* **Souradip Goswami**

1. **Abstract**

DeepSolar Project aimed at using high-resolution satellite images and analyzing it to identify the sizes of solar photovoltaic panels installed in each area and to generate a dataset. The area with solar panels greater than 10 are identified as “high” otherwise “low” as the response variable in the dataset. The response variable is accompanied with features that help describing the social, economic, environmental, geographical, and meteorological aspects of the tile. We preprocess the dataset and apply various supervised machine learning algorithms for training. The accuracy of the trained models is validated and the best model is selected for this dataset. The Random Forest model is found to be achieving the highest accuracy for this dataset maximum number of times and hence this model can later be used to classify unknown data obtained by DeepSolar project with a reasonable accuracy.

1. **Introduction**

Machine Learning is used to build statistical models which are trained on some data to observe its patterns and learn from it. These trained models are then used to predict outcomes for similar data. Machine Learning comprises of two methods: unsupervised learning and supervised learning. Unsupervised learning is used for finding patterns in data where no labels are available. The models try to detect similarity between observations and label the groups observed. In supervised learning, the data has already been provided with labels and this tells the models for what patterns to be looked for in the data.

DeepSolar Project aimed at using high-resolution satellite images and analyzing it to identify the sizes of solar photovoltaic panels in each area. Deep learning was used to analyze the images and build a database about solar power installation patterns along with certain features of the area. This was done to maintain records of solar power installation and for future development of renewable energy in United States. In this project we will be performing supervised machine learning on DeepSolar dataset to predict the solar power system coverage. Each observation corresponds to the number of solar power systems observed in an image tile of a certain area along with certain features of that area. The dataset comprises of 20736 observations with 80 features. In the dataset the response variable tells us about the number of solar panels observed in a given image tile. If the number of solar power systems is less than 11 then the variable is labeled as “low” and if the number is greater than 10 then it has been labeled as “high”. The response variable is accompanied with features that help describing the social, economic, environmental, geographical, and meteorological aspects of the tile. We will use the variables that help describe the number of solar power systems in a particular area and various machine learning algorithms to train several models and select the best model with high accuracy of predicting power. The best model can then be used to predict how many solar panels can be installed in a certain area based on its socio-economic, geographic and environmental features. These predictions can help in decisions regarding energy policy in future.

1. **Exploratory Data Analysis and Data Preprocessing**

On observing the structure of the dataset we can see all the variables are stored as numeric but few variables have levels of data. The variable ‘state’ is converted to a factor with numeric levels. The variables ‘voting\_2012\_dem\_win’ and ‘voting\_2016\_dem\_win’ has data entry as FALSE and TRUE. These variables are converted to factors with levels 0🡪FALSE and 1🡪TRUE. It is observed that no data observation has ‘NA’ (incomplete) data. It is also observed no duplicate data are present.

There are few derived variables having linear relation to other variables. These variables can be removed as they do not provide any new information. The variable ‘employ\_rate’ was a derived variable, i.e. employ\_rate = employment/(employed+unemployed) and so this variable was removed.

We then checked the correlation between the variables that seem to be related from the variable description and remove the variables with high Pearson coefficient.

1. The variables 35-43 (electricity prices and consumption) seemed to be having a very high correlation with the variable ‘State’ and also seemed to be having a constant value for each state. So it could be said they are factors having levels similar to the State variable. So we removed the variables 35-43 as the ‘State’ variable was enough to identify the dependency of the response variable to these variables.

3. The variables ‘average\_household\_income’ and ‘per\_capita\_income’ had a correlation of 0.925 and so we removed ‘per\_capita\_income’ variable.

4. The variable ‘total\_area’ was removed as it had a correlation of 0.999 with the variable ‘land\_area’.

5. The variable ‘housing\_unit\_count’ was removed as it had a correlation of 0.956 with the variable ‘household\_count’.

6. The variables 49, 53, 55 and 60 were removed as they had a correlation of more than 0.87 with the variable ‘earth\_temperature’. The variables 52 and 59 were also removed as they had a correlation of more than -0.93 with the variable ‘earth\_temperature’.

7. The variable ‘voting\_2012\_dem\_percentage’ had a correlation of 0.95 with ‘voting\_2016\_dem\_percentage’ and so the former was removed. Similarly the variable ‘voting\_2012\_gop\_percentage’ was removed because it had a correlation of 0.95 with ‘voting\_2016\_gop\_percentage’.

A new dataset was formed by removing the above mentioned variables and the data was scaled to normalize the skewness of the data distribution.

1. **Methodology**

The scaled dataset is split into training (50%), validation (25%) and test dataset. The training dataset is used to train the different models and validate the results using validation dataset. The best model is selected based on the accuracy calculated during validation and is used to predict outcomes of testing dataset. This entire process of training, validating and testing is iterated over 100 times. It is important to iterate the process externally so that the algorithms can minimize the error margins and in turn increase the model accuracy.

1. **Logistic Regression**

The Logistic Regression is used to predict the outcome of an observation based on the predictor variables when the outcome is binary/categorical in nature. If the outcome is categorical, i.e. discrete in nature, then linear regression model cannot be used as it predicts a continuous outcome. The linear regression model is unbounded and so Logistic Regression is used for such scenarios.

