FIT3152 Assignment 2, Aayan Ahmed Khan, Student ID: 31486347

Question 1

Proportion of days it is warmer than previous day (WarmerTomorrow = 1): 0.5385

Proportion of days it is not warmer than previous day (WarmerTomorrow = 0): 0.4665

Noteworthy in data: It is important to note that most days (more than 50% of the time) there is none or very little rainfall and that the pressure variables have very similar values for all quartiles which would likely mean that these variables are not very good predictors for our target variable.

Omitting attributes from our analysis: We would likely omit the Day, Month, and Year attributes from our analysis as they likely have close to no correlation with predicting the target variable of the next day being hotter. Also, we would omit the Location attribute. Finally, we would also omit the WindGustDir, WindDir9am, WinDir3pm attributes as wind direction has close to no correlation to our target variable.

Description of real-valued attributes: We can see that most variables are distributed evenly around the mean and that the most NA values are in the Evaporation, Sunshine and Cloud9am variables.

```
    MinTemp          MaxTemp         Rainfall        Evaporation         Sunshine       WindGustSpeed    WindSpeed9am    WindSpeed3pm
 Min.   :-3.40   Min.   : 8.90   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :13.00   Min.   : 0.0   Min.   : 0.00
 1st Qu.: 8.70   1st Qu.:19.00   1st Qu.: 0.000   1st Qu.: 2.650   1st Qu.: 5.400   1st Qu.:33.00   1st Qu.: 9.0   1st Qu.:15.00
 Median :12.90   Median :22.70   Median : 0.000   Median : 4.400   Median : 8.800   Median :39.00   Median :15.0   Median :20.00
 Mean   :12.57   Mean   :23.43   Mean   : 1.811   Mean   : 5.141   Mean   : 7.853   Mean   :41.41   Mean   :15.9   Mean   :20.73
 3rd Qu.:16.50   3rd Qu.:27.20   3rd Qu.: 0.800   3rd Qu.: 7.200   3rd Qu.:10.600   3rd Qu.:48.00   3rd Qu.:20.0   3rd Qu.:26.00
 Max.   :27.90   Max.   :44.10   Max.   :84.000   Max.   :19.000   Max.   :13.900   Max.   :96.00   Max.   :59.0   Max.   :59.00
 NA's   :29      NA's   :15      NA's   :37       NA's   :790      NA's   :901      NA's   :109     NA's   :79     NA's   :81
```

```
  Humidity9am      Humidity3pm      Pressure9am      Pressure3pm        Cloud9am        Cloud3pm         Temp9am          Temp3pm
 Min.   : 11.00   Min.   :  5.00   Min.   : 994.6   Min.   : 993.3   Min.   :0.000   Min.   :0.000   Min.   : 0.60   Min.   : 7.70
 1st Qu.: 56.00   1st Qu.: 39.00   1st Qu.:1013.7   1st Qu.:1011.4   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:13.70   1st Qu.:17.50
 Median : 68.00   Median : 54.00   Median :1018.0   Median :1015.9   Median :4.000   Median :4.000   Median :17.70   Median :21.00
 Mean   : 67.79   Mean   : 53.01   Mean   :1018.2   Mean   :1015.9   Mean   :4.099   Mean   :4.084   Mean   :17.73   Mean   :21.59
 3rd Qu.: 81.00   3rd Qu.: 66.00   3rd Qu.:1022.7   3rd Qu.:1020.5   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:21.60   3rd Qu.:25.00
 Max.   :100.00   Max.   :100.00   Max.   :1037.9   Max.   :1034.4   Max.   :8.000   Max.   :8.000   Max.   :38.20   Max.   :42.80
 NA's   :44       NA's   :78       NA's   :293      NA's   :294      NA's   :757     NA's   :850     NA's   :22      NA's   :62
```

```
apply(real_value_predictors, 2, sd, na.rm = TRUE)
      MinTemp       MaxTemp      Rainfall   Evaporation
     5.597280      6.144471      5.892544      3.075633
     Sunshine WindGustSpeed  WindSpeed9am  WindSpeed3pm
     3.759648     12.788190      8.077656      8.906604
  Humidity9am   Humidity3pm   Pressure9am   Pressure3pm
    17.850043     19.711608      6.610455      6.524242
     Cloud9am      Cloud3pm       Temp9am       Temp3pm
     2.856023      2.728316      5.517282      5.808297
```

Question 2

Preprocessing of data: Firstly, we removed the Day, Month, Year and Location columns from our data set. Then we also removed the WindGustDir, WindDir9am, WinDir3pm attributes. Finally, I removed any rows that had missing pieces of data such as NA.

