1. Load the data into R and name the columns to better identify the board (following an ordering from left to right and from top to bottom). Check for missing values.

```
> column_names <- c(
+   "top-left", "top-middle", "top-right",
+   "middle-left", "middle-middle", "middle-right",
+   "bottom-left", "bottom-middle", "bottom-right",
+   "Class"
+ )
> ttt_data <- read.csv(file.choose(), header = FALSE, col.names = column_names)
>
> missing_values_count <- sum(ttt_data == 'b', na.rm = TRUE)
> missing_values_count
[1] 1980

> missing_values <- sum(is.na(ttt_data))
> missing_values
[1] 0
```

There are 1980 empty values in the dataset.

2. Read the "data splitting" section on the caret web page. Then split the data into 70% training and 30% test by keeping the original class proportion (check “createDataPartition()” function).

```
> trainIndex <- createDataPartition(ttt_data$Class, p = 0.7, list = FALSE)
> train_data <- ttt_data[trainIndex, ]
> test_data <- ttt_data[-trainIndex, ]
> nrow(ttt_data)
[1] 958
> nrow(train_data)
[1] 672
> nrow(test_data)
[1] 286
> table(train_data$Class)

negative positive
     233      439
> table(test_data$Class)

negative positive
      99      187
> table(ttt_data$Class)

negative positive
     332      626
```

3. Complete the following table with the final values of accuracy and kappa used for the models

Model	Accuracy	Kappa
Naive Bayes	0.7067982	0.3374644
Decision Tree	0.7684211	0.3949045
Neural Network	0.9843750	0.9649460
Nearest Neighbor	0.9894737	0.9766193
SVM	0.9791667	0.9538462

```
set.seed(42)

ctrl <- trainControl(method = "cv", classProbs = TRUE)

> model_nb <- train(  Class ~ .,
+ data = ttt_data,
+ method = "naive_bayes",
+ trControl = ctrl)
>
> model_dt <- train(  Class ~ .,
+ data = ttt_data,
+ method = "rpart",
+ trControl = ctrl)
>
> model_nn <- train(  Class ~ .,
+ data = ttt_data,
+ method = "nnet",
+ trControl = ctrl);

> model_knn <- train(  Class ~ .,
+ data = ttt_data,
+ method = "knn",
+ trControl = ctrl)
>
> model_svm <- train(  Class ~ .,
+ data = ttt_data,
+ method = "svmLinear",
+ trControl = ctrl)

> models <- list(
+   Naive_Bayes = model_nb,
+   Decision_Tree = model_dt,
+   Neural_Network = model_nn,
+   Nearest_Neighbor = model_knn,
+   SVM_Linear = model_svm
+ )
>
> # Collect the resampling results
> resamp <- resamples(models)
>
> summary(resamp)

Call:
summary.resamples(object = resamp)

Models: Naive_Bayes, Decision_Tree, Neural_Network, Nearest_Neighbor, SVM_Linear
Number of resamples: 10

Accuracy
                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Naive_Bayes      0.5473684 0.6883054 0.7067982 0.7032990 0.7409116 0.8229167    0
Decision_Tree    0.7473684 0.7610341 0.7684211 0.7755271 0.7870490 0.8229167    0
Neural_Network   0.9684211 0.9791667 0.9843750 0.9833004 0.9895833 1.0000000    0
Nearest_Neighbor 0.9789474 0.9791667 0.9894737 0.9885085 0.9974227 1.0000000    0
SVM_Linear       0.9583333 0.9791667 0.9791667 0.9833114 0.9895833 1.0000000    0

Kappa
                        Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Naive_Bayes      0.008735744 0.2830710 0.3374644 0.3339290 0.4243095 0.5866261    0
Decision_Tree    0.328621908 0.3835467 0.3949045 0.4194698 0.4489535 0.5526316    0
Neural_Network   0.928838951 0.9531479 0.9649460 0.9625129 0.9769941 1.0000000    0
Nearest_Neighbor 0.952900347 0.9531479 0.9766183 0.9742784 0.9943008 1.0000000    0
SVM_Linear       0.904903418 0.9531479 0.9538462 0.9625280 0.9767442 1.0000000    0
```

