SVM Digits Recogniser

SVM Case Study

Problem statement:

This problem is about the pattern recognition of the handwritten digits(0-9) recognition.

Goal of the case study:

The goal is to develop a model that can correctly identify the digit (between 0-9) written in an image

Problem solving methodology:

1. Importing the test and training datasets
2. Data preparation regarding the Missing values, outliers, blank values for both training and test sets
3. High level visualisations to print the label data
4. Apply dimensionality reduction technique aka PCA for both train and test set
5. Model building using KSVM algorithm for linear models using C=1 and C=10 , and then using the Vanilla kernel and RBF kernel
6. Note the different outputs and tuning parameter thresholds
7. Execute the cross validation using 3 folds and with sigma as 0.025 and 0.009 using RBF kernel. This is performed on sampled dataset to achieve a better computation time
8. Determine the final model

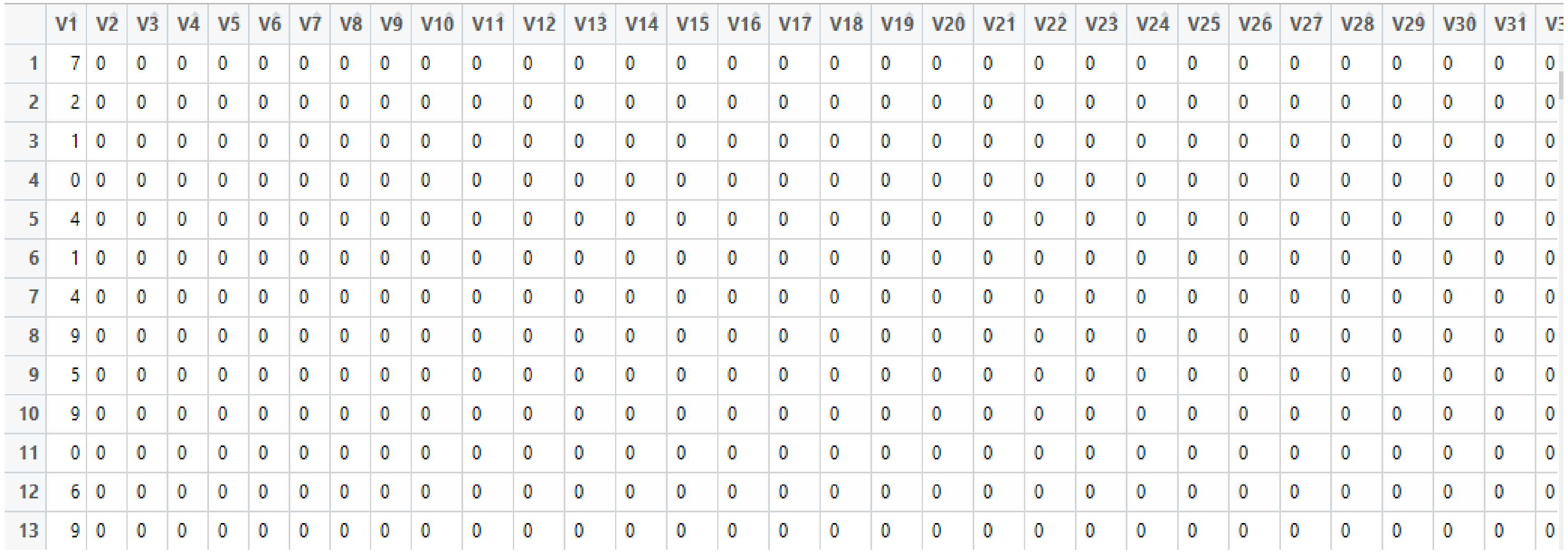
Data Understanding

UpGrad

Goal of the case study:

The goal is to develop a model that can correctly identify the digit (between 0-9) written in an image

# Number of Instances and attributes in training set: 59999 of 785 variables
# Number of Instances and attributes in testing set: 9999 of 785 variables

