Solar Power System Coverage Prediction

Harshad Kumar Elangovan - 19200349

26/04/2020

Abstract

The report describes about the analysis aproach of several supervised classification methods on the deepsolar dataset. This dataset contains several predictor variables with a target variable “solar system count”.This report elaborates the modelling stages of each method in predicting the solar power system coverage of a tile given the collection of predictor features and produces an optimal model based on the training and test data.

Introduction

The dataset on deepsolar contains a subset of the DeepSolar database. Each row of the dataset is a “tile” of interest, that is an area corresponding to a detected solar power system, constituted by a set of solar panels on top of a building or at a single location such as a solar farm. For each system, a collection of features record social, economic, environmental, geographical, and meteorological aspects of the tile (area) in which the system has been detected.

```
library(dplyr)
library(mlbench)
library(rpart)
library(kernlab)
library(ROCR)
library(randomForest)
library(adabag)
library(nnet)
library(foreach)
library(doParallel)
deepsolar<-read.csv("data_project_deepsolar.csv")
```

Exploratory Data Analysis

We will analyze the deepsolar dataset to verify if there are any data redundency in the dataset. For this, we can create correlation matrix among several variables and drop the variables which are dependent on other predictors. By taking the correlation of land_area and water_area with total area, we get the value as ‘one’,which states that these variables are highly dependent on total area variable. So we can remove the “total_area” variable as it produces redundant values.

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

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

```
deepsolar<-select(deepsolar,-total_area)
```

The employee rate is related to variables “employed” and “unemployed”. So it can be dropped.

```
cor((deepsolar$employed/(deepsolar$employed+deepsolar$unemployed)),deepsolar$employ
_rate)
```

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

```
deepsolar<-select(deepsolar,-employ_rate)
```

The “average_household_income” and “per_capita_income” are highly correlated to each other. So, “per_capita_income” can be dropped.

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

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

```
deepsolar<-select(deepsolar,-per_capita_income)
```

Variable “voting_2016_dem_win” can be removed as it is dependent on “voting_2016_dem_percentage” and “voting_2016_gop_percentage” . Similarly,“voting_2012_dem_win” is dependent on “voting_2012_dem_percentage” and “voting_2012_gop_percentage”. So we can remove both the win variables.

```
deepsolar<-subset(deepsolar,select = -c(voting_2016_dem_win,voting_2012_dem_win))
```

The variables related to state variable (i.e) “electricity_price_residential”,“electricity_price_commercial”, “electricity_price_industrial”, “electricity_price_transportation”,“electricity_price_overall”,“electricity_consume_residential”, “electricity_consume_commercial”,“electricity_consume_industrial” and “electricity_consume_total” can be removed as all the values are the same for a state and it only varies with different state values.

```
deepsolar<-subset(deepsolar,select = -c(electricity_price_residential,electricity_p
rice_commercial,
electricity_price_industrial,electricity_price_transportation,electricity_price_ove
rall,electricity_consume_residential,
electricity_consume_commercial,electricity_consume_industrial,electricity_consume_t
otal))
```

Taking the correlation between “occupant_vacant_rate” with (1-(household_count/housing_unit_count)) given a highly correlated value. So we can drop “occupant_vacant_rate” variable aswell.

```
cor((1-(deepsolar$household_count/deepsolar$housing_unit_count)),deepsolar$occupanc
y_vacant_rate)
```

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

```
deepsolar<-select(deepsolar,-occupancy_vacant_rate)
```

Variable “population density” can be dropped as it is highly correlated when dividing “population” by “land and water” variables.

```
cor((deepsolar$population/(deepsolar$land_area+deepsolar$water_area)),deepsolar$pop
ulation_density)
```

```
## [1] 0.9929551
```

```
deepsolar<-select(deepsolar,-population_density)
```