4. Read the "model performance" section on the caret web page. Apply the models to the test dataset. Print the confusion matrix of each model and observe the information it provides.

```
> test_data$Class <- factor(test_data$Class, levels =  unique(pred_nb))
>
>
> pred_nb <- predict(model_nb, newdata = test_data)
> pred_dt <- predict(model_dt, newdata = test_data)
> pred_nn <- predict(model_nn, newdata = test_data)
> pred_knn <- predict(model_knn, newdata = test_data)
> pred_svm <- predict(model_svm, newdata = test_data)
>
> cm_nb <- confusionMatrix(data = pred_nb, reference = test_data$Class)
> cm_dt <- confusionMatrix(data = pred_dt, reference = test_data$Class)
> cm_nn <- confusionMatrix(data = pred_nn, reference = test_data$Class)
> cm_knn <- confusionMatrix(data = pred_knn, reference = test_data$Class)
> cm_svm <- confusionMatrix(data = pred_svm, reference = test_data$Class)
> |
```

```
> cm_nb
Confusion Matrix and Statistics

          Reference
Prediction negative positive
  negative       51       38
  positive       48      149

               Accuracy : 0.6993
                 95% CI : (0.6425, 0.7519)
    No Information Rate : 0.6538
    P-Value [Acc > NIR] : 0.05895

                  Kappa : 0.3195

 Mcnemar's Test P-Value : 0.33180

            Sensitivity : 0.5152
            Specificity : 0.7968
         Pos Pred Value : 0.5730
         Neg Pred Value : 0.7563
             Prevalence : 0.3462
         Detection Rate : 0.1783
   Detection Prevalence : 0.3112
      Balanced Accuracy : 0.6560

       'Positive' Class : negative
```

```
> cm_dt
Confusion Matrix and Statistics

          Reference
Prediction negative positive
  negative       28        0
  positive       71      187

               Accuracy : 0.7517
                 95% CI : (0.6975, 0.8007)
    No Information Rate : 0.6538
    P-Value [Acc > NIR] : 0.0002301

                  Kappa : 0.3402

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.2828
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.7248
             Prevalence : 0.3462
         Detection Rate : 0.0979
   Detection Prevalence : 0.0979
      Balanced Accuracy : 0.6414

       'Positive' Class : negative
```

```
> cm_nn
Confusion Matrix and Statistics

          Reference
Prediction negative positive
  negative       91        0
  positive        8      187

               Accuracy : 0.972
                 95% CI : (0.9456, 0.9878)
    No Information Rate : 0.6538
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.937

 Mcnemar's Test P-Value : 0.01333

            Sensitivity : 0.9192
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9590
             Prevalence : 0.3462
         Detection Rate : 0.3182
   Detection Prevalence : 0.3182
      Balanced Accuracy : 0.9596

       'Positive' Class : negative
```

```
> cm_knn
Confusion Matrix and Statistics

          Reference
Prediction negative positive
  negative       99        0
  positive        0      187

               Accuracy : 1
                 95% CI : (0.9872, 1)
    No Information Rate : 0.6538
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.3462
         Detection Rate : 0.3462
   Detection Prevalence : 0.3462
      Balanced Accuracy : 1.0000

       'Positive' Class : negative
```

```
> cm_svm
Confusion Matrix and Statistics

          Reference
Prediction negative positive
  negative       91        0
  positive        8      187

               Accuracy : 0.972
                 95% CI : (0.9456, 0.9878)
    No Information Rate : 0.6538
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.937

 Mcnemar's Test P-Value : 0.01333

            Sensitivity : 0.9192
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9590
             Prevalence : 0.3462
         Detection Rate : 0.3182
   Detection Prevalence : 0.3182
      Balanced Accuracy : 0.9596

       'Positive' Class : negative
```

Construct a table (similar to the previous one) with the accuracy and the kappa values obtained by each one (to do this you can also use the postResample function). Add the AUC value to the table. It can be calculated using the AUC package.

```
> resamp_nb <- postResample(pred_nb,test_data$Class)
> resamp_dt <- postResample(pred_dt,test_data$Class)
> resamp_nn <- postResample(pred_nn,test_data$Class)
> resamp_knn <- postResample(pred_knn,test_data$Class)
> resamp_svm <- postResample(pred_svm,test_data$Class)
>

> auc_nb <- auc(roc(pred_nb,test_data$Class))
> auc_dt <- auc(roc(pred_dt,test_data$Class))
> auc_nn <- auc(roc(pred_nn,test_data$Class))
> auc_knn <- auc(roc(pred_knn,test_data$Class))
> auc_svm <- auc(roc(pred_svm,test_data$Class))
>
> results_table <- data.frame(
+   Model = c("Naive_Bayes", "Decision_Tree", "Neural_Network", "Nearest_Neighbor", "SVM_Linear"),
+   Accuracy = c(resamp_nb[1], resamp_dt[1], resamp_nn[1], resamp_knn[1], resamp_svm[1]),
+   Kappa = c(resamp_nb[2], resamp_dt[2], resamp_nn[2], resamp_knn[2], resamp_svm[2]),
+   AUC = c(auc_nb, auc_dt, auc_nn, auc_knn, auc_svm)
+ )
```

```
> results_table
             Model  Accuracy     Kappa       AUC
1      Naive_Bayes 0.6993007 0.3195374 0.6559715
2    Decision_Tree 0.7517483 0.3402430 0.6414141
3   Neural_Network 0.9720280 0.9370079 0.9595960
4 Nearest_Neighbor 1.0000000 1.0000000 1.0000000
5       SVM_Linear 0.9720280 0.9370079 0.9595960
```