We use the R command glm() to fit the Logistic Regression model to the training dataset using the ‘solar\_system\_count’ variable as the outcome and the rest as predictor variables. The family “binomial” is used for this model training as the outcome of the data is a binary value: low or high. We select a cutoff threshold of 0.5 and predict the outcome as “low” if the probability of outcome is lower than this threshold else it is classified as “high”.

The ROC curve is plotted and the area under the curve (AUC) is calculated. Good ROC curves are identified as having high AUC values (>0.7). An ROC curve with low AUC value does not yield good discriminatory power while predicting the outcomes. The area under the curve explains the model; if it is 1 then the model is perfect. The sensitivity and specificity values are obtained to calculate the optimal threshold value of probability, which can discriminate the observation as “low” or “high”. We use this optimal cutoff threshold to classify the validation dataset observations. A confusion matrix is created and the accuracy of the prediction of validation dataset is calculated.

1. **Classification Trees**

A Classification Tree has a structure like a flowchart where each node is represented by a question about the attribute, each branch represents the answer of the question and the leaf nodes are the decisions taken after answering the questions and are represented by the class labels. The trees are built using an iterative procedure known as binary recursive partitioning where the data are divided into segments and then the segments from each branch are further divided.

We use the ‘rpart()’ command in R to fit a Classification Tree to the training dataset. The resultant tree is used to label the observations in the validation dataset. A confusion matrix created for the actual and predicted classes is used to calculate the accuracy of this method.

1. **Bagging**

Bagging is an ensemble method which is used as an improvement over the Classification Trees. Numerous random samples of data are taken with replacement from the training dataset and each subset is used to construct a Classification Tree. The trees are constructed parallelly and the outcome of each tree is aggregated to provide the final outcome. A voting is done on the outcome sets to decide on the final outcome and this reduces the variance in the prediction.

We train the model by fitting Bagging on the decision tree classifier using the R command “bagging()”. The trained model is used to classify the observations in the validation dataset and calculate the accuracy of the trained model.

1. **Random Forests**

Random Forest is a very powerful ensemble method which provides improvement over Bagging. For splitting the trees Bagging chooses a predictor variable which minimizes the error. This can result in high correlation between the predictions of each Classification Tree due to similarities in structure. This is rectified by using only a random subset of predictor variables during the splitting of the trees.

We use the R command “randomForest()” with the parameter “importance” set as TRUE. This is to ensure the predictor variables with high importance are selected while splitting as no relevant information is conveyed by unimportant variables. The importance of the variable is selected based on the minimization of the Gini index. The variable which decreases the Gini index most is selected as the root node for the construction of the tree. A graph to show the importance of each variable is plotted. The validation dataset is used to test the accuracy of the model trained using Random Forest classifier.

1. **Boosting**

Boosting is another powerful ensemble method that learns from its previous mistakes. It assigns weights to each observation while constructing a tree. For the next tree construction it increases the weights of the previous misclassified data points. This adaptive learning improves the performance of the baseline classifiers through weight updating and voting to give the final outcome.

We use the command “boosting()” in R to apply AdaBoost.M1 algorithm and calculate alpha as proposed by Breiman. The trained model is applied on validation dataset to calculate the accuracy of the model.

1. **Support Vector Machines**

Support Vector machines are generally used to classify two-label problems by separating them using a hyper-plane. A hyper-plane is the decision boundary that helps distinguish the two categories by maximizing the distances from nearest element of each category. The dimension of a hyper plane is decided by the number of features that help distinguish the two categories.

We use the command “ksvm()” in R to apply support vector machine algorithm to training data. As the data is non-linear we used the kernel “rbfdot” (Radial Basis kernel) for calculating the hyper-plane. The trained model is used to classify the observations in validation dataset and calculate the accuracy.

1. **Best Model Selection over 100 iterations**

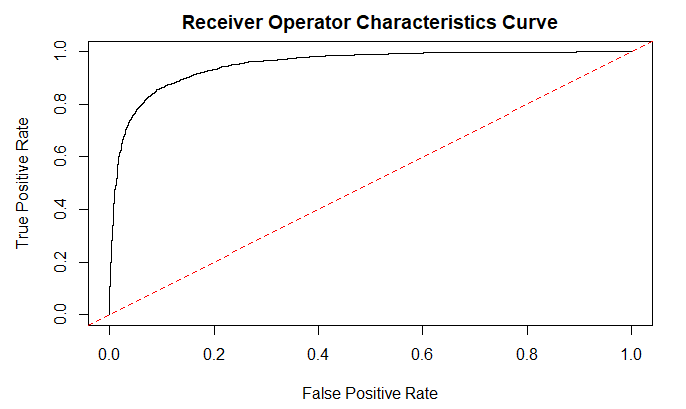
We apply the model training and best model selection for 100 times. It is important to iterate the process externally so that the algorithms can minimize the error margins and in turn increase the model accuracy. We apply parallel programming to run this process efficiently. We randomly sample data for training, testing and validation dataset and then apply all the above selected models on these datasets. The best model in each run is selected based on the accuracy in classifying the validation dataset observations. The best model selected is used to classify the test dataset observations and calculate accuracy.

1. **Selecting the best model**

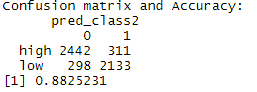
We select the model which has been selected maximum number of times among the 100 runs as the best model. This model is then applied to the test dataset sampled at the beginning and the accuracy of the prediction is calculated.