Standardization of Dataset

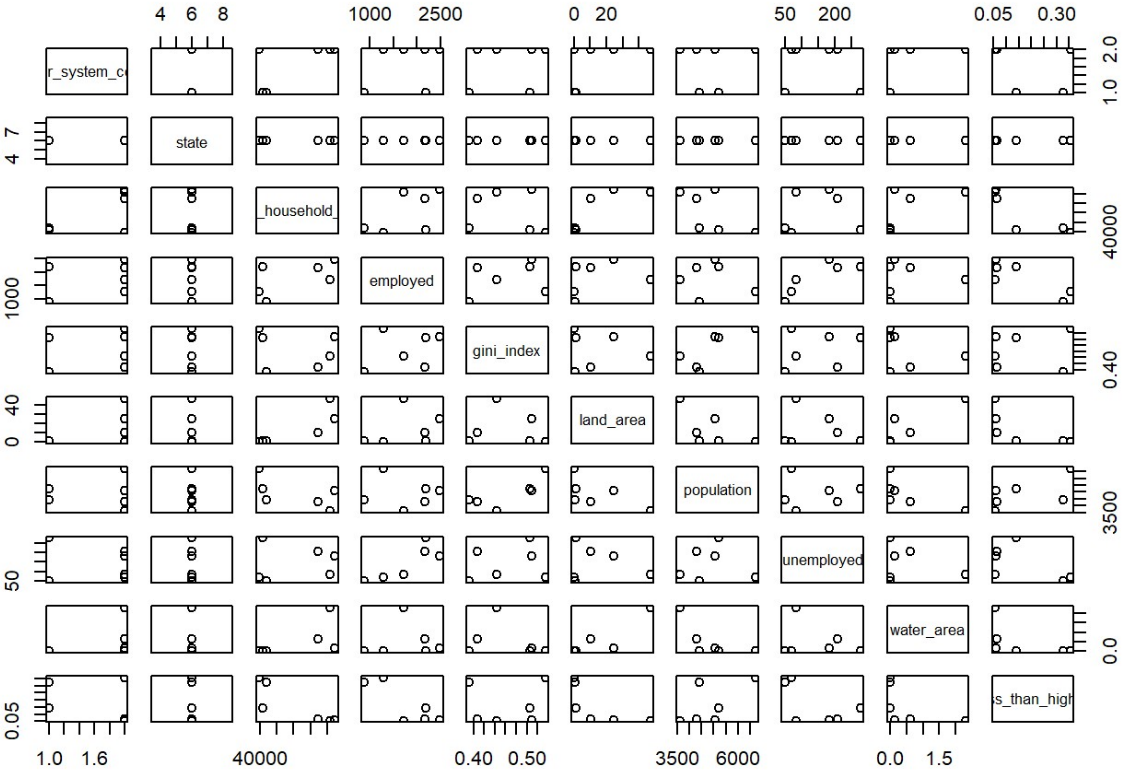
From the sample values of first 10 variables,we can see that the values are widely spread.This can also be seen by pair plotting the data.The plotting range of the variables vary widely. So, we will have to standardize the data before using it for fitting the models.

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

```
##   solar_system_count state average_household_income employed gini_index
## 1                low    ny                108954.82     2168     0.4072
## 2                low    ny                122821.44     1727     0.4434
## 3                low    ny                 39521.76     1279     0.5342
## 4               high    ny                 48647.40      880     0.3928
## 5                low    ny                129044.96     2491     0.5091
## 6               high    ny                 43428.74     2196     0.5058
##    land_area population unemployed  water_area
## 1 10.5840600       4315        206 0.622382000
## 2 47.0470000       3596         83 2.299237000
## 3  0.3999543       6727         68 0.005623192
## 4  0.8971227       4423         49 0.005217785
## 5 24.6710800       5073        181 0.154506900
## 6  1.0900960       5226        276 0.000000000
##   education_less_than_high_school_rate
## 1                           0.06479339
## 2                           0.05585516
## 3                           0.35662526
## 4                           0.32517215
## 5                           0.05925065
## 6                           0.14159869
```

The axis range of the variables vary widely among the variables.

