PREDICTION OF SOLAR POWER SYSTEM COVERAGE

BY

Akanseoluwa Adegoke
Master's Student, Data and Computational Science
University College Dublin.

ABSTRACT

This research explores a solar installation database of the United States. The Image tiles in the dataset shows the number of solar panel systems in all urban locations and it is complemented with features describing social, economic, environmental, geographical and meteorological aspects. We predict whether the deployment of solar systems is high or low in an area. We employ four supervised classification models in our prediction. These are Logistic Regression, Random Forest, Classification Trees and Support Vector Machines. Training and validation are performed on these models using the dataset and the best model is chosen. The performance of all these four models are compared to discover the best predictor model.

INTRODUCTION

This dataset is a subset of the DeepSolar database which is a solar installation database for the United States, 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 the location in the United States employing an advanced machine learning framework. Each image tile records the amount of solar panel systems (in terms of panel surface and the number of solar panels) and is complemented with features describing social, economic, environmental, geographical, and meteorological aspects. The database can be employed to relate key environmental, weather and socioeconomic factors with the adoption of solar photovoltaics energy production. The dataset consists of 20736 observations (rows) and 81 columns. This means that there are 20736 areas detected with solar power systems and 81 variables. All the variables are numerical in form with four exceptions. The non-numerical variables are solar_system_count, the target variable, and it takes two values which are high (area with more than 10 solar systems) and low (area with 10 or lower solar systems). The second variable is state which indicates the state of the US specifically. The third variable is voting 2012_dem_winTrue, which contains information whether the Democrats won the election or not in 2012. The fourth variable is voting_2016_dem_winTrue which contains information whether the democrats won or lost the election in 2016. We wat to predict solar system solar coverage in an area and this will be able to assist the government in planning urban development across the US. Predicting the number of solar systems in an area is also important because researchers discovered that solar power coverage increases with annual house income and it is inversely correlated with the Gini Index. The Gini Index is useful in studying the impacts of inequality and in environmental management.

DATA PRE-PROCESSING

The dataset is loaded and we check the structure of the solar dataset to check for categorical variables. We remove the categorical variables and call the new dataset solar_num. There are 4 categorical variables in the dataset and we remove them. After removing them, the number of the columns in the dataset are 77. We check for multi-collinearity in the dataset because this can affect the accuracy of the supervised learning models. Therefore, we remove variables with correlation greater than 0.9. The dataset is also scaled to standardise the range of the features in the dataset.

METHODS

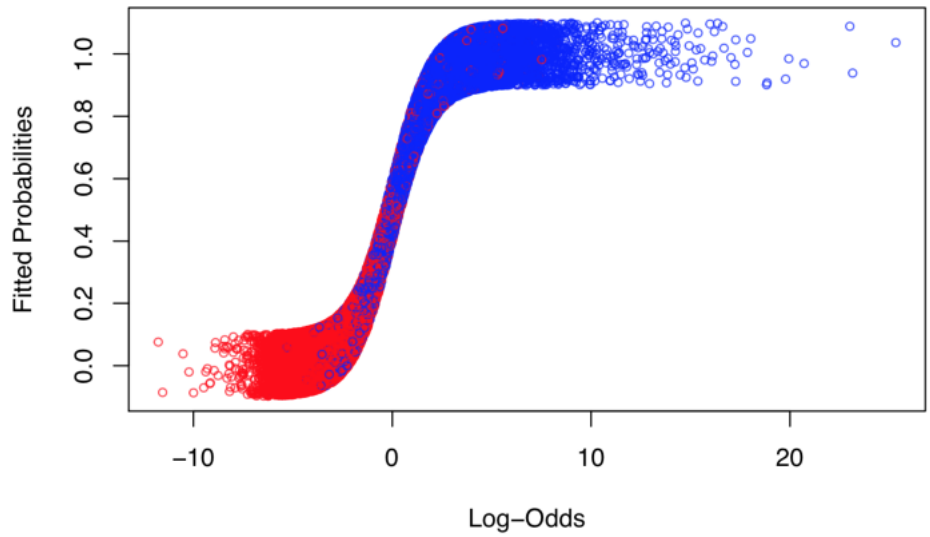
Libraries Used
Library Kernlab is used for Support Vector Machines
Library nnet is used for Multinomial Logistic Regression.
Library randomForest is used for Random Forest Classification.
Library rpart is used for Classification trees.