Question 3

set.seed(31486347) #Student ID as random seed

train.row = sample(1:nrow(weatherdata), 0.7*nrow(weatherdata))

data.train = iris[train.row,]

data.test = iris[-train.row,]

Question 4

All models were implemented with their default settings. The bagging model and the boosting model both were given an mfinal value of 10.

Question 5

Decision Tree Accuracy: 0.6293

```
#Decision Tree Confusion
> print(t1)
               Actual_Class
Predicted_Class  0  1
              0 79 55
              1 41 84
```

Naïve Bayes Accuracy: 0.6448

```
#Naive Bayes Confusion
> print(t1)
               Actual_Class
Predicted_Class  0  1
              0 80 52
              1 40 87
```

Bagging Accuracy: 0.6486

```
#Bagging Confusion
> print(t1)
               Actual_Class
Predicted_Class  0  1
              0 70 41
              1 50 98
```

Boosting Accuracy: 0.6293

```
#Boosting Confusion
> print(t1)
               Actual_Class
Predicted_Class  0  1
              0 69 45
              1 51 94
```

Random Forest Accuracy: 0.7027

```
#Random Forest Confusion
> print(t1)
               Actual_Class
Predicted_Class  0  1
              0 82 41
              1 38 98
```

Question 6

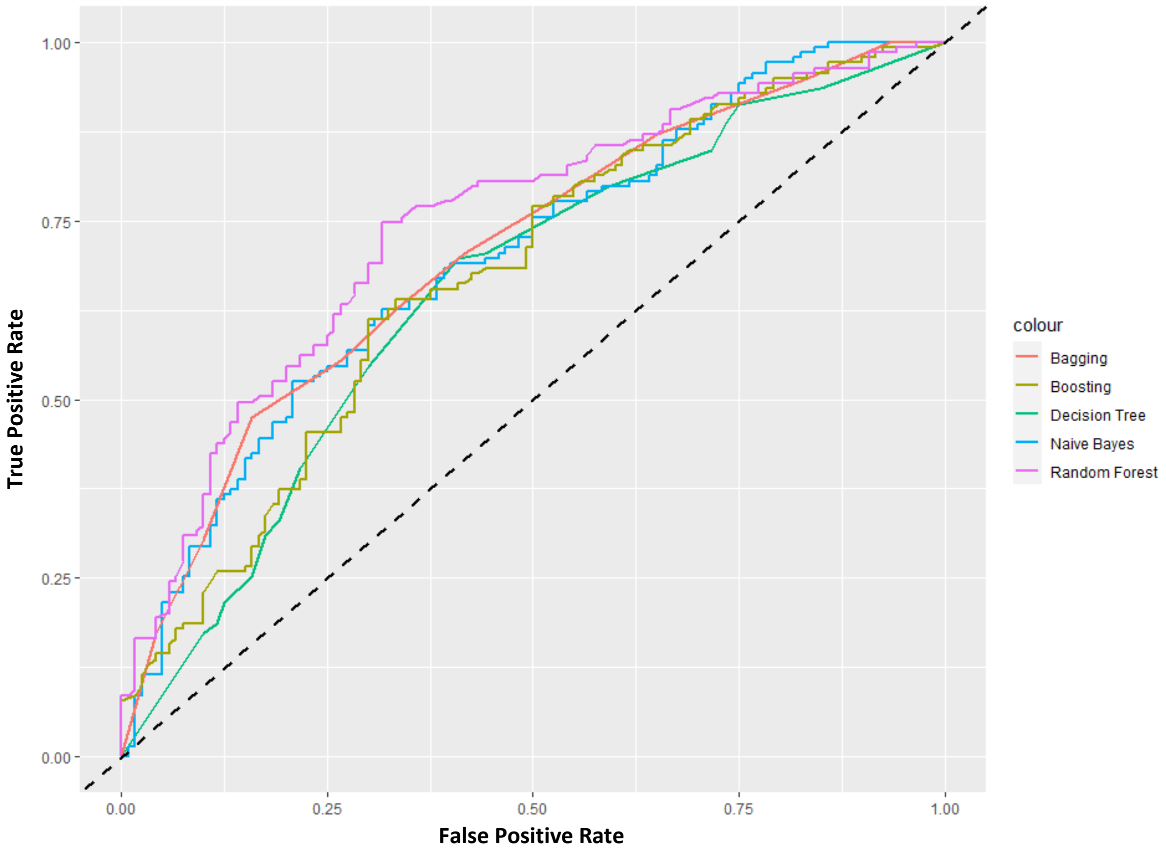
Decision Tree AUC: 0.6534

Naïve Bayes AUC: 0.6974

Bagging AUC: 0.7050

Boosting AUC: 0.6721

Random Forest AUC: 0.7426



Graph: ROC Curves of Different Models

Question 7

The random forest model is the clear single best classifier as it has significantly higher accuracy and a significantly higher AUC than all the other models.

	Accuracy	Area Under Curve (AUC)
Decision Tree	0.6293	0.6534
Naïve Bayes	0.6448	0.6974
Bagging	0.6486	0.7050
Boosting	0.6293	0.6721
Random Forest	0.7027	0.7426

Question 8

Most important variables: Sunshine, MinTemp, Evaporation, Temp3pm, Humidity3pm, Cloud9am

Less important variables: Pressure3pm, Pressure9am, Temp9am, Humidity9am

Least important variables: Rainfall, Cloud3pm, WindGustSpeed, Windspeed3pm, Windspeed9am

