AkshayIthape_SMLProject

Akshay Ithape 19200976

25/04/2020

About the Database/Deep Solar Project

Deep Solar is a deep learning framework that analyses satellite imagery to identify the GPS locations and sizes of solar photovoltaic (PV) panels. Leveraging its high accuracy and scalability, Deep Solar constructed a comprehensive high-fidelity solar deployment database for the contiguous U.S. We demonstrated its value by discovering that residential solar deployment density peaks at a population density of 1000 capita/mile2, increases with annual household income asymptoting at ~$150K, and has an inverse correlation with the Gini index representing income inequality.

We uncovered a solar radiation threshold (4.5 kWh/m2/day) above which the solar deployment is "triggered". Furthermore, we built an accurate machine learning-based predictive model to estimate the solar deployment density at the census-tract level. We offer Deep Solar as a publicly-available database for researchers, utilities, solar developers and policymakers to further uncover solar deployment patterns, build comprehensive economic and behavioural models, and ultimately support the adoption and management of solar electricity. The work has been accepted and published by Joule in December, 2018.

Abstract:

- Solar photovoltaics adoption is rapidly growing worldwide due to its reducing costs and environmental benefits.
- The data is a subset of the Deep Solar database, a solar installation database for the US, built by extracting information from satellite images.
- 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.
- Our aim is to find the best machine learning model which neatly classifies the solar power system coverage.
- We discover that this model is the "random forest classification" model with classification accuracy of approximately 91%.

Introduction:

- Solar photovoltaics (PV) adoption is rapidly growing worldwide due to its reducing costs and environmental benefits.
- With deep penetration of solar energy resources, the electric grid in the U.S. is also undergoing a transformation towards a cleaner energy network.

- The data is a subset of the Deep Solar database, a solar installation database for the US, built by extracting information from satellite images.
- Photovoltaic panel installations are identified from over one billion image tiles covering all urban areas as well as locations in the US by means of an advanced machine learning framework.
- Each image tile records the amount of solar panel systems (in terms of panel surface and number of solar panels) and is complemented with features describing social, economic, environmental, geographical, and meteorological aspects.
- As such, the database can be employed to relate key environmental, weather and socioeconomic factors with the adoption of solar photovoltaics energy production.
- 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.
- The variable 'solar_system_count' is a binary variable indicating the coverage of solar power systems in a given tile. * * The variable takes outcome "low" if the tile has a low number of solar power systems (less than or equal to 10), while it takes outcome "high" if the tile has a large number of solar power systems (more than 10).
- We need to find the best supervised classification method in order to predict the solar power system coverage of the tile with the highest accuracy possible given the range of predictor variables.
- This model will prove very useful in the future installation of the solar panels as it would help in predicting the solar power system coverage of the tile yet to be installed well in advance with the help of the given predictor variable values which will be accurate enough and this will help in determining various factors before installation itself.

About the Database/Deep Solar Project

- DeepSolar is a deep learning framework that analyzes satellite imagery to identify the GPS locations and sizes of solar photovoltaic (PV) panels. Leveraging its high accuracy and scalability, DeepSolar constructed a comprehensive high-fidelity solar deployment database for the contiguous U.S.
- We demonstrated its value by discovering that residential solar deployment density peaks at a population density of 1000 capita/mile2, increases with annual household income asymptoting at ~$150K, and has an inverse correlation with the Gini index representing income inequality.
- We uncovered a solar radiation threshold (4.5 kWh/m2/day) above which the solar deployment is "triggered".
- Furthermore, we built an accurate machine learning-based predictive model to estimate the solar deployment density at the census-tract level.
- We offer DeepSolar as a publicly-available database for researchers, utilities, solar developers and policymakers to further uncover solar deployment patterns, build

comprehensive economic and behavioral models, and ultimately support the adoption and management of solar electricity.

- The work has been accepted and published by Joule in December, 2018.

Methods:

- We will be performing our analysis with the help R software and the code for the analysis is also present with relevant descriptions.
- We begin with loading the required packages in R which contain the functions to be used for implementing the supervised classification methods.
- For our analysis we are going to examine 4 classification techniques viz.

