Assignment 3

Michael Whelan 17338833

2023-04-26

In the assignment the questions are listed by numbers and have a structure. I have decided to tackle this assignment as a mini project so rather than me following the assignment structure I am making the assignment more of a discussion with an appropriate flow. I will start by discussing my choices and motivations and then move on to the modelling making comments along the way and then picking my best one and commenting on that. All questions will be answered but not in the order the question sheet is.

Introduction

When beginning any machine learning project/assignment it is important to first have a think about which direction you want to take it. In this case we are given data and asked to create 3 different models, pick the best one and form a discussion around them. There are some things we need to consider.

The Data: The data has a binary variable with the outcomes of ‘high’ and ‘low’. ‘high’ being a high coverage of solar power systems in a tile and ‘low’ being a low-medium coverage in a tile. Along with this variable there 14 other variables which combine to determine if coverage is high/low.

The Method: What do we want to do with this data? An appropriate method could be to classify the records and there are many models that can do that. Specifically binary classification, so this is the route we will take.

The Models: Now we understand the data and our method let us look at models. We can see the data is labelled which means it would make sense to select a supervised learning method. There are many models to select so determining if the model will be supervised or unsupervised narrows the selection down a lot. So now we are looking for a supervised method that will classify and even when narrowing this down we still have some options.

- Logistic Regression
- Support Vector Machines (SVM)
- Decision trees
- Random Forests
- K-Nearest Neighbors

This is just to name a few. Let us go through each of them giving some pros and cons and finally settling on 3 for this assignment.

1. Logistic Regression

Whenever I hear the word ‘binary’ in machine learning I think Logistic Regression instantly, why? The reason for this is it is one of the most popular algorithms out there as it comes with many advantages.

Advantages

Logistic regressions are very easy to implement, fast to train and interpretable. It is also works well with linearly separable problems. These are classification problems that can be separated by drawing a straight line on a plane and it separates the data into 2 or more classes.

Disadvantages

With this nice feature of linear separability comes the disadvantage that the Logistic regression model assumes linearly separable. So if the data is complex maybe logistic regression is a bit simple for the issue. Which brings us to our next disadvantage being Logistic regression has a limited capacity to model complexity relationships.

Decision

I will decide to take Logistic regression as my first model based on the scenarios above. Our data has many rows but is not too complex and has only 1 binary variable The problem looks to be linearly separable also so I feel Logistic regression would be a good choice.

2. Support Vector Machines (SVM)

Support Vector Machines have a great framework behind them. They are based a lot on linear algebra and the idea behind them is easy to grasp. Let’s look and some pros and cons.

Advantages

They are great for modelling complex decision boundaries (like what we touched on above in logistic regression). They are great at modelling in high dimensional data and versatile with different kernel functions. A kernel function is a function that transforms the input data from a lower dimensional space to a higher one where the classes are linearly separable by a hyper plane.

Disadvantages

SVM’s can be slow to train with large data sets and choosing an appropriate kernel function can be difficult depending on the circumstances. They can also take up a lot of computational resources so if they are limited in the machine learning scenario, SVM’s may not be the best choice.

Decision

For this assignment I don’t think SVM’s are a great fit, I think the data is too simple. Like I said before it has many rows but there are only 15 variables and it seems like the data can be linearly separated. I have decided to reject SVM’s for this assignment.

3. Decision Trees

Decision Trees are very popular algorithm for classification in our case but also for regression too. It gives a graphical representation of a set of rules that can be used to make decisions based on input features. it splits the data based on the most informative features to create a tree-like model.

Advantages

Like logistic regression, Decision Trees are easy to understand and interpret. They can handle a mix of categorical and numerical data and can model complex decision boundaries.

Disadvantage

They have a tenancy to over fitting. Over fitting meaning a model fits the data too closely and fails to generalize on new/unseen data. This can happen when a model is too complex relative to the amount of training data, resulting in high variance and low bias.

Decision

I have decided to pick a Decision Tree model as one of my models as it's easy to interpret and similar to before, the data is not very large or complex.

4. Random Forests

Random Forests are also quite popular, they combine multiple decision trees to make predictions. They randomly select subsets of features and samples from training data to create each decision tree.

Advantages

They can handle high dimensional data and are robust to over fitting. they can also model complex decision boundaries

Disadvantages

They often can be slow to train big data sets and can be difficult to interpret.

Decision

I have decided to go for Random Forests for this assignment. It is similar enough to the decision tree so if we happen to over fit in the decision tree we have the random forest to fall back on.

5. K-Nearest Neighbors (KNN)

KNN is a simple machine learning algorithm is used for classification and regression tasks. It predicts the target value of a new data point based on the k closest data points in the training data, where k is a user-defined parameter. The prediction is based on the majority class or average value of the k neighbors.

Advantages

KNN is easy to implement and can handle nonlinear decision boundaries. They also work well for small data sets.

Disadvantages

They can be slow to predict large data sets and they require a distance metric to be chosen. By a distance metric I mean a function that is used to measure similarity (or dissimilarity) between 2 data points and example being the Euclidean distance.

Decision

The advantages definitely outweigh the disadvantages especially for this assignment I will admit however the classification trees are easier to interpret in my opinion.

Models Chosen

- Logistic Regression
- Decision Trees
- Random Forests

We will now load in our data

```
load("/Users/Michael/Downloads/data_hw3_deepsolar (1).RData")
```

We can have a quick look at the data

```
head(data)
```

```
##       solar_system_coverage average_household_income employ_rate
## 9478                   high                 78800.68   0.9149405
## 19709                  high                100498.66   0.8952381
## 16634                  high                 73678.92   0.9140977
## 3014                   high                 78840.51   0.9434139
## 13740                  high                 83214.07   0.9072407
## 13501                  high                 67139.60   0.9315577
##       population_density housing_unit_count housing_unit_median_value
## 9478           4637.1680               1512                    325800
## 19709           777.9355               3217                    424000
## 16634         10831.8800               1906                    339000
## 3014           1988.0120               2981                    123500
## 13740           740.6628               1985                    220600
## 13501          9717.4950               2099                    277300
##       occupancy_vacant_rate heating_fuel_gas_rate heating_fuel_electricity_rate
## 9478             0.02777778             0.9272109                    0.07278912
## 19709            0.09853901             0.5920690                    0.27758621
## 16634            0.08656873             0.8891442                    0.02642160
## 3014             0.05836967             0.2230139                    0.77377984
## 13740            0.03677582             0.5407950                    0.44194561
## 13501            0.07717961             0.7965927                    0.09963862
##       heating_fuel_oil_rate land_area  water_area air_temperature
## 9478            0.000000000 0.9109008 0.002813140            15.9
## 19709           0.004137931 9.4815050 0.000000000            12.4
## 16634           0.074669730 0.4789567 0.003520094            11.7
## 3014            0.000000000 4.2650640 4.220209000            17.4
## 13740           0.000000000 7.3258710 0.000000000            16.1
## 13501           0.099122354 0.5644459 0.000000000            12.2
##       earth_temperature daily_solar_radiation
## 9478               17.8                  5.14
## 19709              13.6                  5.21
## 16634              11.6                  3.80
## 3014               17.9                  4.70
## 13740              18.3                  5.41
## 13501              12.3                  3.98
```