Logistic Regression

After using the optimal threshold tau to find the value that maximizes the sum of sensitivity and specificity. After plotting the sum of sensitivity and specificity against various values of tau, the optimal value for tau is 0.48809. With the new tau, 9659 observations were correctly classified as high and 8697 observations were correctly classified as low. 1241 were wrongly classified as high and 1139 observations were wrongly classified as low. The classification accuracy with the new tau is 88.52 percent which is slightly higher than the former tau of 0.5. The ROC Curve tells us the area under the curve which will use to measure the performance of our model too. The area under the curve is 0.9517. This number is very high which indicates that our model is performing well.

```
plot(lg, jitter(phat, amount = 0.1), pch = symb[SolarFull$slcou],
     col = adjustcolor(cols[slcou],0.7), cex =0.7, xlab="Log-Odds",
     ylab="Fitted Probabilities")
```

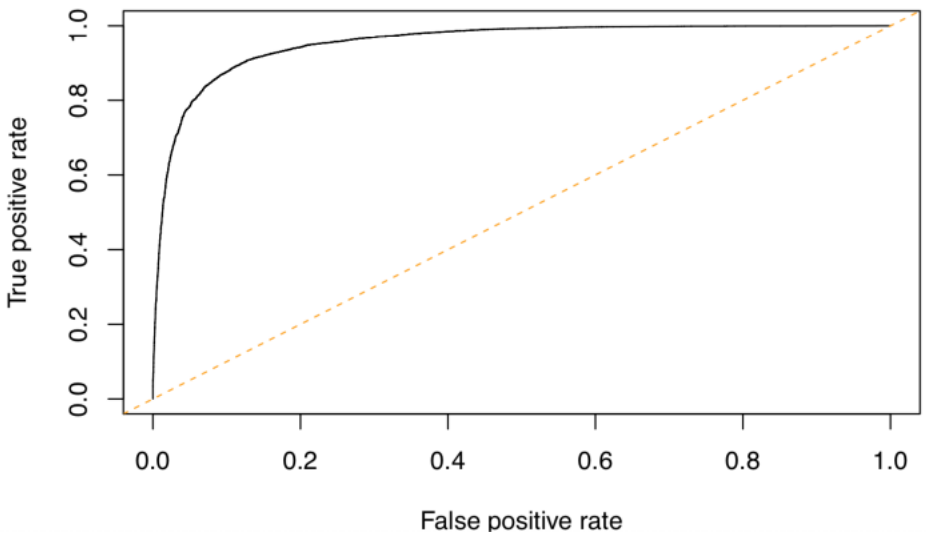
FITTED PROBABILITIES VS LOG-ODDS



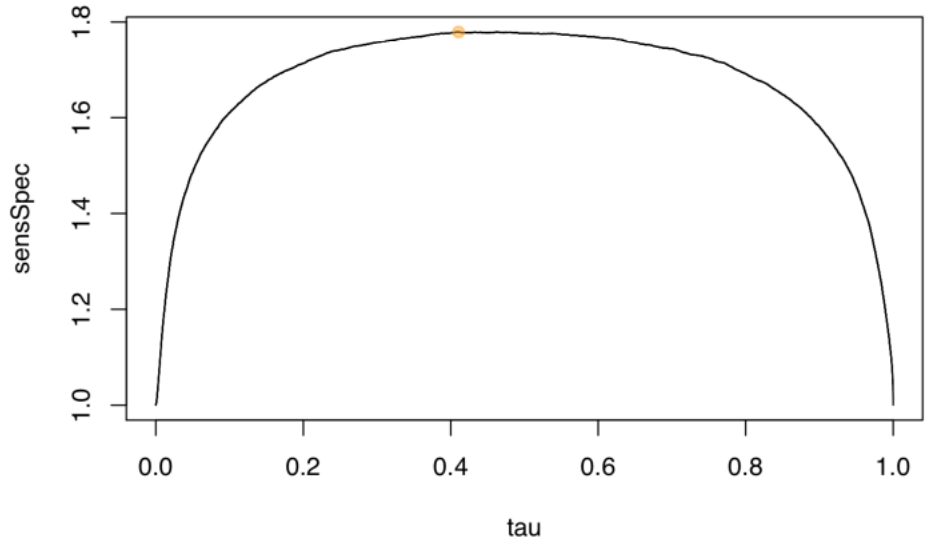
```
predObj = prediction(fitted(fit), SolarFull$solar_system_count)

perf = performance(predObj, "tpr", "fpr")
plot(perf)
abline(0,1, col = "orange", lty = 2)
```