The variables that could be omitted from the data with very little effect on performance include Rainfall, Cloud3pm, WindGustSpeed, Windspeed3pm, Windspeed9am. This is because these variables are consistently rated the least important variables in predicting the target variable by all the models we have used. Furthermore, it makes sense that measurements such as windspeed and rainfall don't help in predicting whether tomorrow will be a hotter day. All the models roughly had the same order for the importance of variables, so the random forest model's predictor importance sums up the overall importance of the predictors.

```
#Random Forest Attribute Importance
> print(warmer.rf$importance[,1][order(-warmer.rf$importance)])
     Sunshine       MinTemp   Humidity3pm   Evaporation       Temp3pm       MaxTemp
    30.238694     23.639238     23.578950     22.680954     22.123985     22.060673
  Humidity9am   Pressure9am   Pressure3pm       Temp9am      Cloud9am  WindSpeed9am
    20.655765     19.747357     18.368686     18.069935     17.996401     13.099704
WindGustSpeed  WindSpeed3pm      Cloud3pm      Rainfall
    12.816350     12.769843     10.377973      9.739401
```

Figure: Relative Importance of Predictors For the Random Forest Model

Question 9

To create a simple classifier that can be used by hand, I simply created a decision tree with only the top 3 most important variables (Sunshine, MinTemp, Cloud9am). The classifier is very easy to use by hand and visualize as it is a very basic tree with just 6 leaf nodes. The model has an accuracy of 0.5869 which is close to the other models but is still noticeably worse than the least accurate model (accuracy of 0.6255) and especially worse than the most accurate model (accuracy of 0.7027). The model has an AUC of 0.6434 which is once again close to the other models but is still a little worse than the least accurate model (AUC of 0.6534) and especially worse than the most accurate model (AUC of 0.7426). This can also be seen in the graph below showing how our simple classifier is close in performance to the other models but it still noticeable worse than the worst model. I gave importance to the model having the least number of variables and also being easy to visualize and thus I chose the decision tree. I chose the attributes as they were described as the most important variables by the other models.