We can also look at the structure of the data

```
str(data)
```

```
## 'data.frame':    10926 obs. of  15 variables:
##  $ solar_system_coverage        : Factor w/ 2 levels "high","low": 1 1 1 1 1 1 1 1
1 1 ...
##  $ average_household_income     : num  78801 100499 73679 78841 83214 ...
##  $ employ_rate                  : num  0.915 0.895 0.914 0.943 0.907 ...
##  $ population_density           : num  4637 778 10832 1988 741 ...
##  $ housing_unit_count           : int  1512 3217 1906 2981 1985 2099 1876 1600 138
4 2231 ...
##  $ housing_unit_median_value    : num  325800 424000 339000 123500 220600 ...
##  $ occupancy_vacant_rate        : num  0.0278 0.0985 0.0866 0.0584 0.0368 ...
##  $ heating_fuel_gas_rate        : num  0.927 0.592 0.889 0.223 0.541 ...
##  $ heating_fuel_electricity_rate: num  0.0728 0.2776 0.0264 0.7738 0.4419 ...
##  $ heating_fuel_oil_rate        : num  0 0.00414 0.07467 0 0 ...
##  $ land_area                    : num  0.911 9.482 0.479 4.265 7.326 ...
##  $ water_area                   : num  0.00281 0 0.00352 4.22021 0 ...
##  $ air_temperature              : num  15.9 12.4 11.7 17.4 16.1 12.2 17.6 15.1 11.
7 15.7 ...
##  $ earth_temperature            : num  17.8 13.6 11.6 17.9 18.3 12.3 19.5 16.4 11.
6 17 ...
##  $ daily_solar_radiation        : num  5.14 5.21 3.8 4.7 5.41 3.98 5.2 4.77 3.8 5.
08 ...
```

Notice the first variable is the binary variable and this is our target variable we will classify.

Before I begin my analysis I will do some pre-requisites

The libraries I will be using are:

```
library(caret)
library(randomForest)
library(ROCR)
library(pROC)
library(rpart)
library(partykit)
library(rpart.plot)
```

Next I will `set.seed` so anyone can reproduce my analysis.

```
set.seed(69)
```

I will now split my data into 3 sets:

- Training (60%)
- Validation (20%)
- Testing (20%)

I picked the simple approach of splitting as the data does not seem too complex. I also originally planned to split the data individually for each model and treat them all as separate however, I will be comparing them at the end so I felt having all the same splits of data would be more appropriate.

```
n_rows <- nrow(data) #complete number of rows in the data
row_indices <- 1:n_rows #indices of the rows

#sampling 60% of the data to make train data
train_indices <- sample(row_indices, floor(0.6 * n_rows), replace = FALSE)

#Setting out the remaining data
remaining_indices <- setdiff(row_indices, train_indices)

#Taking 50% of the remaining data to make the validation data
val_indices <- sample(remaining_indices, floor(0.5 * length(remaining_indices)), repl
ace = FALSE)

#setting variables to each set
train_set <- data[train_indices, ]
val_set <- data[val_indices, ]
test_set <- data[-train_indices, ]
```

I will now show each model individually. I have a similar workflow for each model which follows the next steps (some may have variation):

- Model
- Prediction
- ROC plot
- Confusion Matrix
- Accuracy number

Logistic Regression

First I fitted the logistic regression model using `glm()` and setting `family = binomial` as it is a binary classification. I used the `train_set` which is my training data in this project.

```
# Fit logistic regression model on training data
model <- glm(solar_system_coverage ~ ., data = train_set, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Next, make the predictions using the `val_set` (validation data).

```
# Evaluate model on val data
predictions <- predict(model, newdata = val_set, type = "response")
```

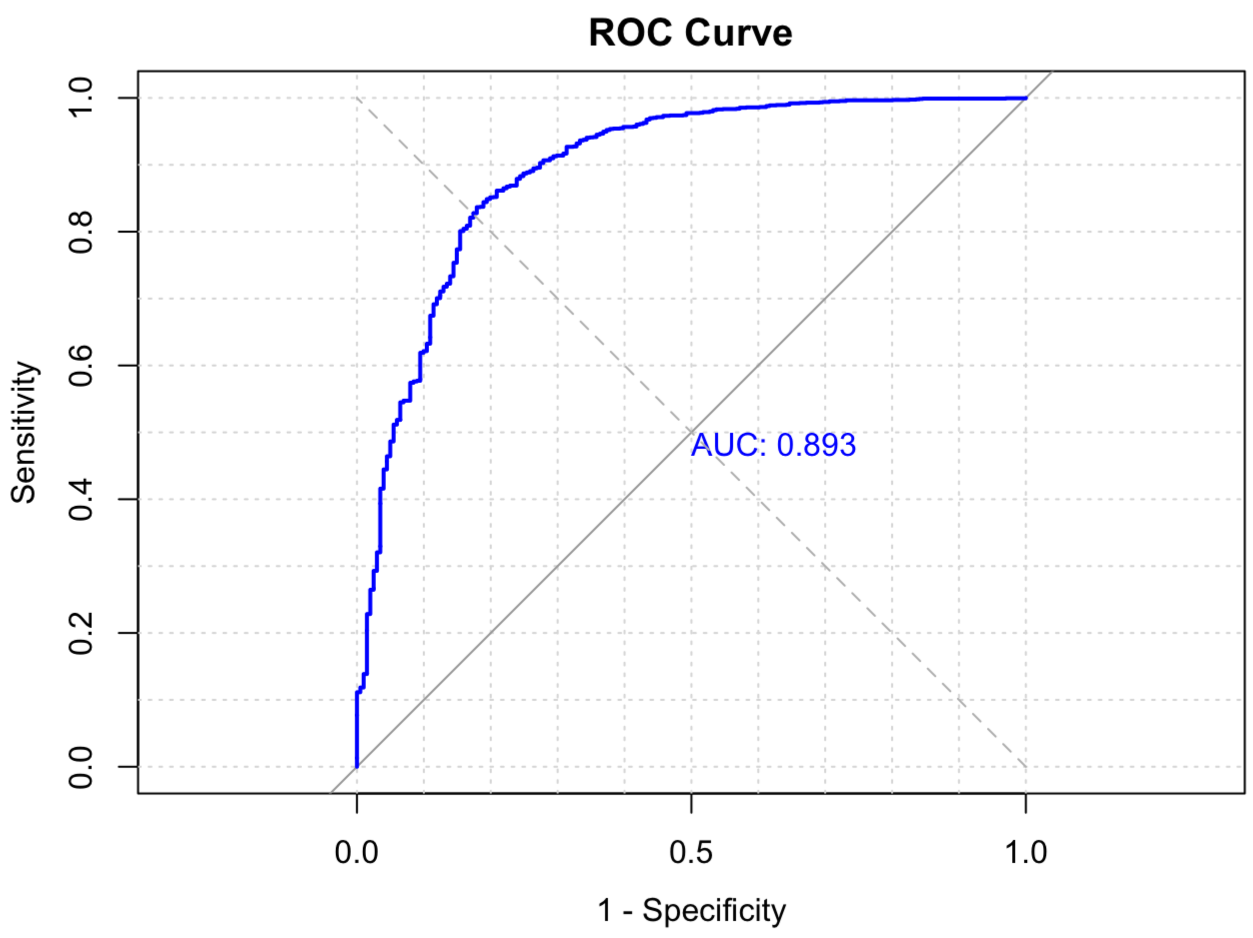
I then began to plot the ROC curve

