# SVM Model Testing

This section begins by loading the training and test data. The svm_resilts table saved at the end of the 'NC_SupportVectorMachine_Methodology.m' script. The parameters that produced the model with the best parameters following bayesian optimisation and grid search are retrieved and the whole training set is trained using these parameters. The resulting model is used to make predictions on the test data, and this is compared to ground truth. Accuracy is assessed and cofusion matrix and ROC are plotted.

## 1. Load Data

```
%The training and test data are loaded

load('Training.mat');
load('Testing.mat');
```

## 2. Train and Test Model

Using the results table saved at the end of

```
%'NC_SupportVectorMachine_Methodology', the parameters that result in the
%best average 10-fold accuracy are retrieved. These are printed.

%load SVM_Results Table
load('SVM_Results.mat')

%find row with best accuracy, and the index, from results table
[best_acc, best_acc_idx] = max(SVM_Results.Avg_Accuracy);

%get specific hyperparameters from best row using the index
best_parameters = SVM_Results{best_acc_idx,:}
best_kernel = best_parameters(1);
best_scale = str2num(best_parameters(3));
best_boxc = str2num(best_parameters(2));

%print the optimum SVM hyperparameters
fprintf('Best Parameters: \n')
```

```
Best Parameters:
```

```
fprintf('Kernel: %s \n',best_kernel)
```

```
Kernel: gaussian
```

```
fprintf('Scale: %2f \n',best_scale)
```

```
Scale: 0.317850
```

```
fprintf('Box Constraint: %2f \n\n',best_boxc)
```

```
Box Constraint: 1181.575900
```

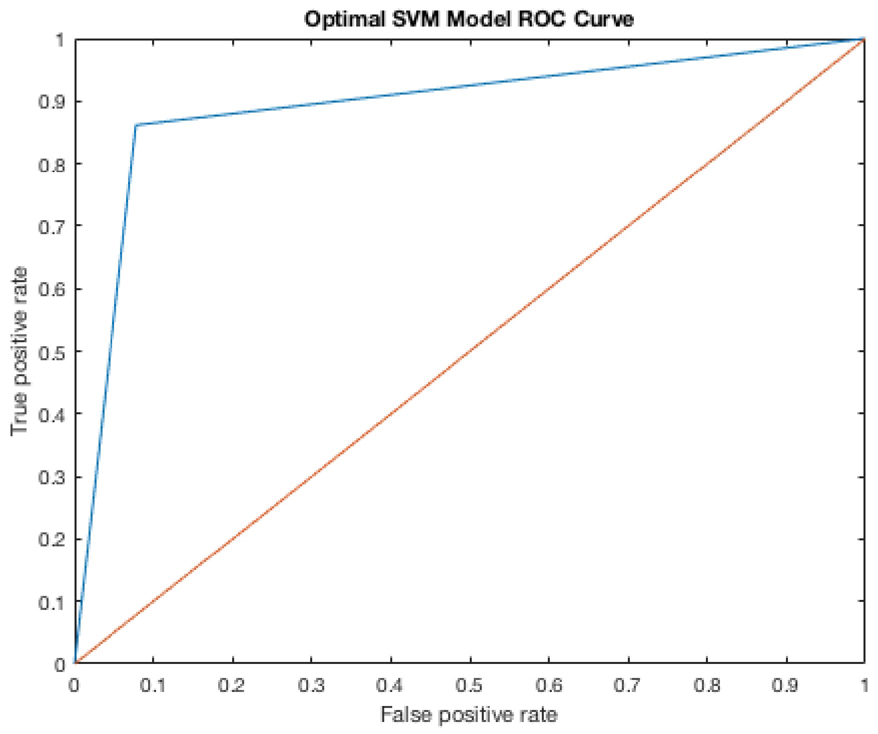
## 3. Train Model

Using the best parameters, a model is fitted to all of the training data and test predictions are generated which are compared against the actual outputs.

```
tic %start times
%use best parameters to fit optimal model on all training data
Best_SVMModel = fitcsvm(Training(:, 1:42),Training(:, 43),'Standardize',true,'KernelFunction',best_kerne
    'KernelScale',best_scale, 'BoxConstraint', best_boxc);
Best_SVMModel_Time = toc; %save time

%fit model to test instances to get predictions on unseen test data
predictions = predict(Best_SVMModel, Testing(:, 1:42));

%compare predictions to test to get accuracy
test_accuracy = sum(predictions == Testing(:, 43))/length(Testing(:, 43))*100;

%print results
fprintf('Test Accuracy: %2f',test_accuracy)
fprintf('\nTime taken: %2f seconds \n',Best_SVMModel_Time)
```