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



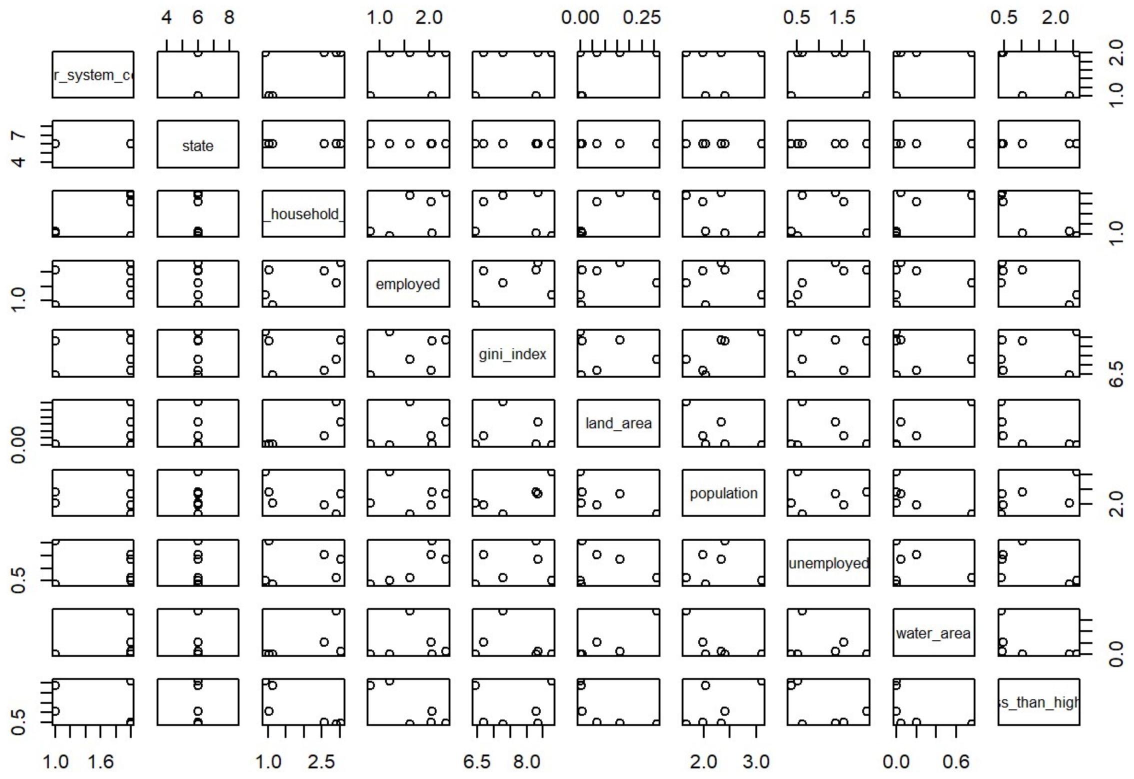
```
deepsolarSdval<-apply(deepsolar[,-(1:2)],2,sd)
deepsolarSdval<-sweep(deepsolar[,-c(1:2)],2,deepsolarSdval,"/")
deepsolarSdval<-cbind(deepsolar[,c(1:2)],deepsolarSdval)
```

After standardizing and looking a the subset of data, we see that the values are now evenly spread. The axis range of all the variables are similar for all the variables.

```
head(deepsolarSdval[,c(1:10)])
```

```
##   solar_system_count state average_household_income  employed gini_index
## 1                low    ny                2.5688107 2.0400911   6.676406
## 2                low    ny                2.8957419 1.6251095   7.269938
## 3                low    ny                0.9317984 1.2035408   8.758684
## 4               high    ny                1.1469520 0.8280812   6.440306
## 5                low    ny                3.0424727 2.3440345   8.347148
## 6               high    ny                1.0239125 2.0664391   8.293041
##     land_area population unemployed  water_area
## 1 0.069411578   1.990863  1.5405803 0.204870386
## 2 0.308540058   1.659129  0.6207192 0.756843178
## 3 0.002622950   3.103716  0.5085411 0.001850994
## 4 0.005883442   2.040692  0.3664487 0.001717546
## 5 0.161796001   2.340591  1.3536167 0.050859260
## 6 0.007148985   2.411182  2.0640785 0.000000000
##   education_less_than_high_school_rate
## 1                            0.4751454
## 2                            0.4095993
## 3                            2.6152185
## 4                            2.3845653
## 5                            0.4344992
## 6                            1.0383772
```

