The DeepSolar Project

Manikandan Ravichandran 19200314

28/04/2020

Abstract:

This reports states the prediction of the solar power system coverage of a tile given the collection of predictor explainatory variables stating that the possibility of the outcome being high or low using supervised classification method.

Introduction:

The Deepsolar dataset is been collected by using deep learning frameworks by identifying the sizes of the solar panels based on geographic and many other factors which are being the predictors of the explanatory variables in this data set. Here in this dataset each observation is a measure of entities of a tile of the satellite map with all of its characteristics been derived in the form of collection of variables such as social, economic and geographical area aspects. The prediction variable here is solar_system_count which is a binomially distributed, the value high and low which categories a less powered solar systems when low and large number of solar systems when high. The target variable being binomial, will the binomial logistic regression be effective in predicting it? yes binomial fitness gives a par level of better accuracy but there might be other fitting models which may acquire better accuracy results.

```
library(kernlab)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(mlbench)
library(nnet)
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
deepsolar=read.csv("data_project_deepsolar.csv",header = T)
```

Data Analysis:

Initially we first find the covariance matrix of the unscaled data to have a better prediction of the correlated variables present in the dataset.

```
deepsolartemp <-deepsolar
# Processing the data to find the covariance of the matrix
deepsolar$solar_system_count<- recode(deepsolar$solar_system_count,'high'=1,'low'=
0) #Conveting the precting variable to binary
deepsolar1=deepsolar[,-c(2,76,79)] # Removing categorical variables
covmatrix<-cov(deepsolar1) # Computing the Covarince matrix
covmatrix[1:10,1:10] # displaying first 10x!0 of the covariance matrix
```

```
##                           solar_system_count average_household_income
## solar_system_count             2.493538e-01             4.629760e+03
## average_household_income       4.629760e+03             1.798990e+09
## employed                       9.405466e+01             8.192780e+06
## gini_index                    -3.714942e-03             1.857297e+02
## land_area                     -4.472656e+00            -3.040062e+05
## per_capita_income              1.231538e+03             6.727679e+08
## population                     1.977870e+02             5.362243e+06
## population_density            -8.799704e+02            -3.834964e+07
## total_area                    -4.550537e+00            -3.066010e+05
## unemployed                     1.031507e+01            -1.412446e+06
##                                employed    gini_index     land_area
## solar_system_count            94.05466  -0.003714942 -4.472656e+00
## average_household_income 8192779.70458 185.729666631 -3.040062e+05
## employed                 1129326.30056 -12.914345934 -7.840249e+03
## gini_index                   -12.91435   0.003719889  2.904085e-01
## land_area                  -7840.24945   0.290408486  2.325095e+04
## per_capita_income        2420069.56591 205.730523159 -1.125395e+05
## population               2131321.36503 -20.954123177 -4.350819e+03
## population_density        260694.89740 115.963909293 -1.497733e+05
## total_area                 -7927.18039   0.296881018  2.335214e+04
## unemployed                 65073.20888  -0.389924299 -7.738572e+02
##                           per_capita_income    population population_density
## solar_system_count             1.231538e+03  1.977870e+02      -8.799704e+02
## average_household_income       6.727679e+08  5.362243e+06      -3.834964e+07
## employed                       2.420070e+06  2.131321e+06       2.606949e+05
## gini_index                     2.057305e+02 -2.095412e+01       1.159639e+02
## land_area                     -1.125395e+05 -4.350819e+03      -1.497733e+05
## per_capita_income              2.938056e+08 -1.320386e+06      -6.579194e+06
## population                    -1.320386e+06  4.697630e+06      -4.433703e+05
## population_density            -6.579194e+06 -4.433703e+05       1.521162e+08
## total_area                    -1.129784e+05 -4.386393e+03      -1.523425e+05
## unemployed                    -7.021585e+05  1.704399e+05       1.781637e+05
##                              total_area    unemployed
## solar_system_count       -4.550537e+00  1.031507e+01
## average_household_income -3.066010e+05 -1.412446e+06
## employed                 -7.927180e+03  6.507321e+04
## gini_index                2.968810e-01 -3.899243e-01
## land_area                 2.335214e+04 -7.738572e+02
## per_capita_income        -1.129784e+05 -7.021585e+05
## population               -4.386393e+03  1.704399e+05
## population_density       -1.523425e+05  1.781637e+05
## total_area                2.346255e+04 -7.810998e+02
## unemployed               -7.810998e+02  1.787993e+04
```

By viewing the covariance matrix, we could predict the correlation between the variables in the data set where in such as average_household_income and per_capita_income having large correlation.

```
cor((deepsolar1$water_area+deepsolar1$land_area),deepsolar1$total_area)
```

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

```
cor(deepsolar1$per_capita_income,deepsolar1$average_household_income)
```

```
## [1] 0.9253816
```

```
cor(((deepsolar1$employed+deepsolar1$unemployed)/deepsolar1$employed),deepsolar1$em
ploy_rate)
```

```
## [1] -0.9844408
```

Similar to these variables we can also remove few of the highly corelated and ineffective variables removed, that would contribute to the change in the intercept values to predict the solar_system_count.

