STAT3006 Assignment 3

Ruyun Qi (44506065)

October 26, 2020

Task 1

1. Linear Discriminant Analysis

When dealing with more than one predictor variable, the LDA classifier assumes that the observations in the kth class are drawn from a multivariate Gaussian distribution $N(\mu_k, \Sigma)$, where μ_k is the class-specific mean vector and Σ is the common covariance matrix for all classes. In addition, we introduce $\hat{\pi}_k$ as the prior probability of observations in kth class.

First we have the Gaussian density function:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)}$$

We can derive the discriminant boundary between class j and class l:

$$\frac{\hat{\pi}_j}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}} e^{-\frac{1}{2}(x-\hat{\mu}_j)^T \hat{\Sigma}^{-1}(x-\hat{\mu}_j)} = \frac{\hat{\pi}_l}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}} e^{-\frac{1}{2}(x-\hat{\mu}_l)^T \hat{\Sigma}^{-1}(x-\hat{\mu}_l)}$$

Then, canceling common terms and taking logs of both sides, we have:

$$\log \frac{\hat{\pi}_j}{\hat{\pi}_l} = x^T \hat{\Sigma}^{-1} (\hat{\mu}_l - \hat{\mu}_j) + \frac{1}{2} \left[\hat{\mu}_j^T \hat{\Sigma}^{-1} \hat{\mu}_j - \hat{\mu}_l^T \hat{\Sigma}^{-1} \hat{\mu}_l \right]$$

Changing the positions of partial terms, we have:

$$x^T\hat{\Sigma}^{-1}\left(\hat{\mu}_l-\hat{\mu}_j\right)=\log\frac{\hat{\pi}_j}{\hat{\pi}_l}+\frac{1}{2}\left[\hat{\mu}_l^T\hat{\Sigma}^{-1}\hat{\mu}_l-\hat{\mu}_j^T\hat{\Sigma}^{-1}\hat{\mu}_j\right]$$

Discriminant function:

$$\hat{\delta}_k(x)=x^T\mathbf{\Sigma}^{-1}\hat{\mu}_k-\frac{1}{2}\hat{\mu}_k^T\mathbf{\Sigma}^{-1}\hat{\mu}_k+\log\hat{\pi}_k$$

where $\hat{\delta}_k$ is the largest discriminant score.

Estimating the parameters:

$$\hat{\pi}_k=\frac{n_k}{n}$$

$$\hat{\mu}_k=\frac{1}{n_k}\sum_{i:y_i=k}x_i$$

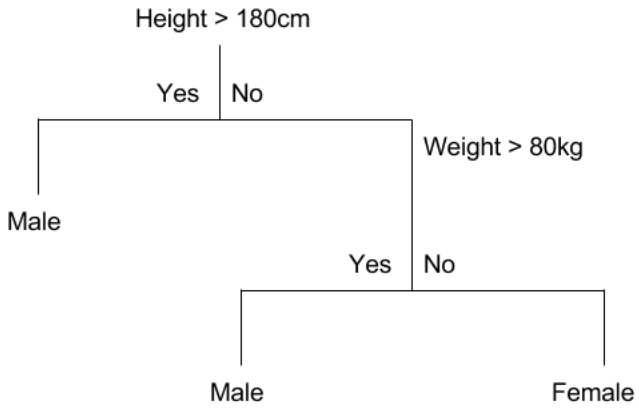
$$\hat{\Sigma}=\frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}\left(x_i-\hat{\mu}_k\right)^2=\sum_{k=1}^{K}\frac{n_k-1}{n-K}\cdot\hat{\Sigma}_k$$

$$\text{where }\ \hat{\Sigma}_k=\frac{1}{n_k-1}\sum_{i:y_i=k}\left(x_i-\hat{\mu}_k\right)^2$$

2. Classification Trees

Decision tree generally gives the outcomes by several 'yes-or-no' tests. It splits the data into partitions, and then splitting it further on each branch. The node is the test for the certain attributes of observations. The branch is the outcome of the test expanding to its further node or leaf. The leaf node is a terminal node which is the prediction.

Figure 1: Example of a decision tree



In Figure 1, we can see the first node is to test whether the height is over 180cm. Its branch of no connects with a leaf node of male and the second node to test whether the weight is over 80kg. It is obvious that a observation will be classified as female if its height is shorter than 180cm and weight is less than 80kg.

In classification tree, we start at the top and split the data on the variable which gives the largest information gain, and repeat this splitting procedure at each child node until the leaves are pure. There are 3 popular ways of measuring the node purity.

Suppose there are G possible classes, then at the mth node:

$$\text{Cross-entropy} = I_C(m) = -\sum_{g=1}^{G} \hat{p}_{mg} \log \hat{p}_{mg}$$

$$\text{Gini index} = I_G(m) = \sum_{g=1}^{G} \hat{p}_{mg}(1 - \hat{p}_{mg})$$

$$\text{Misclassification rate} = I_M(m) = 1 - \hat{p}_{mg*}$$

where $\hat{p}_{mg}$ is the proportion of the training observations in class g out of those which pass through the decision tree to the mth node, and $g^* = \arg\max_g \hat{p}_{mg}$.

Task 2

Question (a):