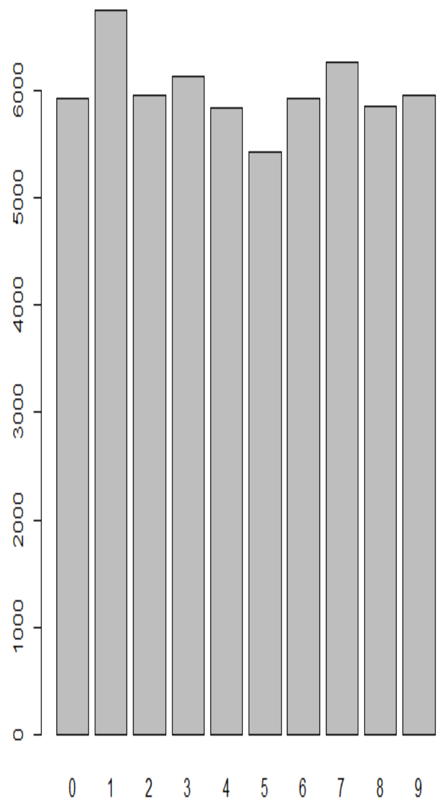


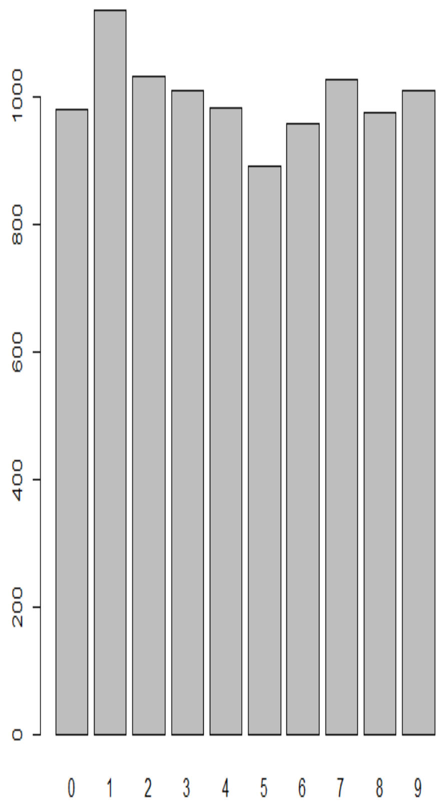
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	V29	V30	V31	V3
1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Data Understanding