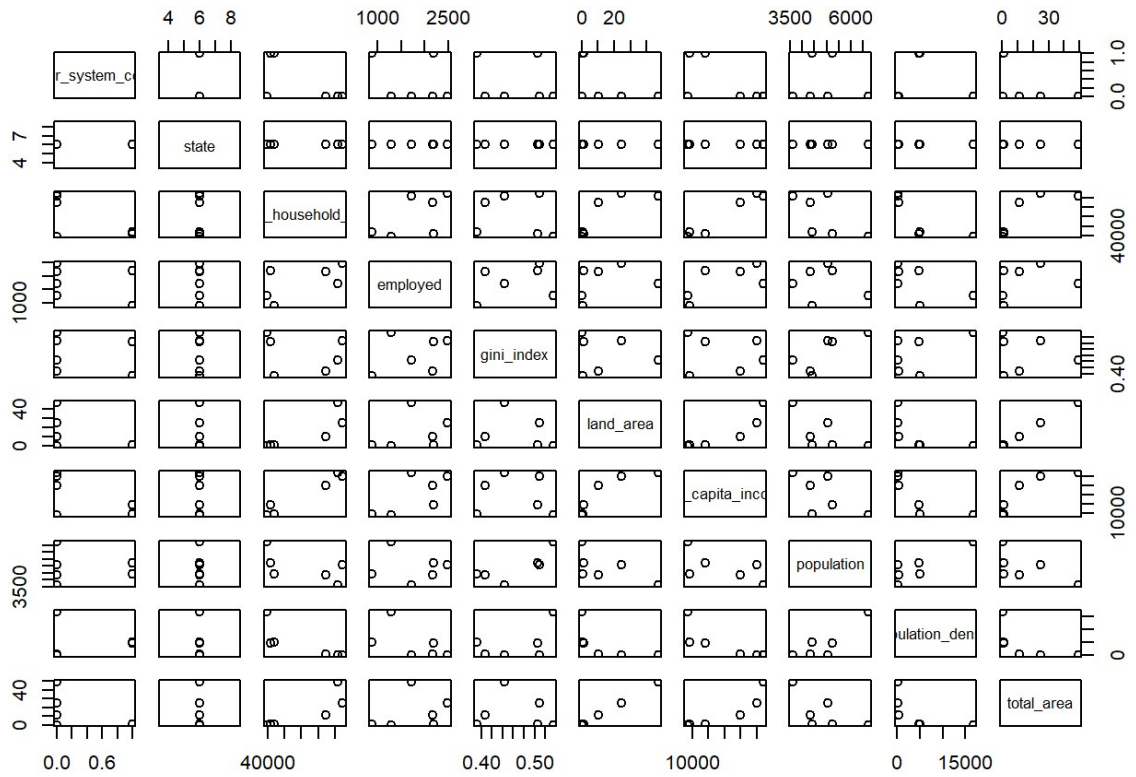
```
table(deepsolar1$solar_system_count)
```

```
##
##     0     1
##  9836 10900
```

Scaling the dataset:

The dataset has a widespread range of values which makes it difficult to predict using supervised classical methods, and this widespread data is been identified in the following pair plots where we could witness the axial limits which vary from one variable to another.