```
roc <- roc(val_set$solar_system_coverage, predictions)
```

```
## Setting levels: control = high, case = low
```

```
## Setting direction: controls < cases
```

```
# Plot ROC curve
plot(roc, main = "ROC Curve",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)
```



We can see from the plot the AUC (Area Under the Curve) is 0.893.

AUC is a metric used in machine learning to evaluate the performance of a binary classification model.

In a binary classification problem, the goal is to predict the class label of a new example based on its features. A model generates a score for each example, which indicates the likelihood of belonging to the positive class. The AUC represents the area under the Receiver Operating Characteristic (ROC) curve, which is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold settings.

Here we have an AUC quite close to 1 which is good. This indicates very good classification performance.We can also see the optimal threshold in the ROC curve which looks like roughly `0.82` .

Now let's check out the accuracy. I will start by creating the confusion matrix and set my threshold.

```
#if predictions are greater than the threshold set right category
predicted_classes <- ifelse(predictions > 0.82, "low", "high")

#Create a table of predicted classes with the the solar system coverage from the val_
set
confusion_matrix <- table(predicted_classes, val_set$solar_system_coverage)
confusion_matrix
```

```
##
## predicted_classes high  low
##              high  145  185
##              low    56 1799
```

Although the `high` is lower than the false positives and negatives, this could indicate there may be an imbalance in the categories (i.e there is a lot of lows compared to highs in the original data). We will keep an eye on this.

Let's look at the accuracy

```
accuracy1 <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
# Print results
print(paste("Accuracy:", accuracy1))
```

```
## [1] "Accuracy: 0.889702517162471"
```

```
auc1 <- auc(roc) #Saving the AUC for later comparison
```

The logistic regression worked well according to accuracy. The accuracy is very high at 89% and an AUC of 89.3%.

Decision Tree

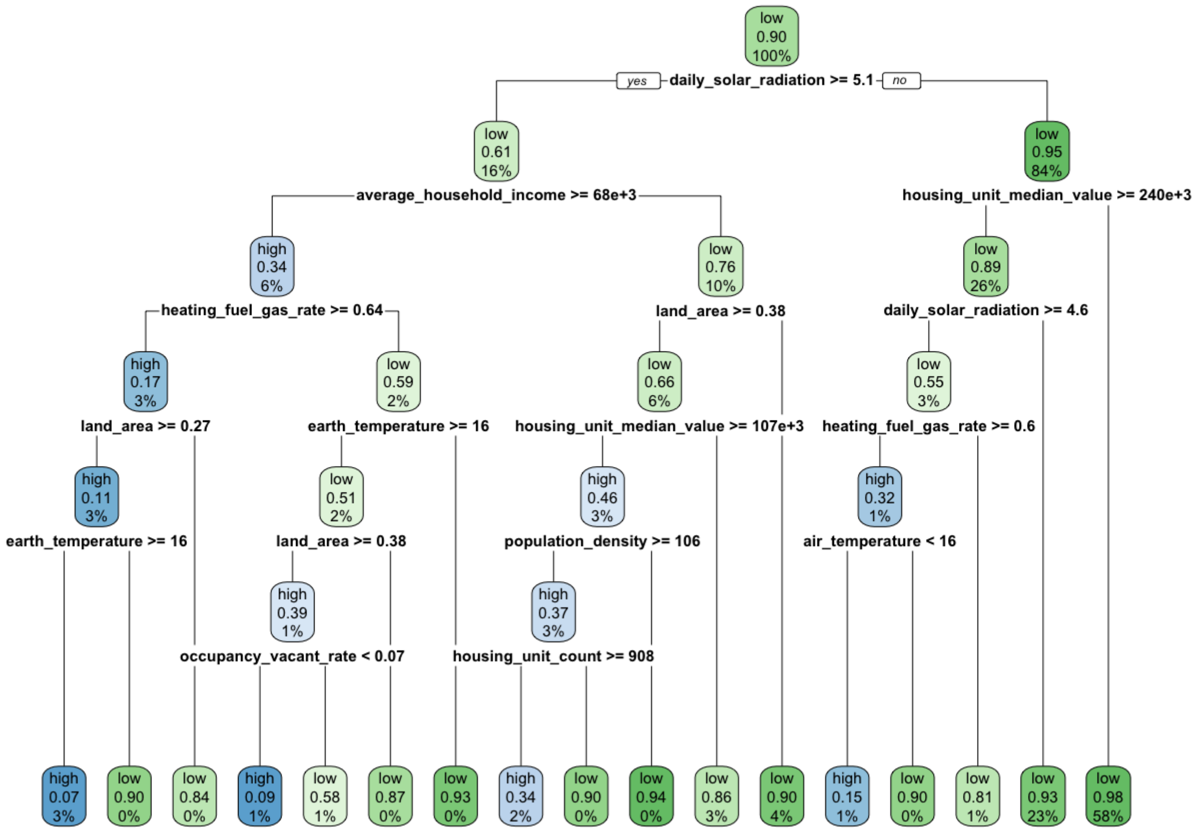
Now we move on to our decision tree. We first create the model like before

```
# Create decision tree model using rpart function
model <- rpart(solar_system_coverage ~ ., data=train_set, method="class")
```

We use our `train_set` and set the `method = class` .

I thought it would be appropriate to plot the decision tree for this case.