1. **Results and Discussion**
2. **Logistic Regression**

The summary of the fitted model suggests that variables 2,3,5,7,8,9,25,29,30,32,33,34,37,38,43,45,47,54,55,56,57 and 58 are found to be affecting the model significantly as the p-value is <0.05. The coefficients were not observed to be having NA values and so we can say none of the columns are linearly dependent on one another. The variables not affecting the model has not been removed from the dataset as the variables might be important for the other classifier algorithms. The ROC curve suggests that the curve is quite far away from the diagonal line, which shows the accuracy is better. We can also see that there is a gain in sensitivity (True positive rate>80%) and it trades off a false positive rate, i.e. specificity, up until about 10% of False positive rate. After 10% of FPR we don’t see much gain in Total Positive Rate while FPR increases. The “AUC” is obtained as 0.9511906 and this shows that the accuracy of the prediction is quite high.

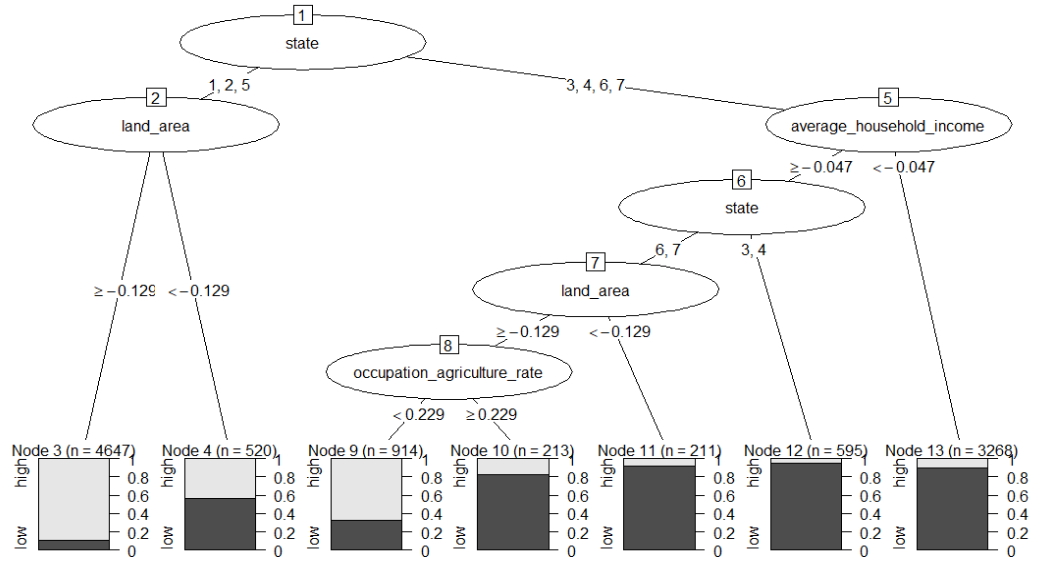


The optimal threshold value based on sensitivity and specificity values is calculated to be 0.4913. The confusion matrix created based on this cutoff threshold is obtained as below and the accuracy of the predictions done for the validation dataset is calculated to be 88.25%.

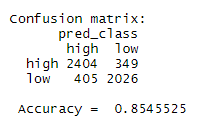


1. **Classification Trees**

On using Classification Tree to train our model, we see that the most important variable is the ‘state’ and forms the root node. We obtain the below tree based on the training:

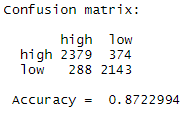


The validation dataset is used to test the tree created and the confusion matrix is created. We can see the accuracy obtained is 85.45%.

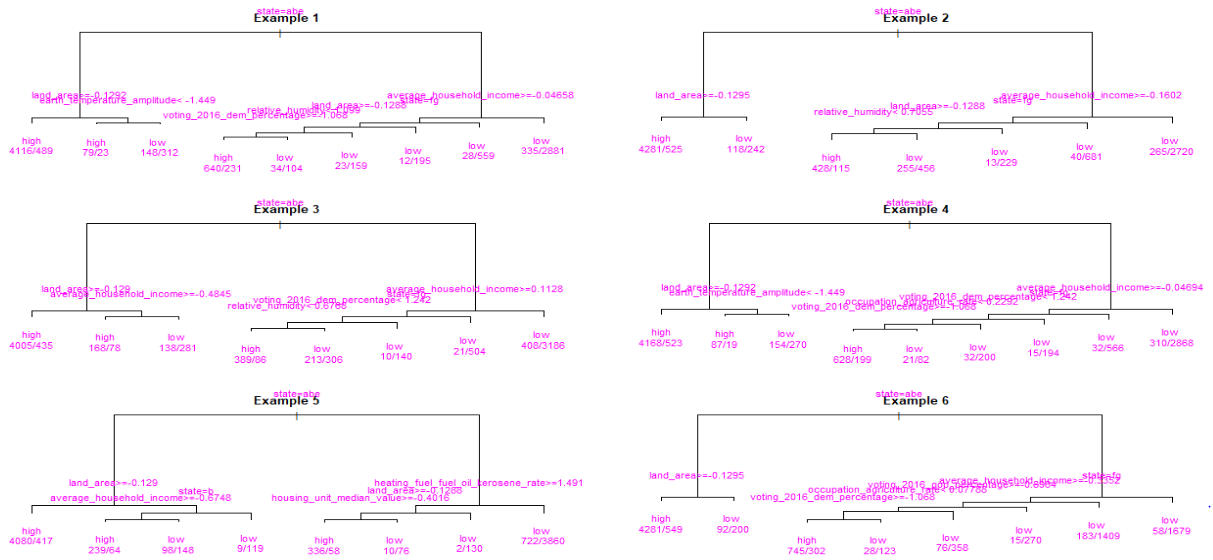


1. **Bagging**

The accuracy of Bagging classifier is found to be more than the accuracy of the Classification Tree.

