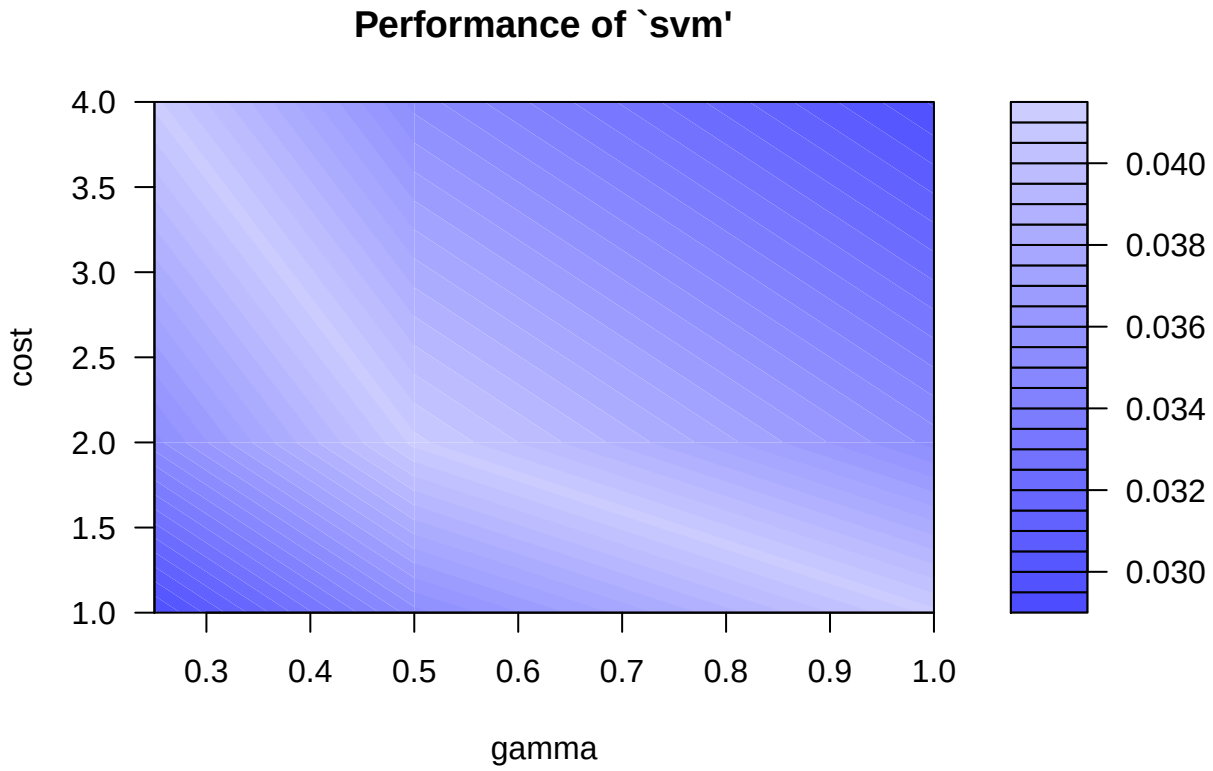
As expected, a SVM classifier with linear kernel was able to lable the data points. The narrow margin boundaries are also shown in the figure. The markers that are inside diamonds, correspond to data points that are either on or inside the margin. For even better performance, the parameters are tuned using the piece of code below:

```
fireforest.svm=tune.svm(y~.,data=dat,gamma=2^(-2:0),cost = 2^(0:2),sampling="cross")
summary(fireforest.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.25    1
```

```
##
## - best performance: 0.02941176
##
## - Detailed performance results:
##   gamma cost      error dispersion
## 1  0.25    1 0.02941176 0.04999039
## 2  0.50    1 0.03529412 0.04960436
## 3  1.00    1 0.04117647 0.04842780
## 4  0.25    2 0.03529412 0.04960436
## 5  0.50    2 0.04117647 0.04842780
## 6  1.00    2 0.03529412 0.04960436
## 7  0.25    4 0.04117647 0.04842780
## 8  0.50    4 0.03529412 0.04960436
## 9  1.00    4 0.02941176 0.04159452
```

```
plot(fireforest.svm)
```



The misclassification error rate of the fitted SVM is calculated here:

```
pred.svm=predict(svmfit,xt)
tab.svm=table(pred.svm,yt)
tab.svm
```

```
##         yt
## pred.svm  1  2
```

```
##         1 42  0
##         2  2 29
```

```
svmfit.error=1-sum(diag(tab.svm))/sum(tab.svm)
svmfit.error
```

```
## [1] 0.02739726
```

```
svmfit$nSV
```

```
## [1] 6 6
```

The misclassification error rate for SVM method is equal to 0.0273973. Number of support vectors are 6 and 6.