```
# Plot decision tree
rpart.plot(model)
```



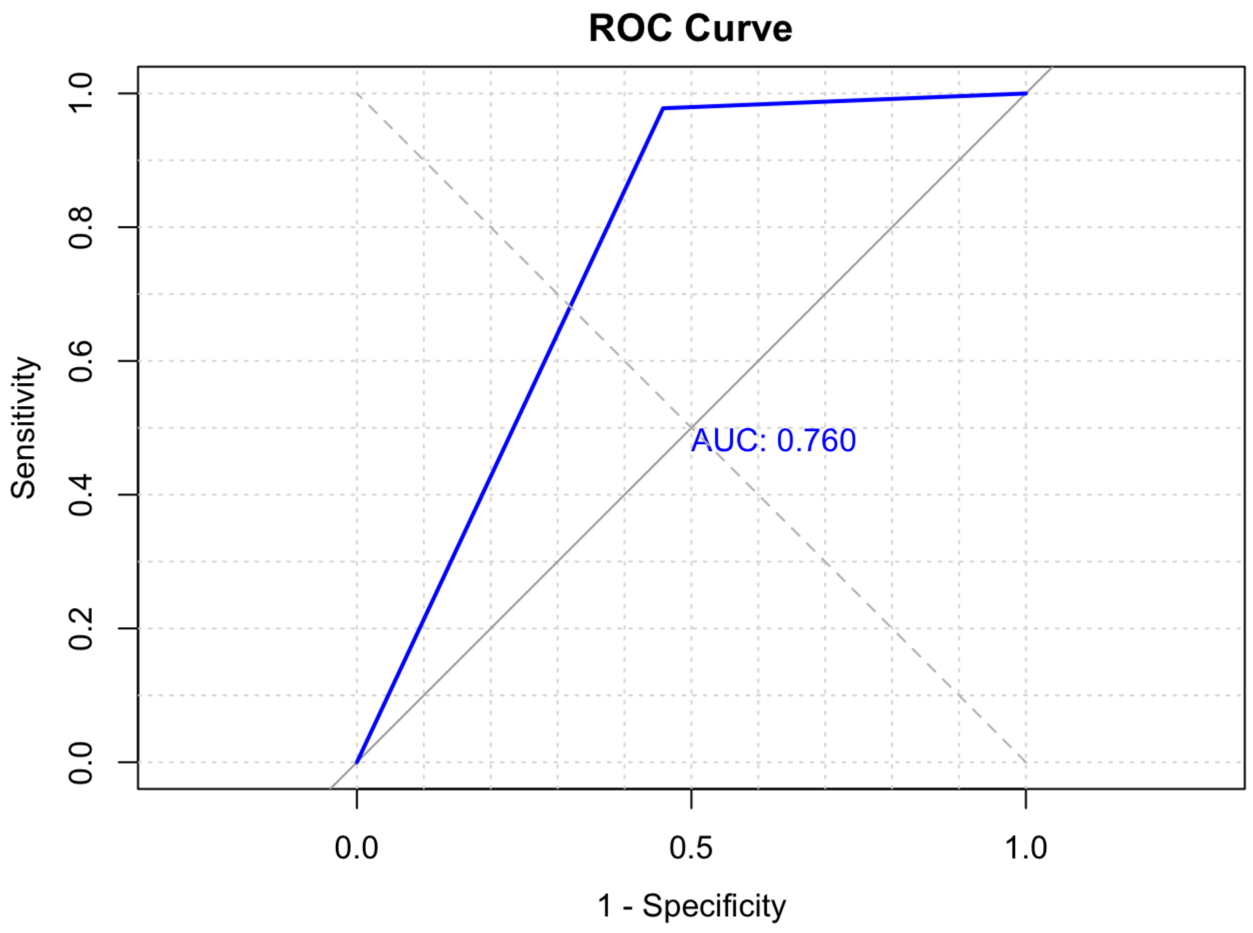
We instantly see `daily_solar_radiation` as our root node and all the different leaf notes. Many nodes appear more than once with `land_area` appearing 3 times.

Next we can make our predictions based on `val_set` .

```
# Make predictions on validation set
val_predictions <- predict(model, newdata=val_set, type="class")
```

```
# Plot ROC curve and calculate AUC for test set predictions
roc_obj1 <- roc(val_set$solar_system_coverage, as.numeric(val_predictions == "high"))

# Plot ROC curve
plot(roc_obj1, main = "ROC Curve",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)
```



```
cat("Test set AUC:", auc(roc_obj1), "\n")
```

```
## Test set AUC: 0.7600556
```

```
auc2 <- auc(roc_obj1)
```

Our ROC curve doesn't look fantastic here. the AUC is quite good at 76% but the curve itself is not the standard shape. The threshold here looks to be about `0.7` . We will use this to calculate our confusion matrix

```
# Generate confusion matrix and accuracy figure for a given threshold
threshold <- 0.7
val_predictions_thresholded <- ifelse(val_predictions  == 'low', 1, 0) >= threshold
confusion_matrix <- table(val_set$solar_system_coverage, val_predictions_thresholded)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Print confusion matrix and accuracy figure
cat("Confusion Matrix (threshold =", threshold, "):\n")
```

```
## Confusion Matrix (threshold = 0.7 ):
```

```
print(confusion_matrix)
```

```
##       val_predictions_thresholded
##        FALSE TRUE
##   high   109   92
##   low     44 1940
```

Again our confusion matrix looks odd with `high-FALSE` although correct is very low.

```
# Evaluate accuracy of validation predictions
val_accuracy <- mean(val_predictions == val_set$solar_system_coverage)
cat("Validation set accuracy:", val_accuracy, "\n")
```

```
## Validation set accuracy: 0.9377574
```

Still a very high accuracy at 94% however this could be misleading.

Random Forest

Finally we reached our last model, the Random Forest. We run the same workflow by starting with our model

```
rf <- randomForest(solar_system_coverage ~ ., data = train_set, importance = TRUE)
```

Now we make our predictions

```
# Make predictions on the val data using the model
predictions3 <- predict(rf,val_set, type = 'prob')
```

Next we sill extract our probabilities. When `type = 'prob'` , the predict function returns a matrix of class probabilities, where each row corresponds to an observation in the new data and each column corresponds to a class (i.e., high or low).