1. LDA

```
Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

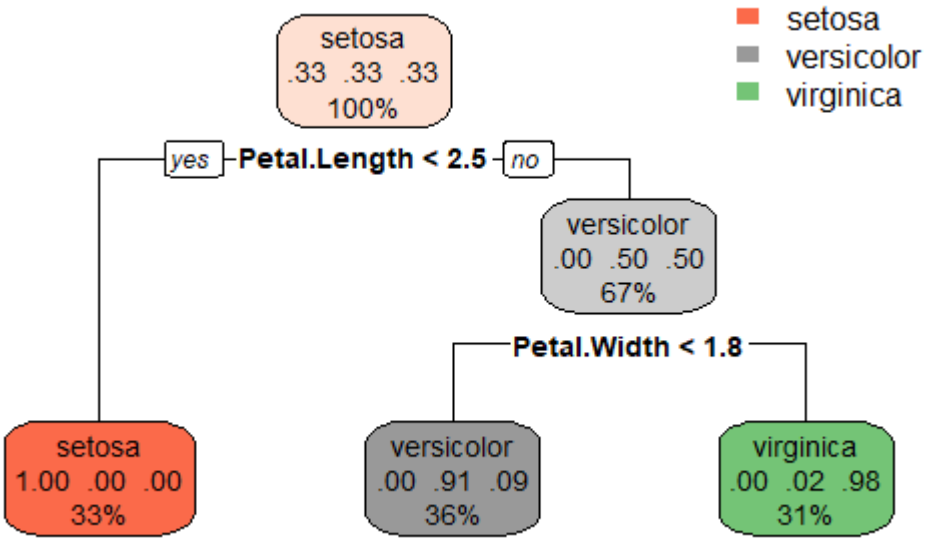
The prior probability of each class is $\frac{1}{3}$, and we have the means of all 4 variables for each class. LDA fitting forms the linear discriminants for 4 variables to classify the observations.

$$\text{LDA1} = 0.8293776 \times \text{Sepal.Length} + 1.5344731 \times \text{Sepal.Width} - 2.2012117 \times \text{Petal.Length} + 2.83918785 \times \text{Petal.Width}$$

$$\text{LDA2} = 0.02410215 \times \text{Sepal.Length} + 2.16452123 \times \text{Sepal.Width} - 0.93192121 \times \text{Petal.Length} - 2.8104603 \times \text{Petal.Width}$$

The first discriminant function have 99.12% separations which is better than the second first discriminant function with 0.88% separations.

2. Classification tree



The prior probability of each class is $\frac{1}{3}$.

The first node splits the observations by testing their Petal.Length. If the Petal.Length < 0.25, then the observation is classified as sentosa.

The second node tests the remaining 67% observations which can be only classified as versicolor or virginica. If the Petal.Width < 1.8, the observation

is classified as versicolor otherwise virginica.
The result shows that 33% of the observations are classified as setosa, 36% as versicolor and 31% as virginica.

Question (b):

	APER
setosa	0.00
versicolor	0.04
virginica	0.02
overall	0.02

Table 1: Apparent Error Estimate for LDA

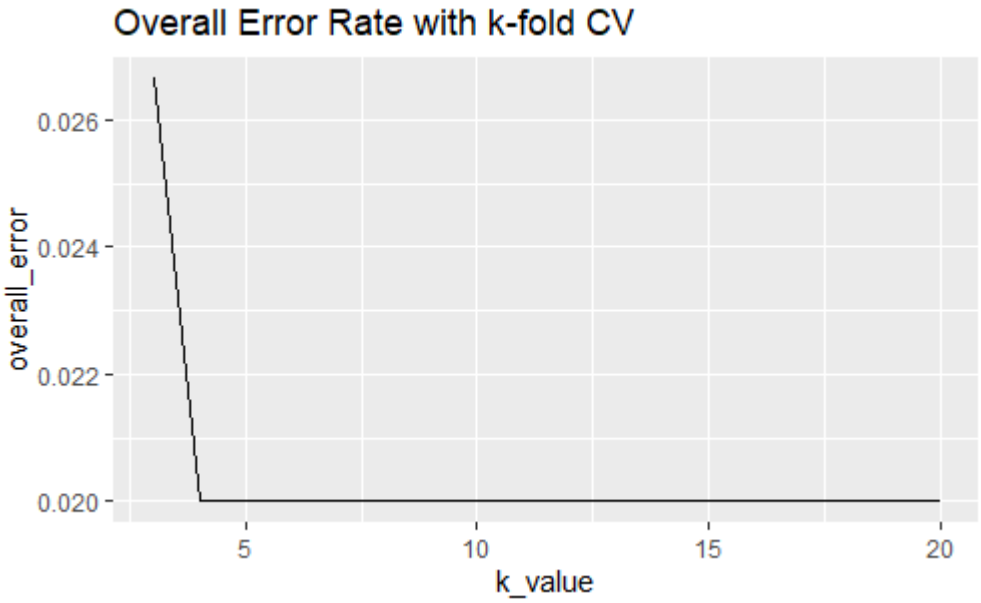
	APER
setosa	0.00
versicolor	0.02
virginica	0.1
overall	0.04

Table 2: Apparent Error Estimate for Classification Tree

Question (c):

I run a for loop for each classifier to test their error rates with k-fold CV where k is from 3 to 20.
I implement binom.test(c(number of misclassification, number of successful classification), conf.level = 0.95) to derive the confidence intervals of all errors. It is a test of a simple null hypothesis about the probability of success in a Bernoulli experiment.

1. LDA

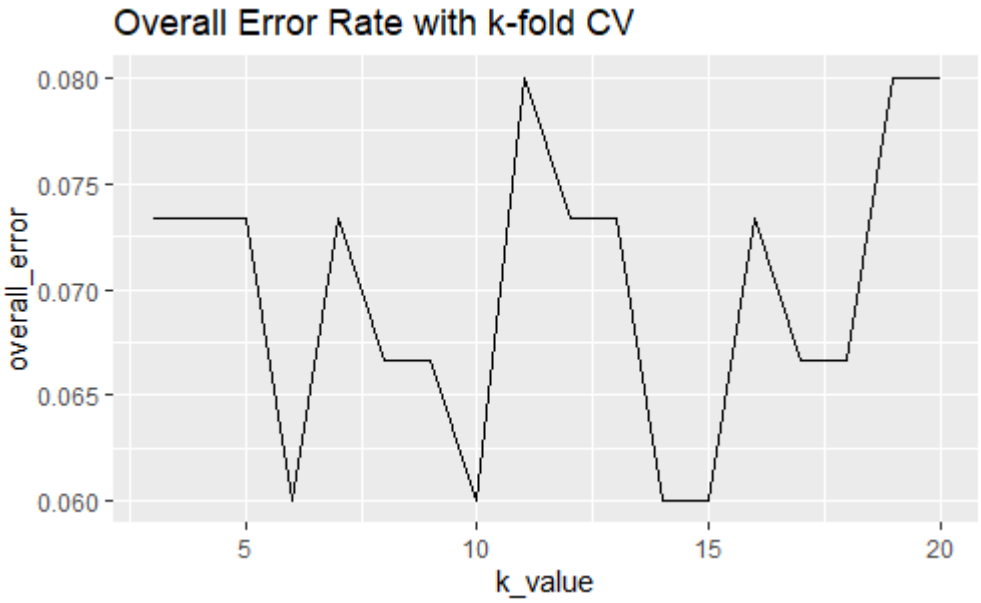


The error becomes steady with k value larger than 5. I would like to choose 10-fold to ensure the optimal error rate.

	error rate	95%CI for error rate