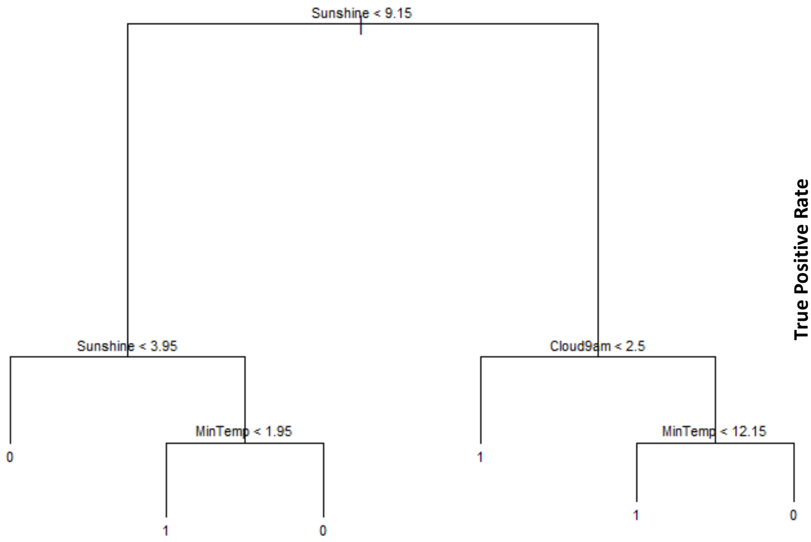
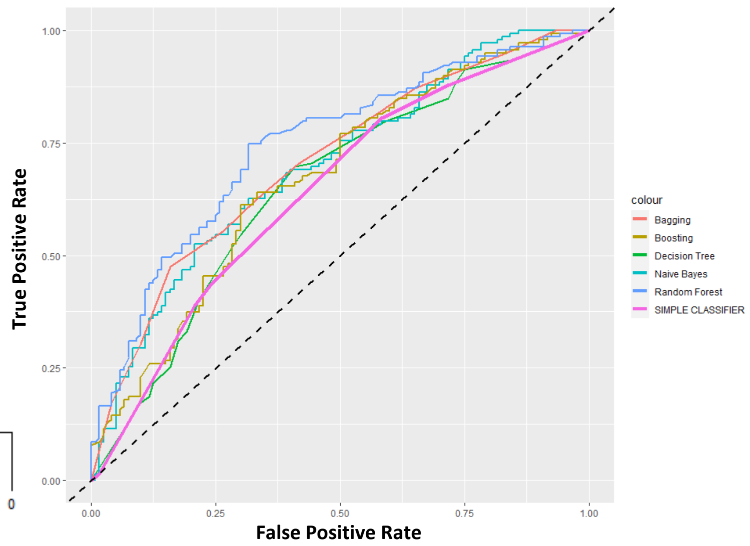