```
# Extract the probabilities of the positive class
pred_prob <- predictions3[,2]
```

We are extracting the probabilities of the positive class from the `predictions3` matrix by using the indexing operator `[,2]` . This selects the second column of the matrix, which contains the probabilities of the positive class.

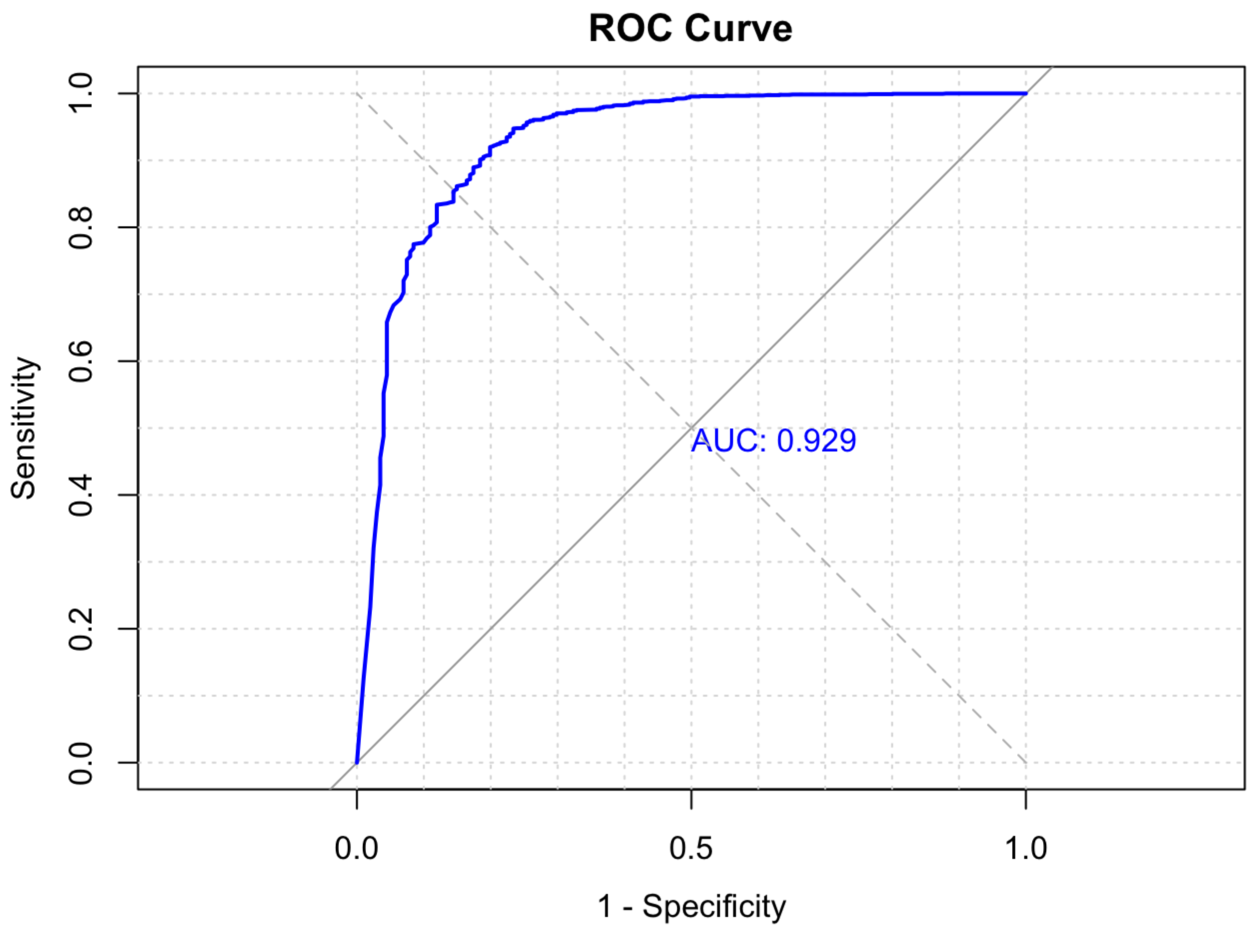
The resulting `pred_prob` variable is a vector of probabilities that can be used to compute the true positive rate (TPR) and false positive rate (FPR) at different threshold values, and to plot the ROC curve which we will do next.

```
# Compute the true positive rate (TPR) and false positive rate (FPR) at different thr
eshold values
roc_obj2 <- roc(val_set$solar_system_coverage, pred_prob)
```

```
## Setting levels: control = high, case = low
```

```
## Setting direction: controls < cases
```

```
# Plot ROC curve
plot(roc_obj2, main = "ROC Curve",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)
```



```
auc3 <- auc(roc_obj2)
```

This ROC curve looks much better than the last with an AUC of 93% which is very good but like I said before the accuracies could be misleading.

```
# Create confusion matrix
pred_class <- ifelse(pred_prob >= 0.75, 1, 0)
actual_class <- val_set$solar_system_coverage
confusion_matrix <- table(pred_class, actual_class)
print(confusion_matrix)
```

```
##           actual_class
## pred_class high  low
##          0  154  116
##          1   47 1868
```

Another confusion matrix quite imbalanced which indicates the data is imbalanced but let us see the accuracy.

```r
# Calculate accuracy
accuracy3 <- sum(diag(confusion_matrix))/sum(confusion_matrix)
print(paste("Accuracy:", accuracy3))
```

```
## [1] "Accuracy: 0.925400457665904"
```

A really high accuracy of 93% for the random forest.

Comparison

Now we have all our model accuracies and AUC numbers, let's make a table.

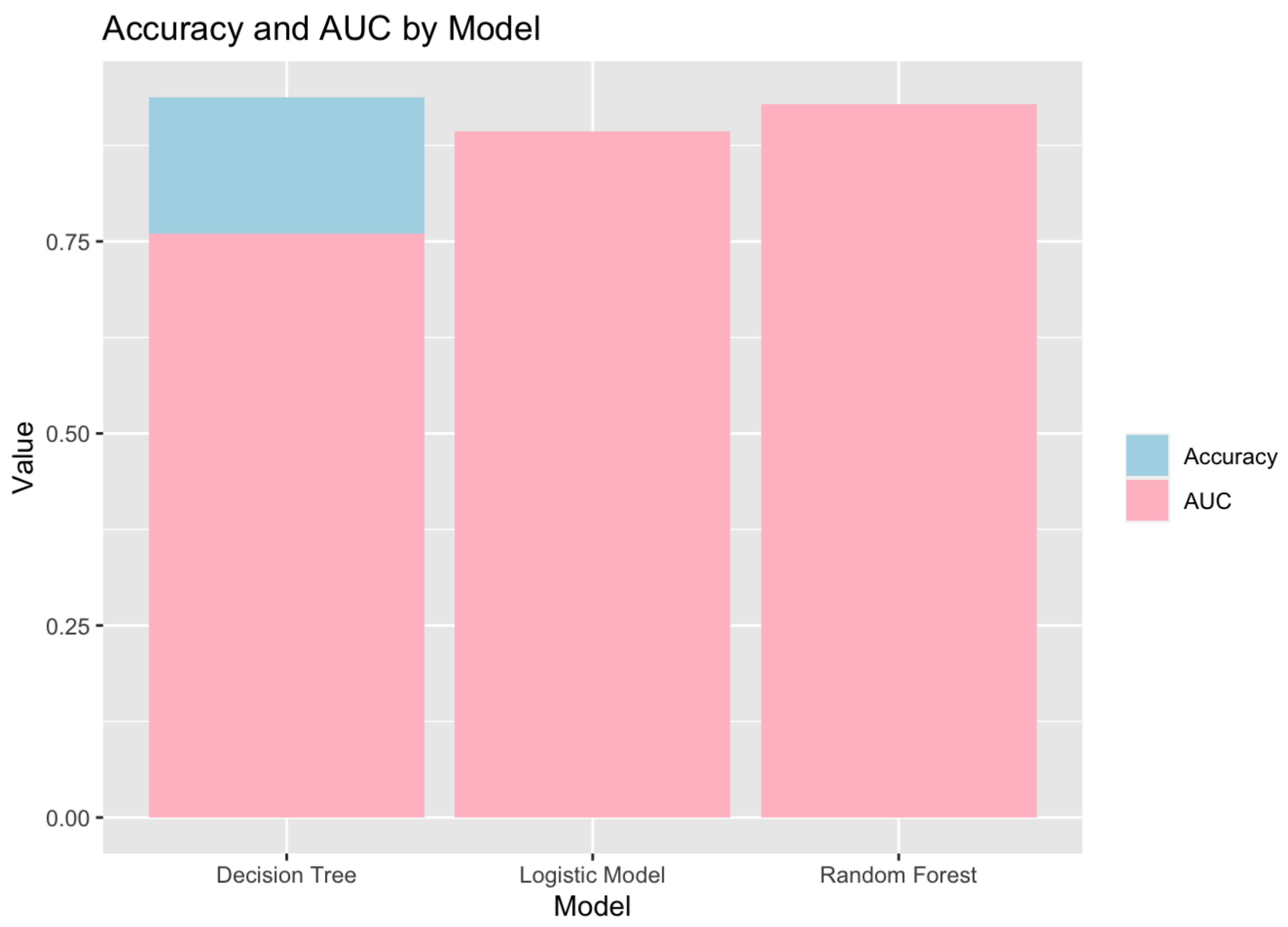
```r
# Create a table to summarize the accuracy results
accuracy_table <- data.frame(
  Model = c("Logistic Model", "Decision Tree", "Random Forest"),
  Accuracy = c(accuracy1, val_accuracy, accuracy3),
  AUC = c(auc1, auc2, auc3)
)