setosa	0.00	(0.00, 0.17)
versicolor	0.05	(0.0013, 0.2487)
virginica	0.05	(0.0013, 0.2487)
overall	0.033	(0.0041, 0.1153)

Table 3: Error Estimate with 10-fold CV for LDA

2. Classification tree



The choice of k value for classification tree is difficult to decide, since it seems unsteady for different k. I would like to choose 10-fold as well for better comparisons.

	error rate	95%CI for error rate
setosa	0.00	(0.00, 0.17)
versicolor	0.05	(0.087, 0.491)
virginica	0.25	(0.0013, 0.2487)
overall	0.1	(0.038, 0.205)

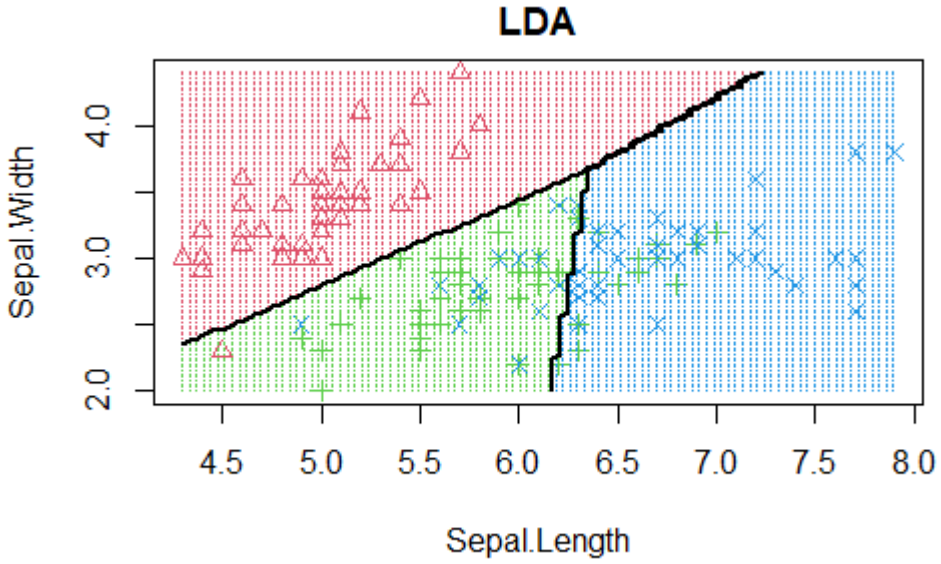
Table 4: Error Estimate with 10-fold CV for Classification Tree

Question (d & f):

The decision boundaries are plotted by evaluating the classifier at evenly spaced grid points, and:
Red Δ represents setosa;
Green + represents versicolor;
Blue $\times$ represents virginica.

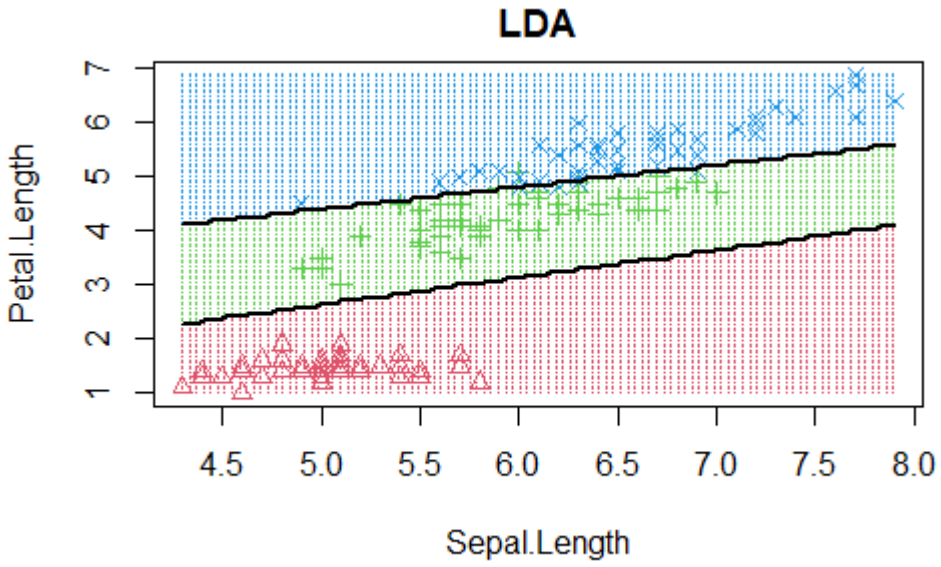
Reference: Hahsler,M.(2017), R Code for Comparing Decision Boundaries of Different Classifiers, https://michael.hahsler.net/SMU/EMIS7332/R/viz_classifier.html

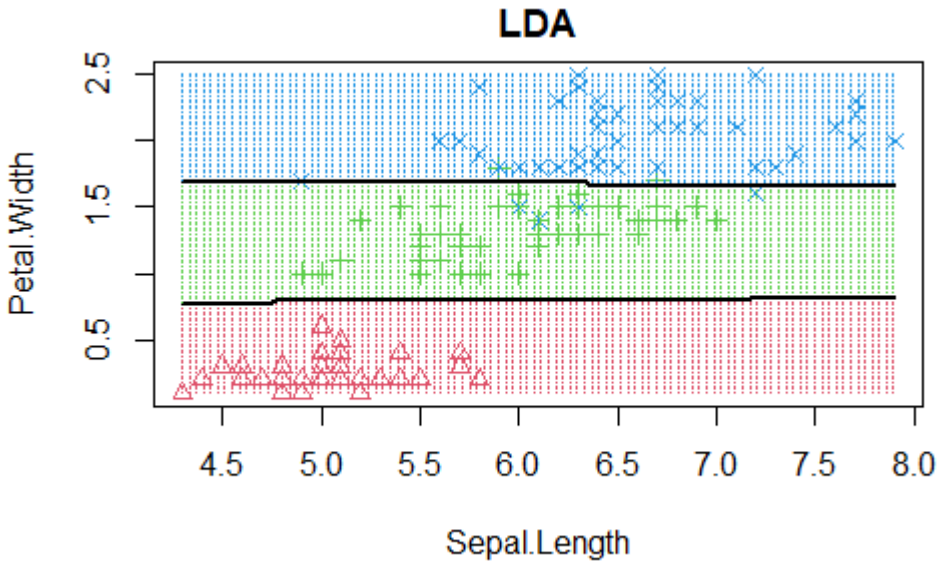
1. LDA



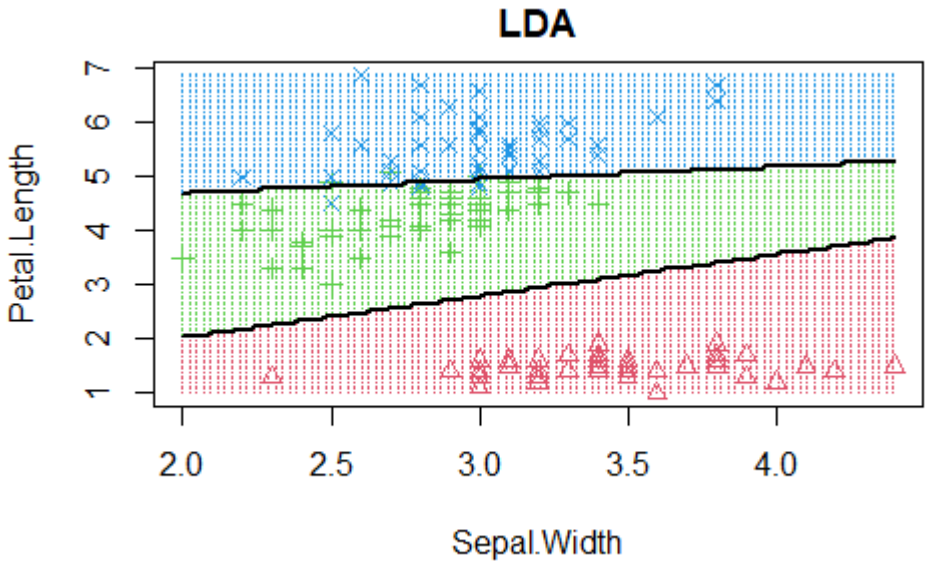
There is an unusual observation in the left-bottom classified as setosa which might be considered as an outlier for versicolor. LDA labels it as setosa because of its small Sepal.Length.
However, the decision boundary based on Sepal.Length and Sepal Width for versicolor and virginica can hardly distinguish the classes of the observations near the

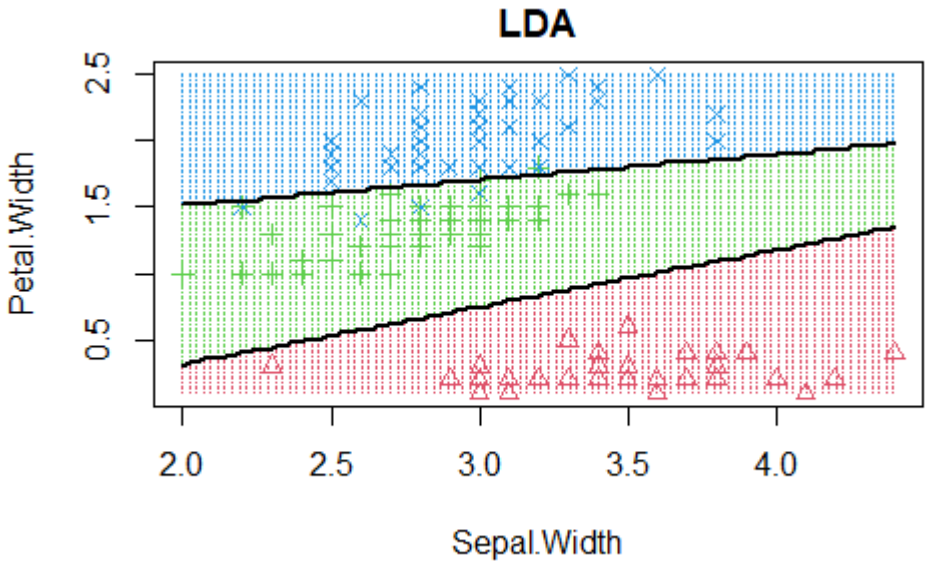
boundary. Therefore, it causes misclassifications.



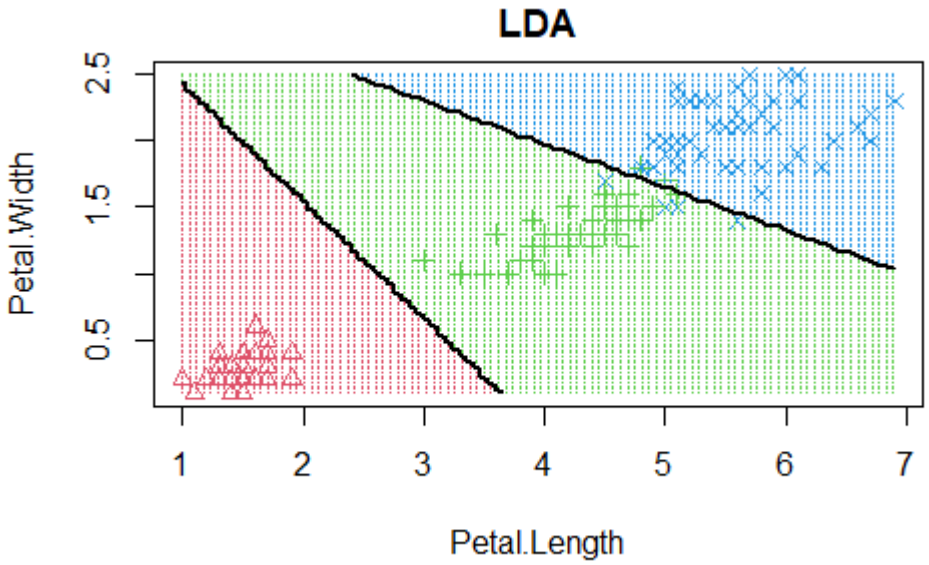


LDA labels the outliers with large Sepal.Length as virginica and the ones with small Sepal.Length as setosa.



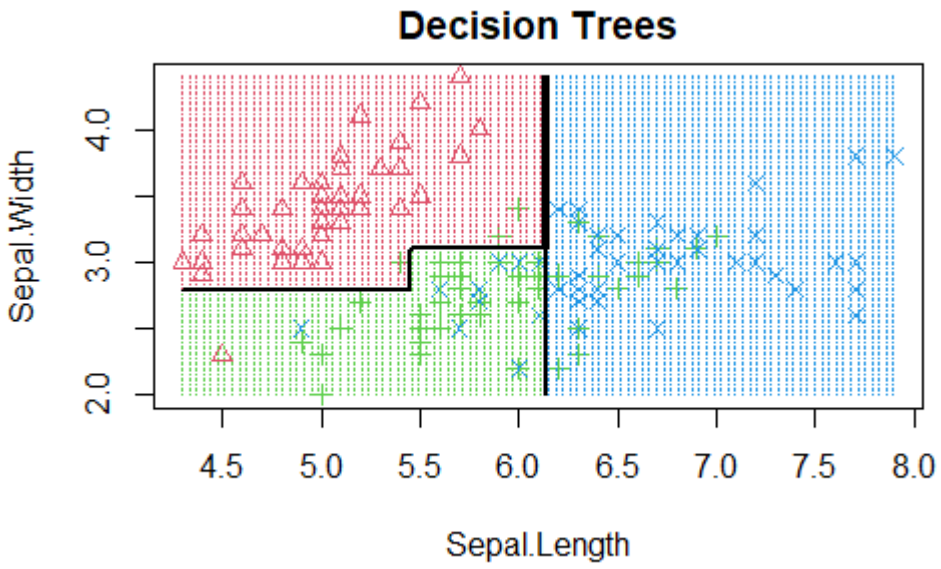


The outliers are classified based on their Petal.Length or Petal Width.
The decision boundaries depend more on Petal.Length or Petal Width, but the measures of versicolor and virginica are closed to each other which might cause misclassifications.

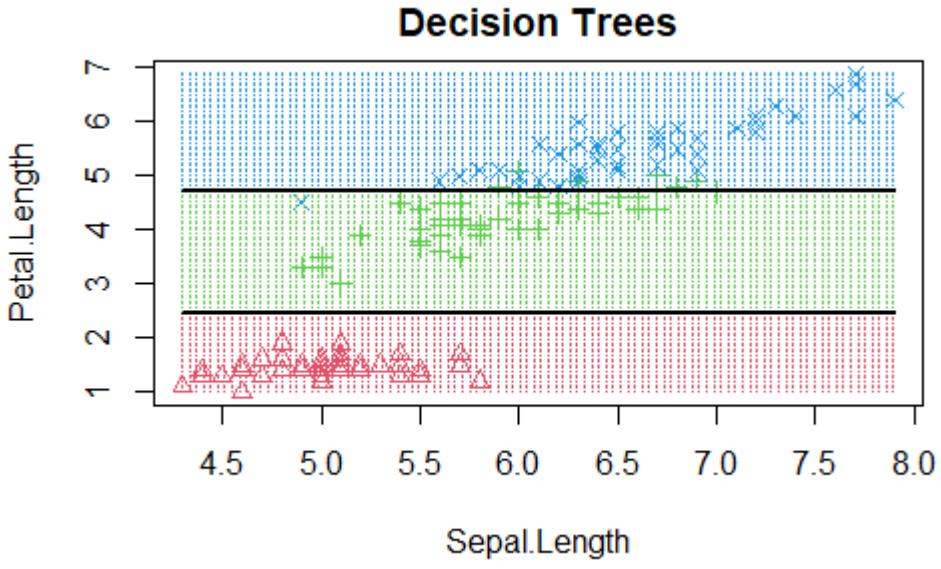


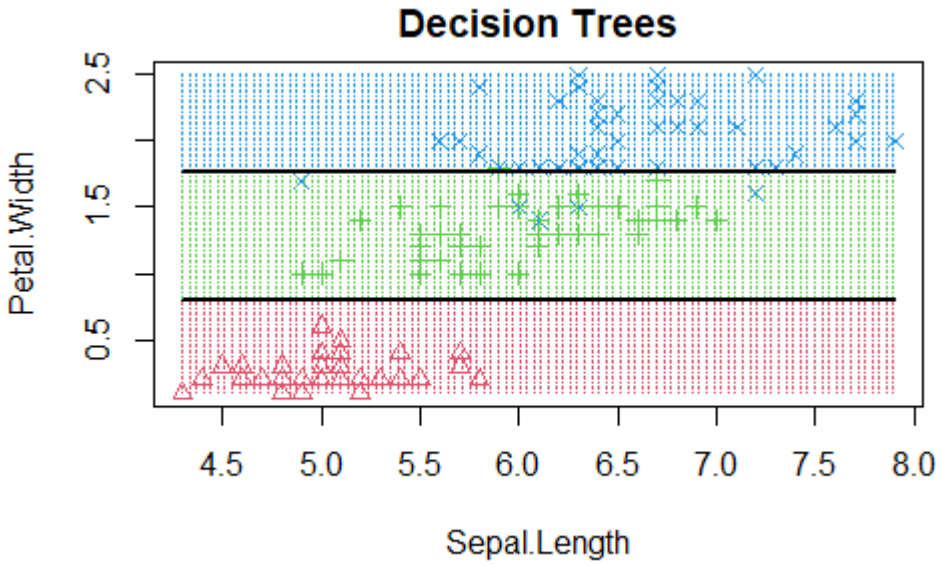