1. **Logistic regression**
2. **Random Forest Classification**
3. **Support vector machines**
4. **Bagging**

- Out of these 4 techniques, we will choose the one with the highest accuracy over 100 replications.

```
# required libraries

library(nnet)
library(randomForest)

library(kernlab)
library(adabag)

solar=read.csv("D:/SEM2/Statistical MAchine Learning/Project/data_project_deepsolar.csv",header = T)

#data preprocessing

solar1=solar[,-c(2,76,79)] # removing the categorical variables
```

- But before we start with this analysis, we have a look at our data by loading it in.
- We observe that there 20736 observations of 81 variables.
- Of these 81 variables 1 variable will be the target variable while 80 are the features.
- Of these 80 features 3 are categorical variables which are the state, and the two variables station of DEM won the election or not in 2016 and 2012.
- In the first step of preprocessing, we get rid of these 3 categorical features as we prefer performing our analysis based on numerical variables only as it will be much easier to do so.

```
# removing highly correlated features
correlation=cor(solar1[,-1])
correlation[upper.tri(correlation,diag = T)]=0
solar1 = solar1[,!apply(correlation,2,function(x) any(abs(x) > 0.88))]
dim(solar1) # dimensions of the reduced data

## [1] 20736    58
```

```r
solar1=scale(solar1)# scaling the data

# final data for analysis
solar1=data.frame(solar[,1],solar1)
colnames(solar1)[1]="solar_system_count"
```

- In the next step of preprocessing we further try to get rid of some more variables in order to reduce complexity.
- For this purpose, we calculate the correlation between all the 77 remaining features.
- After that we examine the pairs of features having a very high correlation and this could mean that either the features are conveying more or less the same kind of information or that one feature could be derived from the other feature.
- Hence, we drop one feature from such pairs.
- In our analysis we have dropped the features who had a correlation coefficient of more than 0.88 with any other feature. * In this process, we drop 19 features which include "average_household_income", "gini_index", "water_area", "education_college_rate","heating_fuel_none_rate","electricity_price_residential","electricity_price_transportation", "electricity_price_overall", "electricity_consume_commercial", "electricity_consume_total","housing_unit_median_value", "elevation", "earth_temperature_amplitude", "frost_days", "wind_speed", "occupancy_vacant_rate", "voting_2016_dem_percentage", "voting_2016_gop_percentage","diversity".
- The new dimensions of our features become 20736*58

- After this we scale our features so that the mean of all of them is 0 and the standard deviation is 1.
- Then we combine our 58 features with the target variable and this is our final data.
- Along with this we also check the proportion of the two categories of our target and observe that it is 53% for "high" and 47% for "low".
- This proportion is more or else balanced.

```r
prop.table(table(solar1$solar_system_count))
```

```
##
##      high       low
## 0.5256559 0.4743441
```

```r
# initial split and removing the test set
set.seed(19200976)
keep=sample(1:nrow(solar1),size=0.75*nrow(solar1))
test=setdiff(1:nrow(solar1),keep)
dat=solar1[keep,]
dat_test=solar1[test,]# test set to be tested on later
```

- In order to find out the best classifier out of the 4 we are going to examine we will split the labeled data into three non-overlapping data sets.
- Data points whose labels are used in the classifier fitting procedure, that is are used to learn the parameters of the classifier are contained in the training set.
- These data points are employed to train the classifier.
- Data points used to test each of the classifiers to see which is best are contained in the validation set.
- Each classifier in turn is tested on these data.
- Data points not used as labeled cases in the fitting procedure or in the validation procedure are contained in the test set.
- The test data set is used to estimate the generalization performance of the best classifier.
- Hence, these data points are used to test the best model.
- To do this we split the original data in the ratio of 75:25 where 25% will be the test data and this will be kept aside.