## 4. ROC Curve

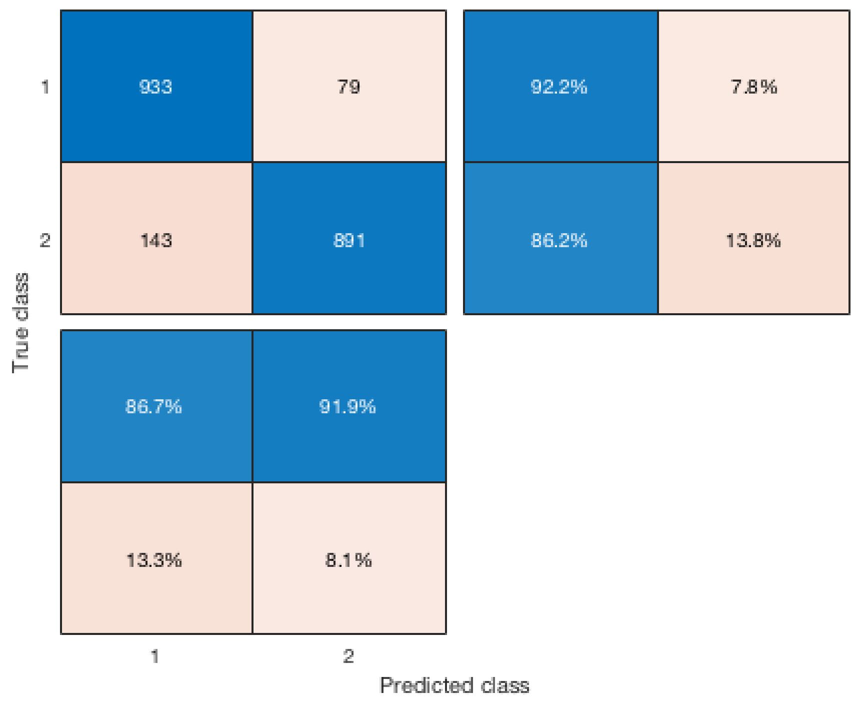
```
%This section plots a ROC curve to compare performance of true positive
%rate and false positive rate

%ROC curve
f2 = figure;
[TP,FP,T,AUC] = perfcurve(Testing(:, 43) , predictions, '1');
plot(TP,FP, [0 1],[0 1])
xlabel('False positive rate')
ylabel('True positive rate')
title('Optimal SVM Model ROC Curve')
```



## 3. Confusion Matrix

A confusion matrix can be polotted to get info on TP, TN, FP and FN. From these, improtant metrics are derived to assess model performance.

```
f1 = figure;
%create confusion matrix
C = confusionmat(Testing(:, 43), predictions);
%plot chart from confusion matrix
D = confusionchart(C,'RowSummary','row-normalized','ColumnSummary','column-normalized');
```



```
accuracy = (C(1,1) + C(2,2)) / (C(1,1) + C(1, 2) + +C(2, 1) + C(2,2));
precision = C(1,1) / (C(1,1) + C(1,2));
sensitivity = (C(1,1)) / (C(1, 1)+C(2, 1));
specificity = (C(2,2)) / (C(2, 2)+C(1, 2));
f1_score = 2 * ((precision * sensitivity)/(precision + sensitivity));

fprintf('\nAccuracy: %2f \n',accuracy) %proportion of correct predictions
```

```
Accuracy: 0.891496
```

```
fprintf('Precision: %2f \n',precision) %proportion of positive results that were correctly classified
```

```
Precision: 0.921937
```

```
fprintf('Sensitivity: %2f \n',sensitivity) %amount of customers leaving that were correctly idenified
```

```
Sensitivity: 0.867100
```

```
fprintf('Specificity: %2f \n',specificity) %amount of customers staying that were correctly identified
```

```
Specificity: 0.918557
```

```
fprintf('F1: %2f \n',f1_score)
```

```
F1: 0.893678
```

```
fprintf('AUC: %2f \n',AUC)
```

```
AUC: 0.891819
```