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



Fitting the methods

There are several supervised classification methods that can be used to predict the solar power system coverage of a tile given the collection of predictor features.We will make use of -Classification Trees -Logistic Regression -Random Forest -Bagging -Boosting -Support Vector Machines

Here, Multinomial Logistic Regression is not required as the target variable has only two variables “low” and “high”. These two values can be updated to 0 and 1 while doing logistic modelling.

We will split the data into training and test data, which will be used for fitting,training and testing the model.

```
set.seed(19200349)
N<-nrow(deepsolarSdval)
trainingdatanum<- sample(1:N,size = 0.75*N)
testdatanum<-setdiff(1:N,trainingdatanum)

#Training data
trainingdata<-deepsolarSdval[trainingdatanum,]
#Test data
testdata<-deepsolarSdval[testdatanum,]

#We will split the training data further into training and validation data
M<-nrow(trainingdata)
trainnum<-sample(1:M,size = 0.75*M)
valnum<-setdiff(1:M,trainnum)
train<-trainingdata[trainnum,]
validation<-trainingdata[valnum,]
```

We will use these data for fitting different models and predict the accuracy of each models . There are two main stages of each model,Fitting the model based on training data.Secondly,predicting the fitted model using the validation data and checking its accuracy. From the accuracy, we can come into conclusion in finding an optimal method given its fitted model.

```
# classification tree
fit1<-rpart(train$solar_system_count~.,data=train)
pred1<-predict(fit1, type = "class",newdata = validation)
table1<-table(validation$solar_system_count,pred1)
accuracy1<-sum(diag(table1))/sum(table1)



#Logistic Regression

deepsolarSdval$solar_system_count<-recode(deepsolarSdval$solar_system_count,'low'=
0,'high'=1)
trainLogisticR<-train
validationLogisticR<-validation

trainLogisticR$solar_system_count<-recode(trainLogisticR$solar_system_count,'low'=
0,'high'=1)
validationLogisticR$solar_system_count<-recode(validationLogisticR$solar_system_cou
nt,'low'=0,'high'=1)

fit2<-glm(trainLogisticR$solar_system_count~.,data=trainLogisticR,family = "binomia
l")
#summary(fit2)
#Prediction object is created using prediction function
predObj<-prediction(fitted(fit2),trainLogisticR$solar_system_count)

#Sensitivity and Specificity calculation for optimal threshold value
sens<-performance(predObj,"sens")
spec<-performance(predObj,"spec")
tau<-sens@x.values[[1]]
#Sum of Sensitivity and Specificity
sensSpec<-sens@y.values[[1]] + spec@y.values[[1]]
#Finding the maximum of Sum of Sensitivity and Specificity
best<-which.max(sensSpec)

pred2<-predict(fit2,type = "response",newdata = validationLogisticR)

#Classification for optimal threshold
pred2<-ifelse(pred2>tau[best],1,0)
table2<-table(validationLogisticR$solar_system_count,pred2)
accuracy2<-sum(diag(table2))/sum(table2)


#Random Forest
fit3<-randomForest(train$solar_system_count~.,data=train, importance=TRUE)
pred3<-predict(fit3, type = "class",newdata = validation)
table3<-table(validation$solar_system_count,pred3)
accuracy3<-sum(diag(table3))/sum(table3)

#Bagging

fit4<-bagging(solar_system_count~.,data = train)
```

```
pred4<-predict(fit4, type = "class",newdata = validation)
table4<-table(validation$solar_system_count,pred4$class)
accuracy4<-sum(diag(table4))/sum(table4)

#Boosting
fit5<-boosting(solar_system_count~.,data = train)
pred5<-predict(fit5, type = "class",newdata = validation)
table5<-table(validation$solar_system_count,pred5$class)
accuracy5<-sum(diag(table5))/sum(table5)

#Support Vector Machines
fit6<-ksvm(solar_system_count~.,data = train)
pred6<-predict(fit6, newdata = validation)
table6<-table(validation$solar_system_count,pred6)
accuracy6<-sum(diag(table6))/sum(table6)

accuracytable <- c(ClassificationTree = accuracy1,LogisticRegression=accuracy2,Rand
omForest = accuracy3, Bagging=accuracy4, Boosting=accuracy5, Supportvector=accuracy
6)
print(accuracytable)
```

```
## ClassificationTree LogisticRegression       RandomForest            Bagging
##          0.8528807          0.8744856          0.8996914          0.8577675
##           Boosting      Supportvector
##          0.8958333          0.8909465
```

```
#Fitting the test data to Random Forest model as this model has the highest accurac
y.
bestmodel <- names( which.max(accuracytable) )

TestdataPred<-predict(fit3, type = "class",newdata = testdata)
testtable<-table(testdata$solar_system_count,TestdataPred)
testaccuracy<-sum(diag(testtable))/sum(testtable)
cat("The accuracy of the test data after running it on Random Forest model is ",tes
taccuracy,"\n")
```

```
## The accuracy of the test data after running it on Random Forest model is  0.8969
907
```