print(accuracy_table)
```

```
##            Model  Accuracy       AUC
## 1 Logistic Model 0.8897025 0.8932756
## 2  Decision Tree 0.9377574 0.7600556
## 3  Random Forest 0.9254005 0.9287371
```

Now let us graph this table

```r
# Plot the graph
ggplot(accuracy_table, aes(x = Model)) +
  geom_bar(aes(y = Accuracy, fill = "Accuracy"), stat = "identity", position = "dodg
e") +
  geom_bar(aes(y = AUC, fill = "AUC"), stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("lightblue", "pink")) +
  labs(x = "Model", y = "Value", fill = "") +
  ggtitle("Accuracy and AUC by Model")
```



From the table and graph we can see the decision tree, although not bad was the weakest performer. Although it had the highest accuracy it had the lowest AUC number. I am aware the AUC isn't terrible but compared to the others it is low.

If a model has high accuracy but a low AUC, it may mean that the model is good at predicting the majority class, but performs poorly in identifying the minority class.

The Logistic model performed very well but the Random Forest was that bit better.

Using Test data on our best model

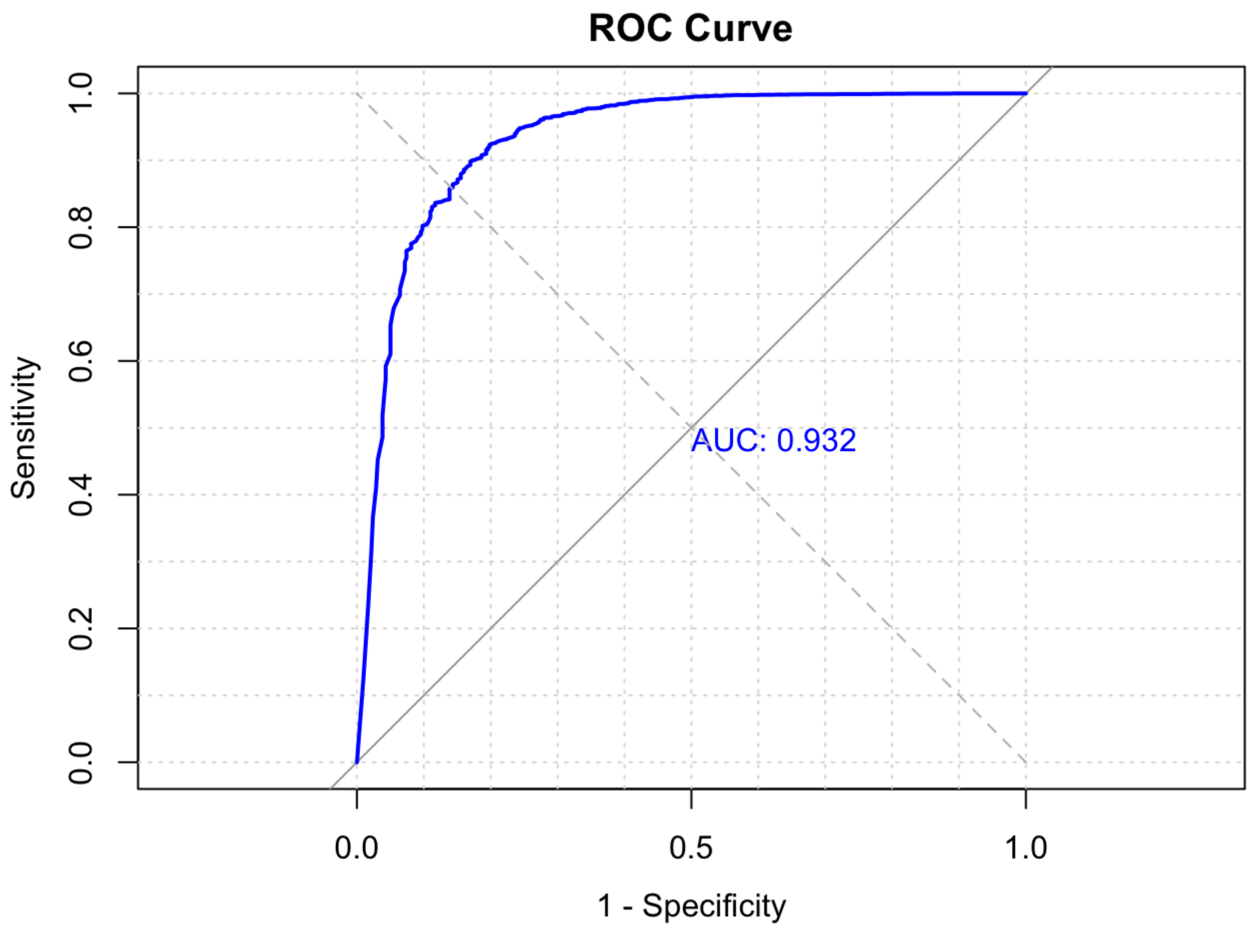
Clearly our best model here was the Random Forest so now lets use our test data on the model

```
predictions_test <- predict(rf,test_set, type = 'prob')
pred_prob_test <- predictions_test[,2]
# Compute the true positive rate (TPR) and false positive rate (FPR) at different thr
eshold values
roc_obj_test <- roc(test_set$solar_system_coverage, pred_prob_test)
```