The outliers are classified based on both their Petal.Length and Petal Width. Since these 2 measure are dominant features for classification, the decision boundaries look more convincing than those of previous pairs.

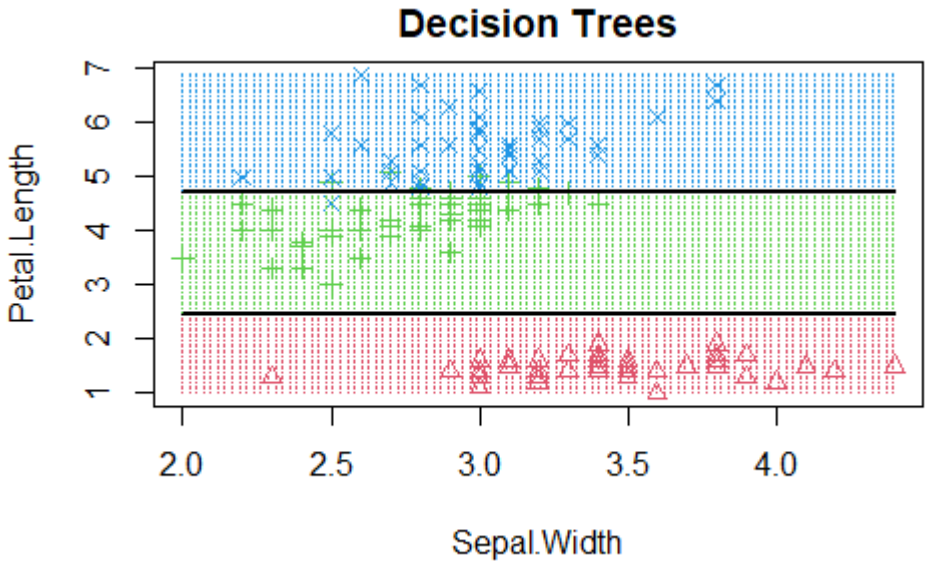
2. Classification tree

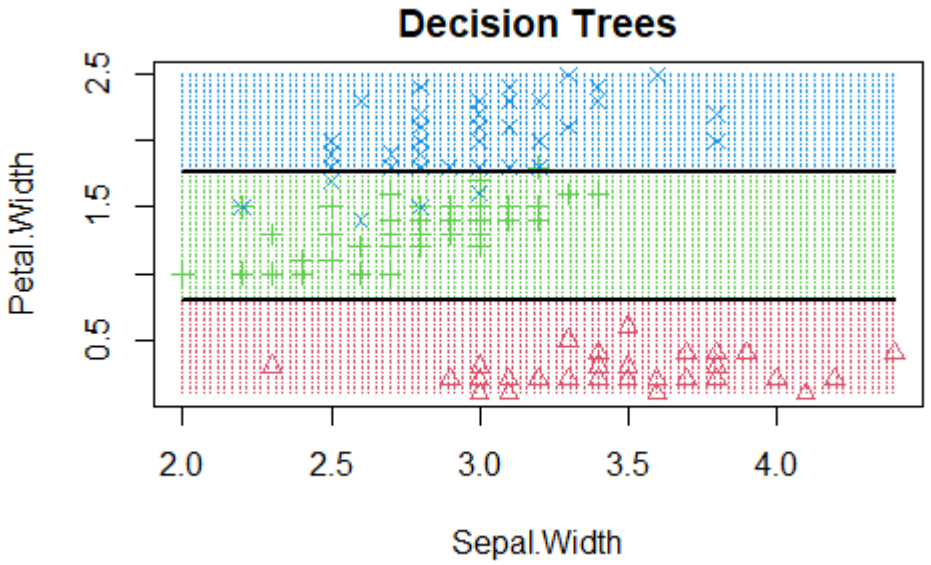


Agian, there is an unusual observation in the left-bottom classified as setosa. Classification tree labels it as setosa because of its small Sepal.Length.

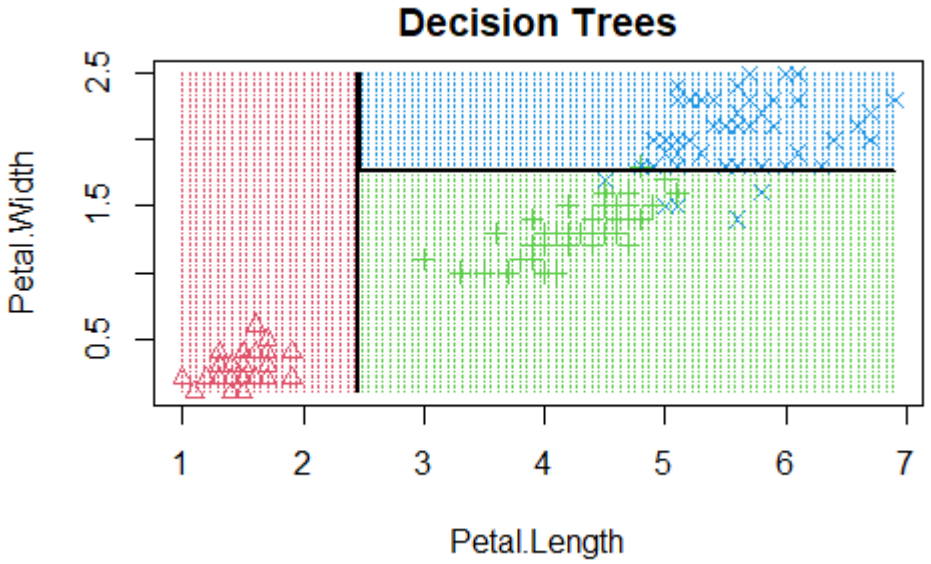



Decision Trees
Petal.Width
0.5
1.5
2.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal.Length


Decision Trees
Petal.Length
1
2
3
4
5
6
7
2.0
2.5
3.0
3.5
4.0
Sepal.Width



The outliers are classified based on their Petal.Length or Petal Width.
The decision boundaries depend more on Petal.Length or Petal Width which is similar with those of LDA.



LDA divide the input space by straight lines and the shape of each space depends on linear discrimiant scores.
Decision Trees divide the input space into axis-parallel rectangles and label each rectangle with one of the k classes.
In the case of Iris data set, the decision boundaries of LDA and classification trees do not make too much differences. I would like to prefer LDA because the performance of classification trees to distinguish versicolor and virginica is less satisfying.

Question (e):

It is not surprising that observations of setosa are not misclassied since their measures are different from those of the other two classes.
From the apparent error rates, some observations of both versicolor and virginica are misclassified. Especially for the classification tree, APER of virginica is 0.1 which is highest compared to other class-specific error rates.
From the CV checks, the 95% CIs for the error rates of both versicolor and virginica for classification tree are higher than those for LDA. After training, the error rates for classification tree are still high, especially classifying virginica.
Therefore, I would prefer LDA in this case rather than classification tree.

Question (g):

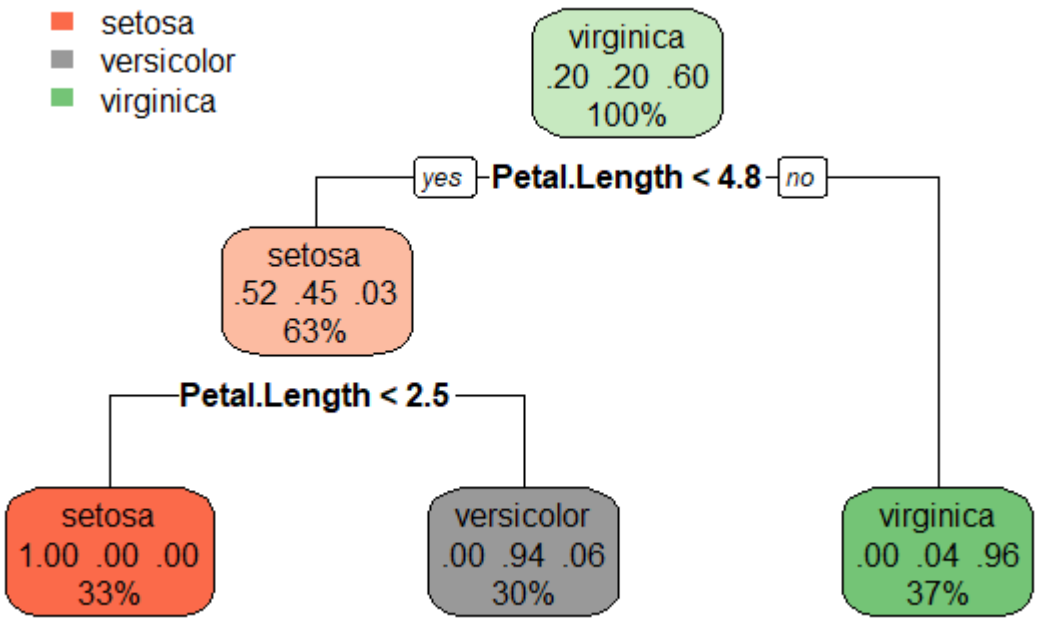
Changing the proportions of classes for LDA can be simply done by setting its parameter of prior probabilities $\hat{\pi}_k$.