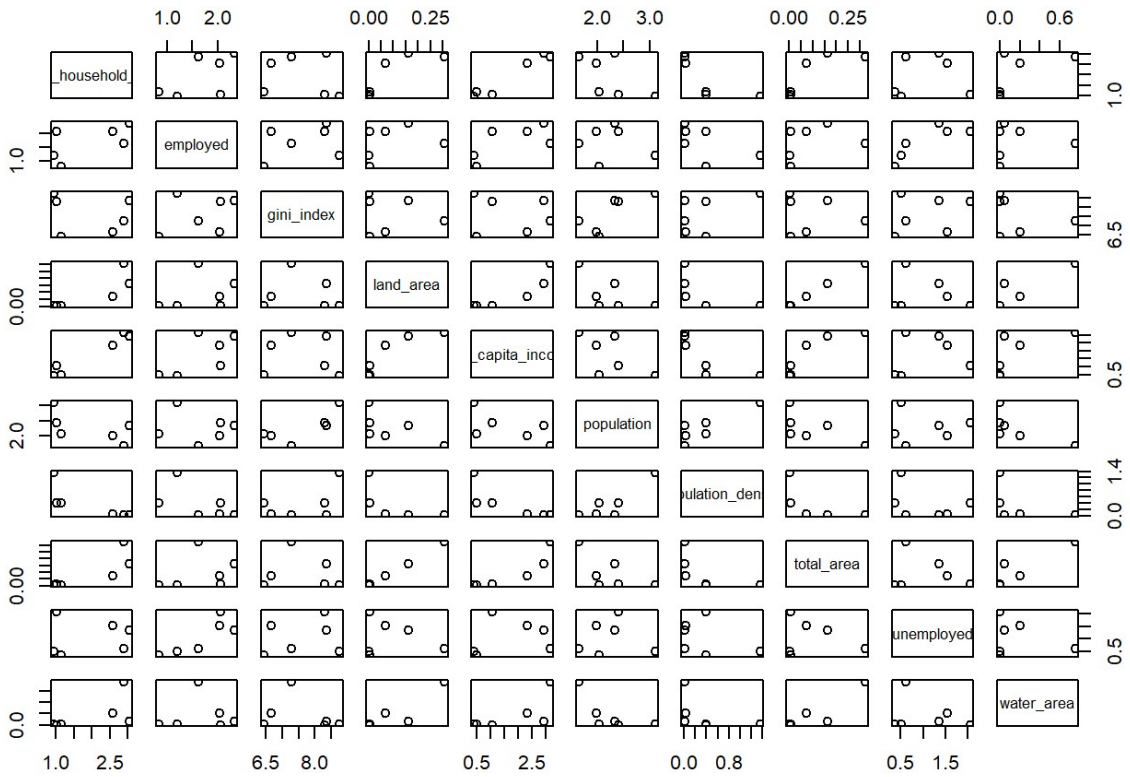
```
pairs(head(deepsolar[,c(1:10)]))
```



In order to scale the data, we initially remove the categorical data and scale only the numerical and integer variable values by dividing its own values with its corresponding standard deviation. Post scaling up we then combine back the respective categorical variable data's back into the scaled dataset.

```
deepsolar1=deepsolar[,-c(1,2,76,79)]
deepsolar1=sweep(deepsolar1,2,apply(deepsolar1,2,sd),"/")
deepsolar1=cbind(deepsolar1,deepsolartemp[,c(1,2,76,79)])
deepsolar1<-as.data.frame(deepsolar1)
```

```
pairs(head(deepsolar1[,c(1:10)]))
```



Post the standardization of the dataset, we could find that the range of values in all the numeric variables are similar and lead to a better fitting of models into the dataset.

Now we are removing the variables which have less or none impact on the predictor variables to explain very less or no change in the prediction of the solar_system_count.

```
deepsolar1<-subset(deepsolar1,select=-c(number_of_years_of_education,race_two_more_
rate,electricity_consume_total,per_capita_income,total_area,electricity_price_resid
ential,electricity_price_commercial,electricity_price_industrial,electricity_price_
transportation,electricity_price_overall,electricity_consume_residential,electricit
y_consume_commercial,electricity_consume_industrial,electricity_consume_total,occup
ancy_vacant_rate,population_density,voting_2016_dem_win,voting_2012_dem_win))
```

Methods:

As we already scaled the data and removed some variables which are inefficient for the prediction. We would now procced with the supervised methods of classification for predicting the solar_system_count.

Processing the data:

Initially we split the data as 75% of observations for the processing and validation (dat) and rest 25% of the data for the testing (dat_test).

The data for processing and validation is further splited into 3 parts for processing and 1 part for the validation using the folds methodology.

```
# to have the same initial split
set.seed(19200314)
D <- nrow(deepsolar1)
keep <- sample(1:D,size=0.75*nrow(deepsolar1))
test <- setdiff(1:D, keep)

dat <- deepsolar1[keep,]   # For the Processing and validation
#str(dat)
dat_test <- deepsolar1[test,] # For the Testing

folds <-rep(1:4,ceiling(nrow(dat)/4))
folds <-sample(folds)# random permute
folds <-folds[1:nrow(dat)]
train <-which(folds!=1)
test <-setdiff(1:nrow(dat), train)
#str(dat_test)
N <- nrow(dat)

traininig_data<-dat[train,]
validation_data<-dat[test,]
```

Binomial Logistic Regression:

AS we know that the solar_system_count which is the predicting variable is a binomial, we initial fit the data for processing with the binomial logistic regression.