ROC CURVE




```
plot(tau, sensSpec, type = "l")
points(tau[best], sensSpec[best], pch = 19, col = adjustcolor("orange", 0.5))
```



```
tau[best]
```

```
##       407
## 0.4105009
```

```
pred = ifelse(fitted(fit) > tau[best], 1, 0)
tab_best = table(SolarFull$solar_system_count, pred)
tab_best
```

```
##       pred
##           0    1
##   high 9508 1392
##   low   916 8920
```

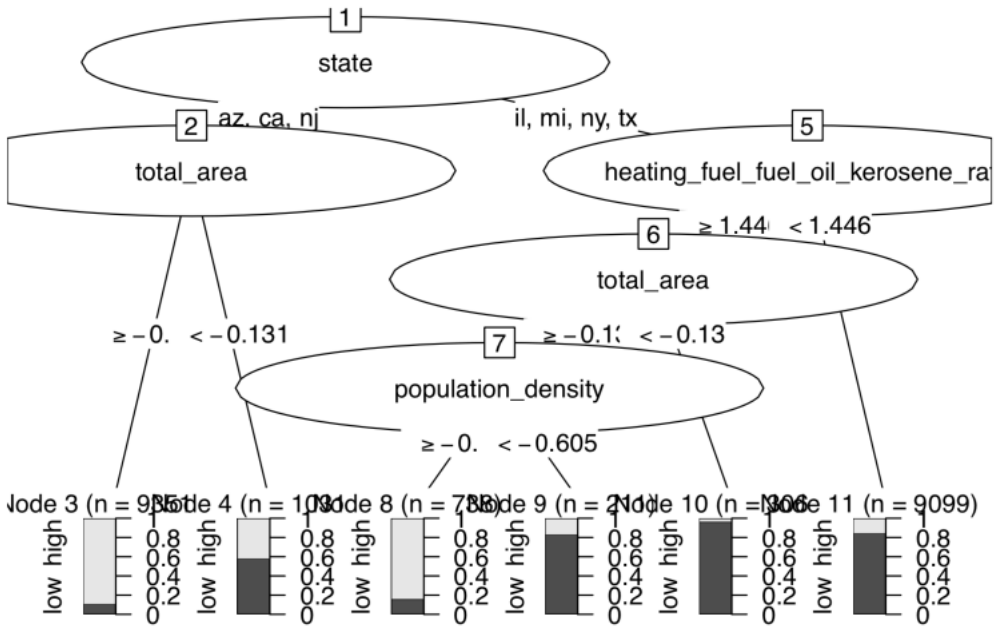
```
# Accuracy
sum(diag(tab_best))/ sum(tab_best)
```

```
## [1] 0.888696
```

Classification Trees

After using the classification trees model on the dataset, a tree is generated. From the tree, we see that the model splits the observations into two categories. The first category contains states such as Arizona, California and New Jersey. The second category contains states such as Illinois, Michigan, New York and Texas. It continues to split into categories and at the end of it, 11 nodes are reached. In the second category, heating_fuel_fuel_oil, total area and population are also included. After computing the performance of our model, our model has an accuracy of 85.45 percent. This figure is lower than that of the Logistic Regression model.The model correctly classifies 8986 observations as high and 8733 observations as low. The model wrongly classifies 1914 as high and 1103 observations as low. We plot the ROC curve and the area under the ROC curve is 0.8719. This value is high but the area under the ROC curve for linear regressions is higher.