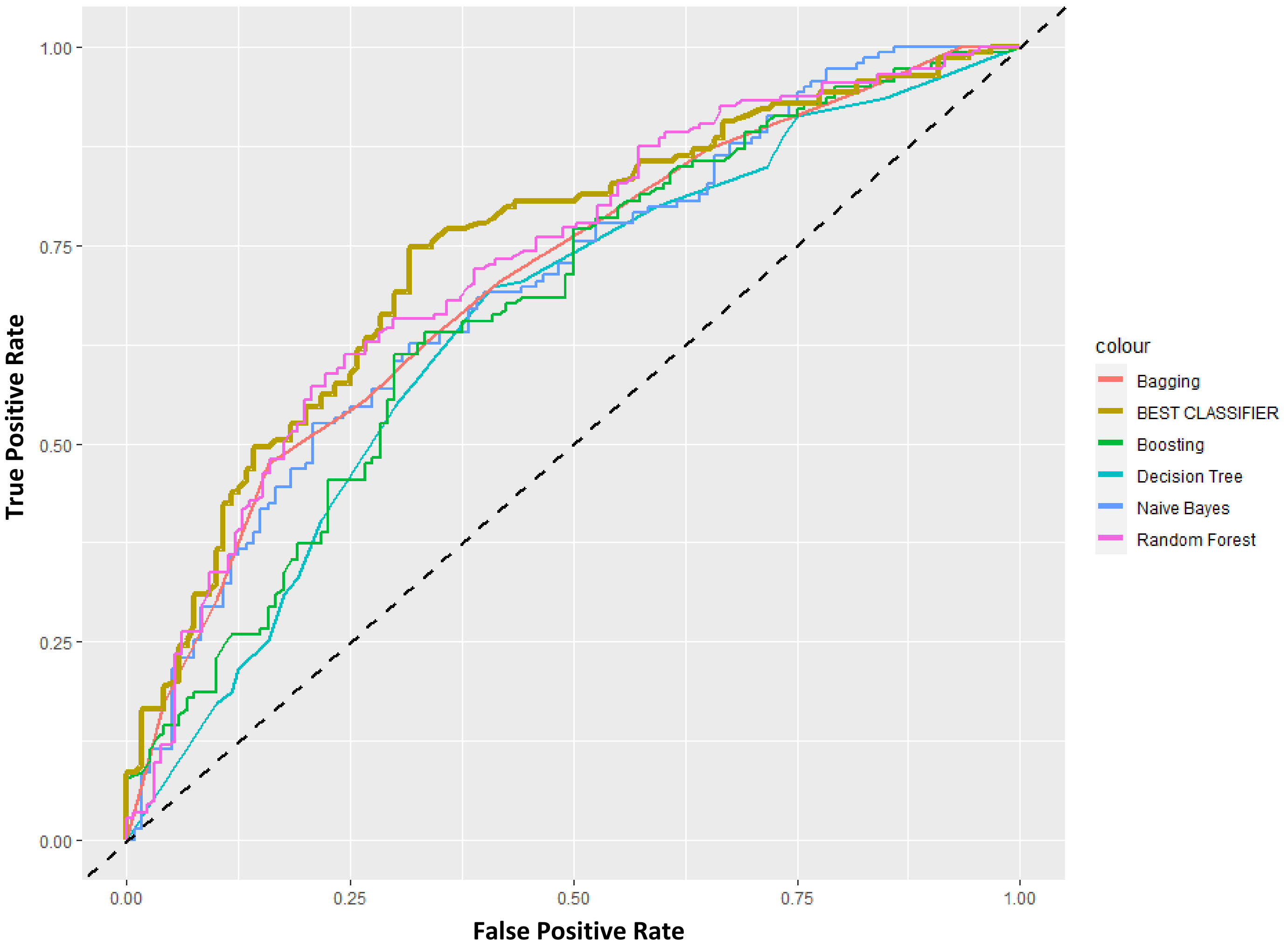
Figure: Decision Tree Produced



Graph: ROC Curves of Different Models, Simple Classifier introduced

Question 10

I aimed to create the best tree-based classifier by modifying a random forest model as the default random forest model was already outperforming all the other models significantly. I modified the model by using significantly more trees in the forest by setting the number of trees to 1000 compared to the 500 trees by default. I also only used the most important predictors determined in Question 8 (Sunshine, MinTemp, Evaporation, Temp3pm, Humidity3pm, Cloud9am) which also meant that less rows were removed due to NAs and there was more training data for the model. The model had an accuracy of 0.7205 which is slightly better than our default random forest model which had an accuracy of 0.7027. This new model also had an AUC of 0.7605 which was also slightly better than the default random forest model which had an AUC of 0.7426. Ultimately, our model was slightly better than the default random forest model whereas it was significantly better than all the other models which is evident in the graph below.



Graph: ROC Curves of Different Models, BEST CLASSIFIER introduced

Question 11

The neural network was trained on the six most important attributes as determined by the other models with 5 hidden layers. Also, the target variable was set to a numerical number rather than a factor and all rows with an NA were removed. The neural network had the exact same training data as all the other models and had an accuracy of 0.7181 which meant that it slightly outperformed the random forest model which had an accuracy of 0.7027. Thus, the neural network was the best performing model out of all the models we have tested. The neural network may be superior to the other models due to being able to learn complex and non-linear relationships between variables and also because it doesn't make any assumptions about the variables being independent unlike the Naïve Bayes classifier.

Neural Network Accuracy: 0.7181

```
#Neural Network Confusion
> print(t1)
        predicted
observed   0   1
       0  80  40
       1  33 106
> cat("\n#Accuracy\n")

#Accuracy
> print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))
[1] 0.7181467
```

Code Appendix

Question 1

```
#QUESTION 1
nrow(WAUS[WAUS$WarmerTomorrow == 1, ]) / nrow(WAUS)
nrow(WAUS[WAUS$WarmerTomorrow == 0, ]) / nrow(WAUS)
real_value_predictors = WAUS[, c(5:9,11, 14:23)]
summary(real_value_predictors)
apply(real_value_predictors, 2, sd, na.rm = TRUE)
```

Question 2

```
#QUESTION 2
#warmerdata = WAUS[, c(5:24)]
warmerdata = WAUS[, c(5:9, 11, 14:24)]
warmerdata = warmerdata[complete.cases(warmerdata),]
warmerdata$WarmerTomorrow = as.factor(warmerdata$WarmerTomorrow)

str(warmerdata)
```

Question 3

```
#QUESTION 3
set.seed(31486347) #Student ID as random seed
train.row = sample(1:nrow(warmerdata), 0.7*nrow(warmerdata))
warmer.train = warmerdata[train.row,]
warmer.test = warmerdata[-train.row,]
```