From the above table, we see that for one iteration with all the models, Random Forest has the highest accuracy when compared with other models and predicting the model with test data,gives a high accuracy as shown above. But, we cannot come to a conclusion using a single iteration. We will have to run all the models again in iterations, with different training samples and find the best model.

We will make use of Hold out sampling cross validation for finding the best model,which can then be used to run the model with test data.For running n iterations effeciently, we will work on parallel coding which uses for each function. This foreach function assigns the accuracy output to the out varible which is then used for finding the optimal model.

```r
library(foreach)
library(doParallel)
cl <- parallel::makeCluster(7)
doParallel::registerDoParallel(cl)

# replicate the process a number of times
R <- 100
out <- matrix(NA, R, 7)
colnames(out) <- c("ClassificationTree","LogisticRegression","RandomForest","Bagging","Boosting","Supportvector", "best")
out <- as.data.frame(out)

#for ( r in 1:R ) {

out<-foreach(r =1:R,.combine = rbind) %dopar% {

library(dplyr)
library(mlbench)
library(rpart)
library(kernlab)
library(ROCR)
library(randomForest)
library(adabag)
library(nnet)


  #We will split the training data further into training and validation data
  M<-nrow(trainingdata)
  trainnum<-sample(1:M,size = 0.75*M)
  valnum<-setdiff(1:M,trainnum)
  train<-trainingdata[trainnum,]
  validation<-trainingdata[valnum,]




  # classification tree
  fit1<-rpart(train$solar_system_count~.,data=train)
  pred1<-predict(fit1, type = "class",newdata = validation)
  table1<-table(validation$solar_system_count,pred1)
  table1
  accuracy1<-sum(diag(table1))/sum(table1)
  #print("Fit1 over")


  #Logistic Regression
  deepsolarSdval$solar_system_count<-recode(deepsolarSdval$solar_system_count,'low'=0,'high'=1)
  trainLogisticR<-train
  validationLogisticR<-validation

  trainLogisticR$solar_system_count<-recode(trainLogisticR$solar_system_count,'low'=0,'high'=1)
  validationLogisticR$solar_system_count<-recode(validationLogisticR$solar_system_count,'low'=0,'high'=1)

  fit2<-glm(trainLogisticR$solar_system_count~.,data=trainLogisticR,family = "binomial")
  summary(fit2)
  #Prediction object is created using prediction function
  predObj<-prediction(fitted(fit2),trainLogisticR$solar_system_count)

  #Sensitivity and Specificity calculation for optimal threshold value
  sens<-performance(predObj,"sens")
  spec<-performance(predObj,"spec")
  tau<-sens@x.values[[1]]
  #Sum of Sensitivity and Specificity
  sensSpec<-sens@y.values[[1]] + spec@y.values[[1]]
  #Finding the maximum of Sum of Sensitivity and Specificity
  best<-which.max(sensSpec)

  pred2<-predict(fit2,type = "response",newdata = validationLogisticR)

  #Classification for optimal threshold
  pred2<-ifelse(pred2>tau[best],1,0)
  table2<-table(validationLogisticR$solar_system_count,pred2)
  accuracy2<-sum(diag(table2))/sum(table2)
  #print("Fit2 over")

  #Random Forest
  fit3<-randomForest(solar_system_count~.,data=train, importance=TRUE)
  pred3<-predict(fit3, type = "class",newdata = validation)
  table3<-table(validation$solar_system_count,pred3)
  accuracy3<-sum(diag(table3))/sum(table3)
  #print("Fit3 over")

  #Bagging

  fit4<-bagging(solar_system_count~.,data = train)
  pred4<-predict(fit4, type = "class",newdata = validation)
  table4<-table(validation$solar_system_count,pred4$class)
  accuracy4<-sum(diag(table4))/sum(table4)
  #print("Fit4 over")

  #Boosting
  fit5<-boosting(solar_system_count~.,data = train)
  pred5<-predict(fit5, type = "class",newdata = validation)
  table5<-table(validation$solar_system_count,pred5$class)
  accuracy5<-sum(diag(table5))/sum(table5)
  #print("Fit5 over")

  #Support Vector Machines
  fit6<-ksvm(solar_system_count~.,data = train)
  pred6<-predict(fit6, newdata = validation)
  table6<-table(validation$solar_system_count,pred6)
  accuracy6<-sum(diag(table6))/sum(table6)
  #print("Fit6 over")

  accuracytable <- c(ClassificationTree = accuracy1,LogisticRegression=accuracy2,Ra
ndomForest = accuracy3, Bagging=accuracy4, Boosting=accuracy5, Supportvector=accura
cy6)

  #out[r,1] <- accuracy1
  #out[r,2] <- accuracy2
  #out[r,3] <- accuracy3
  #out[r,4] <- accuracy4
  #out[r,5] <- accuracy5
  #out[r,6] <- accuracy6
  #out[r,7] <- names( which.max(accuracytable) )
  bestaccuracy<-names( which.max(accuracytable) )


  #To check the iteration number
  #print(r)
  parallelvalues<-c(accuracy1,accuracy2,accuracy3,accuracy4,accuracy5,accuracy6,bes
taccuracy)
}
parallel::stopCluster(cl)
```