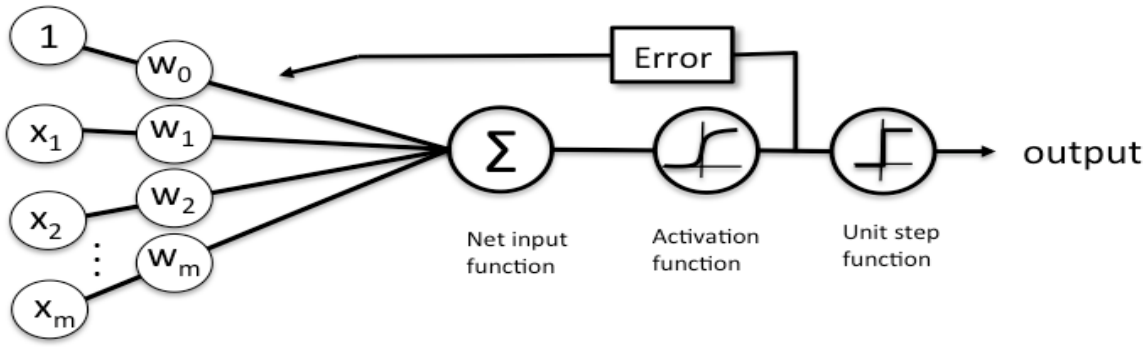
```
R=100
best=rep(NA,R)
acc=matrix(NA,R,4)
N=nrow(dat)

for(r in 1:R)
  {
  train=sample(1:N,size=0.7*N) # training set
  val=setdiff(1:N,train)# validation set

  # logistics regression
  fitlog=multinom(solar_system_count~.,data=dat,subset=train,trace=F)
  predlog = predict(fitlog,newdata=dat[val,])
  tab1=table(predlog,dat$solar_system_count[val])
  acc[r,1]=sum(diag(tab1))/sum(tab1)

  # random forest
  fitran=randomForest(solar_system_count~.,data=dat,subset=train)
  predran=predict(fitran,newdata=dat[val,])
  tab2=table(predran,dat$solar_system_count[val])
  acc[r,2]=sum(diag(tab2))/sum(tab2)

  # support vector machine
  fitsvm=ksvm(solar_system_count~.,data=dat[train,])
  predsvm=predict(fitsvm,newdata=dat[val,])
  tab3=table(predsvm,dat$solar_system_count[val])
  acc[r,3]=sum(diag(tab3))/sum(tab3)

  # bagging
  fitbag=bagging(solar_system_count~.,data=dat[train,])
  predbag=predict(fitbag,newdata=dat[val,])
  tab4=predbag$confusion
```

```
  acc[r,4]=sum(diag(tab4))/sum(tab4)


  # storing the best model in each replication
  i=which.max(acc[r,])
  if(i==1)
    best[r]="logistic"
  if(i==2)
    best[r]="random forest"
  if(i==3)
    best[r]="support vectore machine"
  if(i==4)
    best[r]="bagging"
  }
```

- After setting the test data aside we resort to the method of cross validation using hold out sampling.
- Cross validation is a class of resampling methods that estimate the error by holding out a subset of the training observations from the fitting process.
- Then the model is applied to those held out observations in order to evaluate the predictive performance.
- Here we can see that 75% data excluding the test set is further split randomly in the ratio of 70% and 30%.
- This 70% will constitute to the training set while the 30% will be our validation set.
- We then fit all the models to our training data and evaluate each model's performance by calculating the accuracy of classification in comparison to our original classification in the validation set.
- The model with the highest accuracy is recorded.
- The process is replicated a number of times to account for the sampling variability.
- We will now see the description of all the 4 methods used
-

Logistic regression:

The main task of logistic regression includes modelling a binary/categorical target variable given a set of input variables. The focus is in modelling the probability of an outcome of the target variable given the set of covariates.



Schematic of a logistic regression classifier.

Random Forest classification:

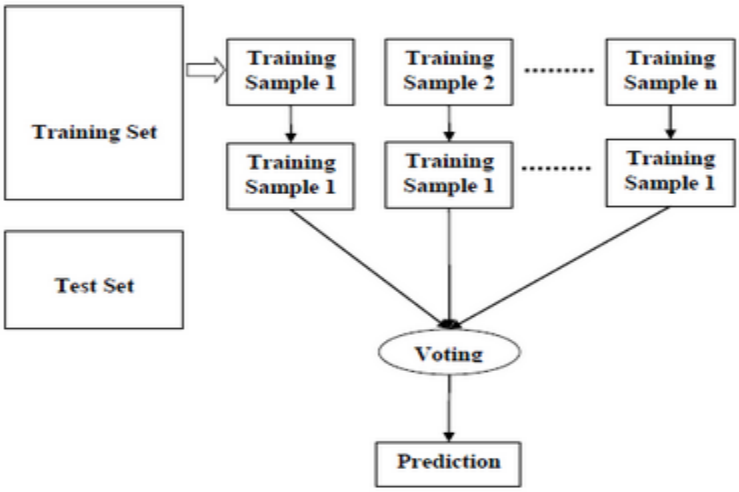
Random forests are a very powerful supervised classification method. The method uses classification trees and bootstrapping extensively. In fact, a "random forest" is simply a random collection of classification trees estimated on random subsets of the data. Random forests provide an improvement over bagging by means of a small tweak that reduces the dependence among the trees. The main idea is to use only a random subset of the predictor variables at each split of the classification tree fitting step. The framework allows also to look at variable importance in terms of classification accuracy

Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps –

- **Step 1** – First, start with the selection of random samples from a given dataset.
- **Step 2** – Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** – In this step, voting will be performed for every predicted result.
- **Step 4** – At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working –



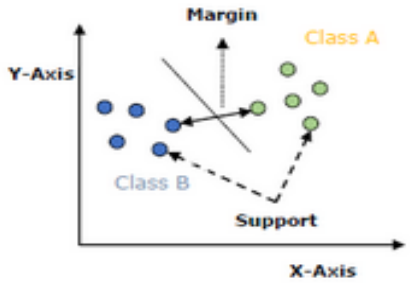
Support vector machines:

Support vector machines (SVM) are another very popular method for classification. In their most basic form, support vector machines are another type of "linear" classifier. So, they are very similar to logistic regression analysis. When used in conjunction with kernels, support vector machines can account for non-linear structure. But the same is true of logistic regression and many other linear classifiers. Suppose that we have two well separated classes in our data.

We can easily draw a line between the data points corresponding to the two classes. In fact, there is a potentially infinite number of lines which can be used to separate the data points from the two classes. For any separating line, we can look at how far it is to the closest point in perpendicular distance from each class. This distance is called the margin.

Working of SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).



The followings are important concepts in SVM –

- **Support Vectors** – Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** – As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** – It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps –

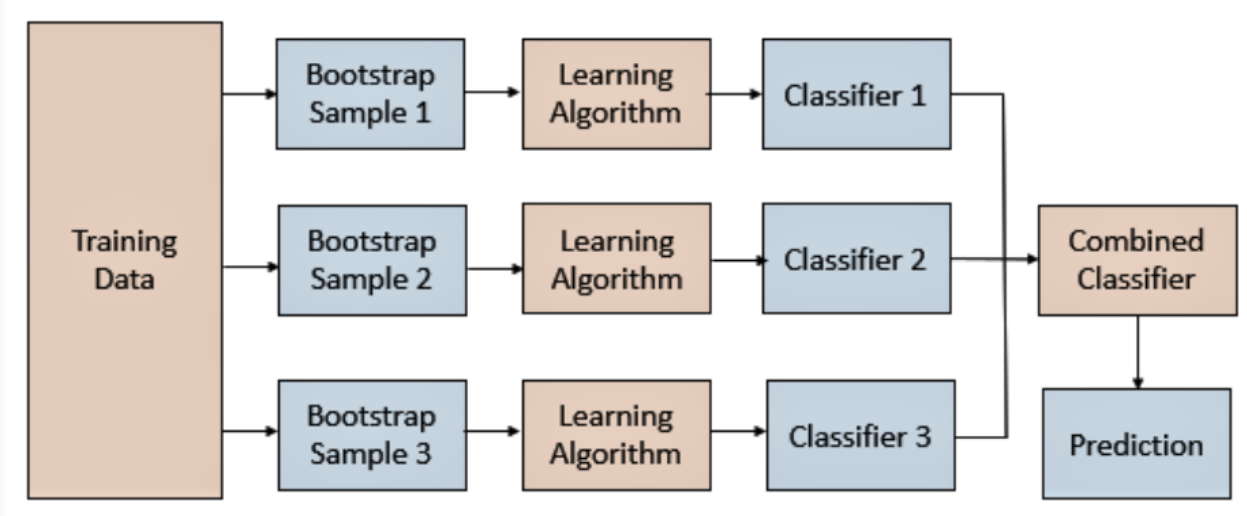
- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