5. Plot the ROC curves of the models. We are going to use the ROCR package:

a. First, we calculate again the predictions on the test set but now setting the "type" parameter of the predict function to "prob".

```
> pred_prob_nb <- predict(model_nb, newdata = test_data, type = "prob")
> pred_prob_dt <- predict(model_dt, newdata = test_data, type = "prob")
> pred_prob_nn <- predict(model_nn, newdata = test_data, type = "prob")
> pred_prob_knn <- predict(model_knn, newdata = test_data, type = "prob")
> pred_prob_svm <- predict(model_svm, newdata = test_data, type = "prob")
```

b. We construct a "prediction()" object for each classifier using the vector of estimated probabilities for the positive class as the first parameter (calculated above), and the vector of actual class labels as the second parameter.

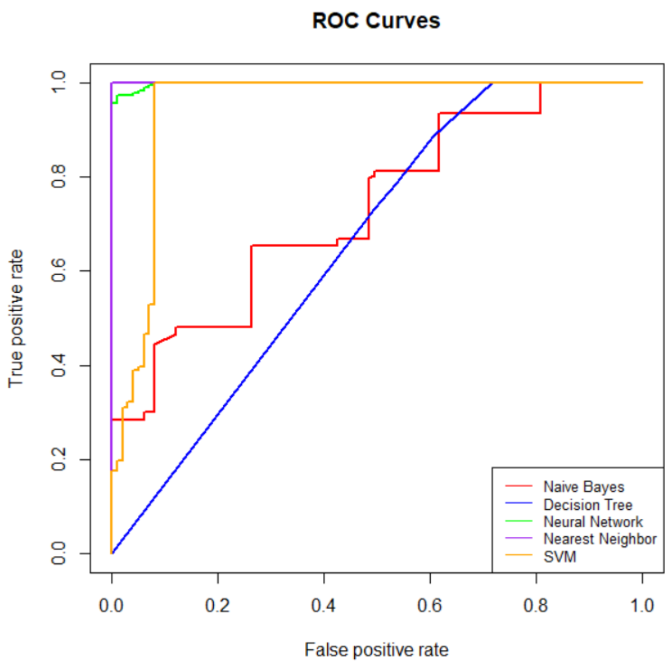
```
> calculate_tpr_fpr <- function(pred_obj) {
+   perf <- performance(pred_obj, "tpr", "fpr")
+   return(perf)
+ }
>
> perf_nb <- calculate_tpr_fpr(pred_obj_nb)
> perf_dt <- calculate_tpr_fpr(pred_obj_dt)
> perf_nn <- calculate_tpr_fpr(pred_obj_nn)
> perf_knn <- calculate_tpr_fpr(pred_obj_knn)
> perf_svm <- calculate_tpr_fpr(pred_obj_svm)
```

c. We calculate the measures we want to plot on the y-axis (TPR) and on the x-axis (FPR) by using the “performance()” function, which also takes the “prediction()” object computed above as a first parameter.

```
> plot(perf_nb, col = "red", main = "ROC Curves", lwd = 2)
> plot(perf_dt, col = "blue", add = TRUE, lwd = 2)
> plot(perf_nn, col = "green", add = TRUE, lwd = 2)
> plot(perf_knn, col = "purple", add = TRUE, lwd = 2)
> plot(perf_svm, col = "orange", add = TRUE, lwd = 2)
```

d. Draw all the curves in the same plot using different colours. Add a legend.

```
> legend("bottomright", legend = c("Naive Bayes", "Decision Tree", "Neural Network", "Nearest Neighbor", "SVM"),
+        col = c("red", "blue", "green", "purple", "orange"), lty = 1, cex = 0.8)
```



Which models should you keep, and which models should you discard, according to the ROC curves?

The best model would be the Nearest Neighbor followed by the Neural Network, the worst performing model is the Decision tree and the Naïve Bayes.