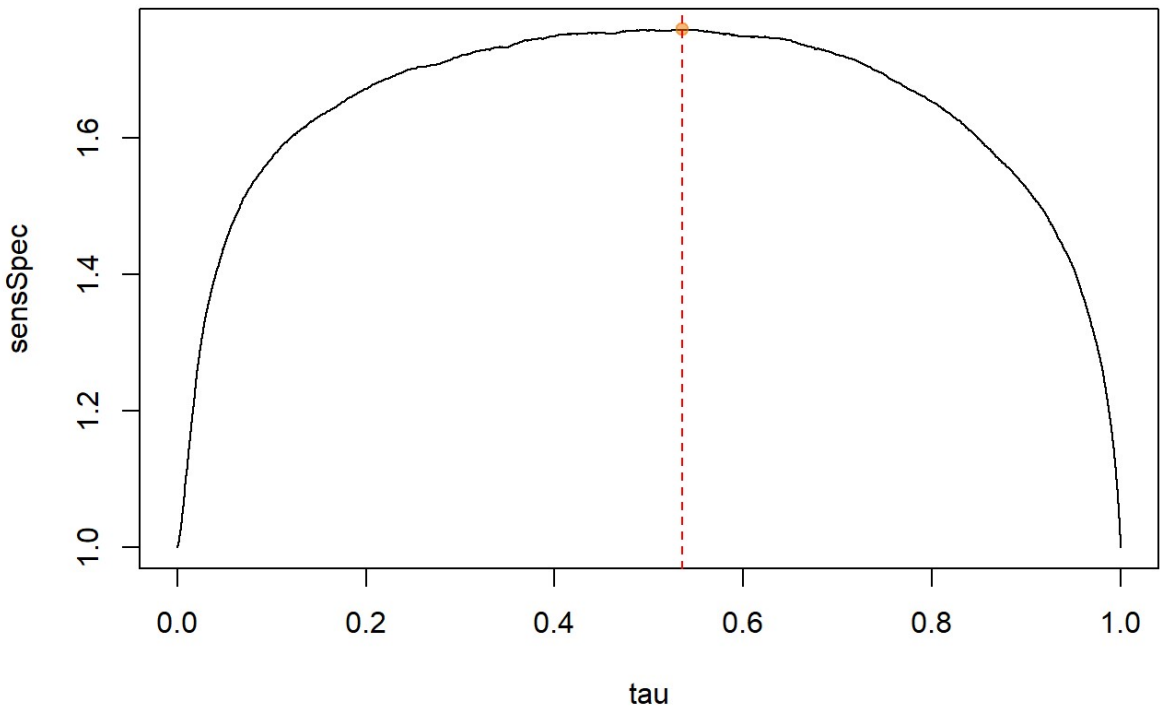
```
# Binomial fitting
fit1=glm(solar_system_count ~ ., data =traininig_data,family="binomial",na.action=n
a.omit)
sumfit <-summary(fit1)

library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
predObj <-prediction(fitted(fit1), traininig_data$solar_system_count)
sens <-performance(predObj,"sens")
spec <-performance(predObj,"spec")
tau <-sens@x.values[[1]]
sensSpec <-sens@y.values[[1]]+spec@y.values[[1]]
best <-which.max(sensSpec)
#plot
plot(tau, sensSpec, type = "l")
points(tau[best], sensSpec[best], pch = 19, col = adjustcolor("darkorange2", 0.5))
abline(v=tau[best],col="red",lwd=1,lty=2)
```



In this process in order to attain the best Tau value, we derive it using the specs and sens performance method , where from the above plot of range of tau values against the sensspec, we could compute the best tau value as

```
tau[best]
```

```
##      5421
## 0.5350022
```

Next we predict the fit with response to the validation data to know the prediction evaluated with the fit which was processed by using the processing data.

```
pred <- predict(fit1, type = "response", newdata = validation_data) # Predicting Lo
gistic Regression
pred <-ifelse(pred>tau[best],1,0)
tabb<-table(validation_data$solar_system_count, pred)
acclog <- sum(diag(tabb))/sum(tabb)
```

The below tabulation prevails that the diagonal entity has predominantly higher value than the other in the tabulation, concluding that the fit is working with the data. But in further case there are numeric values in the non-diagonal which are not less weighted. Hence, we need to derive the accuracy of the fitting model too.

```
tabb
```

```
##       pred
##           0    1
##   high 1797  207
##   low   266 1618
```

```
cat("The accuracy for the  Binomial logistic regression is :",acclog)
```

```
## The accuracy for the  Binomial logistic regression is : 0.8783436
```

The Accuracy of the binomial logistic regression is also nominal and achieves more than 80% of accuracy, still which isn't enough we need to try fitting this data with further models until we achive the best model.

Random forest:

Random forests extend bagging by considering random splits, and this to further make the trees more diverse.