The margin varies from line to line, and some lines have greater margins than others. Support vector machines try to find the separating line which has maximum margin for separating the classes. This is called the maximum separating hyper-plane. The lines that define the margin pass through data points from each class. These lines are called the support vectors. The problem of maximizing the margin can be expressed in terms of inner

products, so the option of using kernels is facilitated. This greatly increases the power of SVMs as a classification methodology.

Bagging:

Bagging is a short name for "bootstrap aggregating". It is a general idea that can be used with any learning method and it is often used for supervised classification. We will look at using bagging in conjunction with classification trees. In general, classification trees suffer from high variance. This means that if we fit a classification tree to different random splits of the data we could obtain quite different results.



Bagging is a general-purpose procedure for reducing the variance of a statistical learning method. The main idea is that averaging multiple predictions reduces variance, thus increasing accuracy. Hence, a natural way to increase the prediction accuracy of a learning method is to take many training sets from the population, fit a separate prediction model on each training set, and average the resulting predictions

- After we're done fitting all these 4 models on the training data, we predict the coverage of each tile in the validation set and compare the same with the observed coverage of the tile. For this, we use the method of cross-tabulation.
- The cross tabulation gives the following quantities.

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

TN - True negatives Number of 0 correctly classified as 0
TP -True positives Number of 1correctly classified as 1.
FN -False negatives Number of 1 wrongly classified as 0.

FP -False positives Number of 0 wrongly classified as 1.

We then calculate the classification accuracy for each model as follows:

Classification accuracy = (TP + TN) / (TP + FP + TN + FN)

- After we finish calculating the accuracies for all the models, we compare them and note the model having the highest accuracy.
- All this is repeated over 100 replications and we store the 100 accuracies for all the 4 models in a matrix and also the best model over the 100 replications.
- **This process which is repeated 100 times takes a long time to end.**

```
mean_acc=apply(acc,2,mean)# mean accuracy of each model
sd_acc=apply(acc,2,sd)# standard deviation in each model's accuracy

# plot of the accuracies at each replication
matplot(acc, type = "l", lty = c(2,3,4,5), col = c("black", "red","blue","mag
enta"), xlab = "Replications", ylab = "Accuracy",ylim=c(0.85,0.92))
abline(h = mean_acc, col = c("black", "red","blue","magenta"))
legend("topleft", fill = c("black", "red","blue","magenta"), legend = c("logi
stic", "ran_ forest","svm","bagging"), bty = "n")
```

```
# proportion of each model being chosen as best
prop.table(table(best))

## best
## random forest
##             1
```

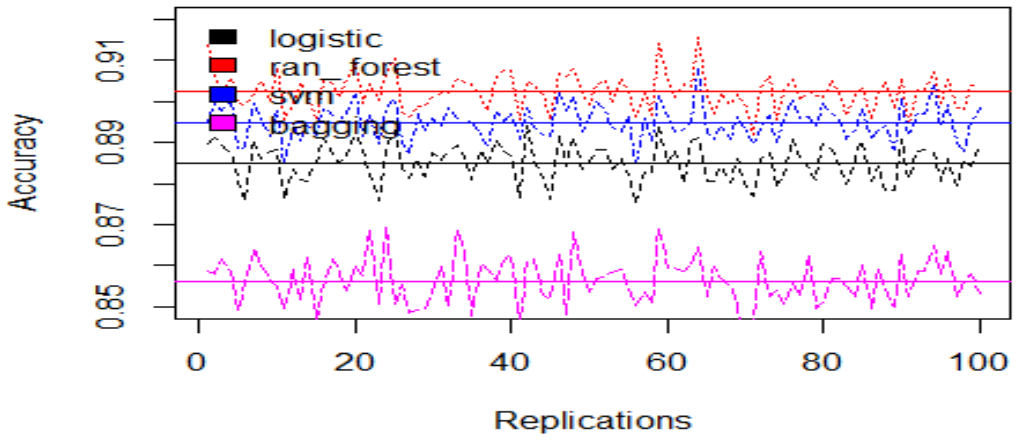
- At the end of this process we have two things in hand, the accuracy matrix and the vector in which we recorded which is the best model over each repetition.
- After that we calculate the mean accuracy for each model and standard deviation of these accuracies as well.
- Then we plot the accuracies for each repetition.
- We also see the proportion of times each model is chosen to be the best model with the highest accuracy.

```r
# testing the generalization performance of the best model
pred = predict(fitran, newdata=dat_test)
tab=table(pred,dat_test$solar_system_count)
tab

##
## pred   high  low
##   high 2518  256
##   low   239 2171

sum(diag(tab))/sum(tab)

## [1] 0.9045139
```

