Homework 4
Applied Machine Learning
Fall 2017
CSCI-P 556/INFO-I 526

Chuck Jia

November 28, 2017

"All the work herein is solely mine."

Directions

Please follow the syllabus guidelines in turning in your homework. I am providing the LaTeX of this document too. This homework is due Monday November 20, 2017 11:59p.m. **OBSERVE THE TIME**. Absolutely no homework will be accepted after that time. Bring a hardcopy to Tuesdays class on the 21th. If you do not bring a hardcopy with the statement of your own work, the homework will not be accepted. All the work should be your own. Within a week, AIs can contact students to examine code; students must meet within three days. The session will last no longer than 5 minutes. If the code does not work, the grade for the program may be reduced. Lastly, source code cannot be modified post due date.

K-Fold Cross Validation for Model Selection

```
1: ALGORITHM k-fold cross validatiaon
2: INPUT
     • training data $\Delta = (\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)$
     • set of parameter values $\Theta$
     • learning algorithm $\mathcal{H}$
     • integer $k$
3: OUTPUT
     • $\theta^* = \text{argmin}_\theta[error(\theta)]$
     • $h_{\theta^*} = \mathcal{H}(\Delta; \theta^*)$
4: Randomly partition $\Delta$ into $\Delta_1, \ldots, \Delta_k$
5: *** $\Delta_1 \cup \Delta_2 \ldots \cup \Delta_k = \Delta$ and $\Delta_i \cap \Delta_j = \varnothing$ for $i \neq j \in [1, 2, \ldots, k]$
6: for $\theta \in \Theta$ do
7:     for $i = 1 \ldots k$ do
8:         *** Train a model for each training set
9:         $h_{i,\theta} = \mathcal{H}(\Delta \setminus \Delta_i; \theta)$
10:    end for
11:    *** Use the trained models over $\Delta_i$ (test data sets) to evaluate the models for each parameter
12:    error($\theta$) $= \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}_{\Delta_i}(h_{i,\theta})$
13: end for
```

K-Nearest Neighbors (KNN) Algorithm in Theory

```
1: ALGORITHM K-nearest neighbors
2: INPUT
     • training data $\Delta$
     • test data $\Delta'$
     • distance metric $d$, i.e., $d : \Delta^2 \to \mathbb{R}_{\geq 0}$
     • integer $k$: nearest neighbors number
3: OUTPUT
     • class label of each $z \in \Delta'$
4: for $z = (\mathbf{x}', y') \in \Delta'$ do
5:     Compute $d(\mathbf{x}, \mathbf{x}')$, the distance between z and every example $(\mathbf{x}, y) \in \Delta$
6:     Select $\Delta_z \subseteq \Delta$, the set of closest $k$ training examples to z
7:     Voting:
     • majority voting: $y' = \operatorname{argmax}_v \sum_{(\mathbf{x_i}, y_i) \in \Delta_z} I(v = y_i)$
     • distance-weighted voting: $y' = \operatorname{argmax}_v \sum_{(\mathbf{x_i}, y_i) \in \Delta_z} w_i \times I(v = y_i)$ where $w_i = \frac{1}{d(\mathbf{x}', \mathbf{x_i})^2}$
8: end for
```

Naive Bayes Classifier

```
1: ALGORITHM Training of naive bayes classifier (continuous attributes)
2: *** training set: $\Delta = \{(\mathbf{x_j}, y_j)\}_{j=1}^m$
3: for $i = 1, \ldots, k$ do
4:     *** class-specific subsets
5:     $\Delta_i \leftarrow \{\mathbf{x_j} | y_j = c_i, j = 1, \ldots, m\}$
6:     *** size of the subsets
7:     $m_i \leftarrow |\Delta_i|$
8:     *** prior probability
9:     $\hat{P}(c_i) \leftarrow m_i / m$
10:    *** mean
11:    $\hat{\mu}_i \leftarrow \frac{1}{m_i} \sum_{\mathbf{x_j} \in \Delta_i} \mathbf{x_j}$
12:    *** centered data
13:    $\mathcal{Z}_i \leftarrow \Delta_i - \mathbb{I}_{m_i} \hat{\mu}_i^T$
14:    *** variance
15:    $\hat{\sigma}_i \leftarrow \frac{1}{m_i} \mathcal{Z}_i^T \mathcal{Z}_i$
16: end for
17: return $\hat{P}(c_i), \hat{\mu}_i, \hat{\sigma}_i$ for all $i = 1, \ldots k$
18:
19: TESTING( $\mathbf{x}$ and $\hat{P}(c_i), \hat{\mu}_i, \hat{\sigma}_i$, for all $i \in [1, k]$):
20: $\hat{y} \leftarrow \operatorname{argmax}_{c_i} \{f(\mathbf{x} | \hat{\mu}_i, \hat{\sigma}_i)\ \hat{P}(c_i)\}$
21: return $\hat{y}$
```

M-estimate of Conditional Probability

If the class-conditional probability for one of the attributes is zero, then overall posterior probability for the class vanishes. This problem can be addressed by using the m-estimate approach for estimating the conditional probability:

$$P(x_i|y_j) = \frac{n_c + m \times p}{n + m}$$

- x_i: training example x_i, y_j**:** class y_j
- n_c : number of training examples from class y_j that take on the value x_i
- n : total number of instances from class y_j
- m : equivalent sample size. m determines the trade-off between the prior probability p and the observed probability n_c/n
- p : user-specified parameter. p can be regarded as the prior probability of observing the attribute value x_i among records with class y_j

---

In this homework, you are asked to implement k-nearest neighbors (KNN), naive bayes classifier and k-fold cross validation for model selection. You will test/compare them over Ionosphere, car evaluation and credit approval data sets. Click on the links below to obtain the data sets.

- Ionosphere Data Set
- Car Evaluation Data Set
- Credit Approval Data Set

Problem 1: K-Fold Cross Validation [25 points]

Implement k-fold cross validation and select $k = 5$ to create 5 training and 5 test data sets from each data set and save these 30 files. You will use these data sets for model comparison and parameter selection.

The `R` code for creating the cross validation files is in the file `Prob1-CreateFiles.R`.
The files generated are stored in the folder `/Data/KFold`.

Problem 2: K-Nearest Neighbors (KNN)[55 points]

2.1 Implement KNN algorithm with two different distance functions. You can either use existing distance functions, i.e., Euclidean or design your own.

2.2 Use the data sets obtained in problem 1 to determine the optimal k over each data set for KNN algorithm. For 5 different k values, plot the test error for each data set. Total number of figures = 3 (data set number) $\times$ 2 (distance function number) = 6. Report the best k and distance function for each data set.

2.3 Use the KNN package in R for validation.

(2.1) In the `R` code, we tested 7 different distance functions. They are the natural metrics induced by the 1-norm, 2-norm, $\cdots$, 6-norm and the ∞-norm in the space $\mathbb{R}^n$. The p-norms are defined as usual:

$$\|(x_1, \cdots, x_n)\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad \text{for } 1 < p < \infty$$

$$\|(x_1, \cdots, x_n)\|_\infty = \max_{i=1,\cdots,n} \{|x_i|\}, \quad \text{for } p = \infty$$

The distance function corresponding to the p-norm is defined as

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

Below we only give the results from using the 1-norm and 2-norm distance functions, as the they are the 2 best performing distance functions among all the functions tested.

Data Pre-processing

Because of the existence of the categorical data, which are not numerical, we need to pre-process the original data set. In data pre-processing, we also engineer the feature scales in order to yield better result.

In our implementation, for different data types, we use the following methods for pre-processing.

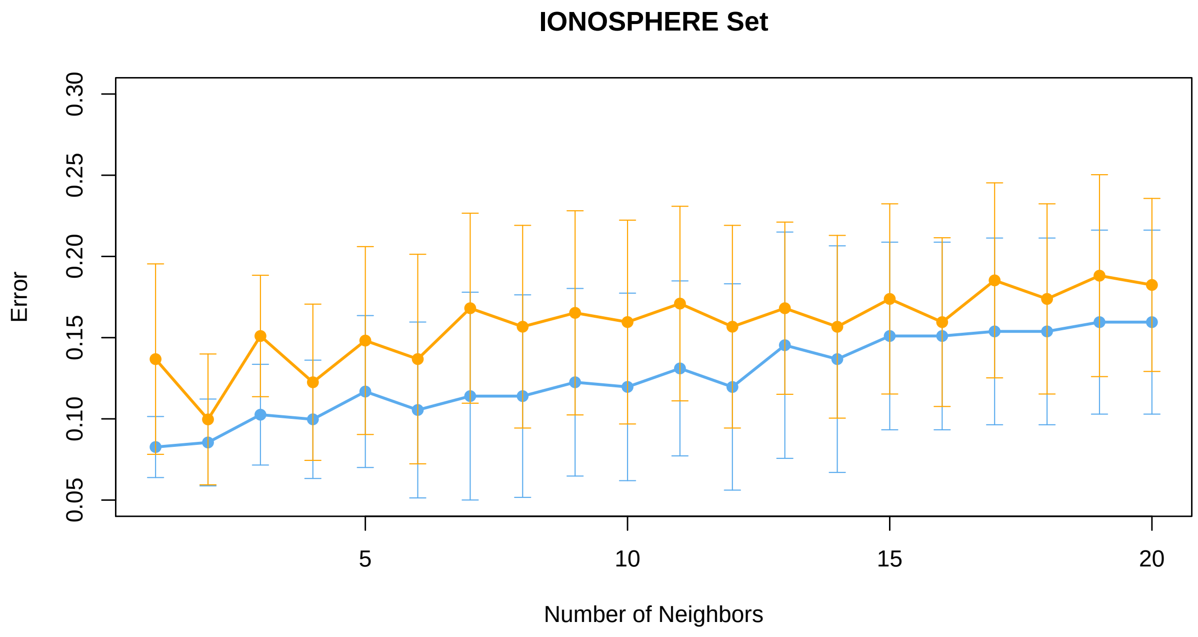
- *Continuous data*: We keep them in continuous form, but we center all the continuous features and scale them to standard deviation 1. This applies to both `Ionosphere` and `Credit Approval Data` sets.
- *Categorical data*: We use dummy variables to transform them to numerical data.
- *Missing data*: Since the proportion of missing data is relatively small, we replace all the `?` with the mean or the median of the feature where the missing data point is located. The use of mean or median depends on the type of the feature. If the feature is continuous, we choose to use the mean for replacement. If the feature is categorical, we use the median.

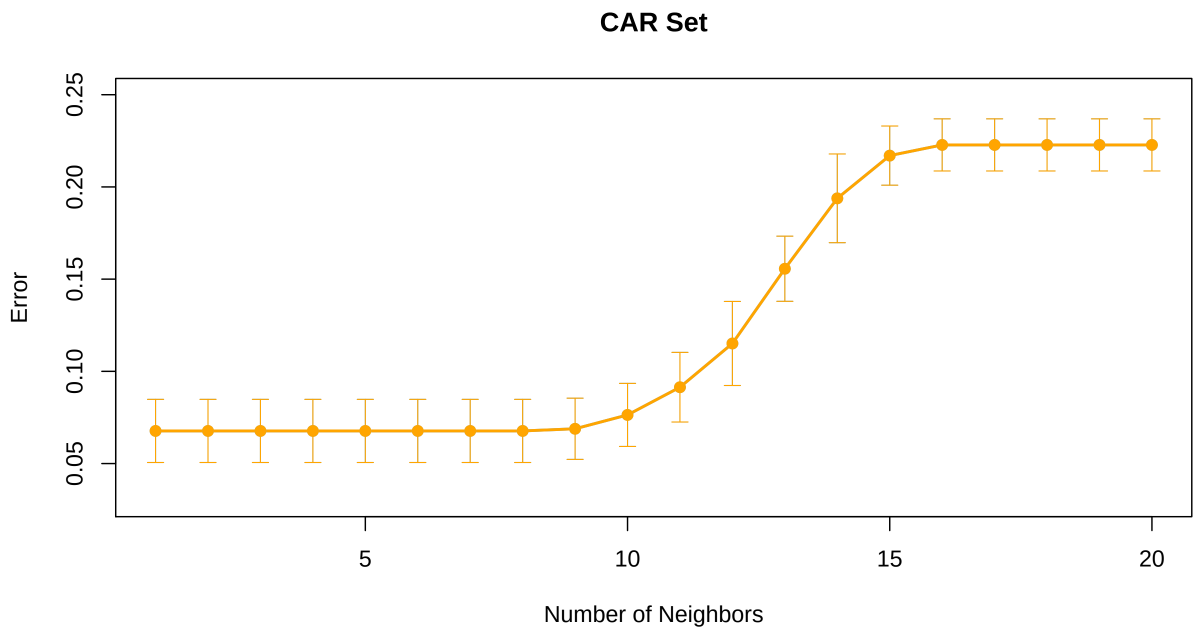
(2.2)

Preliminary Analysis

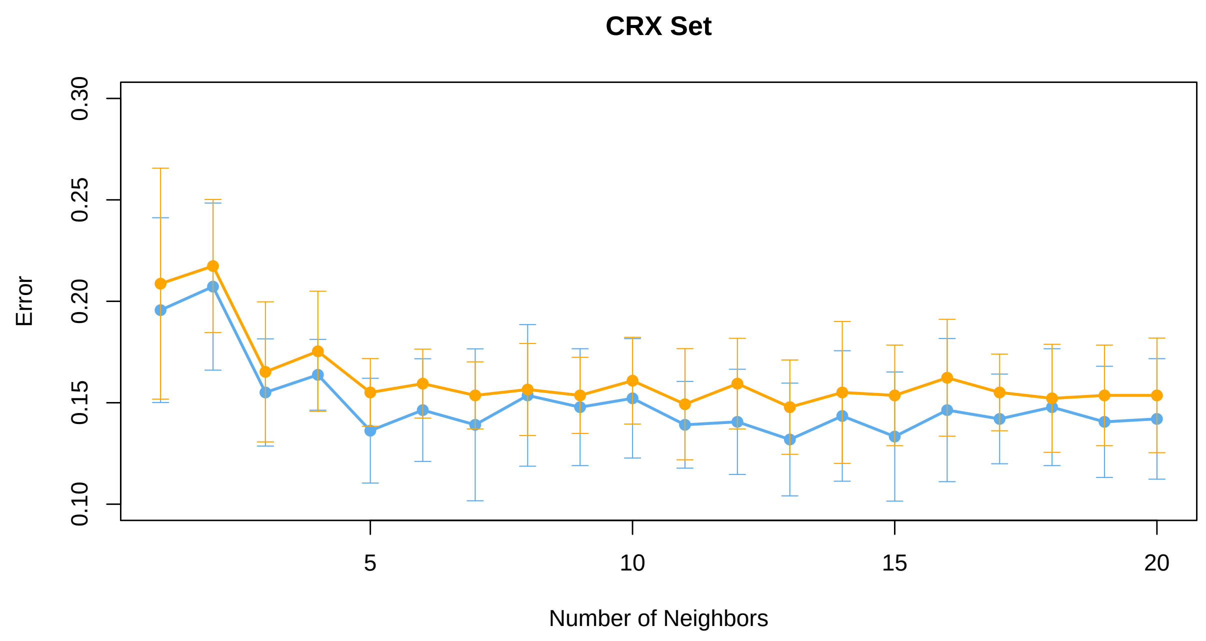
Before we plot the final results on 5 different k's, we need to find the best k values. Here, we plot the average error rates from 20 different k's in order to select the best five k values.

In the 3 graphs below, the `blue lines` represent errors from using the **1-norm** distance function, while the `orange lines` represent errors from using the **2-norm** distance function. The error bars represent the standard deviation across the 5 validation sets for each k value.





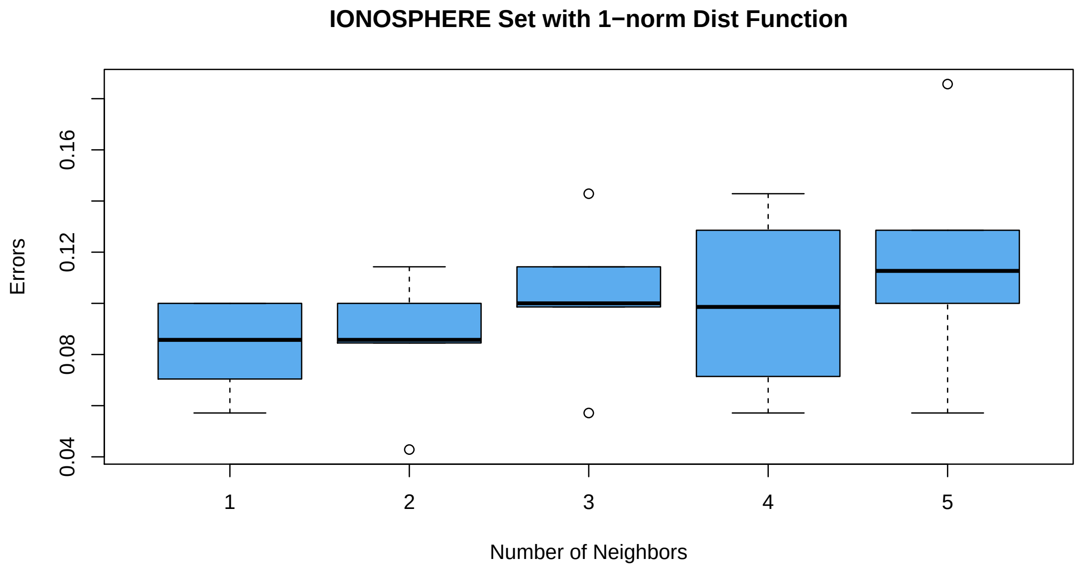
In the car set, the results from the 2 distance functions are very similar, which results in only showing the orange line.

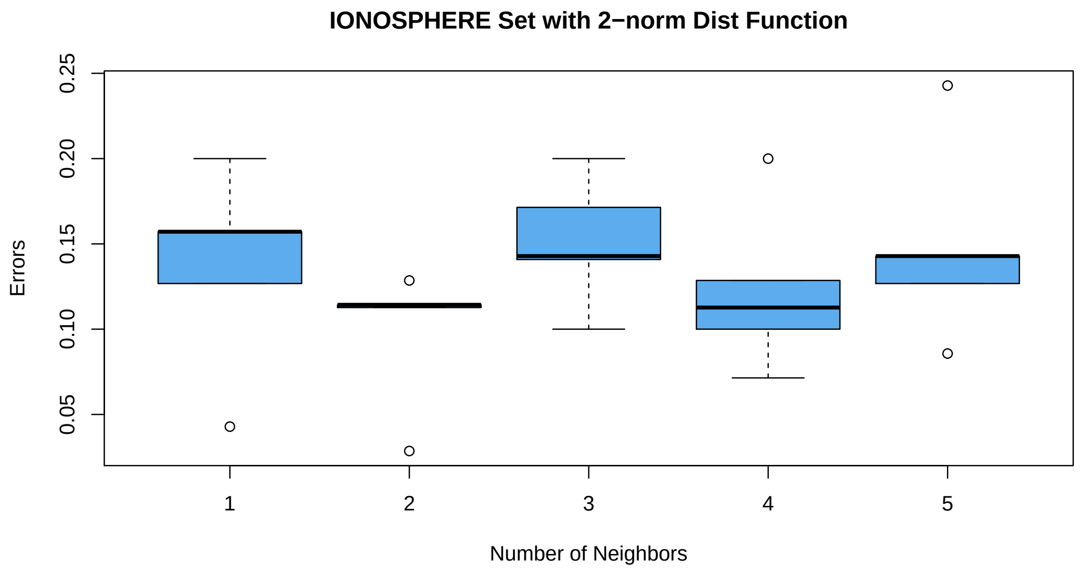


Final Results

From the preliminary results above, we selected 5 best k values for each data set. Now we give the error results using box plots.

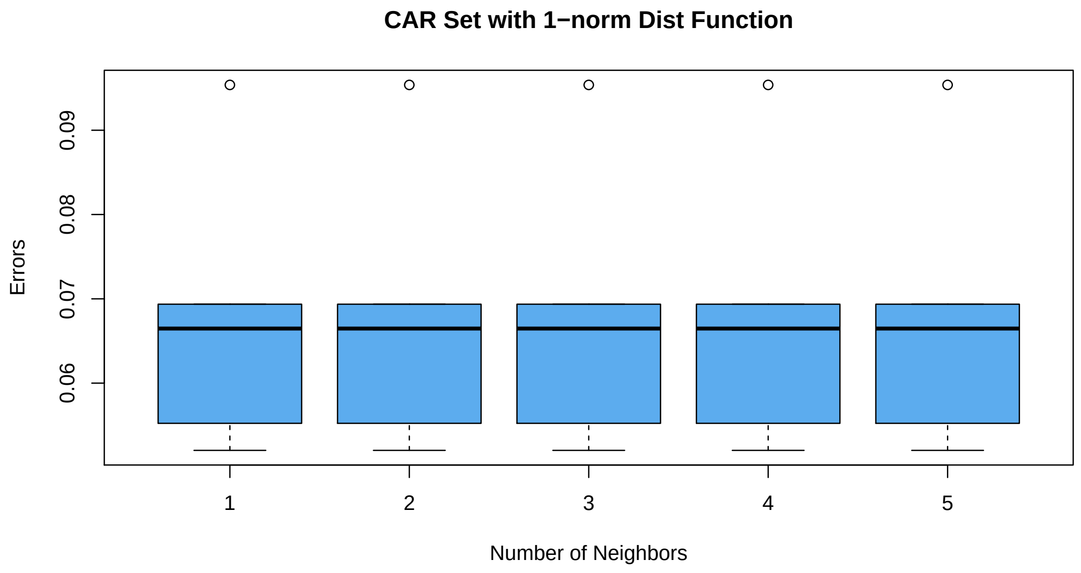
Ionosphere Set

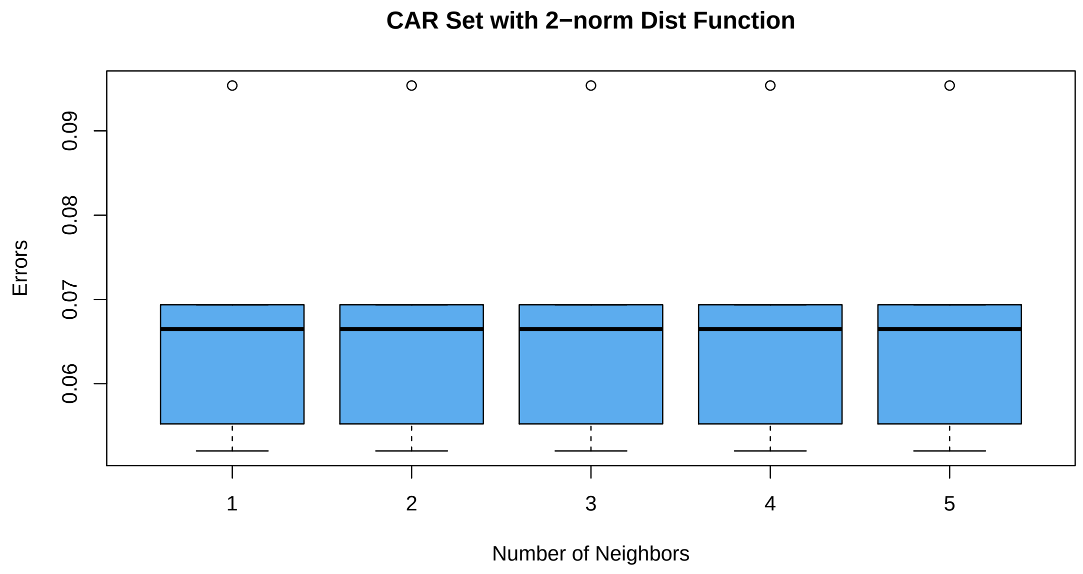




For the Ionosphere set, the best k is 1, with the 1-norm distance function. The error rate is 0.08265594.

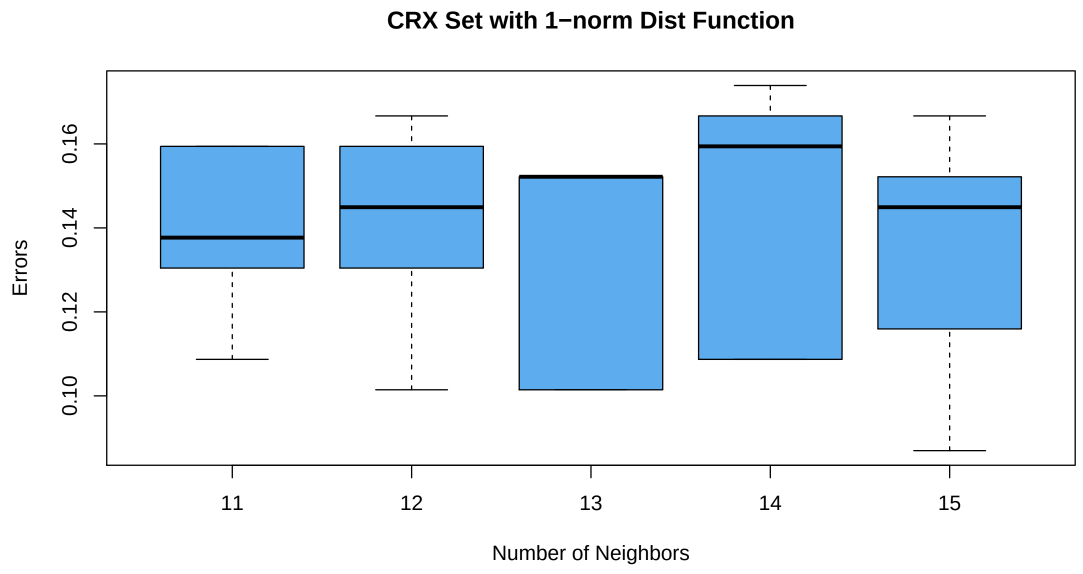
Car Set



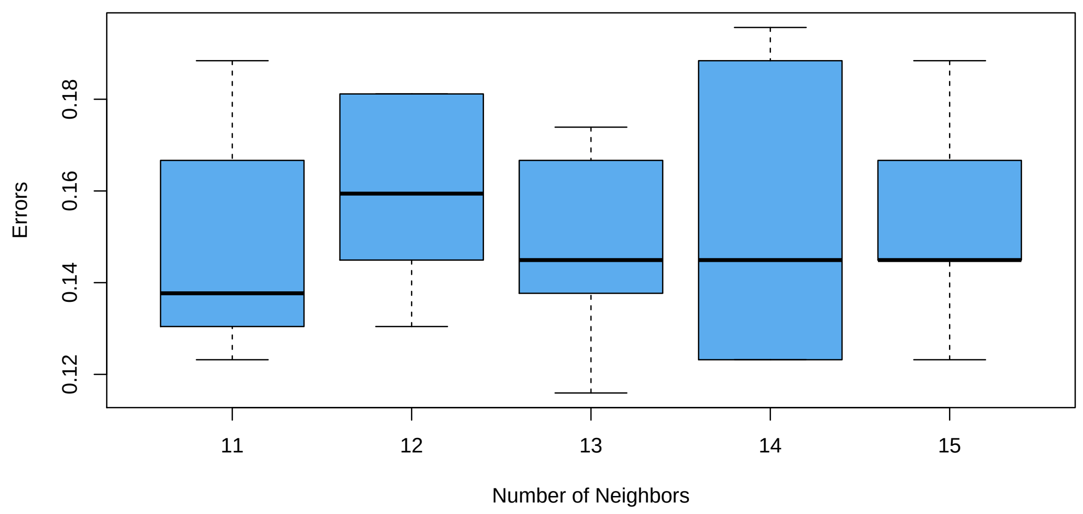


For the Car set, the best k is 1, with the 1-norm distance function. The error rate is 0.06769391.

Credit Approval Data Set



CRX Set with 2−norm Dist Function



For the Crx set, the best k is 13, with the 1-norm distance function. The error rate is 0.1318841.

Summary of Results

Now we summarize the best results from my implementation of KNN.

Our KNN performs best with 1-norm distance function on all of the 3 data sets. For the `Ionosphere` and `Car` sets, the best k is 1, and for the `Credit Approval Data` set, the best k is 13.

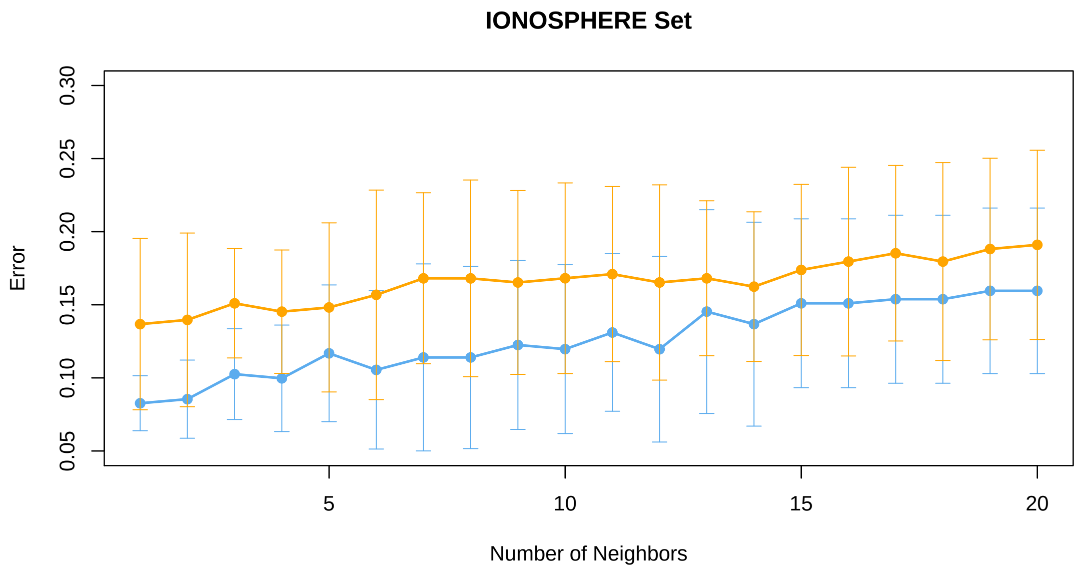
Best Average Error Rate from KNN

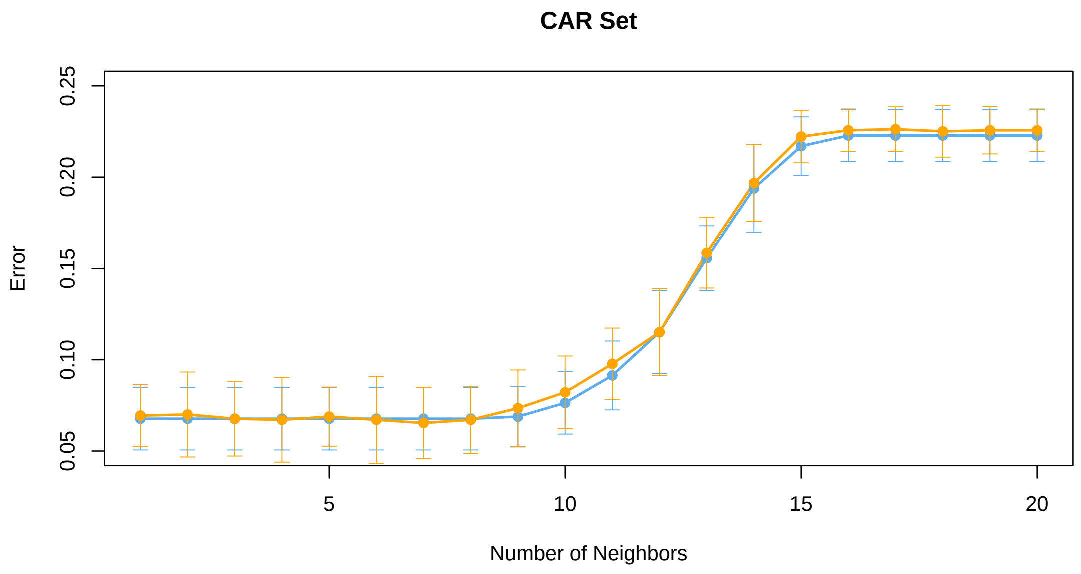
Data Set	**Average Error Rate**
Ionosphere	0.08265594
Car	0.06769391
Credit Approval	0.1318841

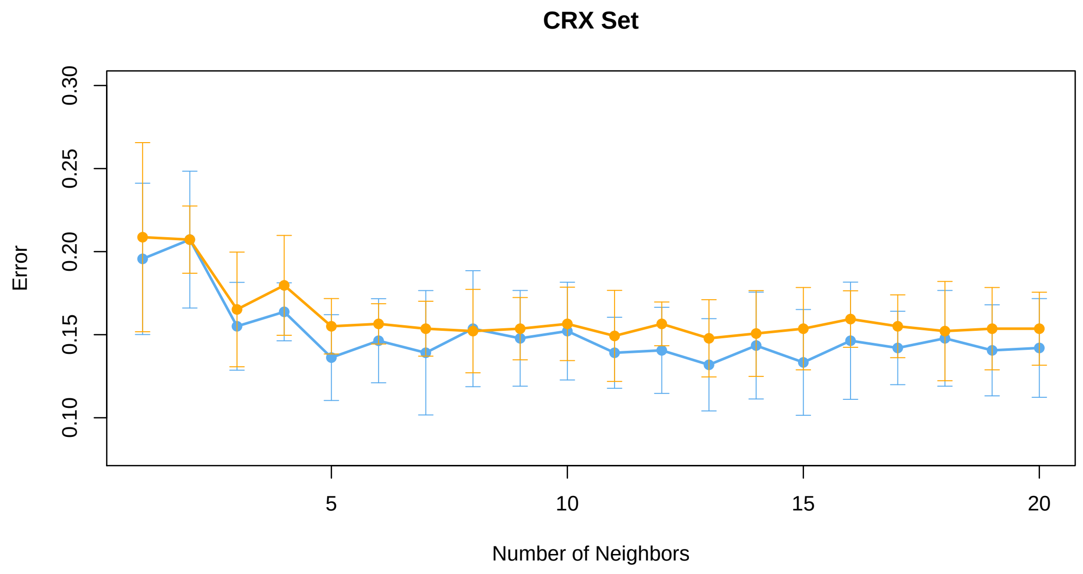
(2.3)

Now we compare the results from my implementation of KNN with the results from using `R` built-in KNN methods for validation.

In the 3 graphs below, the `blue lines` represent errors from using **my implementation** of KNN, while the `orange lines` represent errors from using **built-in** KNN method in R.


IONOSPHERE Set
Error
0.05
0.10
0.15
0.20
0.25
0.30
5
10
15
20
Number of Neighbors


CAR Set
Error
0.05
0.10
0.15
0.20
0.25
5
10
15
20
Number of Neighbors



As we can see, in the 2 data sets with continuous data, `Ionosphere` and `Credit Approval Data Set`, our implementation performs better than the `R` built-in implementation. The difference is in the choice of the distance function. The On these 2 data sets, the KNN method works better with the 1-norm distance function.

In the `Car` data set, our implementation performs very similar with the `R` built-in implementation. This is expected, since this data set is made up of only categorical data. The effect of changing between 1-norm and 2-norm on discrete data is smaller than on continuous data. The slight difference between the results from the two implementations is caused by the difference in the voting method, especially in the ways of handling ties.

Problem 3: Naive Bayes Classifier [55 points]

3.1 Implement Naive Bayes classifier. The Pseudo-code for naive bayes algorithm is provided above. You may need to modify it for categorical variables. To handle unseen feature values, you may need to make use of m-estimate of conditional probability method. There are also other techniques, i.e., Laplace smoothing.

3.2 Train Naive Bayes classifiers over training data sets and test each classifier against corresponding test data. Make a plot that shows the error over each test data. Report the average error rate for 5-fold cross validation for each data sets.

3.3 Use Naive Bayes package in R for validation.

(3.1-3.2)

Data Pre-processing

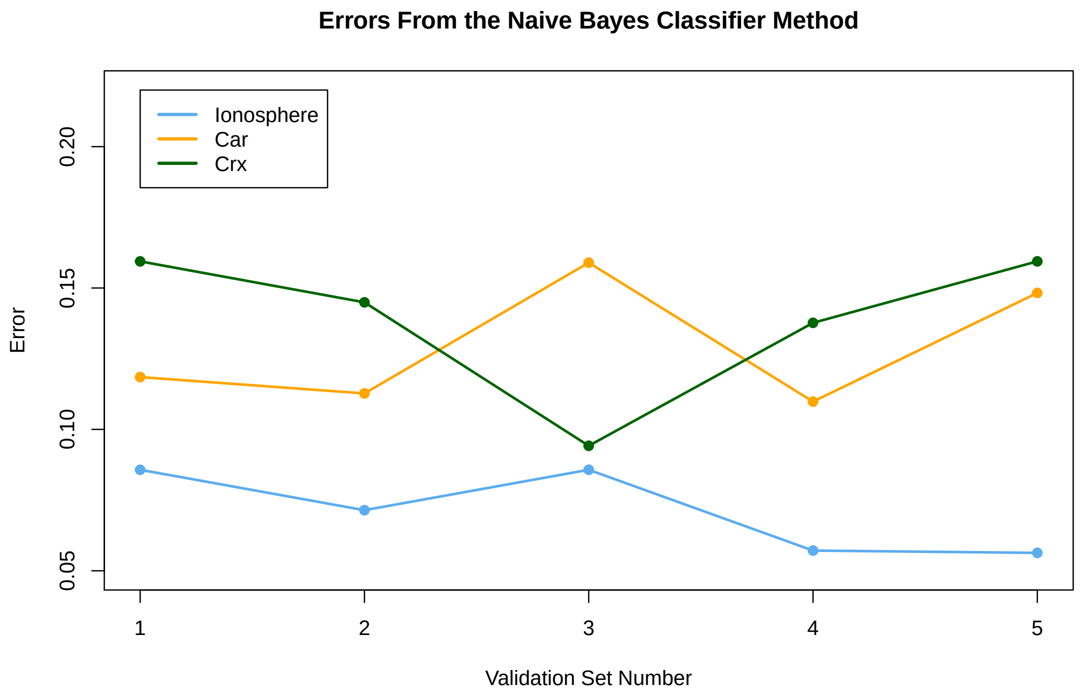
Because of the existence of the categorical data, which are not numerical, we need to pre-process the original data set. In data pre-processing, we also engineer the feature scales in order to yield better results.
In our implementation, for different data types, we use the following methods for pre-processing.

- *Continuous data*: We keep them in continuous form. For the `Credit Approval Data` set, we center all the continuous features and scale them to standard deviation 1.
- *Categorical data*: We use dummy variables to transform them to numerical data.
- *Missing data*: Since the proportion of missing data is relatively small, we replace all the `?` with the mean or the median of the feature where the missing data point is located. The use of mean or median depends on the type of the feature. If the feature is continuous, we choose to use the mean for replacement. If the feature is categorical, we use the median.

Results

Average Error Rates from Naive Bayes Classifier

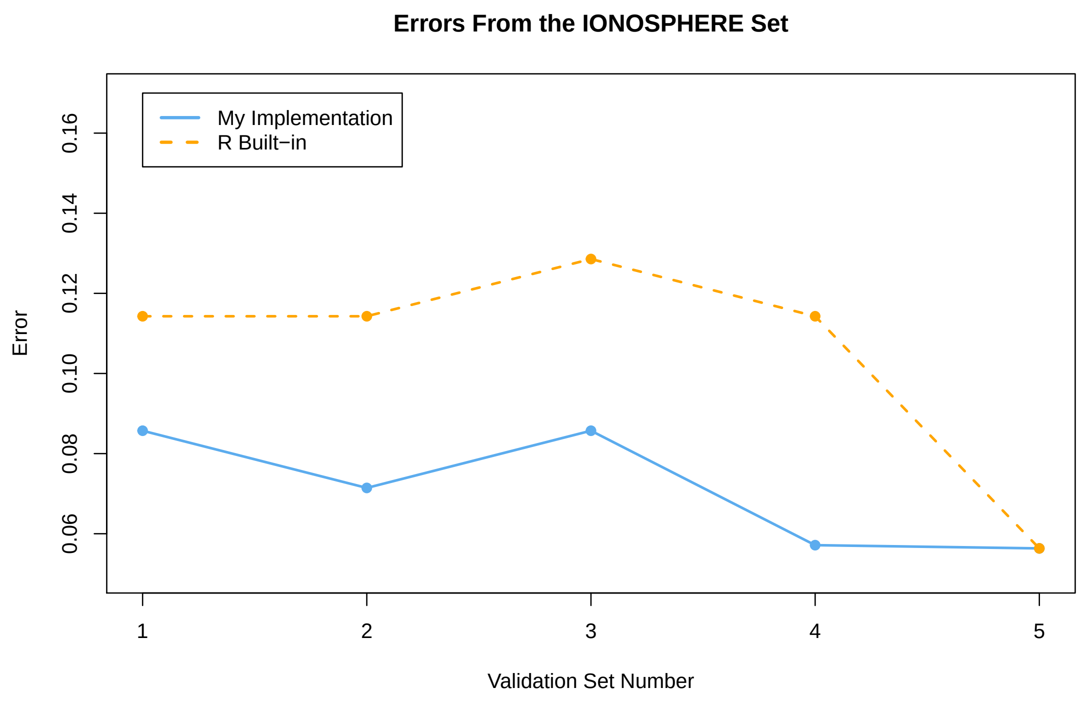
Data Set	Average Error Rate
Ionosphere	0.07126761
Car	0.1296512
Crx	0.1391304

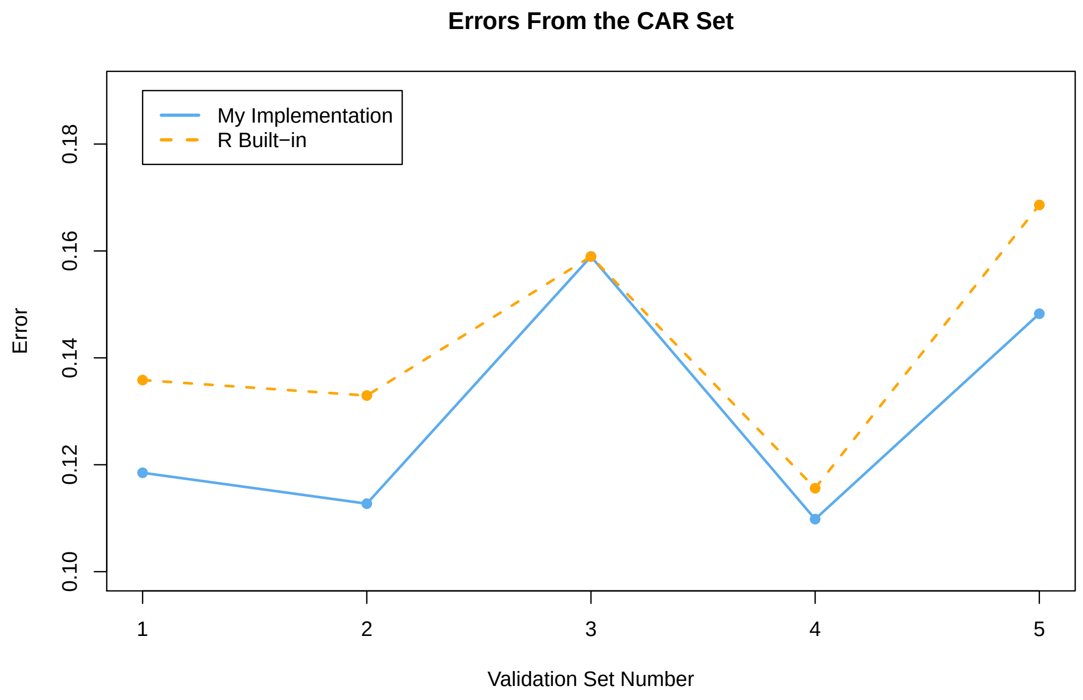


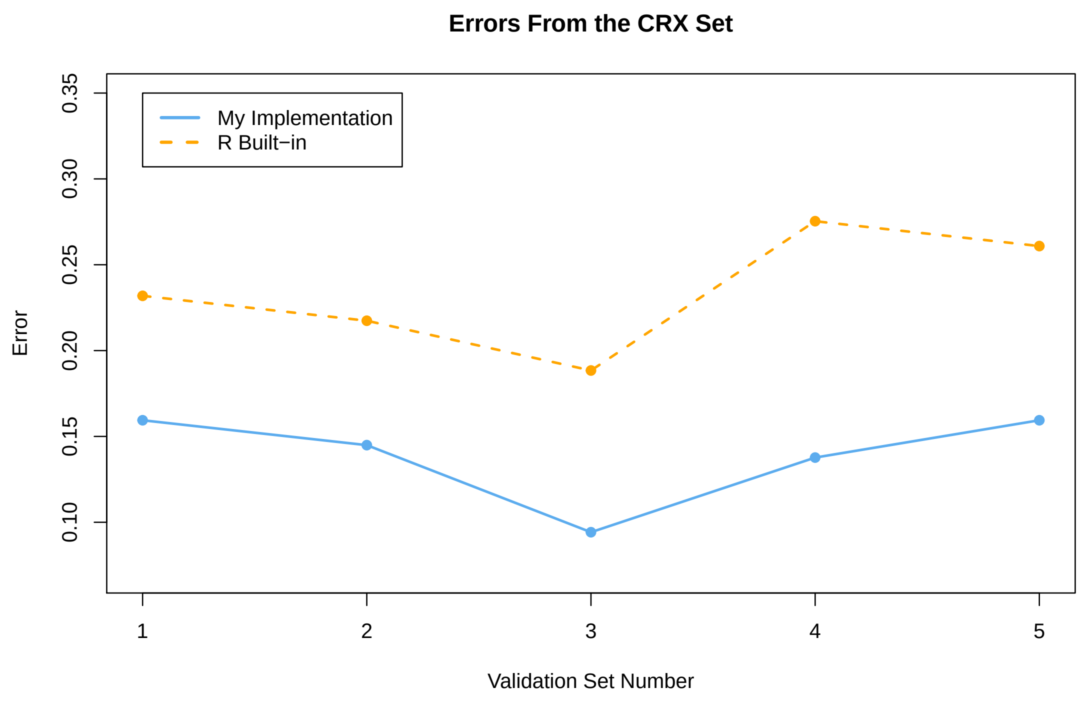
(3.3)

Now we compare the results from my implementation with results from using the `R` built-in implementation.

In the 3 graphs below, the `blue lines` represent errors from using **my implementation** of the Naive Bayes Classifier, while the `orange dotted lines` represent errors from using the **R built-in implementation**.


Errors From the IONOSPHERE Set
My Implementation
R Built−in
Error
0.06
0.08
0.10
0.12
0.14
0.16
1
2
3
4
5
Validation Set Number


Errors From the CAR Set
My Implementation
R Built−in
Error
0.10
0.12
0.14
0.16
0.18
1
2
3
4
5
Validation Set Number



Using the `R` built-in Naive Bayes Method, the average error rate is 0.1055533 for the Ionosphere set, 0.1423915 for the Car set, and 0.2347826 for the Crx set.

We can see from the error plots that, in general, our implementation of the Naive Bayes Classifier performs better than the `R` built-in implementation.

The difference is very pronounced in the `Ionosphere` and `Credit Approval Data` set, where continuous data exist. This is because, in our implementation, we hand picked the class-conditional distributions for the continuous features. For the continuous features, we make the distributional assumption of a normal distribution. But instead of directly using sample mean and variance as the distribution parameters, we experimented various parameter sets and chose the best parameters for each one of the `Ionosphere` and `Credit Approval Data` sets, using exploratory analysis. Note that for each data set, we have only one set of parameters for the distribution assumption, i.e. the parameters are the same across all validation sets.

The difference in the `Car` data set comes from the difference in the treatment of the categorical data. Converting categorical data using dummy variables improved our results. For the R experiments, we used the original data form, since Naive Bayes works with categorical data directly.

Problem 4: Naive Bayes Classifier vs. K-Nearest Neighbors [30 points]

In this question, you are asked to compare Naive Bayes classifier with k-nn algorithm. First, determine the best KNN model for each data set. Then, Make a plot that reveals comparison of two algorithms using test error for each data set. (Total number of figures = 3)

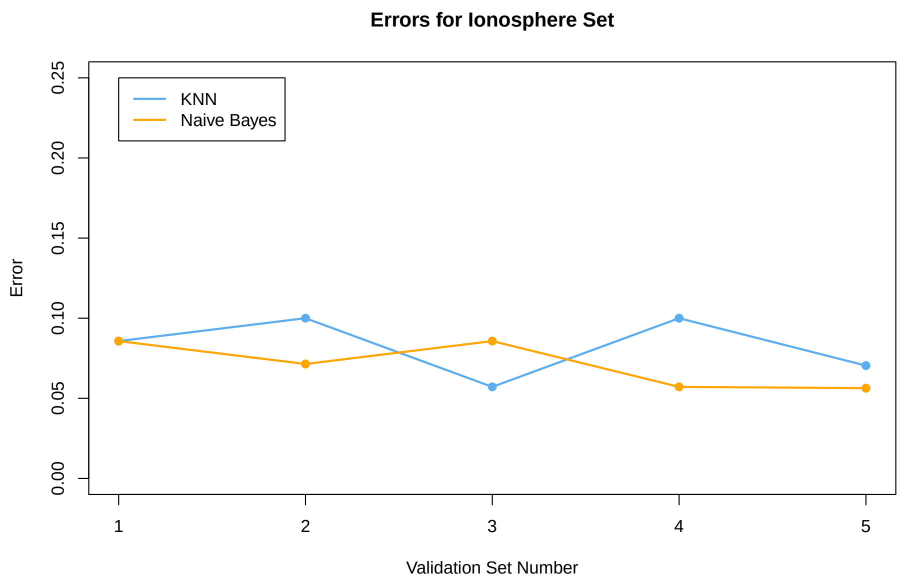
Best Parameters for KNN

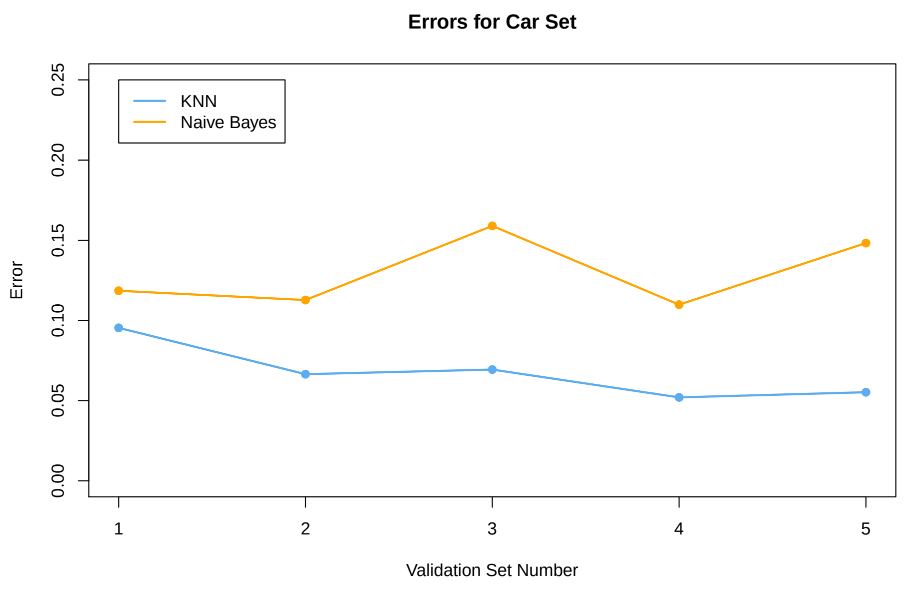
For the Ionosphere data set, the best KNN method is using the 1-norm distance function with $k = 1$.
For the Car data set, the best KNN method is using the 1-norm distance function with $k = 1$.
For the Crx data set, the best KNN method is using the 1-norm distance function with $k = 13$.

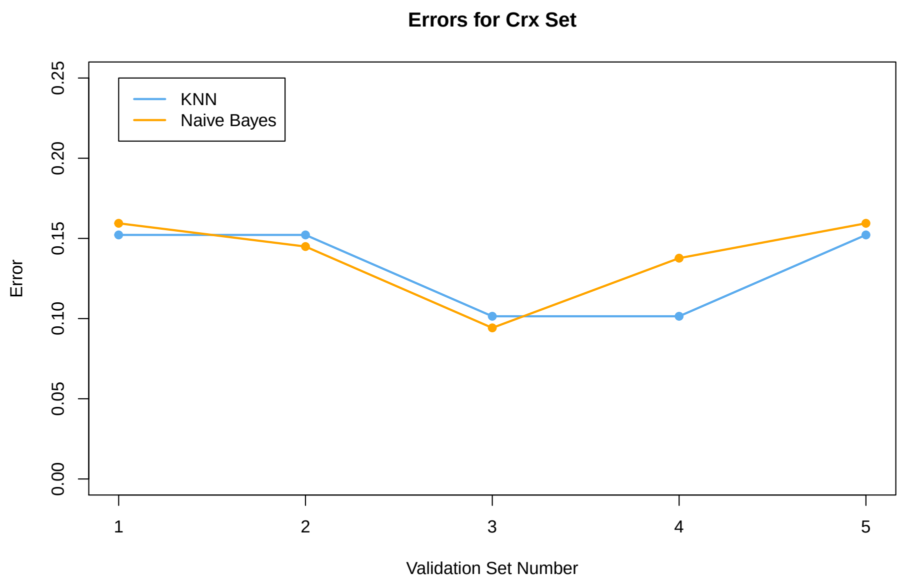
Results Comparison

Average Error Rate

Data Set	KNN	Naive Bayes
Ionosphere	0.08265594	0.07126761
Car	0.06769391	0.1296512
Credit Approval	0.1318841	0.1391304







Conclusion

From our implementations, the KNN method performs better than the Naive Bayes Classifier on the `Car` data set. For the `Ionosphere` and `Credit Approval Data` sets, the two methods have similar performance on the test errors.

Problem 5 [15 points]

From textbook, Chapter 4 exercise 10.g and 13 (only for k-nn and logistic regression)

(10g)

Confusion Matrix

		True Classes		
		Up	Down	Total
Predicted Classes	Up	32	22	54
	Down	29	21	50
	Total	61	43	104

The overall fraction of correct predictions for the held out data is

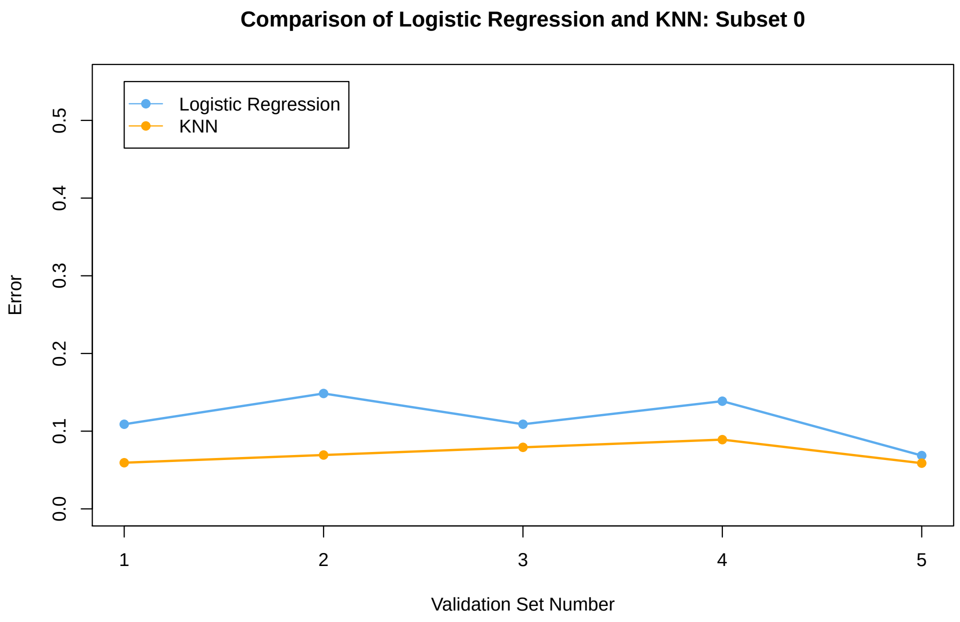
$$\text{Fraction of correct predictions} = \frac{32 + 21}{104} = \frac{53}{104} \approx 0.5096154$$

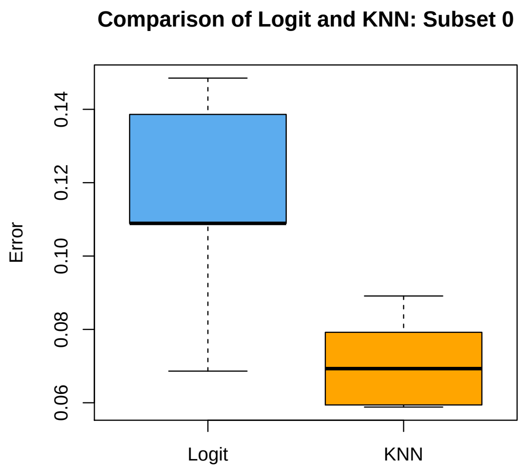
(13)

We experiment the following 6 subsets of predictors: subset 0, subset 1, $\cdots$, subset 5.

Predictor Subset 0

$$\begin{aligned} \text{predictor subset } 0 &= \{\texttt{zn}, \texttt{indus}, \texttt{chas}, \texttt{nox}, \texttt{rm}, \texttt{age}, \texttt{dis}, \texttt{rad}, \texttt{tax}, \texttt{ptratio}, \texttt{black}, \texttt{lstat}, \texttt{medv}\} \\ &= \text{all features as predictors} \end{aligned}$$

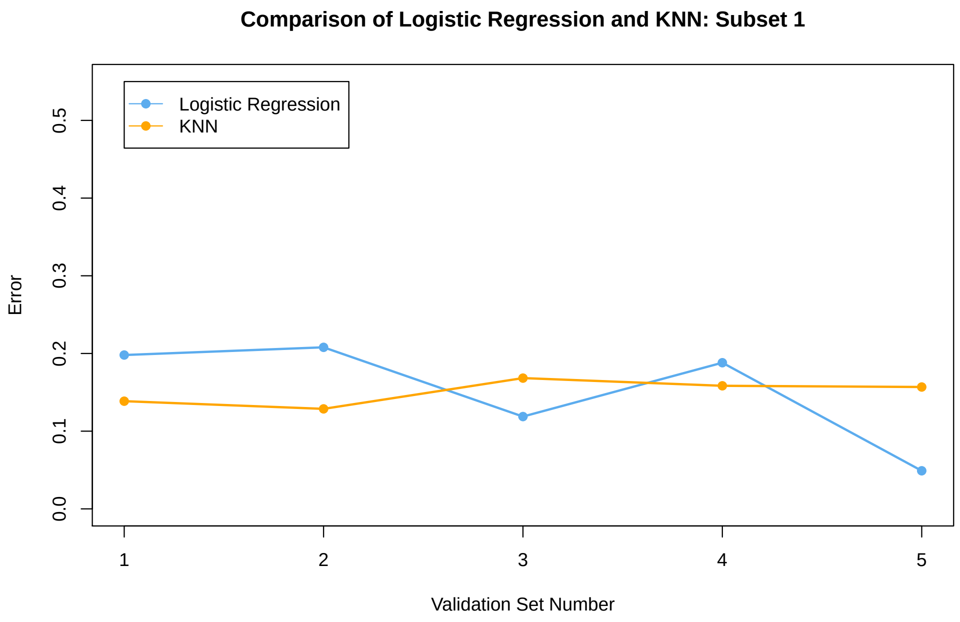


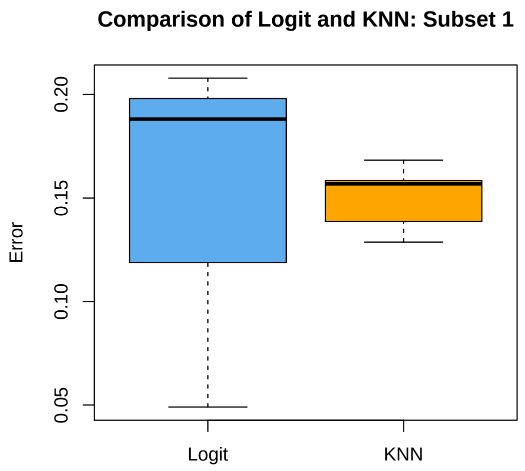


Using this subset of predictors, we can see that KNN performs better across the 5 validation sets. However, the difference in the performance of the logistic regression and KNN are not too different, compared to other subsets we experimented. In fact, as we can see below, this is the subset that the logistic regression performs the best.

Predictor Subset 1

$$\text{predictor subset } 1 = \{\texttt{zn}, \texttt{indus}, \texttt{chas}, \texttt{nox}, \texttt{rm}, \texttt{age}, \texttt{dis}\}$$

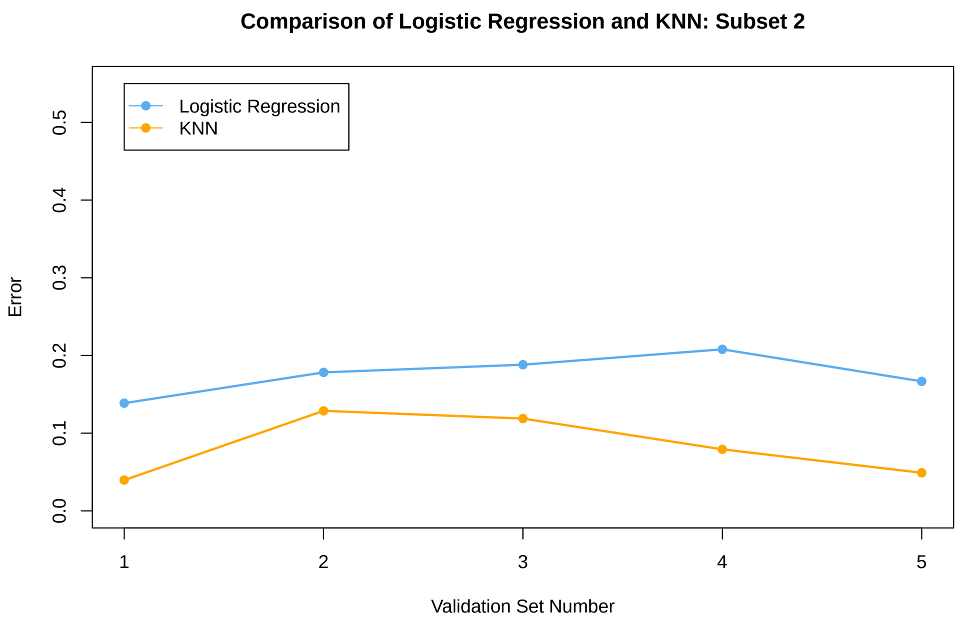


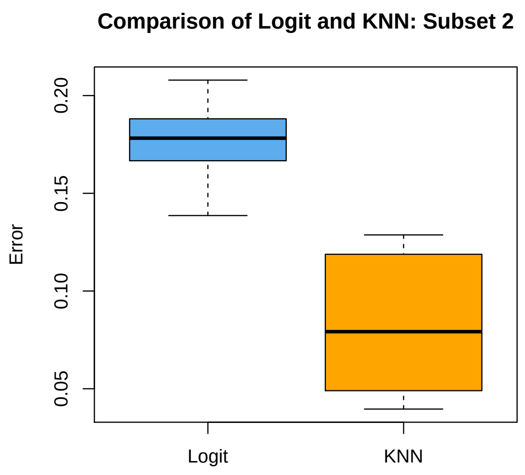


Using this subset of predictors, we can see that the performance of the logistic regression and KNN are similar, with KNN bringing better performance. However, neither of them gives very good error rates, with logit having average error rate of 0.1523782, and KNN having 0.1501844.

Predictor Subset 2

$$\text{predictor subset } 2 = \{\texttt{rad}, \texttt{tax}, \texttt{ptratio}, \texttt{black}, \texttt{lstat}, \texttt{medv}\}$$

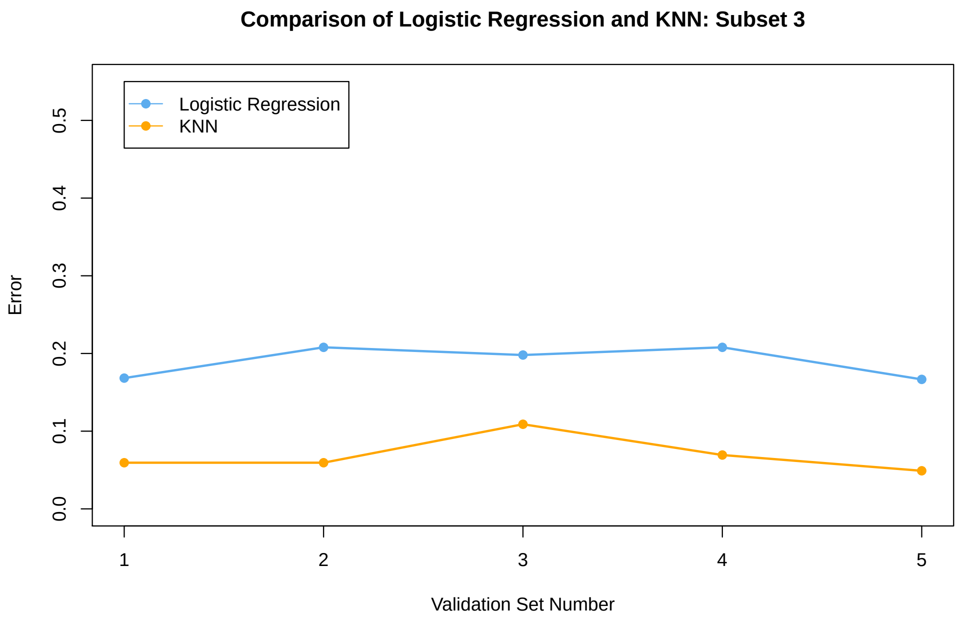


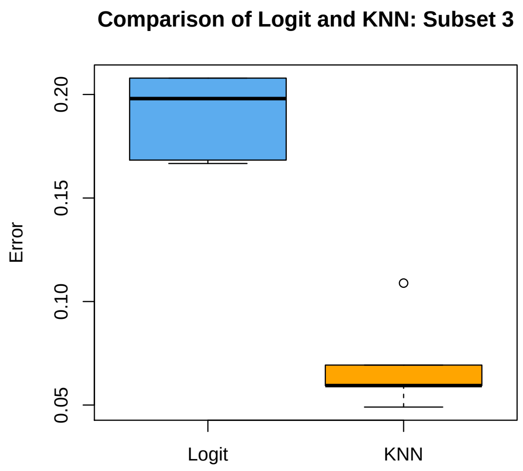


Using this subset of predictors, we can see that KNN performs better across the 5 validation sets, and it gives relatively low error rates, with average error rate being 0.08307126. The logit results have higher error rates, with average error rate being 0.1759076.

Predictor Subset 3

$$\text{predictor subset } 3 = \{\texttt{zn}, \texttt{chas}, \texttt{rm}, \texttt{dis}, \texttt{tax}, \texttt{black}, \texttt{medv}\}$$

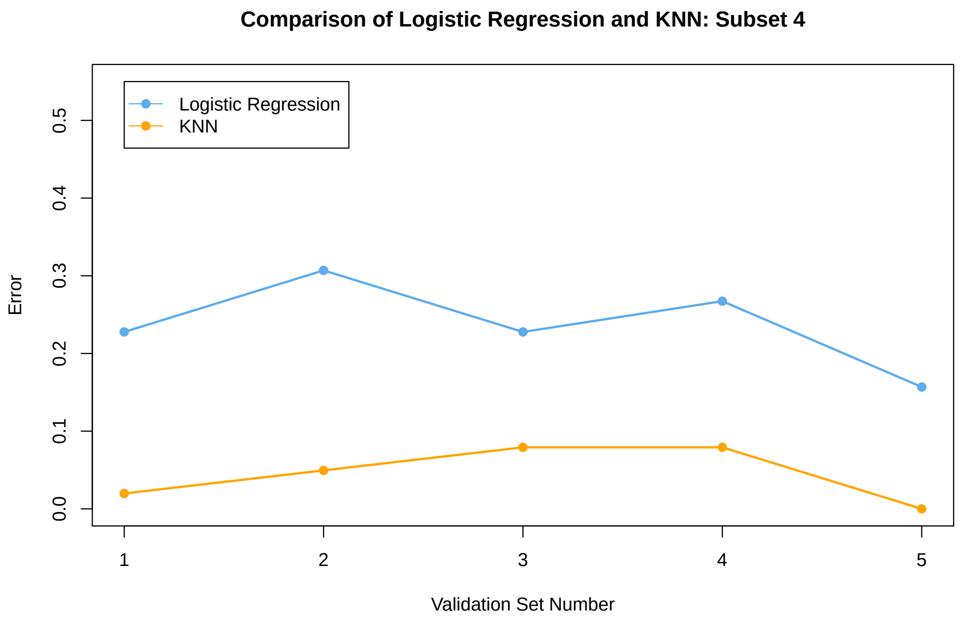


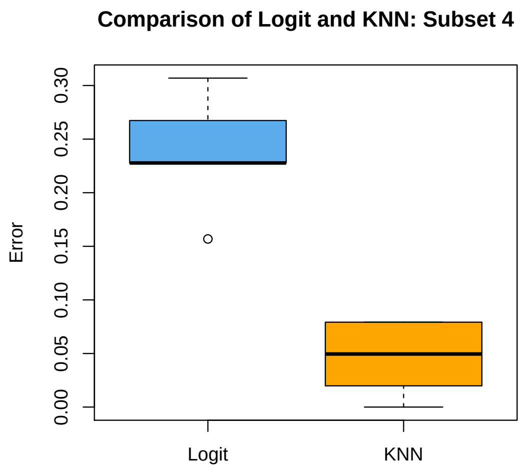


Using this subset of predictors, we can see that KNN performs better across the 5 validation sets. The KNN results are consistent on 5 validation sets and are relative low, with average error being 0.06920986. The logit results are not very satisfactory, with average error being 0.189769.

Predictor Subset 4

$$\text{predictor subset 4} = \{\texttt{tax}\}$$

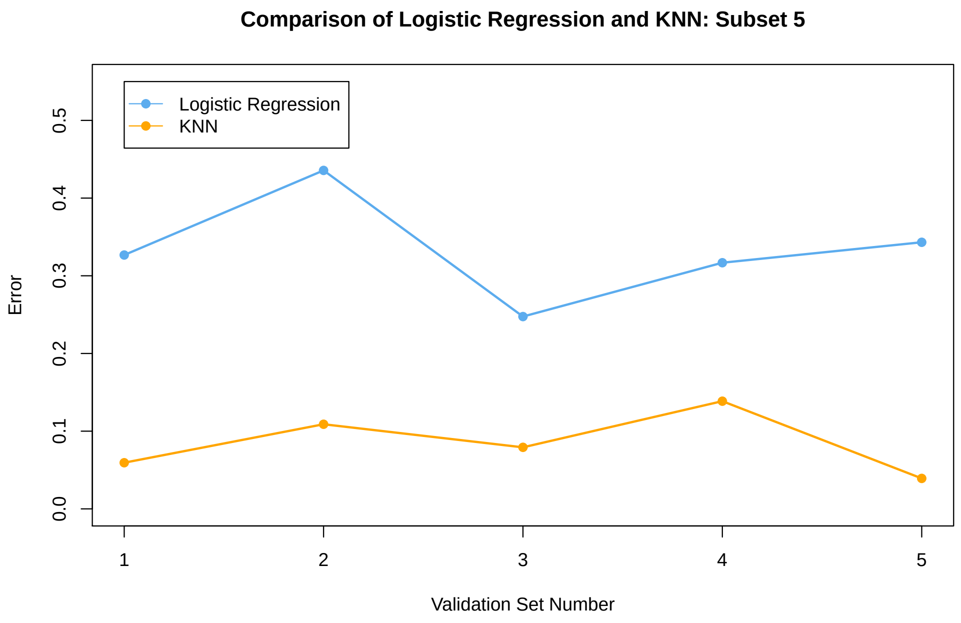


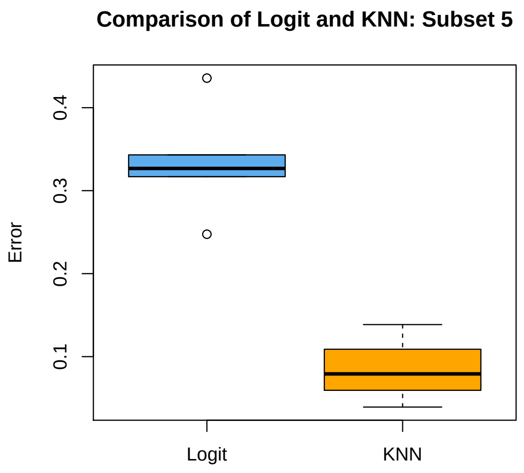


Using this subset of predictors, we can see that KNN performs better across the 5 validation sets. The KNN results are consistent on 5 validation sets and are relative low, with average error being 0.04554455. The logit results are not very satisfactory, with average error being 0.2373131.

Predictor Subset 5

$$\text{predictor subset 5} = \{\texttt{ptratio}\}$$





Using this subset of predictors, we can see that KNN performs better across the 5 validation sets. The KNN results are consistent on 5 validation sets and are relative low, with average error being 0.08507087. The logit results are disappointing, with average error being 0.333974.

Conclusion

For all of the predictor subsets in our experiment, KNN performs better than logistic regression. The performance of logistic regression varies largely with the choice of the subset, while KNN brings good performance even with low number predictors. This shows that this data set does not work as well with the logistic type of generalized linear model, as with the simpler model KNN.

Extra credit (optional) [40 points]

1. You are asked to evaluate the performance of two classifier models, M_1 and M_2. The test set you have chosen contains 26 binary attributes, labeled as A through Z. The table below shows the posterior probabilities obtained by applying the models to the test set. (Only the posterior probabilities for the positive class are shown). As this is a two-class problem, P(-) = 1 - P(+) and P(-|A,...,Z) = 1 - P(+|A,...,Z). Assume that we are mostly interested in detecting instances from the positive class.

   (a) Plot the ROC curve for both M_1 and M_2. Which model do you think is better. Explain your reasons?

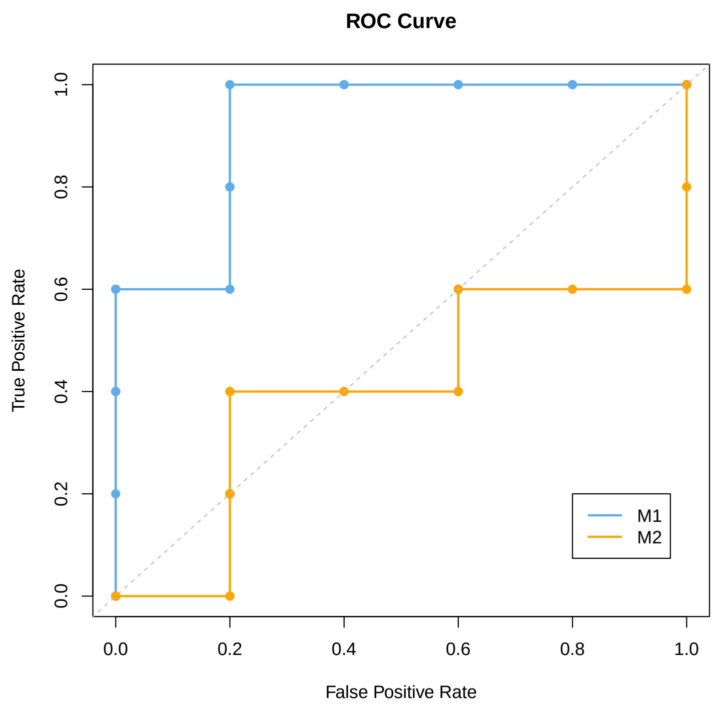
   (b) For model M_1, suppose you choose the cutoff threshold to be $t = 0.5$. In other words, any test instances whose posterior probability is greater than t will be classified as a positive example. Compute the precision, recall, and F-measure for the model at this threshold value.

   (c) Repeat the analysis for part (c) using the same cutoff threshold on model M_2. Compare the F-measure results for both models. Which model is better? Are the results consistent with what you expect from the ROC curve?

   (d) Repeat part (c) for model M_1 using the threshold $t = 0.1$. Which threshold do you prefer, $t = 0.5$ or $t = 0.1$? Are the results consistent with what you expect from ROC curve?

2. Student/s who design/s the best either Naives Bayes classifier or KNN algorithm for the given data sets will receive 20 points.

(a)



The first model performs better.

We can see this by looking at the position of the curves. In this ROC curve, the top left area represents high true positive rate and low false positive rate, which the bottom right area represents low true positive rate and high false positive rate. Therefore, curves near the top left corner represents more desirable performance, while those near the bottom right corner represents poor performance.

In our problem, the M1 curve hugs the top left corner. The M2 curve is entirely below the M1 curve and mostly locates below the diagonal line, towards the bottom right corner. Therefore, M1 performs better than M2.

Alternatively, we can look at the area under the curve or AUC. Obviously, M1 has larger AUC than M2. Therefore, M1 performs better than M2.

(b) Using $t = 0.5$, for model 1

$$\text{TP} = 3, \quad \text{FP} = 1, \quad \text{TN} = 4, \quad \text{FN} = 2$$

Therefore,

$$\text{precision} = \frac{\text{TP}}{\text{P*}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 0.75$$

$$\text{recall} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{FN} + \text{TP}} = \frac{3}{2+3} = 0.6$$

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2}{3} \approx 0.6666667$$

(c) Using $t = 0.5$, for model 2

$$\text{TP} = 1, \quad \text{FP} = 1, \quad \text{FN} = 4$$

Therefore,

$$\text{precision} = \frac{\text{TP}}{\text{P*}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{1}{1+1} = 0.5$$

$$\text{recall} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{FN} + \text{TP}} = \frac{1}{4+1} = 0.2$$

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2}{7} \approx 0.2857143$$

Models with higher precision and recall, i.e. true positive rate, are considered to have the better performance. The F-measure is the harmonic mean of precision and recall. Therefore, it is also positively correlated with model performance.

From the results above, we can see that all three scores, precision, recall and F-measure, are higher in model 1. Therefore, model 1 is better. This is consistent with the result from the ROC curve.

(d) Using $t = 0.1$, for model 1

$$\text{TP} = 5, \quad \text{FP} = 4, \quad \text{TN} = 1, \quad \text{FN} = 0$$

Therefore,

$$\text{precision} = \frac{\text{TP}}{\text{P*}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{5}{5+4} = \frac{5}{9} \approx 0.5555556$$

$$\text{recall} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{FN} + \text{TP}} = \frac{5}{0+5} = 1$$

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{5}{7} \approx 0.7142857$$

From these scores, we can see that $t = 0.1$ is better in recall or true positive rate, while $t = 0.5$ is better in precision. Choosing either over the other would require some kind of trade-off. But if we use the F-measure to make the decision, we would choose $t = 0.1$ over $t = 0.5$, as it has a higher F-measure.

From the ROC, we can see that the point corresponding to $t = 0.1$ for model 1 is $(0.8, 1)$, the point corresponding to $t = 0.5$ for model 1 is $(0.2, 0.6)$. The conclusion we can draw is similar with the one we got from comparing precision, recall and F-measure values.

What to Turn-in (Submission Instructions)

Put the below files in a zipped folder for your submission. The zipped folder should be named as "usename-section number", i.e., hakurban-P556

1. The *tex and *pdf of the written answers to this document.
2. Code and Data
   (a) Question 1: crossValidation.R, output of cross validation: training and test data sets
   (b) Question 2.1: knn.R, Question 2.3: knnValidation.R
   (c) Question 3.1: naiveBayes.R, Question 3.3: naiveBayes-Validation.R
3. A README file that explains how to run your code and other files in the folder