CLASSIFICATION TREE



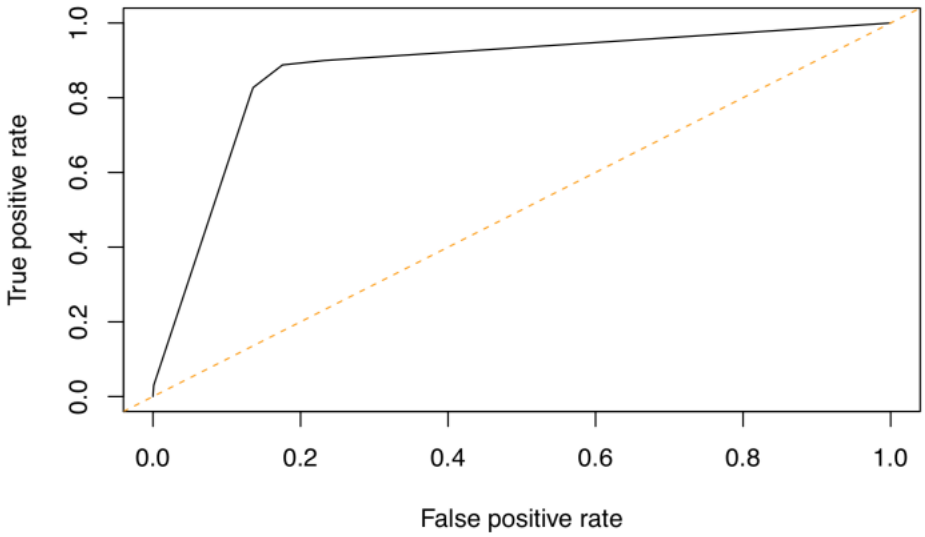
```
class_ct = predict(ct, type ="class")
tab_ct = table(SolarFull$solar_system_count, class_ct)
tab_ct
```

```
##       class_ct
##        high  low
##   high 8986 1914
##   low  1103 8733
```

```
# Accuracy
sum(diag(tab_ct))/ sum(tab_ct)
```

```
## [1] 0.8545042
```

ROC CURVE




```
# Compute the area under the roc curve
auc_ct = performance(predObj_ct, "auc")
auc_ct@y.values
```

```
## [[1]]
## [1] 0.8719242
```

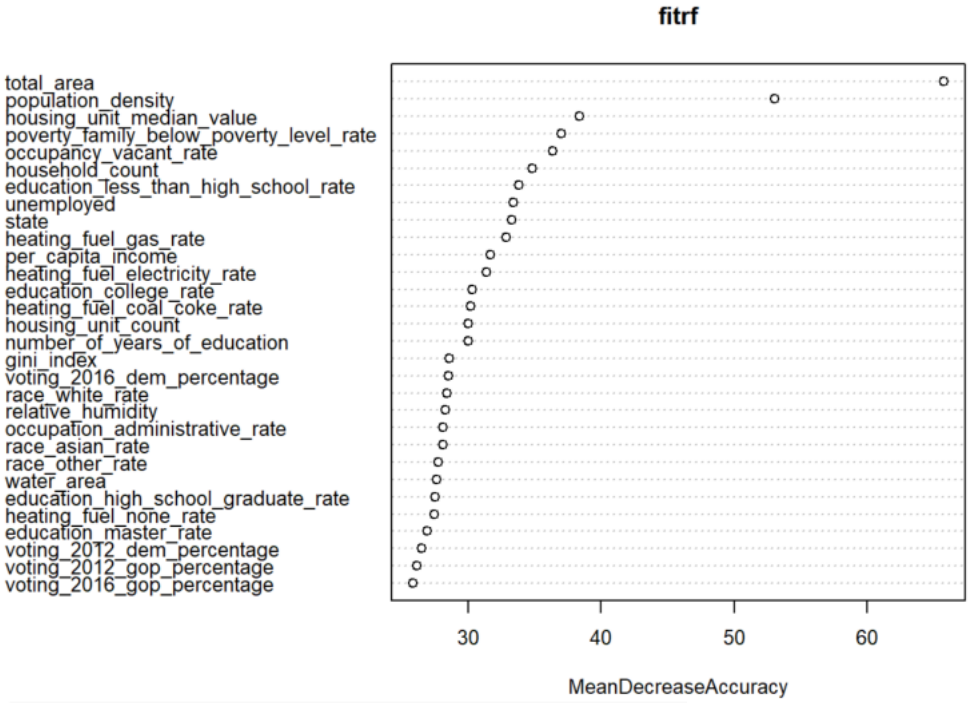
Random Forests

Random Forests uses a random subset of the predictor variables at each split of the classification tree fitting. After applying the random forest model, we discover that the most important variables affecting the deployment of solar power installations are total_area, population_density, housing_unit_median. The model correctly classifies 9939 observations as high and 8785 observations as low. 961 observations were wrongly classified as high and 1051 observations were wrongly classified as low. The classification accuracy of our model is 90.35 percent which is better than the accuracy of the classification trees and Logistic Regression.

```
fitrf=randomForest(solar_system_count~., data = data, importance=TRUE)
fitrf
```

```
varImpPlot(fitrf, type = 1, cex=0.8)
```