```
## Setting levels: control = high, case = low
```

```
## Setting direction: controls < cases
```

```
# Plot ROC curve
plot(roc_obj_test, main = "ROC Curve",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)
```



```r
auc_test <- auc(roc_obj_test)


# Create confusion matrix
pred_class_test <- ifelse(pred_prob_test >= 0.75, 1, 0)
actual_class_test <- test_set$solar_system_coverage
confusion_matrix_test <- table(pred_class_test, actual_class_test)
print(confusion_matrix_test)
```

```
##                  actual_class_test
## pred_class_test high  low
##                0  319  228
##                1  100 3724
```

```r
# Calculate accuracy
accuracy_test <- sum(diag(confusion_matrix_test))/sum(confusion_matrix_test)
print(paste("Accuracy:", accuracy_test))
```

```
## [1] "Accuracy: 0.924959963395104"
```

Carrying out the same operation as we did for the validation data but now for the test data. Let us now compare the validation and test data for the random forest

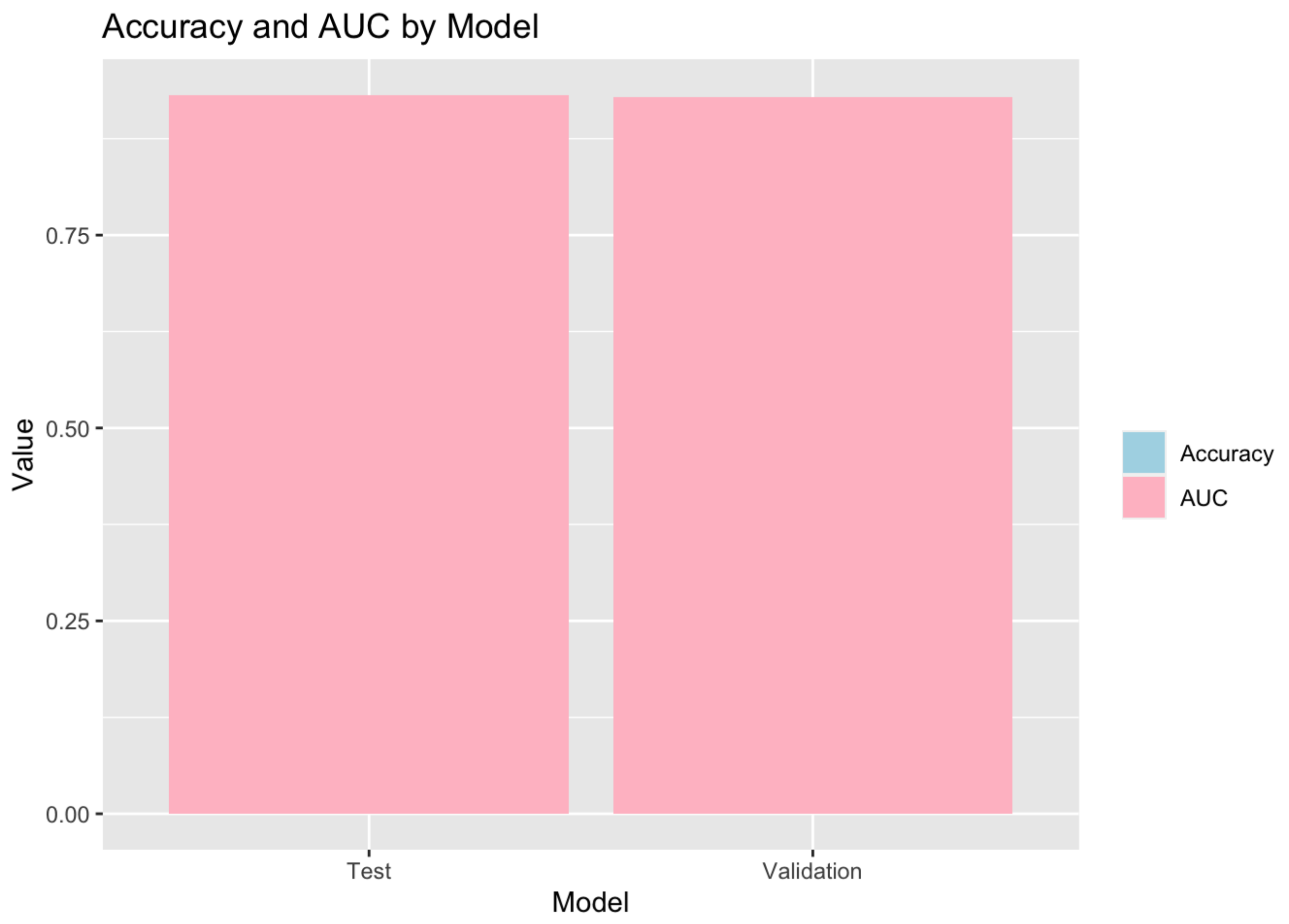
```
# Create a table to summarize the accuracy results
accuracy_table_test <- data.frame(
  Model = c("Validation", "Test"),
  Accuracy = c(accuracy3, accuracy_test),
  AUC = c(auc3, auc_test)
)

print(accuracy_table_test)
```

```
##        Model  Accuracy       AUC
## 1 Validation 0.9254005 0.9287371
## 2       Test 0.9249600 0.9315926
```

All very similar which is a good sign. Let us take a look at the graph

```
# Plot the graph
ggplot(accuracy_table_test, aes(x = Model)) +
  geom_bar(aes(y = Accuracy, fill = "Accuracy"), stat = "identity", position = "dodg
e") +
  geom_bar(aes(y = AUC, fill = "AUC"), stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("lightblue", "pink")) +
  labs(x = "Model", y = "Value", fill = "") +
  ggtitle("Accuracy and AUC by Model")
```



No suprise at all, the figures are similar making the model look great in validation and test.

After these 3 models it's evident there is an imbalance in the data set and there is some sort of bias. By looking at the data we can see there is a lot more 'low' than 'high' and something I should have done before I began.

```
table(data$solar_system_coverage)
```

```
##
## high  low
## 1090 9836
```

Clearly an imbalance. To correct this what can we do? We can use boosting and bagging methods to improve this.

I will create a bag model (`bag_model`) and a boost model (`boost_model`) to try and correct this.

First store our data in a variable

```
# load data, rename and convert class column
dat0 <- data
```

I will also use K-Folds cross validation which I did not do previously.

```
# split
train_val <- createDataPartition(dat0$solar_system_coverage, p = 0.80, list = FALSE)
dat <- dat0[train_val,]
dat_test <- dat0[-train_val,]

train_ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 10)
```

Next define our bagging and boosting models

WARNING! This took me about an hour to run the 2 models

```
# Define the bagged and boosted models
bag_model <- train(solar_system_coverage ~ ., data = dat, method = "treebag",
                   trControl = train_ctrl, preProcess = c("center", "scale"),
                   tuneLength = 10)
```

```
boost_model <- train(solar_system_coverage ~ ., data = dat, method = "gbm",
                     trControl = train_ctrl, preProcess = c("center", "scale"),
                     tuneLength = 10)
```

Next make our predictions for the bagging model.