- After we have found the best model after all this process, we use this best model to predict the coverage of the tiles in the test set in order to find the generalization performance of the model.
- To find the generalization performance we again calculate the classification accuracy using the cross-tabulation method thus conclude our analysis.

Results and discussion:

- So far, we have seen the methodology that we have used in the entire analysis process.
- Now we will discuss and interpret all the results that we have obtained from this analysis.
- In the first part which was of preprocessing, we have already seen that how we have reduced the number of predictor variables by removing the highly correlated ones and thus getting rid of 19 numeric variables and 3 categorical variables.
- Later we also saw that after scaling, the mean and standard deviation of all the variables become 0 and 1 respectively.
- Now let's move on to the fitting and assessment of the 4 models over 100 replications.
- We'll look at the graph of accuracies.



- This graph clearly shows that the accuracy is lowest for bagging, a bit higher for the logistic regression, a bit higher for support vector machines than logistic regression and the highest for the random forest.
- Also, there is large variation in the accuracies of all the methods over 100 replications.
- The 4 horizontal lines represent the mean accuracies if the 4 models.
- Thus, we can say that the mean accuracy for bagging is approximately 85.5%, the mean accuracy for logistic regression is approximately 88.2%, the mean accuracy for support vector machines is approximately 89.1% and the mean accuracy for random forest is 89.9% which is the highest.
- Hence, the main observation from the graph is the fact that random forest visually has the highest accuracy over all the 100 repetitions.
- After the graph, we look at the proportion of times each model was chosen as the best model over the 100 repetitions and we find that random forest was chosen as the best model 100% of the times which confirms our observation from the graph visual.
- Thus, we have found the best model out of the 4 models which is the random forest classification model.
- We now check the generalization performance of the random forest model on the test set and calculate the classification accuracy of the same. We get the following cross-tabulation between the predicted and actual classification

ACTUAL

	High	Low
PREDICTED High	2518	256
PREDICTED Low	239	2171

- From this table, we can see that the total number of observations in the test set are 5184.

- From these 5184 observations, 2518 observations actually classified as “high” are classified as “high” by our model and 2171 observations actually classified as “low” are classified as “low”.
- However, 256 observations actually classified as “low” are wrongly classified as “high” by our model while 239 observations actually classified as “high” are wrongly classified as “low” by our model.
- Thus, using our accuracy formula, we get the accuracy of this classification to be approximately 91%.
- Hence, we can say that our model has a very neat and good generalized performance.

Conclusion:

- Thus, in our entire analysis we have seen that in what ways we can preprocess our large dataset and then we have seen in detail the hold-out sample cross validation techniques in order to compare the classification performance of the 4 models.
- Also, we have seen how we can implement logistic regression, random forest, support vector machines and bagging models for classification using R software.
- During the analysis we also concluded the random forest classification model is the best model for classification with generalized classification accuracy of approximately 91%.
- This tells us that if we want to classify whether an unknown tile has low coverage or a high coverage given the set of predictor variables, the random forest model has 91% chance of correctly classifying that tile.

References: Following are the references for the material used for this project:

(1) Lecture materials for the course STAT-30270 Statistical machine learning

(2) https://stackoverflow.com/questions/18275639/remove-highly-correlated-variables

(3) https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm (Images and theory for the project)