The random forest method which in turn is supposed to be a higher accuracy model which fits the model by recursive fitting against the splitting the data within itself is evaluated now:

```
# Random forest fitting
fit2=randomForest(solar_system_count~.,data=traininig_data)
#predict the classification of the test data observations in the dropped fold
pred2 <- predict(fit2, type = "class", newdata = validation_data) # Predicting  Ran
dom Forest
tab2 <- table(validation_data$solar_system_count, pred2)
accrandmf <- sum(diag(tab2))/sum(tab2)
```

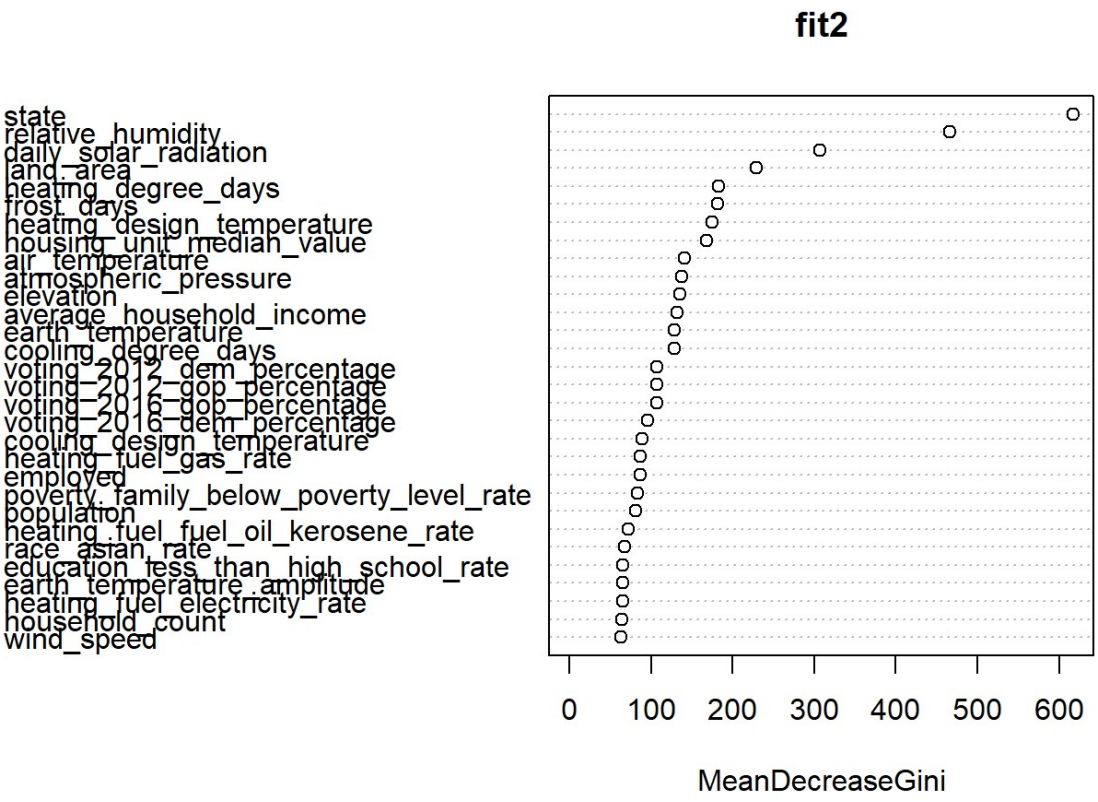
The Random forest fitting now computes a higher accuracy than the binomial logistic regression.

```
cat("The accuracy for the  Random forest fitting is :",accrandmf)
```

```
## The accuracy for the  Random forest fitting is : 0.8971193
```

The variable importance plot , which is visualised by the random forest fit object describes the best suited explainatory variables that has higher influence on the solar_system_count.

```
varImpPlot(fit2)
```



Bagging:

Bagging is a method that uses bootstrapping to increase the stability of a classificationmethod. Here we will use this procedure for supervised classification of the deepsolar data.

```
# Bagging fitting

fit3=bagging(solar_system_count~.,data=dat,subset=train)
# predict the classification of the test data observations in the dropped fold
pred3 <- predict(fit3, type = "class", newdata = validation_data) # Predicting  Bag
ging
tab3 <- table(validation_data$solar_system_count, pred3$class)
accbaging <- sum(diag(tab3))/sum(tab3)
```

```
cat("The accuracy for the  Bagging fitting is :",accbaging)
```

```
## The accuracy for the  Bagging fitting is : 0.8626543
```

Support vector:

```
# support vector machine fitting
fit4=ksvm(solar_system_count~.,data=traininig_data)
# predict the classification of the test data observations in the dropped fold
pred4 <- predict(fit4,newdata=validation_data) # Predicting  Bagging
tab4 <- table(validation_data$solar_system_count, pred4)
accsuppvec <- sum(diag(tab4))/sum(tab4)
```

```
cat("The accuracy for the  support vector machine fitting is :",accsuppvec)
```

```
## The accuracy for the  support vector machine fitting is : 0.8863169
```

Boosting:

Boosting uses a collection of weak classifiers on weighted versions of the training data to enhancethe overall prediction performance, placing more importance on misclassified observations at each iterations.

```
# Boosting vector machine fitting
fit6 <- boosting(solar_system_count~.,data = traininig_data,coeflearn = "Breiman",b
oos = FALSE)
  # predict the classification of the test data observations in the dropped fold
pred6 <- predict(fit6, newdata = validation_data) # Predicting  Bagging
tab6 <- table(validation_data$solar_system_count, pred6[["class"]])
accboost <- sum(diag(tab6))/sum(tab6)
```

```
cat("The accuracy for the  Boosting fitting is :",accboost)
```

```
## The accuracy for the  Boosting fitting is : 0.8922325
```

overall accuracy:

On a single iteration, we could predict that the Random forest fitting supervised method generates the higher accuracy being a optimal model.

```
acc <- c(binomial = acclog,RandomForest = accrandmf,Bagging=accbaging,Supportvector
=accsuppvec,Boosting=accboost)
acc
```

```
##      binomial  RandomForest       Bagging Supportvector      Boosting
##     0.8783436     0.8971193     0.8626543     0.8863169     0.8922325
```

For 100 iterations in each classical supervised model fitting:

Now we couldn’t conclude the optimal model for the dataset by a single iteration of data fitting, in order to get a concise average accuracy of all the fitting model, we now execute the fitting and evaluate the accuracy against 100 iterations.

```
#####################
library(foreach)
library(doParallel)

#setup parallel backend to use many processors
cl <- parallel::makeCluster(5)
doParallel::registerDoParallel(cl)

# replicate the process a number of times
R <- 100
acca <- matrix(NA, 1, 5)
outa <- vector("list", R)
out2 <- vector("list", R)


out2<-foreach(r =1:R,.combine = rbind) %dopar% {
  library(randomForest)
  library(adabag)
  library(kernlab)
  # Data Pre-processing
  keep <- sample(1:D,size=0.75*nrow(deepsolar1))
test <- setdiff(1:D, keep)
dat <- deepsolar1[keep,]   # For the Processing and validation
dat_test <- deepsolar1[test,] # For the Testing

foldsL <-rep(1:4,ceiling(nrow(dat)/4))
foldsL <-sample(foldsL)# random permute
foldsL <-foldsL[1:nrow(dat)]
trainL <-which(foldsL!=1)
testL <-setdiff(1:nrow(dat), trainL)
N <- nrow(dat)

traininig_dataL<-dat[trainL,]
validation_dataL<-dat[testL,]
  # Binomial fitting
  fit1L=glm(solar_system_count ~ ., data =traininig_dataL,family="binomial")
  # predict the classification of the test data observations in the dropped fold
  pred1L <- predict(fit1L, type = "response", newdata = validation_dataL) # Predict
ing Multinomial Regression
  tab1La <- table(validation_dataL$solar_system_count, pred1L)
  acca[1,1]  <- sum(diag(tab1La))/sum(tab1La)
  # Random forest fitting
  fit2L=randomForest(solar_system_count~.,data=traininig_dataL)
  #predict the classification of the test data observations in the dropped fold
  pred2L <- predict(fit2L, type = "class", newdata = validation_dataL) # Predicting
Random Forest
  tab2La <- table(validation_dataL$solar_system_count, pred2L)
  acca[1,2] <- sum(diag(tab2La))/sum(tab2La)
  # Bagging fitting
  fit3L=bagging(solar_system_count~.,data=dat,subset=trainL)
  # predict the classification of the test data observations in the dropped fold
  pred3L <- predict(fit3L, type = "class", newdata = validation_dataL) # Predicting
```

```
Bagging
  tab3La <- table(validation_dataL$solar_system_count, pred3L$class)
  acca[1,3] <- sum(diag(tab3La))/sum(tab3La)

  # support vector machine fitting
  fit4L=ksvm(solar_system_count~.,data=traininig_dataL)
  # predict the classification of the test data observations in the dropped fold
  pred4L <- predict(fit4L,newdata=validation_dataL) # Predicting  Bagging
  tab4La <- table(validation_dataL$solar_system_count, pred4L)
  acca[1,4] <- sum(diag(tab4La))/sum(tab4La)

  # Boosting vector machine fitting
  fit6L <- boosting(solar_system_count~.,data = traininig_dataL,coeflearn = "Breima
n",boos = FALSE)
  # predict the classification of the test data observations in the dropped fold
  pred6L <- predict(fit6L, newdata = validation_dataL) # Predicting  Bagging
  tab6La <- table(validation_dataL$solar_system_count, pred6L[["class"]])
  acca[1,5] <- sum(diag(tab6La))/sum(tab6La)
  ######
  outa=acca
  ##
}
parallel::stopCluster(cl)
```

Results and discussion:

Mean :

By execution of all the classical supervised modelling for the 100 iterations, now we could evaluate the accuracy of the each models by finding the mean of the accuracy for each models.

```
# Calculating Mean on all iterations
meanAcc <- colMeans(out2) # estimated mean accuracy
cat("### OVERALL ACCURACY ####","\n")
```

```
## ### OVERALL ACCURACY ####
```