Question 4

```
#decision tree
warmer.tree = tree(WarmerTomorrow ~. ,data=warmer.train)
plot(warmer.tree)
text(warmer.tree, pretty=0, cex=0.7)

#naive bayes
warmer.bayes = naiveBayes(WarmerTomorrow ~. ,data=warmer.train)

#bagging
warmer.bag <- bagging(WarmerTomorrow ~. ,data=warmer.train, mfinal=10)

#boosting
warmer.boost <- boosting(WarmerTomorrow ~. ,data=warmer.train, mfinal=10)

#Random Forest
warmer.rf <- randomForest(WarmerTomorrow ~. ,data=warmer.train, na.action = na.exclude)
```

Question 5

```
#decision tree
warmer.tree.pred = predict(warmer.tree, warmer.test, type = "class")
t1=table(Predicted_Class = warmer.tree.pred, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Decision Tree Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))


#naive bayes
warmer.bayes.pred = predict(warmer.bayes, warmer.test)
t1=table(Predicted_Class = warmer.bayes.pred, Actual_Class = warmer.test$WarmerTomorrow)
```

```
cat("\n#Naive Bayes Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))

#bagging
warmer.bag.pred = predict.bagging(warmer.bag, warmer.test)
t1=table(Predicted_Class = warmer.bag.pred$class, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Bagging Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))

#boosting
warmer.boost.pred = predict.boosting(warmer.boost, warmer.test)
t1=table(Predicted_Class = warmer.boost.pred$class, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Boosting Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))

#Random Forest
warmer.rf.pred = predict(warmer.rf, warmer.test)
t1=table(Predicted_Class = warmer.rf.pred, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Random Forest Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))
```

Question 6

```
#decision tree
warmer.tree.pred = predict(warmer.tree, warmer.test, type = "vector")
warmer.tree.pred = prediction(warmer.tree.pred[, 2], warmer.test$WarmerTomorrow)
warmer.tree.perf = performance(warmer.tree.pred, "tpr", "fpr")

warmer.tree.fpr = warmer.tree.perf@x.values
warmer.tree.tpr = warmer.tree.perf@y.values
df.tree = data.frame(warmer.tree.fpr, warmer.tree.tpr)

auc <- performance(warmer.tree.pred, measure = "auc")
cat("\n#AUC of Decision Tree\n")
print(auc@y.values[[1]])


#Naive bayes
warmer.bayes.pred = predict(warmer.bayes, warmer.test, type='raw')
warmer.bayes.pred = prediction(warmer.bayes.pred[, 2], warmer.test$WarmerTomorrow)
warmer.bayes.perf = performance(warmer.bayes.pred, "tpr", "fpr")

warmer.bayes.fpr = warmer.bayes.perf@x.values
warmer.bayes.tpr = warmer.bayes.perf@y.values
df.bayes = data.frame(warmer.bayes.fpr, warmer.bayes.tpr)

auc <- performance(warmer.bayes.pred, measure = "auc")
cat("\n#AUC of Naive Bayes\n")
print(auc@y.values[[1]])

#bagging
warmer.bag.pred = predict.bagging(warmer.bag, warmer.test)
warmer.bag.pred = prediction(warmer.bag.pred$prob[, 2], warmer.test$WarmerTomorrow)
warmer.bag.perf = performance(warmer.bag.pred, "tpr", "fpr")

warmer.bag.fpr = warmer.bag.perf@x.values
```

```
warmer.bag.tpr = warmer.bag.perf@y.values
df.bag = data.frame(warmer.bag.fpr, warmer.bag.tpr)

auc <- performance(warmer.bag.pred, measure = "auc")
cat("\n#AUC of Bagging\n")
print(auc@y.values[[1]])

#boosting
warmer.boost.pred = predict.boosting(warmer.boost, warmer.test)
warmer.boost.pred = prediction(warmer.boost.pred$prob[, 2], warmer.test$WarmerTomorrow)
warmer.boost.perf = performance(warmer.boost.pred, "tpr", "fpr")

warmer.boost.fpr = warmer.boost.perf@x.values
warmer.boost.tpr = warmer.boost.perf@y.values
df.boost = data.frame(warmer.boost.fpr, warmer.boost.tpr)

auc <- performance(warmer.boost.pred, measure = "auc")
cat("\n#AUC of Boosting\n")
print(auc@y.values[[1]])

#Random Forest
warmer.rf.pred = predict(warmer.rf, warmer.test, type='prob')
warmer.rf.pred = prediction(warmer.rf.pred[, 2], warmer.test$WarmerTomorrow)
warmer.rf.perf = performance(warmer.rf.pred, "tpr", "fpr")

warmer.rf.fpr = warmer.rf.perf@x.values
warmer.rf.tpr = warmer.rf.perf@y.values
df.rf = data.frame(warmer.rf.fpr, warmer.rf.tpr)

auc <- performance(warmer.rf.pred, measure = "auc")
cat("\n#AUC of Random Forest\n")
print(auc@y.values[[1]])

#Graph
library(ggplot2)

graph = ggplot() +
  geom_line(data=df.tree, aes(x=df.tree[[1]], y=df.tree[[2]], color = "Decision Tree"), size=0.9)+
  geom_line(data=df.bayes, aes(x=df.bayes[[1]], y=df.bayes[[2]], color = "Naive Bayes"), size=0.9)+
  geom_line(data=df.bag, aes(x=df.bag[[1]], y=df.bag[[2]], color = "Bagging"), size=0.9)+
  geom_line(data=df.boost, aes(x=df.boost[[1]], y=df.boost[[2]], color = "Boosting"), size=0.9)+
  geom_line(data=df.rf, aes(x=df.rf[[1]], y=df.rf[[2]], color = "Random Forest"), size=0.9)+
  xlab("")+
  ylab("")+
  geom_abline(intercept=0, slope=1, linetype="dashed", color="black", size=1)+
  ggtitle("")
graph
```