The accuracy of each method will be added to the out variable. This variable is used for summarizing the efficiency and result of the optimal model.

Results and discussion

The out data frame has the accuracy values of each model fitted based on the training data. From the values of out variable, we can see that the best model among the six mothods is “RandomForest”. This can also be seen by plotting the values of each model to a graph.

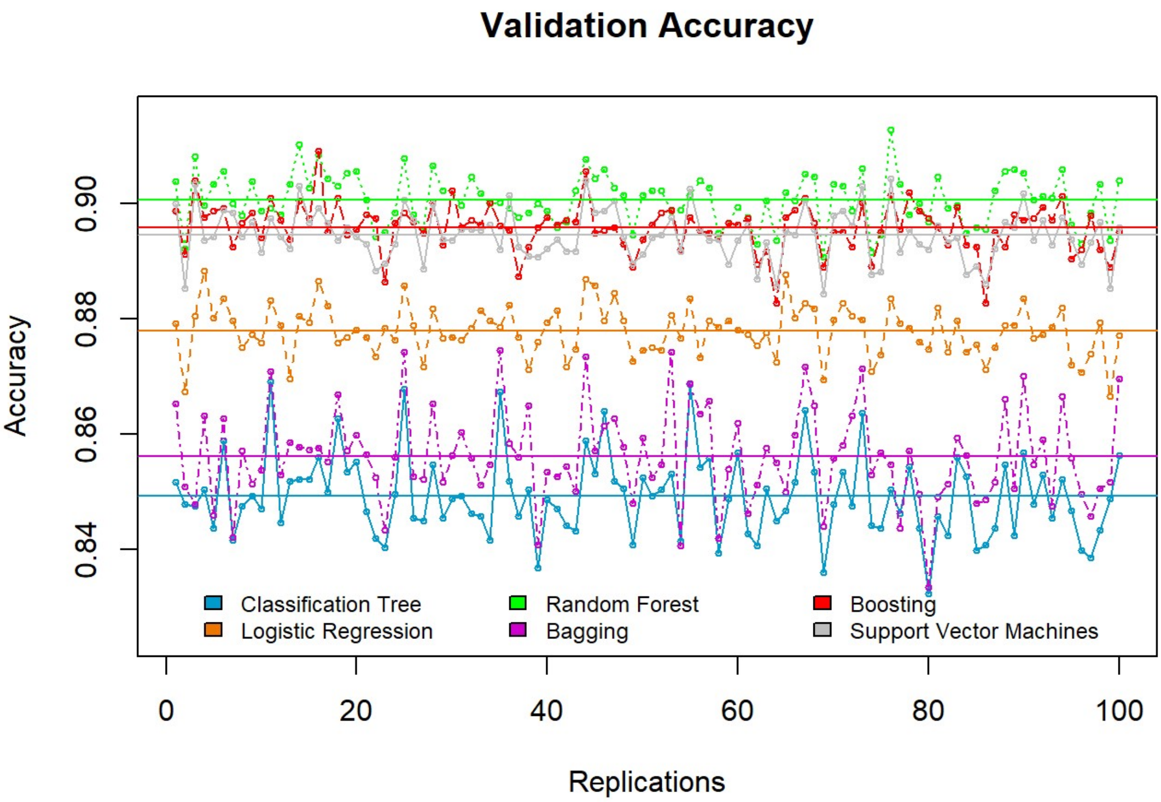
```
outdata <- matrix(NA, R, 7)
colnames(outdata) <- c("ClassificationTree","LogisticRegression","RandomForest","Ba
gging","Boosting","Supportvector", "BestModel")
#outdata <- as.data.frame(outdata)
outdata<-out
outdata <- as.data.frame(outdata)
```

```
table(outdata[,7])
```

```
##
##      Boosting  RandomForest Supportvector
##            10            88             2
```