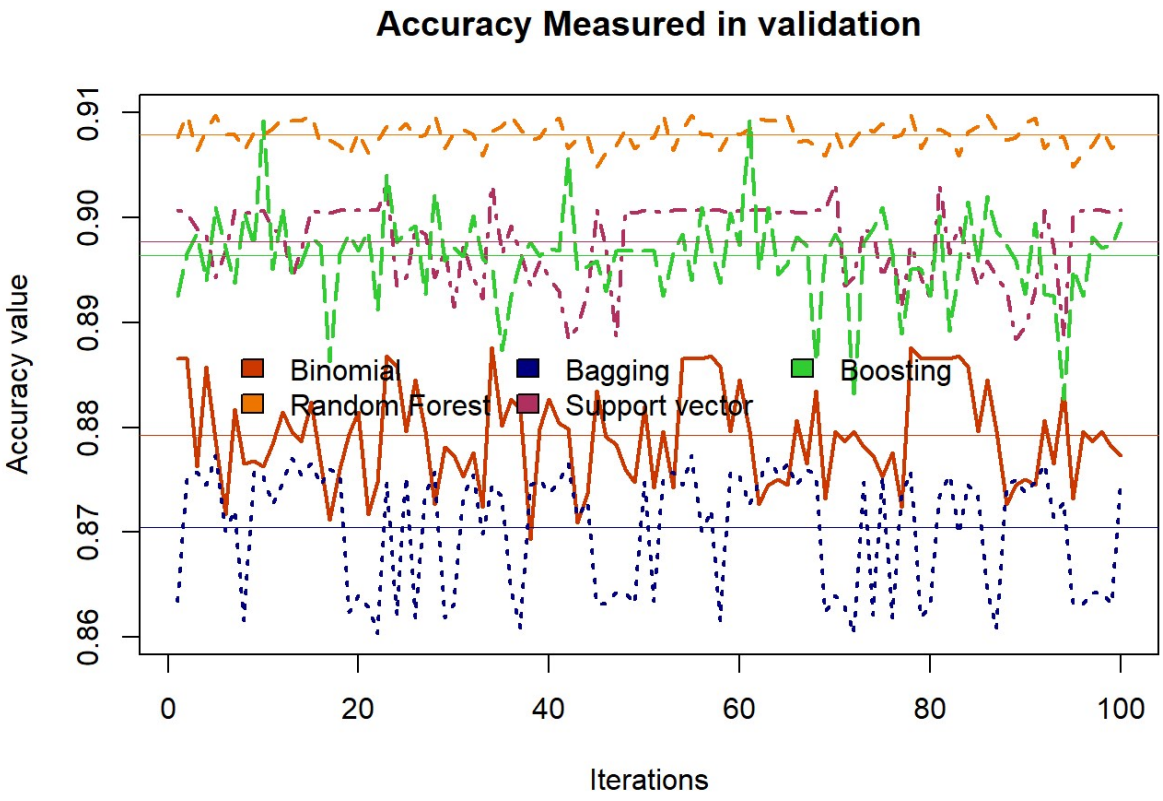
```
cat("Binomial Regression","\t","Random Forest","\t","Bagging","\t","Support vecto
r","\t","Boosting","\t",
    "\n",meanAcc[1],"\t\t",meanAcc[2],"\t",meanAcc[3],"\t",meanAcc[4],"\t\t",meanAc
c[5],"\n")
```

```
## Binomial Regression   Random Forest   Bagging     Support vector      Boosting
##  0.879285         0.9078858   0.8705041   0.8977289       0.8964198
```

Here on the findings above, we could conclude that the best model of supervised classical fitting is defined to be the random forest for this dataset of deep solar predicting the solar_system_count.

Matplot:

```
#Mat plot
matplot(out2[,1:5], type = "l", pch= 1, cex= 0.5, Ity= 2.5,lwd=2,
        col = c("orangered3","darkorange2","navy","maroon","limegreen"),
        ylab="Accuracy value", xlab = "Iterations",
        main="Accuracy Measured in validation")
abline(h = c(mean(out2[,1]),mean(out2[,2]), mean(out2[,3]), mean(out2[,4]), mean(ou
t2[,5])),
       col= c("orangered3","darkorange2","navy","maroon","limegreen"),lwd=0.5, Ity=
2)
legend(5,0.889, fill = c("orangered3","darkorange2","navy","maroon","limegreen"),
       legend = c("Binomial","Random Forest","Bagging","Support vector","Boostin
g"), bty = "n",ncol=3)
```



The Mat plot for the above fitting all the methods of fitting, we could see that the random forest peaks the accuracy and its mean median being more than 90% of accuracy.