```
Prior probabilities of groups:
    setosa versicolor  virginica
       0.2        0.2        0.6

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.828443  0.04615616
Sepal.Width   1.476351  2.20457427
Petal.Length -2.175642 -0.99014657
Petal.Width  -2.884992  2.76342116

Proportion of trace:
   LD1    LD2
0.9922 0.0078
```

As we can see, the change of prior probabilities cause the change of the linear discriminants in LDA. However, it does not affect too much on LDA1 which is our dominant classification factor.

2. Classification tree

Changing the proportions of classes for classification tree can be simply done by setting its parameter $\hat{p}_{1g}$, which is the proportion of the training observations passing through the decision tree to the first node.



The change of proportions cause the change of node because the node purity measurements include the proportion parameter $\hat{p}_{mg}$. We can see that the first node is changed into testing out if the observation belongs to virginica which takes the largest proportion 0.6 among 3 classes.

Task 3

Question (a):

The train set has 60000 rows and 785 columns while the test set has 10000 rows and 785 columns.
The first column of both data sets is the label which denotes the actual digit of each observation.

	label <int>	X1x1 <int>	X1x2 <int>	X1x3 <int>	X1x4 <int>	X1x5 <int>	X1x6 <int>	X1x7 <int>	X1x8 <int>
1	5	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	9	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	0	0	0

6 rows | 1-10 of 785 columns

The numbers of observations for each digit in the training set:

```
   0    1    2    3    4    5    6    7    8    9
5923 6742 5958 6131 5842 5421 5918 6265 5851 5949
```

and the numbers of observations for each digit in the test set:

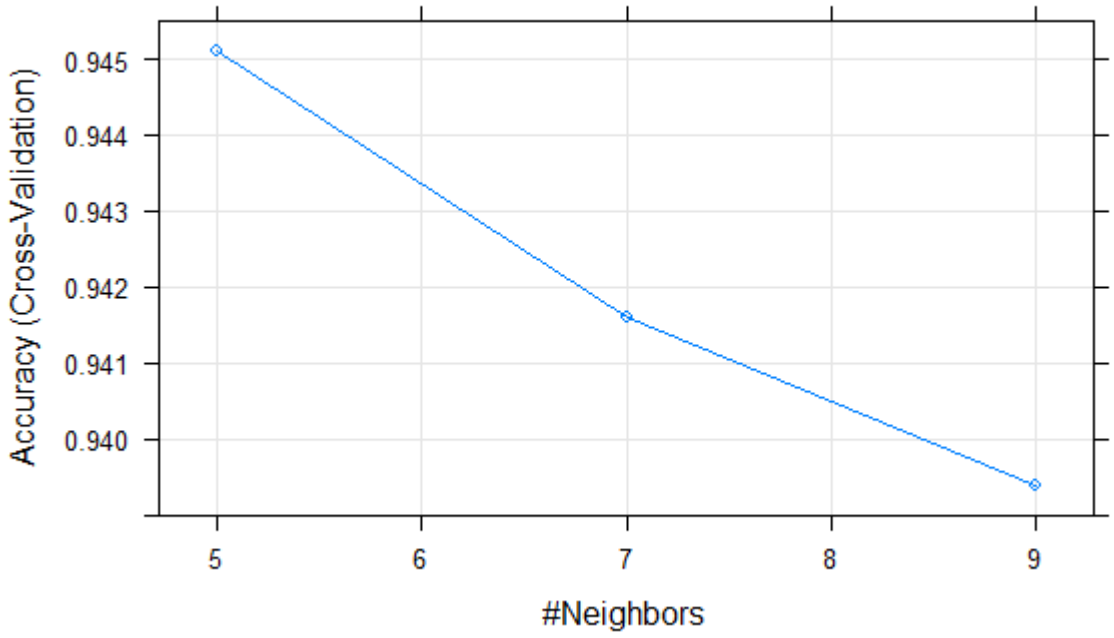
```
   0    1    2    3    4    5    6    7    8    9
 980 1135 1032 1010  982  892  958 1028  974 1009
```

Since the original MNIST data is too large for computing on my laptop, I sample part of these 2 sets. Now my new training set has 10000 rows and my new test set has 1000 rows.

Question (b & c):

3. KNN

Firstly, train the knn model with 5-fold CV and derive the optimal value for k. When k = 5, the knn model has the highest accuracy which is around 94.5%



Then, predict the labels for the test set after training on the training data set. The following is the confusion matrix of predictions for the test set:

	0	1	2	3	4	5	6	7	8	9
0	89	0	0	0	0	0	0	0	0	0
1	0	106	2	1	0	1	1	0	3	1
2	0	1	94	0	0	0	0	0	0	0
3	0	0	0	98	0	1	0	0	1	3
4	0	0	0	0	90	1	0	0	3	1
5	0	0	0	2	0	82	1	0	1	0
6	0	0	0	0	1	1	103	0	1	0
7	0	0	3	1	0	0	0	109	0	1
8	0	0	0	0	0	0	0	0	86	0
9	0	0	0	1	2	1	0	2	1	104

and the confusion matrix of predictions for the training set:

	0	1	2	3	4	5	6	7	8	9
0	962	0	9	0	2	3	1	0	2	4
1	2	1161	14	7	15	3	2	10	19	1
2	2	2	957	5	0	1	0	0	2	1
3	0	0	3	977	0	7	0	1	10	7
4	0	1	1	0	929	1	0	1	3	9
5	1	0	0	13	0	886	2	0	17	3
6	7	0	0	0	5	6	946	0	5	0
7	0	1	24	9	2	0	0	1042	3	9
8	1	1	4	3	1	1	0	0	874	4
9	2	1	1	12	22	4	0	7	9	935

According to the confusion matrix, calculate out the error rates for each digit and the overall error rate. Similarly with Task 2 c), form the 95% CIs for all error rates by binomial test.

Error rates estimation for test set:

digit	error_rate	CI_left	CI_right
0	0.000000000	0.000000000	0.04060086
1	0.009345794	0.000236587	0.05097254
2	0.050505051	0.016599911	0.11393582
3	0.048543689	0.015947594	0.10965615
4	0.032258065	0.006702441	0.09138779
5	0.057471264	0.018921890	0.12904246
6	0.019047619	0.002315137	0.06711576
7	0.018018018	0.002189563	0.06357332
8	0.104166667	0.051094201	0.18323317
9	0.054545455	0.020277721	0.11494663
overall	0.039000000	0.027877149	0.05293094

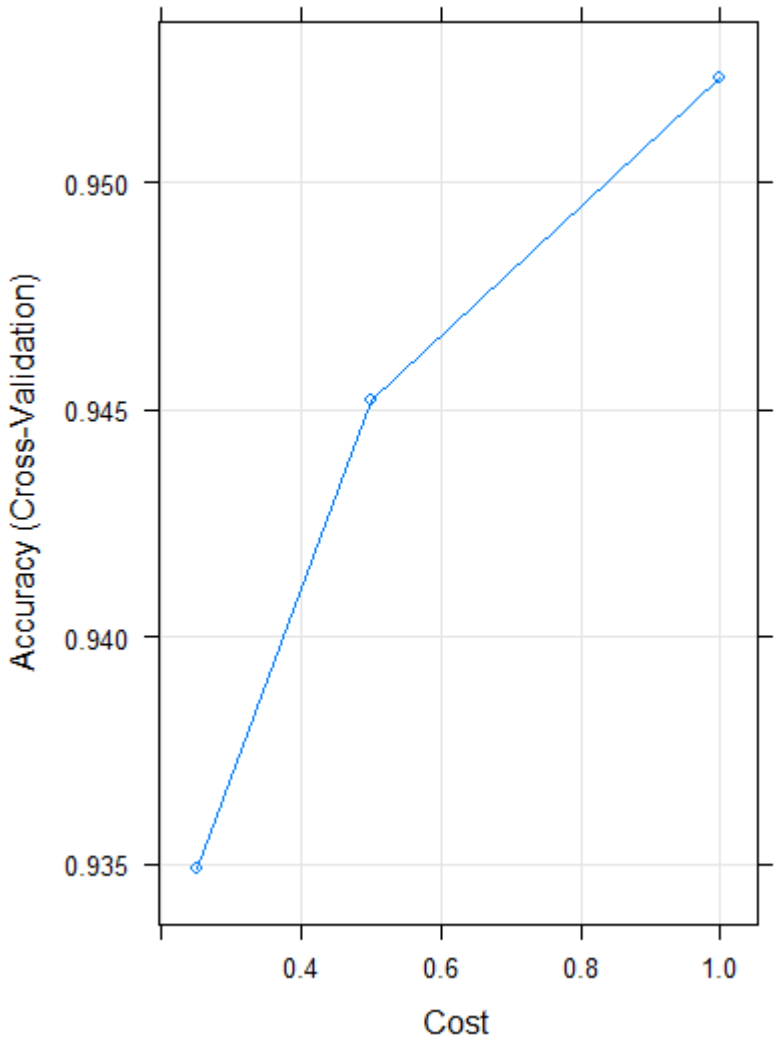
Error rates estimation for training set:

	digit	error_rate	CI_left	CI_right
1	0	0.015353122	0.008617820	0.02519638
2	1	0.005141388	0.001889066	0.01115682
3	2	0.055281343	0.042025636	0.07119020
4	3	0.047758285	0.035537504	0.06265027
5	4	0.048155738	0.035594359	0.06352517
6	5	0.028508772	0.018705578	0.04149429
7	6	0.005257624	0.001709274	0.01222657
8	7	0.017907634	0.010815126	0.02782424
9	8	0.074152542	0.058258525	0.09276136
10	9	0.039054471	0.027782828	0.05321353
11	overall	0.033100000	0.029680211	0.03679509

We can see that most of the class-specific error rates for training set are slightly smaller than those for test set, and the overall error rate of training set is slightly smaller than that of test set. This is expected since the training set has already been exposed to the model before predicting, while the test set has not been used by the model before.

