CS 498 DAF Homework 1
Atul Nambudiri - nambudi2

3.1a:
Data Pre-processing: I didn't preprocess the data. However, any value that was NA in any calculations I ran, I just ignored.
Implementation: For this question, I did 10 rounds(splits) of Naive Bayes, with an 80/20 split of training data to testing data in each round
For each round, I split the training data into two separate sections, those that were classified as 1, and those that were classified as 0. I found the mean/sd of each feature for the two sections. I then found P(xi|y) for each feature xi. I did this by getting the Log probabilities of the Normal Distribution for each of the features. I summed all of these values up, and then added in Log(P(y)) for both the y=1 and y=0 sections. This gave me the probabilities of each sample being 1, or 0.
I then classified each sample as either 1 or 0, based upon which of these was higher.

I repeated the process for the testing set, using the mean and sd deviation I had calculated earlier. I then calculated the accuracy for each round, and stored the values.
Results: I took the mean and standard deviation of the accuracy results for each of the 10 rounds I ran.
Training Set Accuracies: [0.7622150, 0.7719870, 0.7524430, 0.7719870, 0.7768730, 0.7638436, 0.7654723, 0.7654723, 0.7540717, 0.7622150]
Training Set Mean: 0.764658
Training Set SD: 0.007687189
Testing Set Accuracies: [0.7777778, 0.7254902, 0.7712418, 0.7712418, 0.7124183, 0.7124183, 0.7581699, 0.7516340, 0.7973856, 0.7385621]
Testing Set Mean: 0.751634
Testing Set SD: 0.02890305
The accuracy for both the training and testing set are fairly close, around 75%

3.1b:

Data Pre-processing: I didn't preprocess the data. However, any value that was NA in any calculations I ran, I just ignored.

Implementation: Once I read my data in from the file, I looped through all the rows and columns. For any feature in columns 3, 5, 6, and 8, if the value was 0, I just set the value to be NA. Other than that, the implementation is the same as 3.1a

Results: I took the mean and standard deviation of the accuracy results for each of the 10 trials I ran.

Training Set Accuracies: [0.7508143, 0.7589577, 0.7573290, 0.7654723, 0.7508143, 0.7719870, 0.7687296, 0.7638436, 0.7638436, 0.7475570]

Training Set Mean: 0.7599349

Training Set SD: 0.008240463

Testing Set Accuracies: [0.8039216, 0.7647059, 0.7189542, 0.6928105, 0.7908497, 0.6993464, 0.7320261, 0.6993464, 0.7712418, 0.7581699]

Testing Set Mean: 0.7431373

Testing Set SD: 0.04017821

The accuracy for both the training and testing set are fairly close, around 76%. The preprocessing did not make any substantial difference in the accuracy of the classifier.

3.1c:

Data Pre-processing: None

Implementation: This time, I only did one split. I split the data 80/20 into a training and testing set. I then used the caret implementation of Naive Bayes, which uses a Normal Distribution by default, and trained a classifier on the training set. I used 10-fold K-fold Cross-Validation for the classifier. I found a predicted accuracy from this. I then tested the classifier on the test data that I held back, and verified that my predicted accuracy matched up with expected results.

Results: The classifier predicted the accuracy of itself to be 0.7733051

When I tested out the classifier on the held back testing data, I got the following confusion matrix:

	Reference	
Prediction	0	1
0	86	26
1	17	24

This corresponds to an overall accuracy of 0.719. The accuracy that was a predicted was a little higher than the actual, but not by too much.

The classifier seems to classify negative(0) example better than positive(1) examples. This could be due to a smaller number of positive examples in the training/testing sets.

3.1d:
Data Pre-processing: None
Implementation: This time, I only did one split. I split the data 80/20 into a training and testing set. I then used svmlight to train an svm classifier on the training set, with the default parameters. I then tested the classifier on the test data that I held back, and calculated the accuracy from it.
Results: The classifier predicted the accuracy of itself to be 0.7733051
When I tested out the classifier on the held back testing data, I got the following confusion matrix:

	Reference	
Prediction	0	1
0	90	26
1	7	30

This corresponds to an overall accuracy of 0.7843, which is similar to what I got from the Naive Bayes classifiers.

3.3a:
Data Pre-processing: Before I ran my code, I looked through the data for any missing attributes which are symbolized by a questions marked. I removed any row that this appeared in.
Implementation: Once I read the data in, I looked at all the rows in the data. If any row was classified as class > 0, I changed the class to be 1. This way, I could run Naive Bayes with just two classes, 0, and 1, with 1 representing all values greater than 1.

For this question, I did 10 rounds(splits) of Naive Bayes, with an 85/15 split of training data to testing data in each round.
I used the caret implementation of Naive Bayes, which uses a Normal Distribution by default, and trained a classifier on the training set. I used 10-fold K-fold Cross-Validation for the classifier. I then tested the classifier on the test data that I held back, and calculated the accuracy of the classifier. I stored this value for all 10 rounds
Results: I took the mean and standard deviation of the accuracy results for each of the 10 rounds I ran.
Accuracies: [0.8444444, 0.8888889, 0.8444444, 0.8888889, 0.7555556, 0.8222222, 0.7777778, 0.8666667, 0.7777778, 0.8222222]
Mean: 0.8288889
SD: 0.04690708

3.3b:
Data Pre-processing: Before I ran my code, I looked through the data for any missing attributes which are symbolized by a questions marked. I removed any row that this appeared in.
Implementation: Almost the same as 3.3b. The only difference is that I didn't change all classes > 0 to be equal to 1. This way, I can find the accuracy of classification of all 5 classes.
Results: I took the mean and standard deviation of the accuracy results for each of the 10 rounds I ran.
Accuracies: [0.5555556, 0.6444444, 0.5777778, 0.5777778, 0.6222222, 0.6888889, 0.5333333, 0.4888889, 0.5111111, 0.5777778]
Mean: 0.5777778
SD: 0.06108305

This is the Confusion Matrix from one round:

	Reference				
Prediction	0	1	2	3	4
0	19	5	2	0	0
1	2	2	3	2	0
2	0	1	1	2	0
3	0	1	2	1	2
4	0	0	0	0	0

These results are significantly worse than the previous one. It appears the classifier is good at classifying objects of class 0 correctly, but not much else. I attribute this partially to the fact that there are much less sample of class 1-4 than of class 0, making it harder to classify them.