Question 7

No code

Question 8

```
#QUESION 8
#Attribute importance

#Decision Tree
cat("\n#Decision Tree Attribute Importance\n")
print(summary(warmer.tree))

#Naive Bayes
#e1071 package does not have importance method, so we use the caret package
library(caret)
cat("\n#Naive Bayes Attribute Importance\n")
Grid = data.frame(usekernel=TRUE,laplace = 0,adjust=1)

mdl = train(WarmerTomorrow ~ .,data=warmer.train,method="naive_bayes",
        trControl=trainControl(method="none"),
        tuneGrid=Grid)
```

```
varImp(mdl)

#Bagging
cat("\n#Baging Attribute Importance\n")
print(warmer.bag$importance[order(-warmer.bag$importance)])

#Boosting
cat("\n#Boosting Attribute Importance\n")
print(warmer.boost$importance[order(-warmer.boost$importance)])

#Random Forest
cat("\n#Random Forest Attribute Importance\n")
print(warmer.rf$importance[,1][order(-warmer.rf$importance)])

warmer.rf$importance + warmer.boost$importance
```

Question 9

```
#QUESTION 9
#our chosen model is a decision tree
simple.classifier = tree(WarmerTomorrow ~ Sunshine + MinTemp + Cloud9am,data=warmer.train)
plot(simple.classifier)
text(simple.classifier, pretty=0, cex=0.7)

#checking confusion matrix
simple.classifier.pred = predict(simple.classifier, warmer.test, type = "class")
t1=table(Predicted_Class = simple.classifier.pred, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Decision Tree Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))

#getting tpr and fpr and AUC
#decision tree
simple.classifier.pred = predict(simple.classifier, warmer.test, type = "vector")
simple.classifier.pred = prediction(simple.classifier.pred[, 2], warmer.test$WarmerTomorrow)
simple.classifier.perf = performance(simple.classifier.pred, "tpr", "fpr")

simple.classifier.fpr = simple.classifier.perf@x.values
simple.classifier.tpr = simple.classifier.perf@y.values
df.simple.classifier = data.frame(simple.classifier.fpr, simple.classifier.tpr)

auc <- performance(simple.classifier.pred, measure = "auc")
cat("\n#AUC of Decision Tree\n")
print(auc@y.values[[1]])

#GRAPH
graph = ggplot() +
  geom_line(data=df.tree, aes(x=df.tree[[1]], y=df.tree[[2]], color = "Decision Tree"), size=0.9)+
  geom_line(data=df.bayes, aes(x=df.bayes[[1]], y=df.bayes[[2]], color = "Naive Bayes"), size=0.9)+
  geom_line(data=df.bag, aes(x=df.bag[[1]], y=df.bag[[2]], color = "Bagging"), size=0.9)+
  geom_line(data=df.boost, aes(x=df.boost[[1]], y=df.boost[[2]], color = "Boosting"), size=0.9)+
  geom_line(data=df.rf, aes(x=df.rf[[1]], y=df.rf[[2]], color = "Random Forest"), size=0.9)+
  geom_line(data=df.simple.classifier, aes(x=df.simple.classifier[[1]], y=df.simple.classifier[[2]], color = "SIMPLE CLASSIFIER"), size= 1.5)+
  xlab("")+
  ylab("")+
  geom_abline(intercept=0, slope=1, linetype="dashed", color="black", size=1)+
  ggtitle("")
graph
```