4. SVM

I choose the Radial basis function kernel which depends on the distance from the origin or from some point. When the cost $C = 1$, the SVM model has the highest accuracy which is around 95.2% and the tuning parameter γ is held constant at a value of 1.61729e-07.



The following is the confusion matrix of predictions for the test set:

	0	1	2	3	4	5	6	7	8	9
0	89	0	0	0	0	0	0	0	0	0
1	0	104	1	0	0	1	0	0	0	1
2	0	2	94	1	0	0	0	0	2	0
3	0	0	0	98	0	0	0	0	2	2
4	0	0	1	0	92	1	0	2	1	1
5	0	0	0	2	0	83	1	0	0	0
6	0	0	0	0	0	1	104	0	2	0
7	0	0	2	1	0	0	0	107	0	0
8	0	1	1	0	0	1	0	0	89	0
9	0	0	0	1	1	0	0	2	0	106

and the confusion matrix of predictions for the training set:

	0	1	2	3	4	5	6	7	8	9
0	970	0	5	1	2	1	0	0	0	3
1	0	1154	1	3	2	1	1	3	6	2
2	0	7	993	8	1	2	1	2	4	1
3	0	2	1	980	0	2	0	2	2	8
4	2	1	4	0	956	0	1	6	2	10
5	1	0	2	18	1	903	2	1	3	2
6	1	0	1	1	3	1	946	0	1	1
7	0	0	4	5	0	0	0	1043	1	6
8	2	1	1	5	1	2	0	0	923	4
9	1	2	1	5	10	0	0	4	2	936

Error rates estimation for test set:

digit	error_rate	CI_left	CI_right
0	0.00000000	0.0000000000	0.04060086
1	0.02803738	0.0058197288	0.07975466
2	0.05050505	0.0165999113	0.11393582
3	0.04854369	0.0159475941	0.10965615
4	0.01075269	0.0002721974	0.05845816
5	0.04597701	0.0126673758	0.11355345
6	0.00952381	0.0002410929	0.05192238
7	0.03603604	0.0099043659	0.08969989
8	0.07291667	0.0298178423	0.14447971
9	0.03636364	0.0099952043	0.09049205
overall	0.03400000	0.0236586481	0.04718944

Error rates estimation for training set:

digit	error_rate	CI_left	CI_right
0	0.007164790	0.002885327	0.01470618
1	0.011139674	0.005944369	0.01897385
2	0.019743337	0.012100423	0.03032793
3	0.044834308	0.033008296	0.05935247
4	0.020491803	0.012560761	0.03147120
5	0.009868421	0.004522119	0.01865036
6	0.005257624	0.001709274	0.01222657
7	0.016965127	0.010084807	0.02668011
8	0.022245763	0.013821966	0.03380481
9	0.038026721	0.026913054	0.05203611
overall	0.019600000	0.016973683	0.02251108

Similarly with the result of KNN, most of the class-specific error rates for training set are smaller than those for test set, and the overall error rate of training set is much smaller than that of test set. Fitting the model decreases more error rates for SVM than for KNN.

Question (d):

3. KNN

Advantages:

- KNN is a simple algorithm to understand and implement.
- KNN does not require classes to be linearly separable.

Disadvantages:

- KNN takes longer time for predicting since it needs all data from the training set during prediction.
- KNN does not work well with large or high-dimensional datasets
- KNN is sensitive to noisy or irrelevant attributes. It requires us to manually impute missing values and remove outliers, otherwise it causes less meaningful distance numbers.

4. SVM

Advantages:

- SVM scales well to high dimensional data