UpGrad

1.Check for missing values and blanks 2. Check for Outliers   3. Visualisations

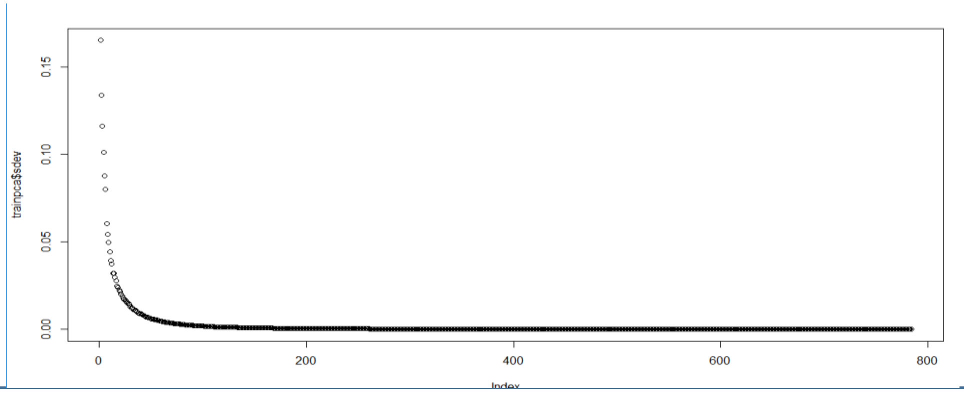


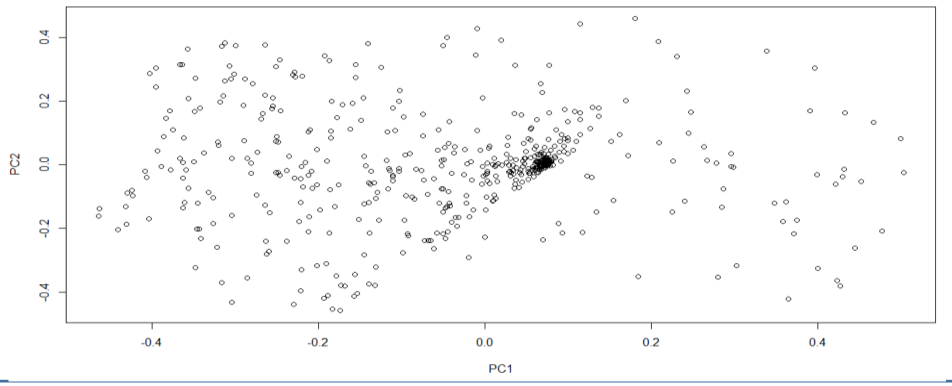


Data Preparation

1. Dimensionality of the data set and its reduction
2. Why PCA
3. Code for PCA:

```
label_total <- as.factor(train_Data[[1]])          #converting the class column to factor type
trainreduced <- train_Data[,2:785]/255  #Normalising the data , i.e convert the pixel range from 0-255 to 0-1
traincov <- cov(trainreduced)          # Applying co-variance
trainpca <- prcomp(traincov)          # Applying the PCA via the prcomp method
plot(trainpca$sdev)                                        #Visualising the SD to view the top predictors
plot(trainpca$x)                                           # Visualising the x , this shows the distibution of the principal
components and the distribution is elliptiical (Non-linear)
trainext <- as.matrix(trainreduced) %*% trainpca$rotation[,1:60]   #Since the PCA will be aligned in 2D, i.e convertng the
wide format to long format, transposing it again for matrix multiplication
trainFinal <- data.frame(label_total,trainext)#Getting the final dataframe on Training set
```





Data Modelling

#Linear model, Model1: With C=1,Accuracy : 0.9783

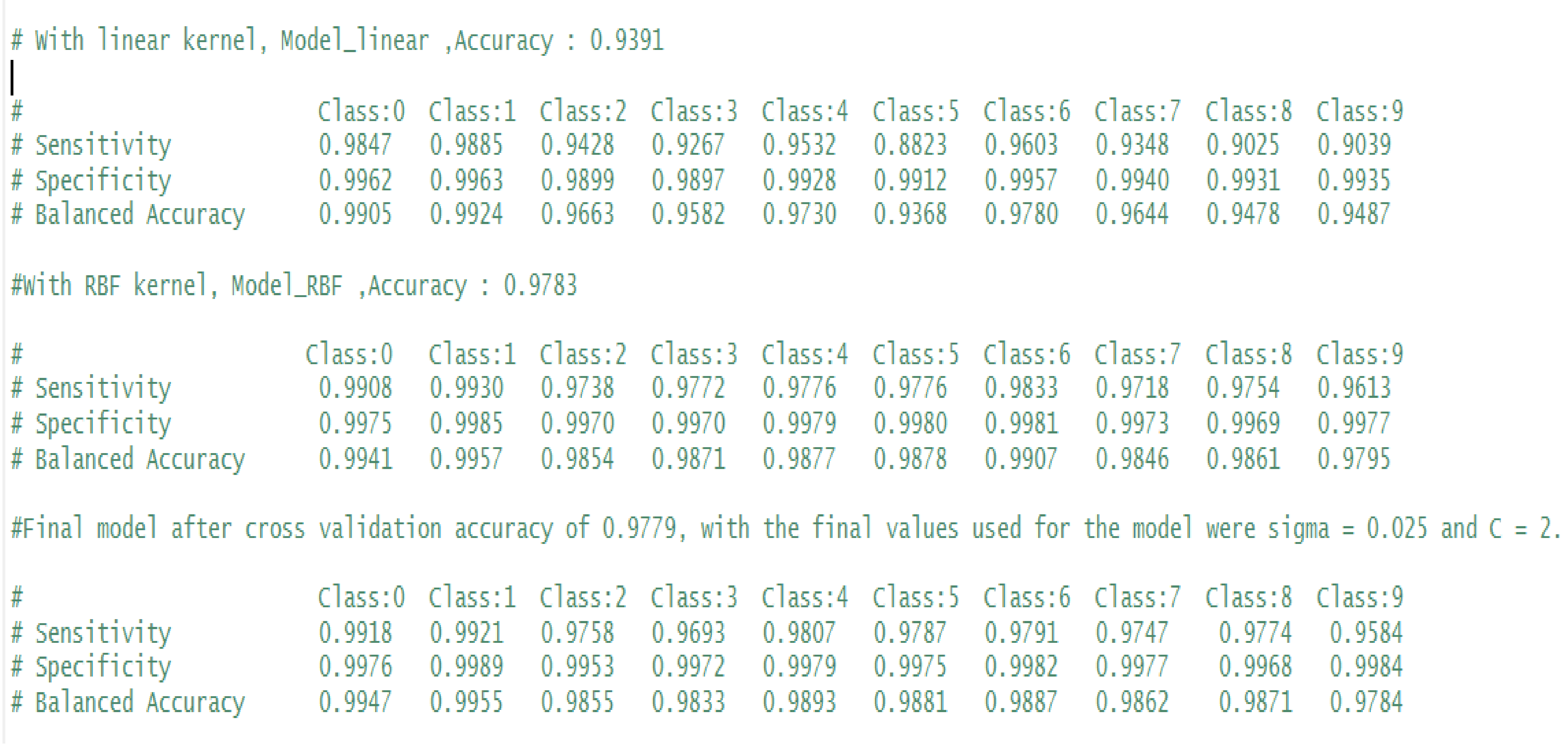
#	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
# Specificity	0.9908	0.993	0.9738	0.9772	0.9776	0.9776	0.9833	0.9718	0.9754	0.9613
# Sensitivity	0.9975	0.9985	0.997	0.997	0.9979	0.998	0.9981	0.9973	0.9969	0.9977
# Bal Acc	0.9941	0.9957	0.9854	0.9871	0.9877	0.9878	0.9907	0.9846	0.9861	0.9795