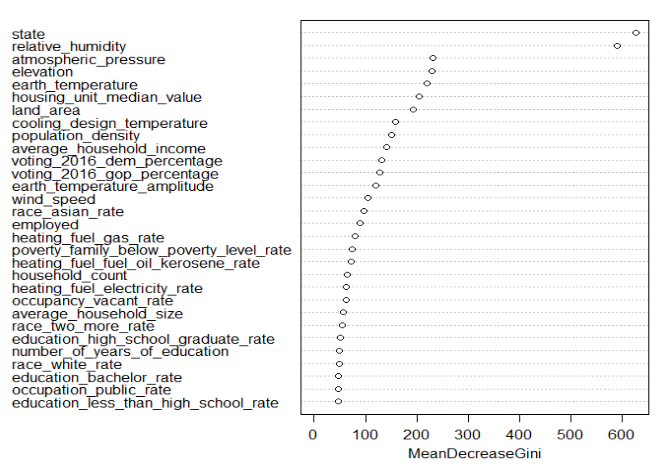


The below plot shows example of 6 trees constructed using Bagging classifier and the variable ‘state’ is selected as the root node in each of the trees:

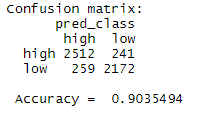


1. **Random Forest**

We can observe from the variable importance plot that the ‘state’ variable decreases the Gini index most and is also found to be having the highest importance in Classification Trees. This shows ‘state’ variable is an important factor and is used as the root node in deciding how many solar-panels are observed in an image tile.

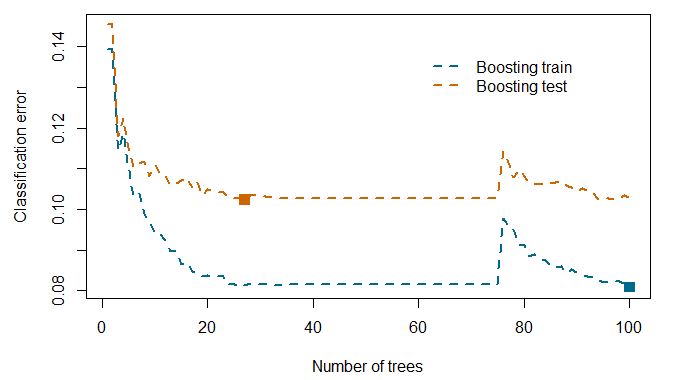


The accuracy based on the confusion matrix is calculated to be 90% which shows improvement over Classification Trees and Bagging.

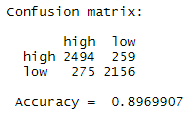


1. **Boosting**

The error path of the Boosting method fitted to training and validation dataset is plotted to check the overfitting of the model to training dataset.

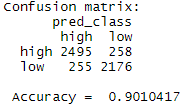


It is observed from the graph that model over-fits the training data but the error margins are not much so we proceed with the same number of trees. The accuracy of classifying the validation dataset is obtained as below:



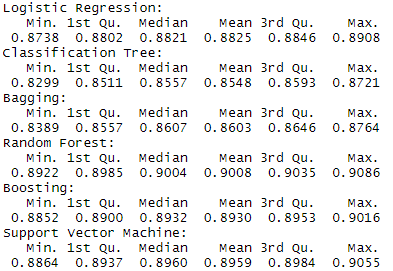
1. **Support Vector Machines**

On using non-linear kernel “rbfdot”, the accuracy from the confusion matrix for support vector machine is obtained as below.

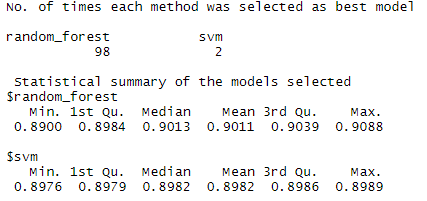


1. **Best Model Selection over 100 iterations**

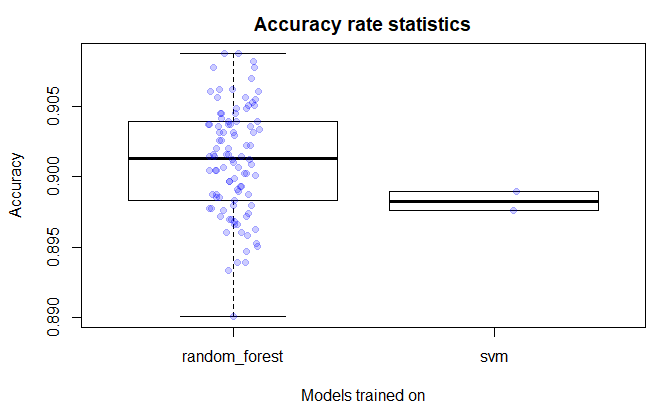
On running the models multiple times we can see the accuracy for Classification Trees is the lowest (85.48%) and the Random Forest is the highest (90.08%). The statistical summary of accuracy for validating each model is calculated. The accuracy ranged between 83-87% for Classification Trees and Bagging, 87-89% for Logistic Regression and 88-91% for Random Forest, Boosting and SVM. The summaries are obtained as below:

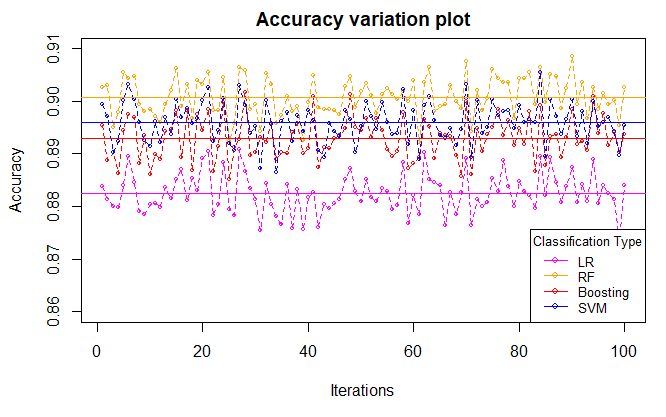
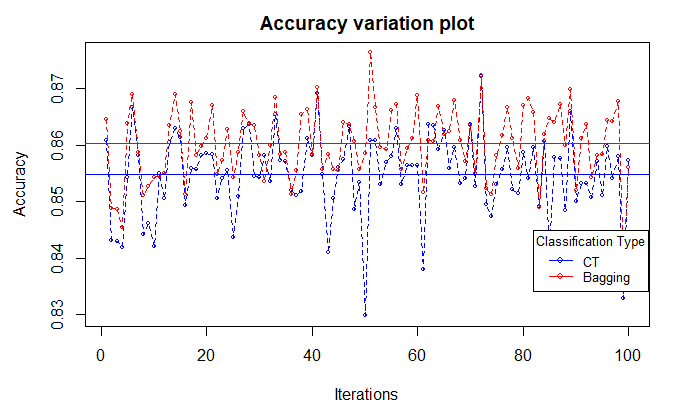


Based on the above accuracies, the Random Forest is selected as the best method 98 times and SVM twice. The statistical summary of the test accuracies is obtained as below:



The model trained on Random Forest had accuracy varying from 89-91% and for SVM it was 89.76-89.89%.

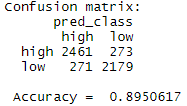




We can see there is high variation in accuracy for Classification Trees and Bagging. The models trained on SVM, Boosting and Random Forest appear to be having less variation in accuracy. The model trained on Random Forest seems to be having higher accuracy followed by SVM and Boosting.

1. **Best Model selection and Test Accuracy**

The model trained on Random Forest is selected as the best model and is applied on the test dataset for final classification. The accuracy of classifying the test observations is 89.42%



1. **Conclusion**

We applied exploratory data analysis on the DeepSolar dataset and retained the relevant variables. On applying the 6 discussed machine learning algorithms we see that Random Forest is selected 98 times as the best model. We can see from the validation accuracies, the models trained on Classification Trees have the lowest performance compared to the others. The accuracy increased slightly for Bagging which shows it is an improvement over Classification Trees. The accuracies ranged for Random Forest, Boosting and SVM from 88-91% but Boosting and SVM were more computationally expensive compared to Random Forest as they took more time to run. It is evident from the outcomes that Random Forest is a powerful ensemble method which provides improvement over Bagging. The model trained with Random Forest is able to predict the number of solar panels that can be installed in an area in the image tile based on its socio-economic, geographic and environmental conditions.

1. **References**
2. <https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/>
3. <https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac>
4. <https://www.geeksforgeeks.org/ml-bagging-classifier/>
5. <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>
6. **Appendix**

**R Code:**

#installing required packages

if(!require(ggplot2))

{

install.packages("ggplot2")

library(ggplot2)

}

if(!require(GGally))

{

install.packages("GGally")

library(GGally)

}

if(!require(corrplot))

{

install.packages("corrplot")

library(corrplot)

}

if(!require(nnet))

{

install.packages("nnet")

library(nnet)

}

if(!require(car))

{

install.packages("car")

library(car)

}

if(!require(ROCR))

{

install.packages("ROCR")

library(ROCR)

}

if(!require(rpart))

{

install.packages("rpart")

library(rpart)

}

if(!require(partykit))

{

install.packages("partykit")

library(partykit)

}

if(!require(randomForest))

{

install.packages("randomForest")

library(randomForest)

}

if(!require(adabag))

{

install.packages("adabag")

library(adabag)

}

if(!require(kernlab))

{

install.packages("kernlab")

library(kernlab)

}

##Load dataset

solar.data<-read.csv(file.choose(),header=TRUE)

##EDA and Data Preprocessing

str(solar.data[,c(1,2,76,79)])

#Checking for Null and duplicated observations

table(is.na(solar.data))

table(duplicated(solar.data))