- SVM works well with unstructured or semi-structured data like images

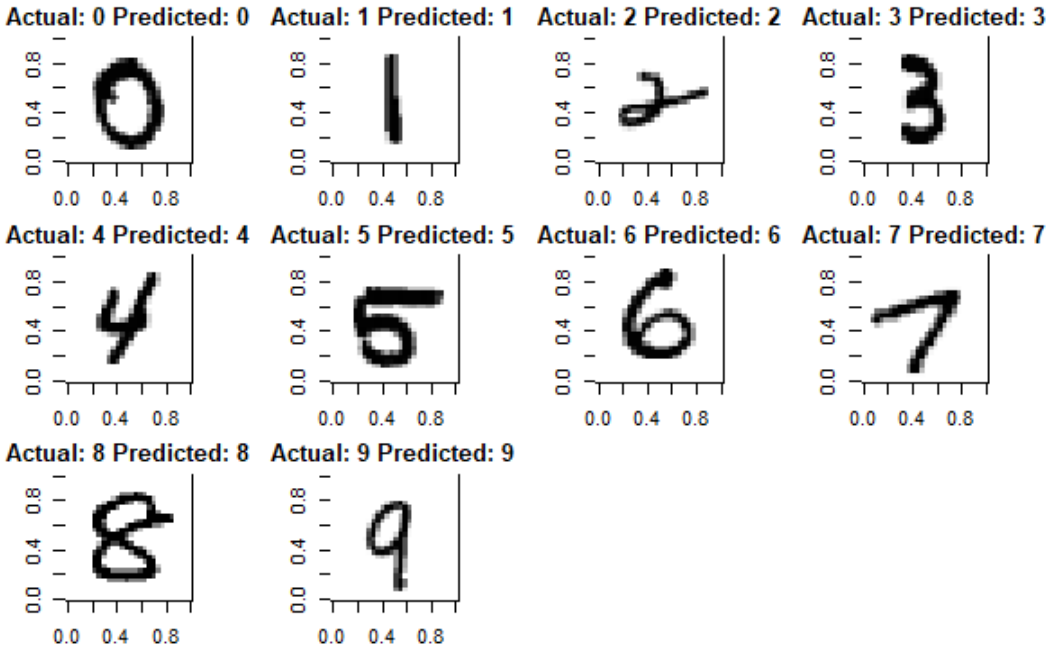
Disadvantages:

- SVM takes a long training time for large datasets
- SVM is difficult to interpret with the model of classifying hyperplanes
- It is hard to interpret and visualize the impact of SVM hyperparameters.

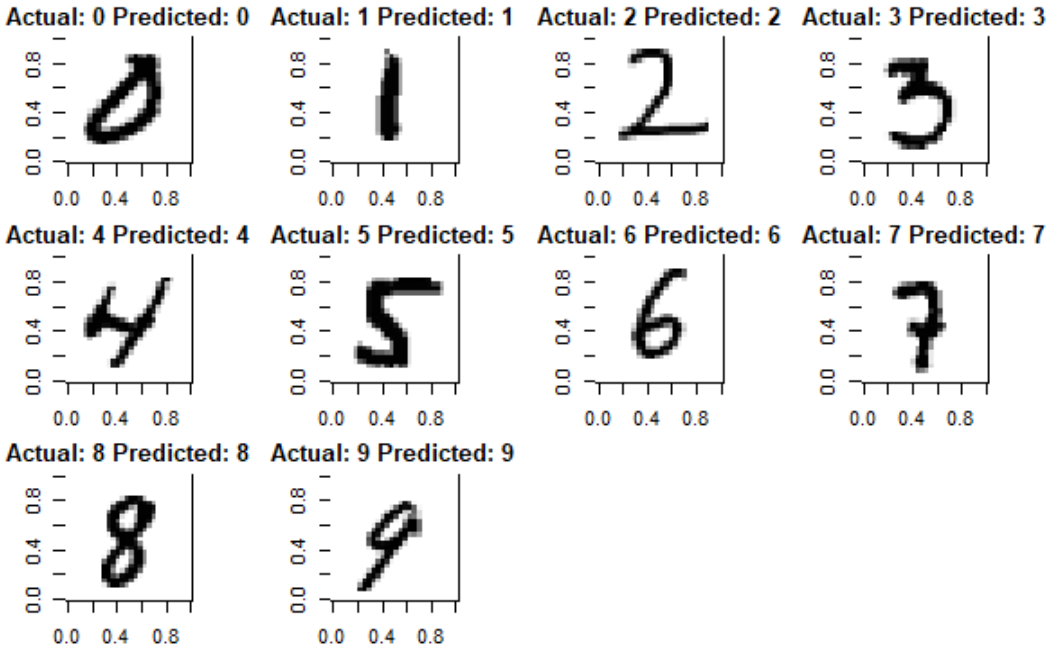
Question (e):

I compared the actual labels and the predicted labels and visualize the data to ensure if they are predicted successfully. I choose the successful examples from the test set to ensure the certainty because the test set was firstly predicted by the KNN model.

3. KNN



4. SVM

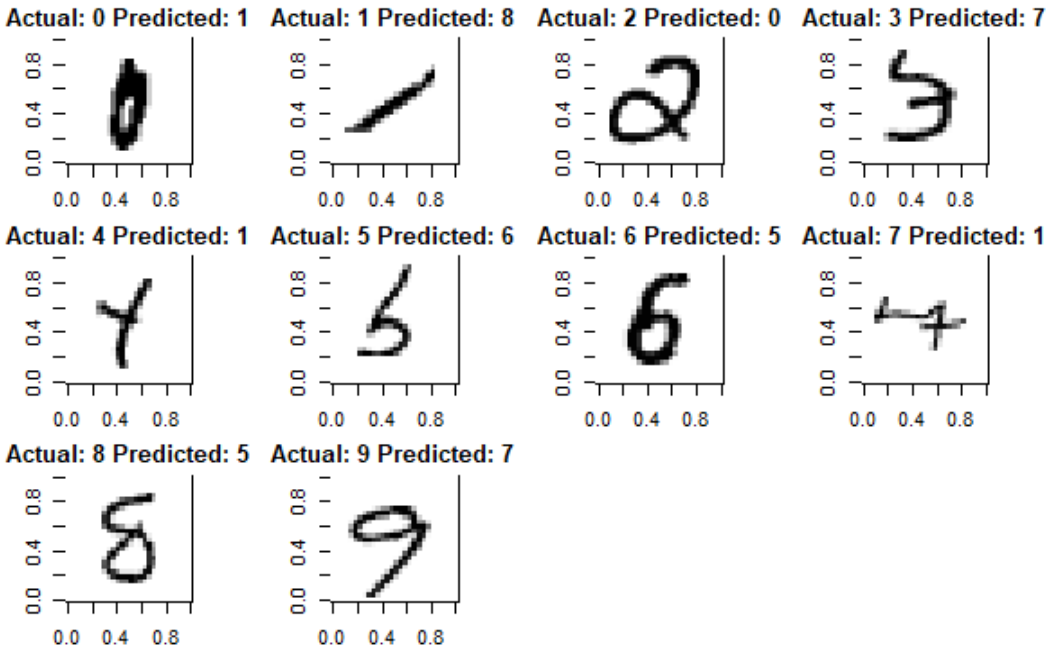


These hand-written digits are not difficult to distinguish by our eyes since they are written clearly and similar with the printed digits. Also, these hand-written digits are clearly different from each other which makes the classification more easier.

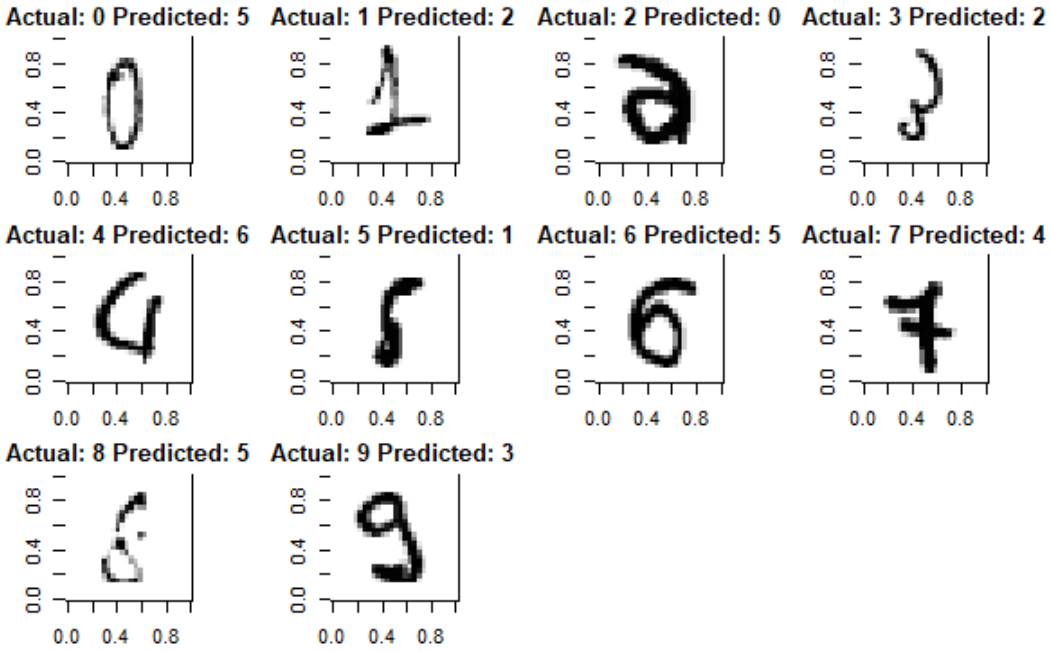
Question (f):

I choose the worst examples from the training. The training set was been fitted once by the models, but the models still failed to classify some digits. Therefore, these digits could be considered as the worst examples.

3. KNN



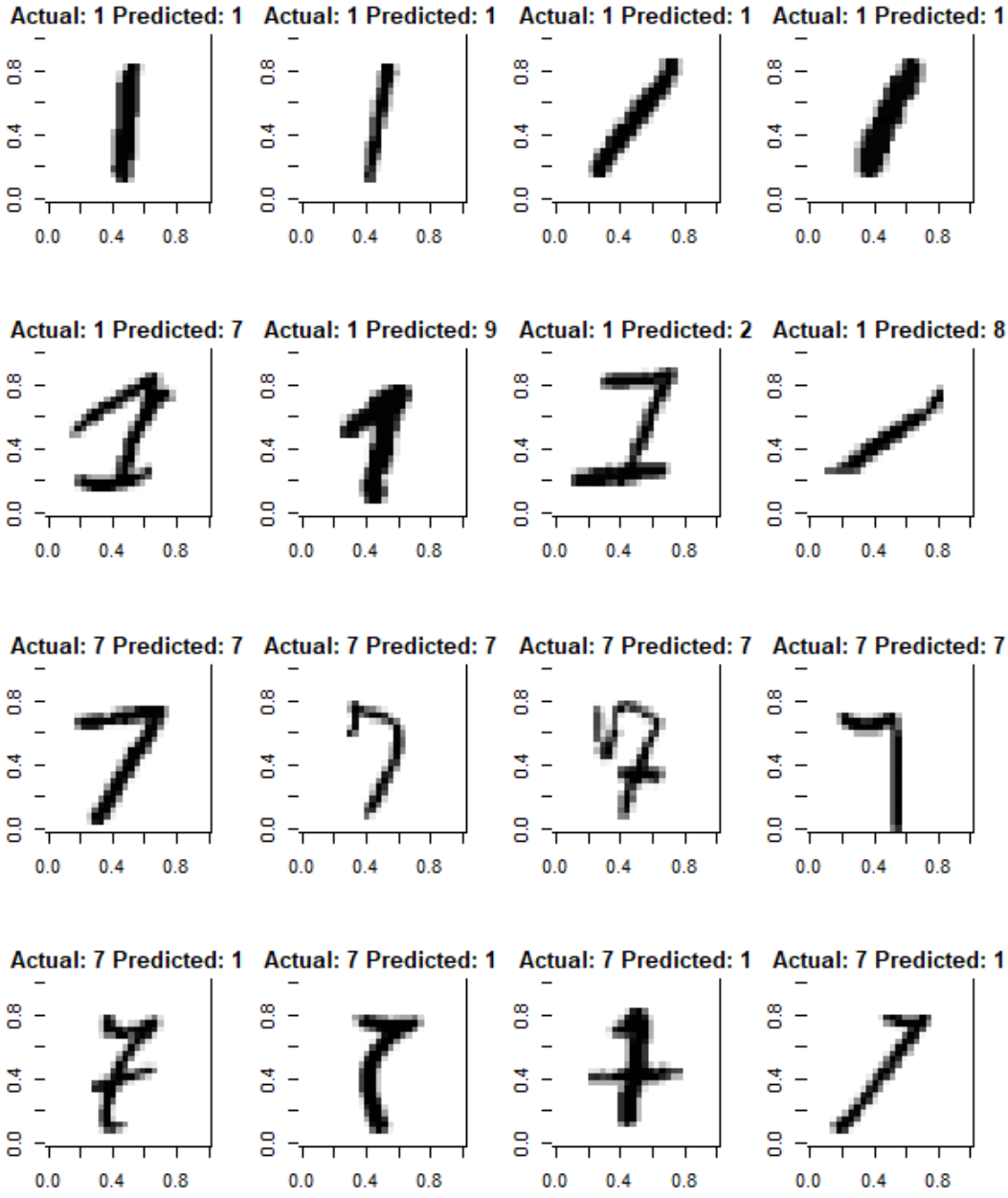
4. SVM



Clearly, most of these digits can hardly be distinguished even by our eyes. Some of them are blurry while some have weird curves or lines might increase their similarity with other digits.

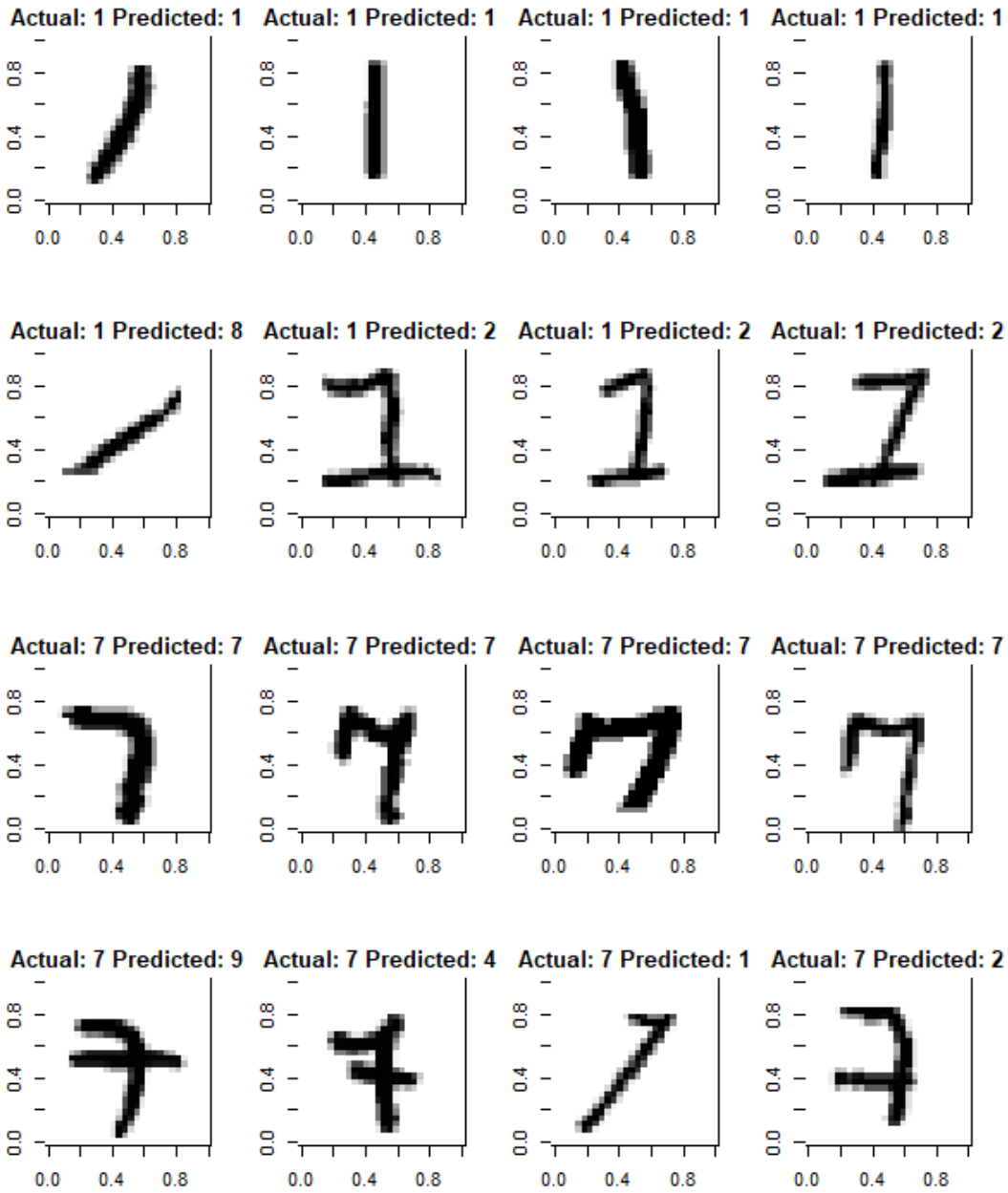
Question (g):

3. KNN



KNN looks at the new data point's distance from all other data points in training set, and then assign the class of the k closest training data points to that new test data point.
In this case, KNN compared a new digit to any other digits in training set. The digit 1 written in a simple vertical line can be easily classified as 1, while the digit 7 written with a long enough line on the top of the backslash can be classified correctly. We cans see the digit 7 with shorter line on the top is classified as 1.

4. SVM


Actual: 1 Predicted: 1
Actual: 1 Predicted: 1
Actual: 1 Predicted: 1
Actual: 1 Predicted: 1
Actual: 1 Predicted: 8
Actual: 1 Predicted: 2
Actual: 1 Predicted: 2
Actual: 1 Predicted: 2
Actual: 7 Predicted: 7
Actual: 7 Predicted: 7
Actual: 7 Predicted: 7
Actual: 7 Predicted: 7
Actual: 7 Predicted: 9
Actual: 7 Predicted: 4
Actual: 7 Predicted: 1
Actual: 7 Predicted: 2
0.0
0.4
0.8

SVM take groups of observations and construct boundaries to predict which group future observations belong to based on their measurements. Classes are divided by a dividing plane that maximizes the margin between different classes. We can see that digit 1 is more likely to be misclassifed as 2 instead of 7. The differneces between digit 1 and 7 for SVM are similar with those for KNN, but SVM misclassified 1 as 7 less than KNN.