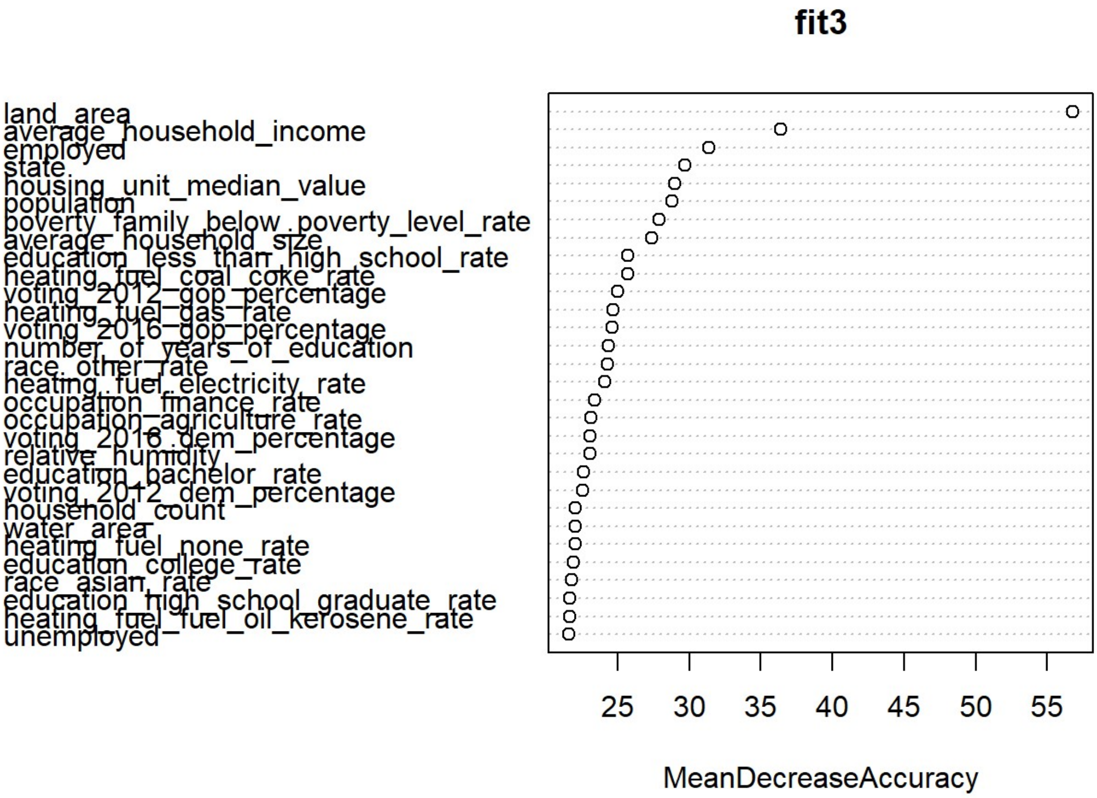
The table data shows that Random Forest model dominated the cross validation as it has the highest count among all the iteration.

```
matplot(outdata[,1:6], type = "o", pch= 1, cex= 0.5, Ity= 2,col = c("deepskyblue3",
"darkorange2","green","magenta3","red","grey"),ylab="Accuracy",xlab = "Replication
s",main="Validation Accuracy",ylim = c(0.825,0.915))
abline(h = c(mean(as.numeric(outdata[,1])), mean(as.numeric(outdata[,2])), mean(as.
numeric(outdata[,3])), mean(as.numeric(outdata[,4])), mean(as.numeric(outdata[,
5])),mean(as.numeric(outdata[,6]))),col= c("deepskyblue3", "darkorange2","green","m
agenta3","red","grey"))
legend("bottom", fill = c("deepskyblue3", "darkorange2","green","magenta3","red","g
rey"),    legend = c("Classification Tree","Logistic Regression","Random Forest","B
agging","Boosting","Support Vector Machines"), bty = "n",ncol=3,cex = 0.75)
```



The plot of “Random Forest” curve shows that it has the highest accuracy among all the other models. We can look at the variable importance plot for further analysis of the model.

```
varImpPlot(fit3,type = 1)
```



The variable importance plot suggests that the top three variables contributing to the accuracy of the random forest model are: “land_area”,“average_household_income” and “housing_unit_median_value”. We can now fit the test data in Random Forest model and check the accuracy of it.

```
predtest<-predict(fit3,type = "class",newdata = testdata)
  testtable1<-table(testdata$solar_system_count,predtest)