#Convert FALSE/TRUE to numerical factor levels

solar.data$voting\_2012\_dem\_win<-toupper(solar.data$voting\_2012\_dem\_win)

solar.data$voting\_2016\_dem\_win<-toupper(solar.data$voting\_2016\_dem\_win)

solar.data$voting\_2016\_dem\_win<-ifelse(solar.data$voting\_2016\_dem\_win=="FALSE",0,1)

table(solar.data$voting\_2016\_dem\_win)

solar.data$voting\_2016\_dem\_win<-factor(solar.data$voting\_2016\_dem\_win,levels = c(0,1))

solar.data$voting\_2012\_dem\_win<-ifelse(solar.data$voting\_2012\_dem\_win=="FALSE",0,1)

solar.data$voting\_2012\_dem\_win<-factor(solar.data$voting\_2012\_dem\_win,levels = c(0,1))

table(solar.data$voting\_2012\_dem\_win)

#Check structure of converted variables

str(solar.data[,c(1,2,76,79)])

##Correlation check

cor(solar.data[,c(44,46)])

cor(solar.data[,c(49:60)])

cor(solar.data[,c(74,75,77,78)])

cor(solar.data[,c(6,10)])

cor(solar.data[,c(3,4,7,8,9)])

######Splitting Training, Validation and Test dataset

set.seed(19200368)

#Remove variables with high correlation

new.df<-solar.data[,-c(7,8,10,27,35:43,46,49,52,53,55,59,60,77,78)]

#Scale Data

new.df[,-c(1,2,56,57)]<-scale(new.df[,-c(1,2,56,57)])

#CV split

N <- nrow(new.df)

train\_samps <- sample(1:N, size = 0.50\*N)

validation\_samps<- sample( setdiff(1:N, train\_samps), size = 0.25\*N )

test\_samps <- setdiff(1:N, union(train\_samps, validation\_samps))

train<-new.df[train\_samps,]

validation<-new.df[validation\_samps,]

test<-new.df[test\_samps,]

####Logistic Regression

#Fit Multinomial Logistic Regression to training dataset

fit.lr<-glm(solar\_system\_count~.,data=train, family = binomial)

summary(fit.lr)

#Fit the model into validation dataset and classify the observations

pred\_scores<-predict(fit.lr,validation,"response")

tau=0.5

pred\_class<-ifelse(pred\_scores>tau,1,0)

#create confusion matrix for the predicted classes and the actual classes

tab<-table(validation$solar\_system\_count,pred\_class)

#Plot ROC curve to check AUC

predictedObject<-prediction(pred\_scores,validation$solar\_system\_count)

perform<-performance(predictedObject,"tpr","fpr")

plot(perform, xlab="False Positive Rate", ylab="True Positive Rate", main="Receiver Operator Characteristics Curve")

abline(0,1,col="red",lty=2)

auc<-performance(predictedObject,"auc")

auc@y.values

#Select Optimal tau based on specificity and sensitivity

sensitivity<-performance(predictedObject,"sens")

specificity<-performance(predictedObject,"spec")

tau<-sensitivity@x.values[[1]]

SS <- sensitivity@y.values[[1]] + specificity@y.values[[1]]

best <- which.max(SS)

tau[best]

#Classify based on optimal tau

pred\_class2<-ifelse(pred\_scores>tau[best],1,0)

#create confusion matrix for the predicted classes and the actual classes

tab2<-table(validation$solar\_system\_count,pred\_class2)

#calculate accuracy and misclassification error based on the confusion matrix

accuracy<-sum( diag(tab2) ) / sum(tab2)

cat("Confusion matrix and Accuracy: \n")

tab2

accuracy

####Classification Trees

#Fit Classification tree to training dataset

fit.ct <- rpart(solar\_system\_count ~ ., data = train)

#Plot the tree modeled

plot( as.party(fit.ct), cex = 0.5 )

#Get the summary of trained model

fit.ct

summary(fit.ct)

#Predict outcome of validation dataset

pred\_class<-predict(fit.ct,validation,type = "class")

#Create confusion matrix and calculate validation accuracy

tab<-table(validation$solar\_system\_count,pred\_class)

accuracy<-sum( diag(tab) ) / sum(tab)

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)

#####Bagging

#Fir Bagging to training dataset

fit.bag<-bagging(solar\_system\_count~.,data=train)

#Fit the model into validation dataset and classify the observations

pred\_class<-predict(fit.bag,validation,type="class")

#create confusion matrix for the predicted classes and the actual classes

tab<-table(validation$solar\_system\_count,pred\_class$class)

accuracy<-sum( diag(tab) ) / sum(tab)

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)

#Plot 6 trees created by bagging method

par(mfrow=c(3,2))

for (j in 1:6)

{

plot(fit.bag$trees[[j]],main=paste("Example ",j,sep=""))

text(fit.bag$trees[[j]],use.n=TRUE,xpd=TRUE,col="magenta")

}

######Random Forest

#Fit Random Forest algorithm to training dataset

fit.rf<-randomForest(solar\_system\_count~.,data=train,importance=TRUE)

#Fit the model into validation dataset and classify the observations

pred\_class<-predict(fit.rf,validation)

#create confusion matrix for the predicted classes and the actual classes

tab<-table(validation$solar\_system\_count,pred\_class)

accuracy<-sum( diag(tab) ) / sum(tab)

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)

#Plot the Gini index to check the important variables

varImpPlot(fit.rf,main="Random Forest")

####### Boosting

fit.boost<-boosting(solar\_system\_count~.,data=train,

coeflearn = "Breiman",boos = FALSE)