Question 10

```
#QUESTION 10
#Since there are less variables, less rows are removed due to NAs
warmerdata = WAUS[, c(9, 5, 8, 23, 17, 20, 24)]
warmerdata = warmerdata[complete.cases(warmerdata),]
warmerdata$WarmerTomorrow = as.factor(warmerdata$WarmerTomorrow)
set.seed(31486347) #Student ID as random seed
train.row = sample(1:nrow(warmerdata), 0.7*nrow(warmerdata))
```

```
warmer.train = warmerdata[train.row,]
warmer.test = warmerdata[-train.row,]

#I chose to use a random forest model with a 1000 tree (default is 500)

best.classifier <- randomForest(WarmerTomorrow ~ Sunshine + MinTemp + Evaporation + Temp3pm + Humidity3pm + Cloud9am, data=warmer.train,
na.action = na.exclude, ntree=1000)

#get accuracy
best.classifier.pred = predict(best.classifier, warmer.test)
t1=table(Predicted_Class = best.classifier.pred, Actual_Class = warmer.test$WarmerTomorrow)

cat("\n#Random Forest Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))


#get AUC
best.classifier.pred = predict(best.classifier, warmer.test, type='prob')
best.classifier.pred = prediction(best.classifier.pred[, 2], warmer.test$WarmerTomorrow)
best.classifier.perf = performance(best.classifier.pred, "tpr", "fpr")

best.classifier.fpr = best.classifier.perf@x.values
best.classifier.tpr = best.classifier.perf@y.values
df.best.classifier = data.frame(best.classifier.fpr, best.classifier.tpr)

auc <- performance(best.classifier.pred, measure = "auc")
cat("\n#AUC of Random Forest\n")
print(auc@y.values[[1]])

graph = ggplot() +
  geom_line(data=df.tree, aes(x=df.tree[[1]], y=df.tree[[2]], color = "Decision Tree"), size=0.9)+
  geom_line(data=df.bayes, aes(x=df.bayes[[1]], y=df.bayes[[2]], color = "Naive Bayes"), size=0.9)+
  geom_line(data=df.bag, aes(x=df.bag[[1]], y=df.bag[[2]], color = "Bagging"), size=0.9)+
  geom_line(data=df.boost, aes(x=df.boost[[1]], y=df.boost[[2]], color = "Boosting"), size=0.9)+
  geom_line(data=df.rf, aes(x=df.rf[[1]], y=df.rf[[2]], color = "BEST CLASSIFIER"), size=1.5)+
  geom_line(data=df.best.classifier, aes(x=df.best.classifier[[1]], y=df.best.classifier[[2]], color = "Random Forest"), size= 0.9)+
  xlab("")+
  ylab("")+
  geom_abline(intercept=0, slope=1, linetype="dashed", color="black", size=1)+
  ggtitle("")
graph
```

Question 11

```
#QUESTION 11
library(neuralnet)

warmerdata = WAUS[, c(5:9, 11, 14:24)]
warmerdata = warmerdata[complete.cases(warmerdata),]
warmerdata$WarmerTomorrow = as.numeric(warmerdata$WarmerTomorrow)

str(warmerdata)

set.seed(31486347) #Student ID as random seed
train.row = sample(1:nrow(warmerdata), 0.7*nrow(warmerdata))
warmer.train = warmerdata[train.row,]
warmer.test = warmerdata[-train.row,]

#train neural net
warmer.nn = neuralnet(WarmerTomorrow ~ Sunshine + MinTemp + Evaporation + Temp3pm + Humidity3pm + Cloud9am, warmer.train, hidden=5)

#get predictions
warmer.nn.pred = compute(warmer.nn, warmer.test[, c(5, 1, 4, 16, 10, 13, 17)] )

#round results to integers
warmer.nn.pred = as.data.frame(round(warmer.nn.pred$net.result,0))

# plot confusion matrix
t1 = table(observed = warmer.test$WarmerTomorrow, predicted = warmer.nn.pred$V1)
```

```
cat("\n#Neural Network Confusion\n")
print(t1)

cat("\n#Accuracy\n")
print((t1[1,1] + t1[2,2]) / (t1[1,1]+t1[1,2]+t1[2,1]+t1[2,2]))
```