RANDOM FOREST MODEL




```
# Confusion Matrix
fitrf$confusion
```

```
##      high  low class.error
## high 9930  970  0.08899083
## low  1049 8787  0.10664904
```

```
# Accuracy
sum(diag(fitrf$confusion[,1:2]))/sum(fitrf$confusion[,1:2])
```

```
## [1] 0.9026331
```

Support Vector Machines

Support Vector Machines are based on Linear Algebra and it works by finding the best plane to separate two classes. After applying the SVM, the accuracy of our model is 91.23 percent. This is the highest among the models. SVM correctly classifies 9966 observations as high and 8890 observations as low. SVM wrongly classifies 934 observations as high and 946 observations as low.

```
class_svm = predict(fitsvm)

tab_svm = table(SolarFull$solar_system_count, class_svm)
tab_svm
```

```
##       class_svm
##         high   low
##   high 10005   895
##   low    913  8923
```

```
sum(diag(tab_svm))/ sum(tab_svm)    # accuracy
```

```
## [1] 0.9128086
```

RESULTS AND DISCUSSION

Support Vector Machines have the best accuracy among the four models used for prediction. Furthermore, we split the dataset into training, validation and test subsets. We apply all models and iterate over them 50 times. Every time the iteration is performed, a different sample for training and validating the data is used. Every time the iteration is done, the accuracy of the four super supervised learning methods are compared with one another, the name and maximum value of the best model is stored. After checking the results, Random Forest performs best with a mean accuracy of 89.98 %. When we applied the models individually, Random Forest and SVM had very similar accuracies. Therefore, we can conclude that Random Forest is the best classifier.

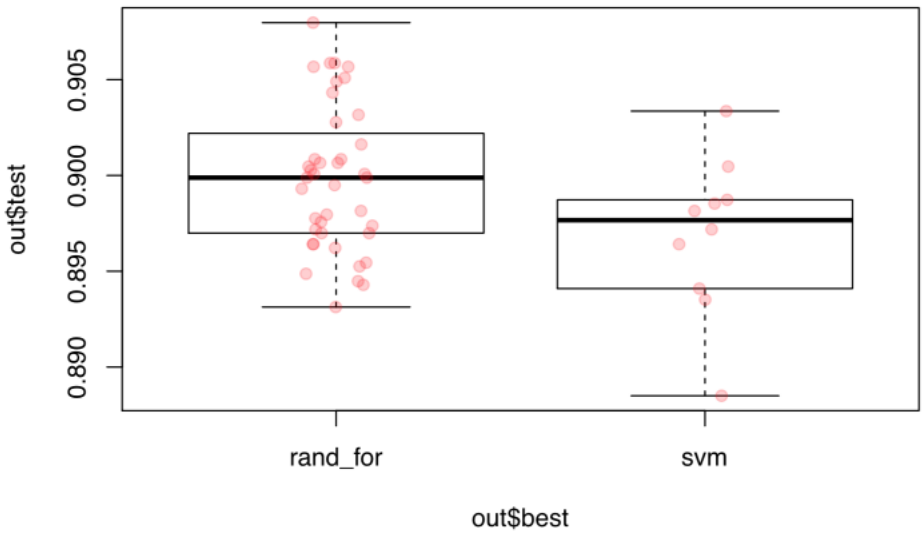
```
# How many times each classifier was selected
table(out[,5])/ 50
```

```
##
## rand_for      svm
##      0.8      0.2
```

```
# Summary test accuracy of the selected classifiers
tapply(out[,6], out[,5], summary)
```

```
## $rand_for
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8931  0.8970  0.8999  0.8998  0.9019  0.9080
##
## $svm
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8885  0.8947  0.8977  0.8969  0.8987  0.9034
```

BOX PLOT FOR ACCURACY OF THE ITERATIONS OF THE MODELS



CONCLUSION

Four supervised learning methods were applied to predict if the solar power system coverage was high or low. The methods applied are Logistic Regression, Classification Trees, Random Forests and Support Vector Machines. After comparing the four methods of classification, the best was Random Forest as it had the highest accuracy among other models.

REFERENCES

Calvin Lee Kwan. Influence of local environmental, social, economic and political variables on the spatial distribution of residential solar PV arrays across the United States. Energy Policy, 47: 332–344, 2012.