#Check if the model has been overfitted on training dataset

eBoostTrain <- errorevol(fit.boost, train)$error

# compute test classification error as function of number of trees

eBoostTest <- errorevol(fit.boost, validation)$error

# plot error paths

mat <- cbind(eBoostTrain, eBoostTest)

cols <- c("deepskyblue4", "darkorange3")

matplot(mat, type = "l", lty = rep(2:1, each = 2), col = cols,

lwd = 2, xlab = "Number of trees", ylab = "Classification error")

legend(x = 60, y = 0.14, cex = 1,

legend = c( "Boosting train", "Boosting test"),

lty = rep(2:1, each = 2), col = cols, lwd = 2, bty = "n")

points(apply(mat, 2, which.min), apply(mat, 2, min), col = cols,

pch = rep(c(15, 17), each = 2), cex = 1.5)

# Fit the model into validation dataset and classify the observations

pred\_class<-predict(fit.boost,validation,type="class")

#create confusion matrix for the predicted classes and the actual classes and calculate the accuracy

tab<-table(validation$solar\_system\_count,pred\_class$class)

accuracy<-sum( diag(tab) ) / sum(tab)

#Display the confusion matrix and accuracy

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)

###### Support Vector Machines

#Train the model using SVM on training dataset and Non-linear kernel

fit.svm <- ksvm(solar\_system\_count~.,data=train,kernel="rbfdot")

# Fit the model into validation dataset and classify the observations

pred\_class<-predict(fit.svm,validation)

#create confusion matrix for the predicted classes and the actual classes

tab<-table(validation$solar\_system\_count,pred\_class)

accuracy<-sum( diag(tab) ) / sum(tab)

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)

##### Running the entire process 100 times to select best model

##### Applying parallel programming to run the code efficiently on available processes

accuracy<-rep(NA,8)

no\_cores <- detectCores() - 1

registerDoParallel(no\_cores)

accuracy\_mat<-foreach (i=1:100,.combine = 'rbind',.multicombine = TRUE,.packages = c("nnet","car","ROCR","partykit","randomForest","adabag","kernlab","rpart","MASS")) %dopar% {

N <- nrow(new.df)

train\_samps2 <- sample(1:N, size = 0.50\*N)

validation\_samps2<- sample( setdiff(1:N, train\_samps2), size = 0.25\*N )

test\_samps2 <- setdiff(1:N, union(train\_samps2, validation\_samps2))

train2<-new.df[train\_samps2,]

validation2<-new.df[validation\_samps2,]

test2<-new.df[test\_samps2,]

#####Logistic####

fit.lr2<-glm(solar\_system\_count~.,data=train2, family = binomial)

pred\_scores<-predict(fit.lr2,validation2,"response")

tau=0.5

pred\_class<-ifelse(pred\_scores>tau,1,0)

predictedObject<-prediction(pred\_scores,validation2$solar\_system\_count)

sensitivity<-performance(predictedObject,"sens")

specificity<-performance(predictedObject,"spec")

tau<-sensitivity@x.values[[1]]

SS <- sensitivity@y.values[[1]] + specificity@y.values[[1]]

best <- which.max(SS)

pred\_class2<-ifelse(pred\_scores>tau[best],1,0)

tab2<-table(validation2$solar\_system\_count,pred\_class2)

accuracy[1]<-sum( diag(tab2) ) / sum(tab2)

####Decision tree

fit.ct2 <- rpart(solar\_system\_count ~ ., data = train2)

pred\_class<-predict(fit.ct2,validation2,type = "class")

tab<-table(validation2$solar\_system\_count,pred\_class)

accuracy[2]<-sum( diag(tab) ) / sum(tab)

#####Bagging

fit.bag2<-bagging(solar\_system\_count~.,data=train2)

pred\_class<-predict(fit.bag2,validation2,type="class")

tab<-table(validation2$solar\_system\_count,pred\_class$class)

accuracy[3]<-sum( diag(tab) ) / sum(tab)

####Randomforest

fit.rf2<-randomForest(solar\_system\_count~.,data=train2,importance=TRUE)

pred\_class<-predict(fit.rf2,validation2)

tab<-table(validation2$solar\_system\_count,pred\_class)

accuracy[4]<-sum( diag(tab) ) / sum(tab)

######Boosting

fit.boost2<-boosting(solar\_system\_count~.,data=train2,

coeflearn = "Breiman",boos = FALSE)

pred\_class<-predict(fit.boost2,validation2,type="class")

tab<-table(validation2$solar\_system\_count,pred\_class$class)

accuracy[5]<-sum( diag(tab) ) / sum(tab)

#####SVM

fit.svm2 <- ksvm(solar\_system\_count~.,data=train2,kernel="rbfdot")

pred\_class<-predict(fit.svm2,validation2)

tab<-table(validation2$solar\_system\_count,pred\_class)

accuracy[6]<-sum( diag(tab) ) / sum(tab)

acc<-c(logistic=accuracy[1],decision\_tree=accuracy[2],

bagging\_method=accuracy[3],random\_forest=accuracy[4],

boosting\_method=accuracy[5],svm=accuracy[6])

best<-names( which.max(acc) )

switch(best,

logistic = {

pred\_test\_scores<-predict(fit.lr2,test2,"response")

tau=0.5

pred\_test\_class<-ifelse(pred\_test\_scores>tau,1,0)

predictedObject<-prediction(pred\_test\_scores,test2$solar\_system\_count)

sensitivity<-performance(predictedObject,"sens")

specificity<-performance(predictedObject,"spec")

tau<-sensitivity@x.values[[1]]