Data analysis

If we compare the misclassification error rates of different methods, we can see that CART, Bagging, RF, PRIM and SVM have similar performance. CART and RF Bagging has exactly same performance. Between all these methods Mars has the worst performance.

```
library(knitr)

df <- data.frame(Method = c("Logistic regression","CART","Bagging","Mars(default)",
"Mars(deg=2)","RF","SVM","PRIM"),
value = c(mc.glm,tree.misrate, mc.bag, mc.mars1, mc.mars2, mc.rf,
svmfit.error,mc.prim))

print(kable(df))
```

```
##
##
## |Method              |     value|
## |:-------------------|---------:|
## |Logistic regression | 0.0136986|
## |CART                | 0.0273973|
## |Bagging             | 0.0136986|
## |Mars(default)       | 0.0136986|
## |Mars(deg=2)         | 0.0136986|
## |RF                  | 0.0136986|
## |SVM                 | 0.0273973|
## |PRIM                | 0.0273973|
```

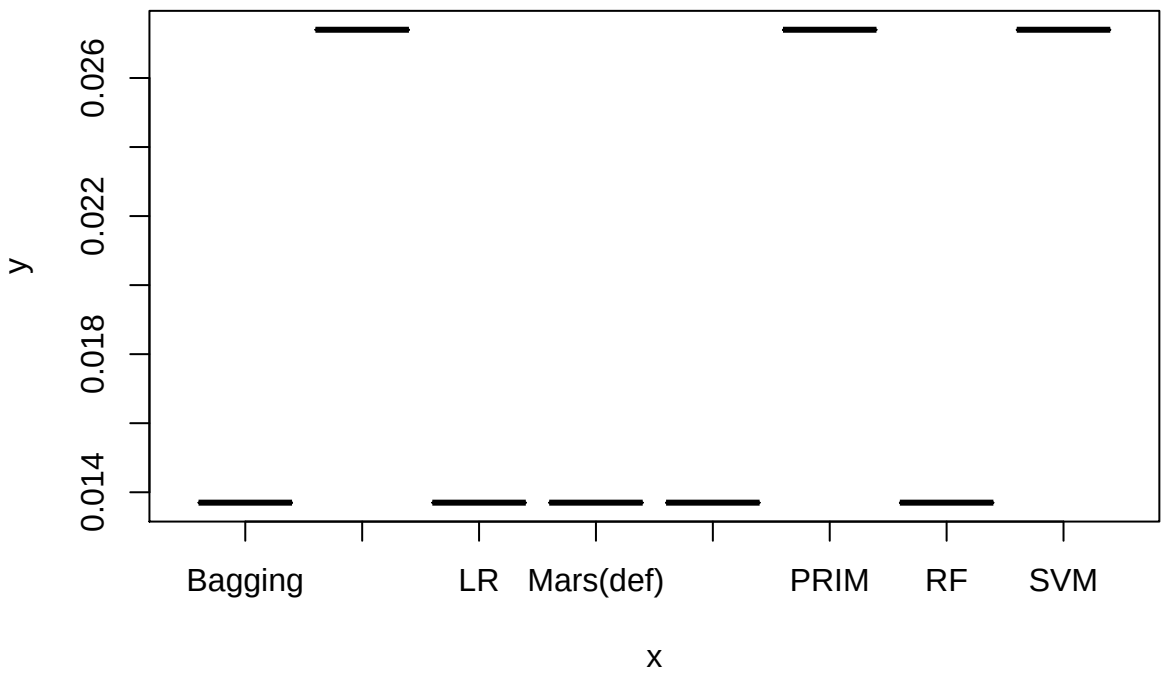
Results

We applied differnt methods to our dataset. It seems that our models perform well in classifying the response variable “fire” and “not fire”. In CART method the tree using default values for rpart.control resulted in single split tree but when I used rpart.control with values less than default values, I got deeper tree. After pruning single split tree was the best tree with least misclassification error rate.

The results gathered from bagging method was slightly better than CART method. Therefore, there is not much improvement in Bagging method in comparison to CART.

We have similar performance for MARS method using default degree and degree=2. Generally, logistic regression, Bagging, MARS and random forest had slightly better performance in comparison to CART, SVM and PRIM.

```
x_plot=c("LR","CART","Bagging","Mars(def)","Mars(deg=2)","RF","SVM","PRIM")
x_plot=factor(x_plot)
y_plot=c(mc.glm,tree.misrate, mc.bag, mc.mars1, mc.mars2, mc.rf,svmfit.error,mc.prim)
plot(x_plot,y_plot,col="red",pch=4)
```



Discussion and conclusion

Based on the dataset that we have, almost all the methods had similar performances. The important conclusion that we got from this research is that the predictors FMCC and ISI have the most effect on classification performance. We confirmed this outcome -both analytically and visually- by evaluating the classification power of SVM using just these two predictors. The SVM misclassification error shows that the result is almost the same as the case considering all the predictors. This means we can be certain that we can make our decision and do our analyses without taking all the predictors into account and consider only FMCC and ISI.