James, G., 2013. An Introduction To Statistical Learning: With Applications In R (springer Texts In Statistics). Springer.

Jiafan.2020.Home – DeepSolar. [ONLINE] Available at http://web.stanford.edu/group/deepsolar/home. [Accessed 15 April 2020].

Jordan M Malof, Leslie M Collins, Kyle Bradbury, and Richard G Newell. A deep convolutional neural network and a random forest classifier for solar photovoltaic array detection in aerial imagery, 2016c.

APPENDIX

```
# represents comments

SolarFull = read.csv("data_project_deepsolar.csv")

# Structure of the solar dataset
str(SolarFull)

# Dimenision of the dataset
dim(SolarFull)

# Removing Categorical variables
solar_num = SolarFull[,-c(1,2,76,79)]

# Scaling the Solar Dataset
solar_num = scale(solar_num)

corr_solar = cor(solar_num)

for (i in 1:nrow(corr_solar)) {
  for (j in 1:ncol(corr_solar)) {
    if(corr_solar[i,j] > 0.85){
      if(i!=j){
        vec = c(i,j,corr_solar[i,j])
        print(vec)
      }
    }
  }
}

solar_num = solar_num[, -c(1,2,4,6,14,34,37,38,40,42,47,50,51,53,56,72,73,76)]

# Adding categoriacl variables back into the dataset
SolarFull = cbind(SolarFull[,c(1,2,76,79)], solar_num)

## Logistic Regression

# We fit a logistic regression where the response is function of all the
# other variables.
fit = glm(solar_system_count ~., data = SolarFull, family = "binomial")
summary(fit)

w = coef(fit)
exp(w)

# Standard Errors
sm = summary(fit)
se = sm$coef[,2]
```

```
# Confidence Limits for w
wLB = w - 1.96* se
wUB = w + 1.96 * se

ci = cbind(lb = wLB, w=w, ub = wUB)
ci

exp(ci)

# Estimated logg-odds for each observation and the estimated Probabilities
lg = predict(fit)
head(lg)

phat = predict(fit, type = "response")
head(phat)

slcou = as.integer(SolarFull$solar_system_count)

symb = c(19,17)
cols = c("red","blue")

plot(lg, jitter(phat, amount = 0.1), pch = symb[SolarFull$slcou], col = adjustcolor(cols[slcou],0.7), cex =0.7, xlab="Log-Odds", ylab="Fitted Probabilities")

tau = 0.5
p = fitted(fit)
pred = ifelse(p > tau, 1, 0)

# Cross tabulation between observed and predicted
tab = table(SolarFull$solar_system_count, pred)
tab

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

library(ROCR)
predObj = prediction(fitted(fit), SolarFull$solar_system_count)

perf = performance(predObj, "tpr", "fpr")
plot(perf)
abline(0,1, col = "orange", lty = 2)

area = performance(predObj, "auc")
area@y.values

# The optimal threshold taub can be found maximizing the sum of sensitivity
# and specificity for different values of tau.

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(tau, sensSpec, type = "l")
points(tau[best], sensSpec[best], pch = 19, col = adjustcolor("orange", 0.5))

tau[best]

pred = ifelse(fitted(fit) > tau[best], 1, 0)
tab_best = table(SolarFull$solar_system_count, pred)
tab_best

# Accuracy
sum(diag(tab_best))/ sum(tab_best)

## CLASSIFICATION TREES

library(rpart)
library(partykit)

ct = rpart(solar_system_count ~., data = SolarFull)
# Generating classification trees
plot(as.party(ct), cex = 0.2)

ct

summary(ct)

class_ct = predict(ct, type ="class")
tab_ct = table(SolarFull$solar_system_count, class_ct)
tab_ct

# Accuracy
sum(diag(tab_ct))/ sum(tab_ct)

phat_ct = predict(ct)
head(phat_ct)

predObj_ct = prediction(phat_ct[,2], SolarFull$solar_system_count)
roc_ct = performance(predObj_ct, "tpr", "fpr")
plot(roc_ct)
abline(0,1, col = "orange", lty=2)

# Compute the area under the roc curve
auc_ct = performance(predObj_ct, "auc")
auc_ct@y.values
```

```
## RANDOM FORESTS
 library(randomForest)

# Random Forest Model
fitrf = randomForest(solar_system_count ~., data = SolarFull, importance = TRUE)