SS <- sensitivity@y.values[[1]] + specificity@y.values[[1]]

best <- which.max(SS)

pred\_class2<-ifelse(pred\_test\_scores>tau[best],1,0)

tab2<-table(test2$solar\_system\_count,pred\_class2)

accBest<-sum( diag(tab2) ) / sum(tab2)

},

decision\_tree = {

pred\_ct<-predict(fit.ct2,test2,type = "class")

tab<-table(test2$solar\_system\_count,pred\_ct)

accBest<-sum( diag(tab) ) / sum(tab)

},

random\_forest = {

pred\_rf<-predict(fit.rf2,test2)

tab<-table(test2$solar\_system\_count,pred\_rf)

accBest<-sum( diag(tab) ) / sum(tab)

},

bagging\_method = {

pred\_bag<-predict(fit.bag2,test2,type="class")

tab<-table(test2$solar\_system\_count,pred\_bag$class)

accBest<-sum( diag(tab) ) / sum(tab)

},

boosting\_method = {

pred\_boost<-predict(fit.boost2,test2,type="class")

tab<-table(test2$solar\_system\_count,pred\_boost$class)

accBest<-sum( diag(tab) ) / sum(tab)

},

svm = {

pred\_svm<-predict(fit.svm2,test2)

tab<-table(test2$solar\_system\_count,pred\_svm)

accBest<-sum( diag(tab) ) / sum(tab)

}

)

accuracy[7] <- best

accuracy[8] <- accBest

return(accuracy)

}

##Statistical Summary of validation and test accuracies

colnames(accuracy\_mat) <- c("val\_logistic", "val\_decisiontree","val\_bagging",

"val\_random\_forest","val\_boosting","val\_svm",

"best","test")

accuracy\_mat<-as.data.frame(accuracy\_mat)

accuracy\_mat[,1]<-as.numeric(as.character(accuracy\_mat[,1]))

accuracy\_mat[,2]<-as.numeric(as.character(accuracy\_mat[,2]))

accuracy\_mat[,3]<-as.numeric(as.character(accuracy\_mat[,3]))

accuracy\_mat[,4]<-as.numeric(as.character(accuracy\_mat[,4]))

accuracy\_mat[,5]<-as.numeric(as.character(accuracy\_mat[,5]))

accuracy\_mat[,6]<-as.numeric(as.character(accuracy\_mat[,6]))

accuracy\_mat[,8]<-as.numeric(as.character(accuracy\_mat[,8]))

str(accuracy\_mat)

#Statistical summary of both models

cat("Logistic Regression: \n")

summary(accuracy\_mat[,1])

cat("Classification Tree: \n")

summary(accuracy\_mat[,2])

cat("Bagging:\n")

summary(accuracy\_mat[,3])

cat("Random Forest: \n")

summary(accuracy\_mat[,4])

cat("Boosting:\n")

summary(accuracy\_mat[,5])

cat("Support Vector Machine: \n")

summary(accuracy\_mat[,6])

#Plotting the accuracy rate of the best model selected in each iteration

table(accuracy\_mat[,7])

tapply(accuracy\_mat[,8], accuracy\_mat[,7], summary)

boxplot(accuracy\_mat[,8] ~ accuracy\_mat[,7], xlab="Random Forest", ylab="Accuracy",main="Accuracy rate statistics")

stripchart(accuracy\_mat[,8] ~ accuracy\_mat[,7], add = TRUE, vertical = TRUE,

method = "jitter", pch = 19, col = adjustcolor("blue", 0.2))

#Plotting the variation in accuracy

meanAcc<-colMeans(accuracy\_mat[,1:6])

matplot(accuracy\_mat[,c(2,3)],type="o",cex=0.5,lty=2,lwd=1,pch=1,col=c("blue","red"),

ylab="Accuracy",xlab="Iterations",main="Accuracy variation plot")

abline(h=meanAcc[c(2,3)],col=c("blue","red"))

legend("right",col=c("blue","red"),lty=1,pch=1,cex=0.8,title ="Classification Type",legend =c("CT","Bagging"))

matplot(accuracy\_mat[,c(1,4,5,6)],type="o",cex=0.5,lty=2,lwd=1,pch=1,col=c("magenta","orange","brown","darkgreen"),

ylab="Accuracy",xlab="Iterations",main="Accuracy variation plot")

abline(h=meanAcc[c(1,4,5,6)],col=c("magenta","orange","brown","darkgreen"))

legend("right",col=c("magenta","orange","brown","darkgreen"),lty=1,pch=1,cex=0.8,title ="Classification Type",legend =c("LR","RF","Boosting","SVM"))

######### Applying the best model selected on the test dataset

#Fit the model into validation dataset and classify the observations

pred\_class<-predict(fit.rf,test[,-1])

#create confusion matrix for the predicted classes and the actual classes

tab<-table(test$solar\_system\_count,pred\_class)

#Calculate accuracy of test dataset classification

accuracy<-sum( diag(tab) ) / sum(tab)

cat("Confusion matrix: \n")

tab

cat("\n Accuracy = ", accuracy)