#Linear model, Model2: With C=10,Accuracy : 0.9839 [Even though it looks like a highly accurate model, this also poses a risk of overfitting the data]

#	Class: 0	Class: 1	Class: 2	Class: 3	Class: 4	Class: 5	Class: 6	Class: 7	Class: 8	Class: 9
# Sensitivity	0.9939	0.9938	0.9806	0.9842	0.9847	0.9832	0.9864	0.9786	0.9825	0.9703
# Specificity	0.9981	0.9993	0.9975	0.9983	0.9982	0.9979	0.9987	0.9981	0.9978	0.9981
# Bal acc	0.996	0.9966	0.9891	0.9912	0.9915	0.9905	0.9926	0.9884	0.9902	0.9842

Data Modelling

UpGrad



```
# With linear kernel, Model_linear ,Accuracy : 0.9391

#                       Class:0  Class:1  Class:2  Class:3  Class:4  Class:5  Class:6  Class:7  Class:8  Class:9
# Sensitivity           0.9847   0.9885   0.9428   0.9267   0.9532   0.8823   0.9603   0.9348   0.9025   0.9039
# Specificity           0.9962   0.9963   0.9899   0.9897   0.9928   0.9912   0.9957   0.9940   0.9931   0.9935
# Balanced Accuracy     0.9905   0.9924   0.9663   0.9582   0.9730   0.9368   0.9780   0.9644   0.9478   0.9487

#With RBF kernel, Model_RBF ,Accuracy : 0.9783

#                      Class:0   Class:1  Class:2  Class:3  Class:4  Class:5  Class:6  Class:7  Class:8  Class:9
# Sensitivity           0.9908   0.9930   0.9738   0.9772   0.9776   0.9776   0.9833   0.9718   0.9754   0.9613
# Specificity           0.9975   0.9985   0.9970   0.9970   0.9979   0.9980   0.9981   0.9973   0.9969   0.9977
# Balanced Accuracy     0.9941   0.9957   0.9854   0.9871   0.9877   0.9878   0.9907   0.9846   0.9861   0.9795

#Final model after cross validation accuracy of 0.9779, with the final values used for the model were sigma = 0.025 and C = 2.

#                       Class:0  Class:1  Class:2  Class:3  Class:4  Class:5  Class:6  Class:7  Class:8  Class:9
# Sensitivity           0.9918   0.9921   0.9758   0.9693   0.9807   0.9787   0.9791   0.9747    0.9774   0.9584
# Specificity           0.9976   0.9989   0.9953   0.9972   0.9979   0.9975   0.9982   0.9977    0.9968   0.9984
# Balanced Accuracy     0.9947   0.9955   0.9855   0.9833   0.9893   0.9881   0.9887   0.9862    0.9871   0.9784
```