```
# Make predictions on the validation set using bagging
val_predictions_bag <- predict(bag_model, newdata = dat_test)

# Evaluate the bagged model using ROC curve, AUC, and accuracy
val_roc_bag <- roc(dat_test$solar_system_coverage,
                   predict(bag_model, newdata = dat_test, type = "prob")[, 2])
```

```
## Setting levels: control = high, case = low
## Setting direction: controls < cases
```

```
val_auc_bag <- auc(val_roc_bag)
val_acc_bag <- confusionMatrix(val_predictions_bag, dat_test$solar_system_coverage)$o
verall[1]
```

Then make our predictions for the boosting model.

```
# Make predictions on the validation set using boosting
val_predictions_boost <- predict(boost_model, newdata = dat_test)

# Evaluate the boosted model using ROC curve, AUC, and accuracy
val_roc_boost <- roc(dat_test$solar_system_coverage,
                     predict(boost_model, newdata = dat_test, type = "prob")[, 2])
```

```
## Setting levels: control = high, case = low
```

```
## Setting direction: controls < cases
```

```
val_auc_boost <- auc(val_roc_boost)
val_acc_boost <- confusionMatrix(val_predictions_boost, dat_test$solar_system_coverag
e)$overall[1]
```

Finally, print our results

```
# Create a data frame with the results
results_df_bag_boost <- data.frame(
  Model = c("Bagging", "Boosting"),
  AUC = c(val_auc_bag, val_auc_boost),
  Accuracy = c(val_acc_bag, val_acc_boost)
)

results_df_bag_boost
```

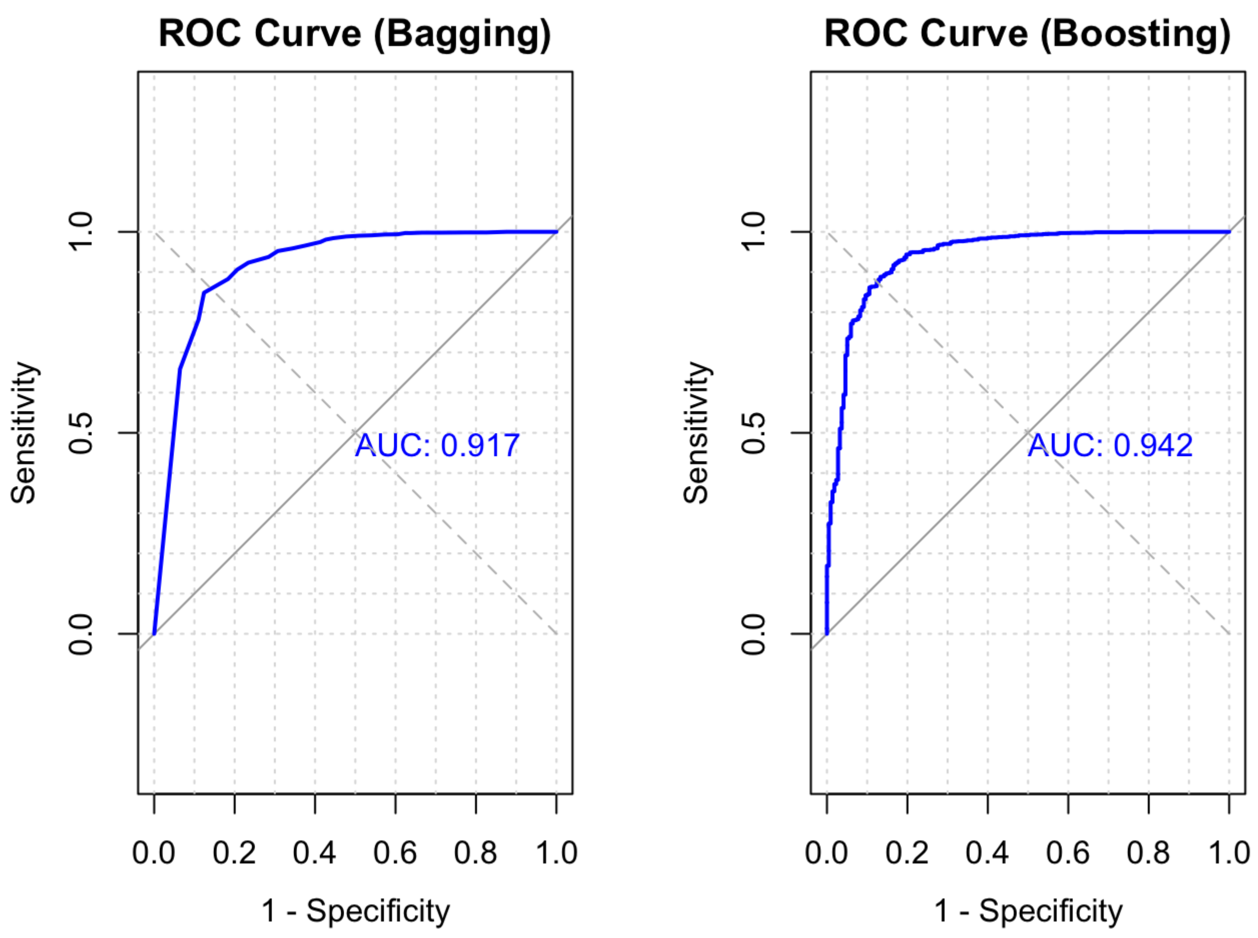
```
##      Model       AUC  Accuracy
## 1  Bagging 0.9173717 0.9414188
## 2 Boosting 0.9416962 0.9455378
```

Not only did we account for the imbalance using the booting and bagging method but our accuracies and AUC numbers were much better than the previously.

To finish I will compare the ROC curves too.

```
# Plot ROC curves
par(mfrow = c(1, 2))
plot(val_roc_bag, main = "ROC Curve (Bagging)",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)

plot(val_roc_boost, main = "ROC Curve (Boosting)",
     print.auc = TRUE, legacy.axes = TRUE, grid = TRUE, col = "blue")
lines(x = c(0, 1), y = c(0, 1), col = "gray", lty = 2)
```



The curves look very well too compared to some of the ones we had previously.

Conclusion

So to sum up we had 3 models taking a very simplistic approach possibly some errors too. We had imbalanced data and found out a bit late into the assignment yet we still analyzed our models as asked on the assignment. We recognized our error and corrected it by introducing bagging and boosting to our assignment along with K-Folds cross validation on the fly. We can now confidently say the new models introduced are the best ones with the boosting model having the highest accuracy and AUC.