fitrf

varImpPlot(fitrf, type =1)

# Confusion Matrix
fitrf$confusion

# Accuracy
sum(diag(fitrf$confusion[,1:2]))/sum(fitrf$confusion[,1:2])


## SUPPORT VECTOR MACHINES
fitsvm = ksvm(solar_system_count ~., data = SolarFull)
fitsvm

class_svm = predict(fitsvm)

tab_svm = table(SolarFull$solar_system_count, class_svm)
tab_svm

sum(diag(tab_svm))/ sum(tab_svm)   # accuracy

# Kernelized support vector machines can be applied using the function ksvm,
# and we compare this method with logistic regression and classification trees.

library(rpart)
set.seed(19201513)

# Replicate the process a number of times
R = 50
out = matrix(NA, 50, 6)
colnames(out)=c("logistic","classtree", "rand_For", "svm", "best", "test")
out = as.data.frame(out)

for (r in 1:R) {
  # split the data into training, validation and test sets
  N = nrow(SolarFull)
  train = sample(1:N, size = 0.50 *N)
  val = sample(setdiff(1:N, train), size = 0.25*N)
  test = setdiff(1:N, union(train, val))

  # fit classifiers to only the training data
  fitLog = multinom(solar_system_count ~., data = SolarFull[train,], trace = FALSE)
  fitCt = rpart(solar_system_count ~., data = SolarFull[train,])
  fitRF = randomForest(solar_system_count ~., data = SolarFull[train,])
```

```
fitSVM = ksvm(solar_system_count ~., data = SolarFull[train,])

# fit on validation data

# Logistic Regression
predValLog = predict(fitLog, type ="class", newdata = SolarFull[val,])
tabValLog = table(SolarFull$solar_system_count[val], predValLog)
accLog = sum(diag(tabValLog))/sum(tabValLog)

# Classification trees
predValCt = predict(fitCt, type = "class", newdata = SolarFull[val,])
tabValCt = table(SolarFull$solar_system_count[val], predValCt)
accCt = sum(diag(tabValCt))/sum(tabValCt)

# Random Forest
predValRF = predict(fitRF, newdata = SolarFull[val,])
tabValRF = table(SolarFull$solar_system_count[val], predValRF)
accRF = sum(diag(tabValRF))/ sum(tabValRF)

# Support Vector Machines
predValSVM = predict(fitSVM, newdata = SolarFull[val,])
tabValSVM = table(SolarFull$solar_system_count[val], predValSVM)
accSVM = sum(diag(tabValSVM))/ sum(tabValSVM)

# Compute Accuracy
acc = c( log = accLog,class_tree = accCt, rand_for = accRF, svm = accSVM)
out[r,1] = accCt
out[r,2] = accLog
out[r,3] = accRF
out[r,4] = accSVM

# Use the method that did best on the validation data
# to predict the test data

best = names(which.max(acc))
switch (best,
    logistic = {
      predTestLog = predict(fitLog, type ="class", newdata = SolarFull[test,])
      tabTestLog = table(SolarFull$solar_system_count[test], predTestLog)
      accBest = sum(diag(tabTestLog))/sum(tabTestLog)
    },
    class_tree = {
      predTestCt = predict(fitCt, type = "class", newdata = SolarFull[test,])
      tabTestCt = table(SolarFull$solar_system_count[test], predTestCt)
      accBest = sum(diag(tabTestCt))/sum(tabTestCt)
    },
  rand_for = {
    predTestRF = predict(fitRF, newdata = SolarFull[test,])
    tabtestRF = table(SolarFull$solar_system_count[test], predTestRF)
    accBest = sum(diag(tabtestRF))/sum(tabtestRF)
```

```
    },
    svm = {
      predTestSVM = predict(fitSVM, newdata = SolarFull[test,])
      tabTestSVM = table(SolarFull$solar_system_count[test], predTestSVM)
      accBest = sum(diag(tabTestSVM))/sum(tabTestSVM)
      }
  )

  out[r,5] = best
  out[r,6] = accBest
}


# How many times each classifier was selected
table(out[,5])/ 50

# Summary test accuracy of the selected classifiers
tapply(out[,6], out[,5], summary)

# Plotting
boxplot(out$test~out$best)
stripchart(out$test ~ out$best, add = TRUE, vertical = TRUE, method = "jitter", pch = 19, col
= adjustcolor("red", 0.2))
```