  testaccuracy1<-sum(diag(testtable1))/sum(testtable1)
  cat("The test accuracy is ",testaccuracy1)
```

```
## The test accuracy is  0.8967978
```

From the test accuracy value, we can see that the test data is highly fitted to the Random Forest model.

```
print(testtable)
```

```
##       TestdataPred
##        high  low
##   high 2429  262
##   low   272 2221
```

```
cat("The overall accuracy of the model is: ",testaccuracy,"\n","The Solar System co
unt high accuracy:",sum(diag(testtable[1]))/sum(testtable[1,]),"\n","The Solar Syst
em count low accuracy:",sum(diag(testtable[2]))/sum(testtable[2,]))
```

```
## The overall accuracy of the model is:  0.8969907
##  The Solar System count high accuracy: 0.9026384
##  The Solar System count low accuracy: 0.1091055
```

Conclusion

The analysis show that the supervised classification method “Random Forest” is the best model among the other classification methods. It provides the best accuracy among other models. Three most important variables that influence the accuracy of solar data are “land_area”, “average_household_income” and “housing_unit_median_value”.From the table of test data fitting, there is approximately 90% accuracy of Solar system count being high and approximately 10% accuracy of it being low.

References

-Parallel processing methods using foreach and parallel packages (https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf)

-Recode and select function from dplyr package (https://cran.r-project.org/web/packages/dplyr/dplyr.pdf)