Now we could perform the final testing of the dataset using the best acquired model of fitting as we could derive the tabulation how well the fit has been optimal with the test data.

```
cat(" ##### Testing Data ####","\n")
```

```
##  ##### Testing Data ####
```

```
Random_Forest_testfit = predict(fit2, newdata=dat_test)

tb=table(Random_Forest_testfit,dat_test$solar_system_count)
vec <- sum(diag(tb))/sum(tb)
cat(" Random Forest Accuracy in Test Data","\n",vec,"\n")
```

```
##  Random Forest Accuracy in Test Data
##  0.9014275
```

```
cat(" #####","\n")
```

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

```
cat(" Overall Comparison of the Test data with the Fit","\n")
```

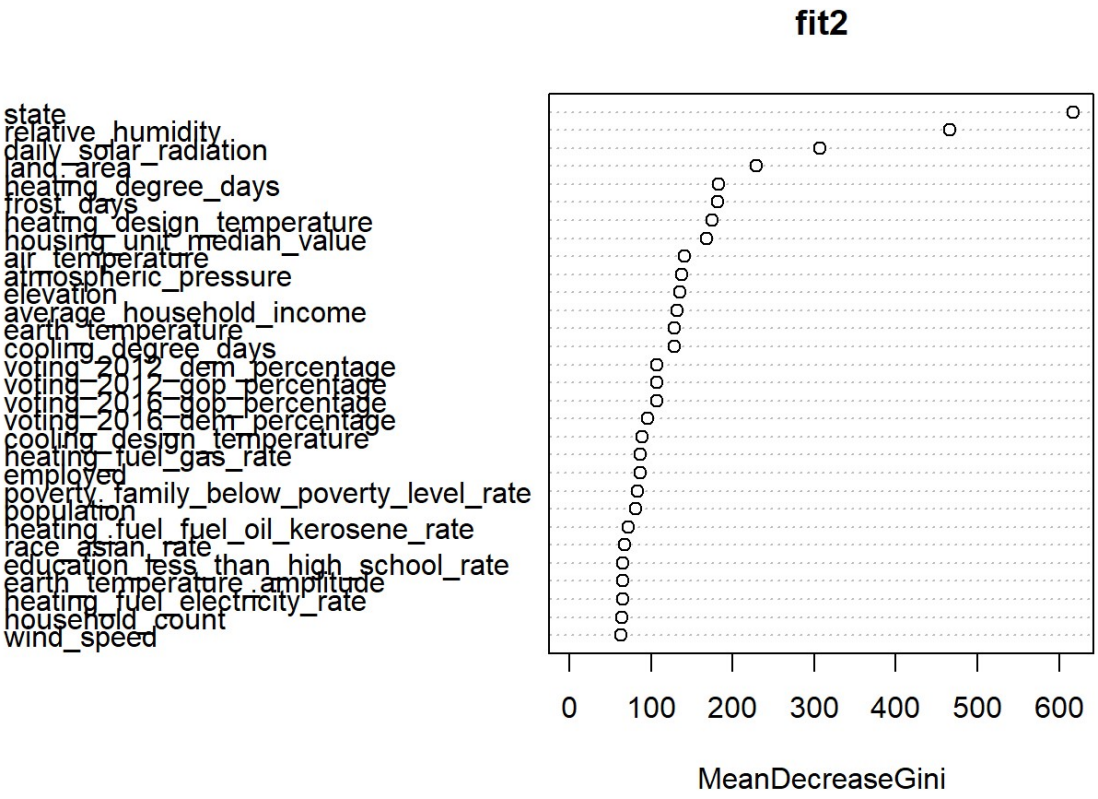
```
##  Overall Comparison of the Test data with the Fit
```

```
tb
```

```
##
## Random_Forest_testfit high  low
##                  high 2480  252
##                  low   259 2193
```

The Random forest supervised classical model fitting has evaluated a accuracy more than 90% on the test data, were in by examining the tabulation we could see 4666(2481+2185) observations out of 5184 in the test data is said to be fitted correctly and only about 518 observations out of 5184 has been differently fitted by the random forest when compared to the initial data.

```
varImpPlot(fit2)
```



Conclusion:

Here I would like to conclude by proving the prediction that the best optimal model for the deepsolar dataset is the Random Forest classical supervised fitting model, which adheres a better accuracy in 100 iteration fitting of the dataset and better explanation of variables which influences more to the prediction variable of solar_system_count using variable importance plot determines that variables such as state,relative_humidity and daily_solar_radiations leads to three of the top best influencers to the solar_system_count. we could also visualise the mat plot where the accuracy level of the random forest remains to be stable and high as compared to its peer models.

References:

Online web portal reference (https://privefl.github.io/blog/a-guide-to-parallelism-in-r/)

Foreach package (https://cran.r-project.org/web/packages/foreach/vignettes/foreach.html)

Do Parellel (https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf)