Machine Learning

Assignment 2

Shadab Zafar
2017MCS2076

Q1: Naive Bayes

a.)

Naive Bayes Algorithm Implementation

Training Accuracy: 68.90%
Testing Accuracy: 37.95%

For the first part, I tried various ways of cleaning the data from just lower casing the text to removal of HTML tags like **
** and even removing everything except alphanumeric characters but none of them made any significant change to the accuracy.

b.)

Random Accuracy: 12.5%
Majority Accuracy: 20.08%

Testing Accuracy: 37.95%

Improvement:

Over Random: 25.45%
Over Majority: 17.87%

c.)

Confusion Matrix

Diagonal values indicate the classes that are correctly classified. In our case, Class 1 has the highest diagonal value which means that our classifier is able to distinguish class 1 from all other classes correctly.

As we can observe from the first and last diagonal values, the classifier mostly predicts class 1 and 10 correctly. We also see that reviews from classes 1-4 get predicted into 1 and from classes 7-10 get predicted into 10.

At first I thought that this was happening because of the prior probabilities of these classes, so I tried removing the effect of the priors, but that resulted in the same accuracy, showing that is not the case.

d.)

After Cleaning the Training & Testing Data

Testing Accuracy: 38.3%

The accuracy does not change much even after stopword removal and stemming. This shows that even after cleaning most of the classes have similar words which is not enough to distinguish amongst the classes.

e.)

Feature Engineering

At first, I tried keeping only the most common 5000 words in each class, but the accuracy decreased (to 29%, ~9%loss), that indicates that maybe each class has same set of words. I also tried with 3500 most common words per class and 3500 least common words, but that wasn't much good either. (to 30%, ~8% loss)

After using both unigrams and bigrams: Training: 92.988%, Testing: 37.992%, which shows that the model has overfit!

Using only bigrams resulted in: Training: 99.556%, Testing: **39.324%**

I then tried TF-IDF score: Training: 76.376%, Testing: 35.568%.
TF-IDF with bigrams only: Training: 99.976%, Testing: 29.772%.
TF-IDF with both uni and bigrams: Training: 99.916%, Testing: 37.292%

All of these methods overfit :(

Just to confirm whether my implementations were correct, I tried scikit_learn's TF-IDF & Naive Bayes methods and found accuracies: Training: 40.820%, Testing: 34.248% (using uni/bi/tri grams!)

Q2: SVM

a.)

SVM using Pegasos with SGD

I implemented randomised mini-batch gradient descent with a batch size of 100 with the convergence criteria: `all(abs(`W_{new} `-` W_{old}`) < 10**-3)`

b.)

One-vs-One Classification
(using Pegasos)

Without scaling the data, it takes ~7 minutes to train classifiers and find the accuracy, which comes out to be:

Train Accuracy: 98.44%
Test Accuracy: 92.23%

But after scaling, total run time reduces to ~1 minute, and accuracies come out to be:

Train Accuracy: 93.31%
Test Accuracy: 92.17%

So we find that without scaling it takes longer for the model to build and it overfits a bit.

These results use C = 1.0 and in case of ties, we choose the label with the highest digit value.

c.)

Multi-class SVM (using libsvm)

Linear Kernel:

Unscaled data:

Training Time: ~55 seconds

Training Accuracy: 100%
Testing Accuracy: 91.31%

Scaled data:

Training Time: ~45 seconds

Training Accuracy: 98.79%
Testing Accuracy: 92.78%

Comparing the accuracy of linear case with Pegasos, we see that it is almost similar.

Gaussian Kernel:

Unscaled data:

Training Time: ~10 minutes

Training Accuracy: 100% (~5 minutes)
Testing Accuracy: 8.92% (~3 minutes)

Scaled data:

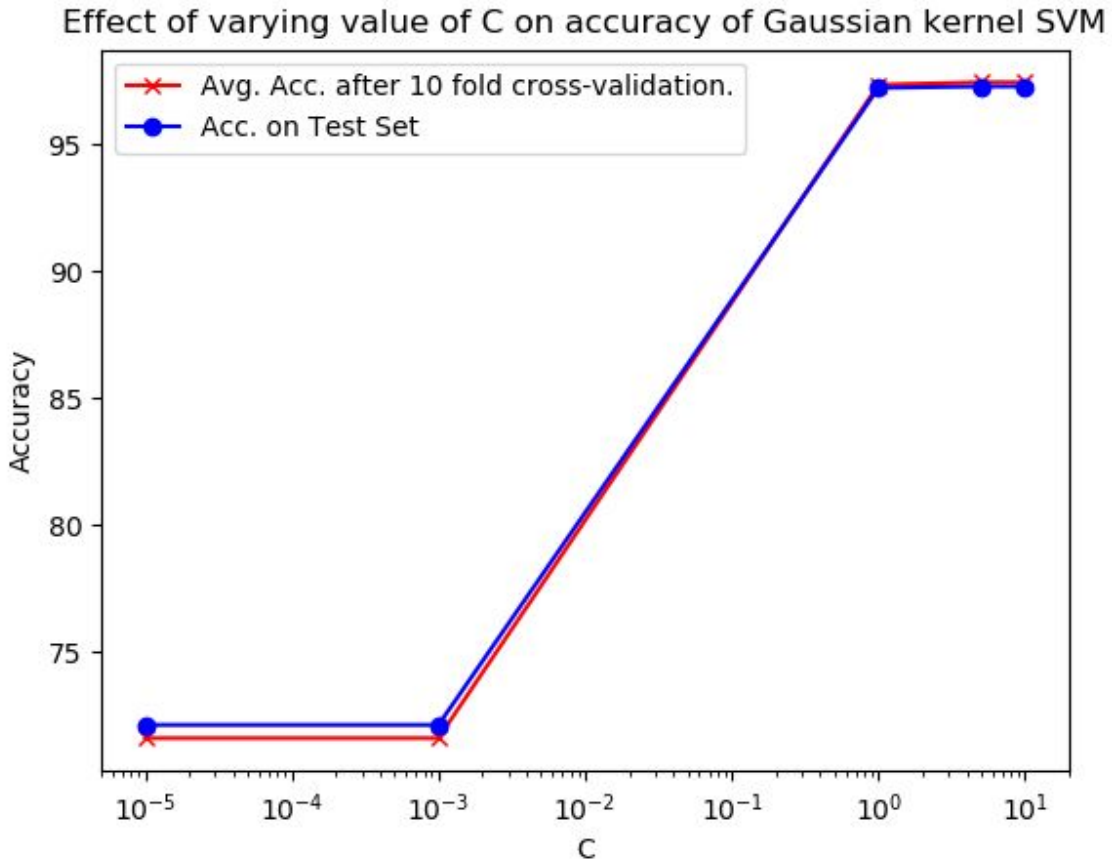
Training Time: ~3 minutes

Training Accuracy: 99.92% (~3 minutes)
Testing Accuracy: 97.23% (~1 minute)

d.)

Model Selection using 10-fold Cross Validation

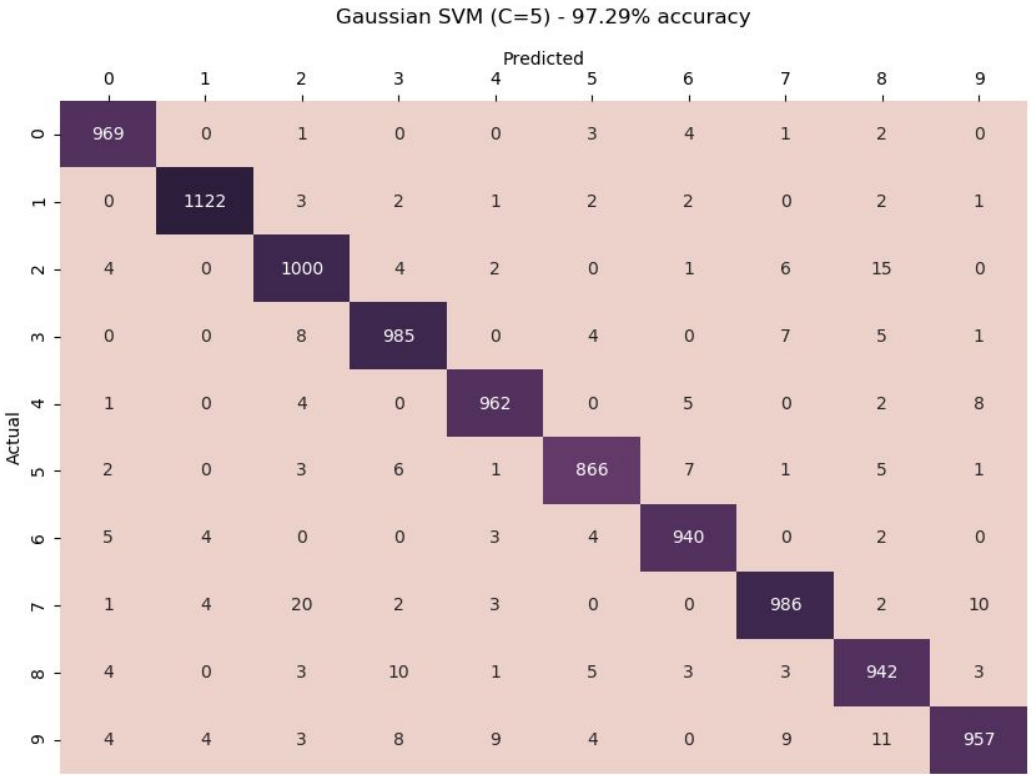
C	0.00001	0.001	1	5	10
CV. Acc.	71.59	71.59	97.355	97.455	97.455
Test. Acc.	72.11	72.11	97.23	97.29	97.29



We observe that low values of C result in poor while higher values give good accuracy. Both C = 5 or C = 10 give same accuracy on validation set and test set. Both of these take around the same time to run!

e.)

Confusion matrix



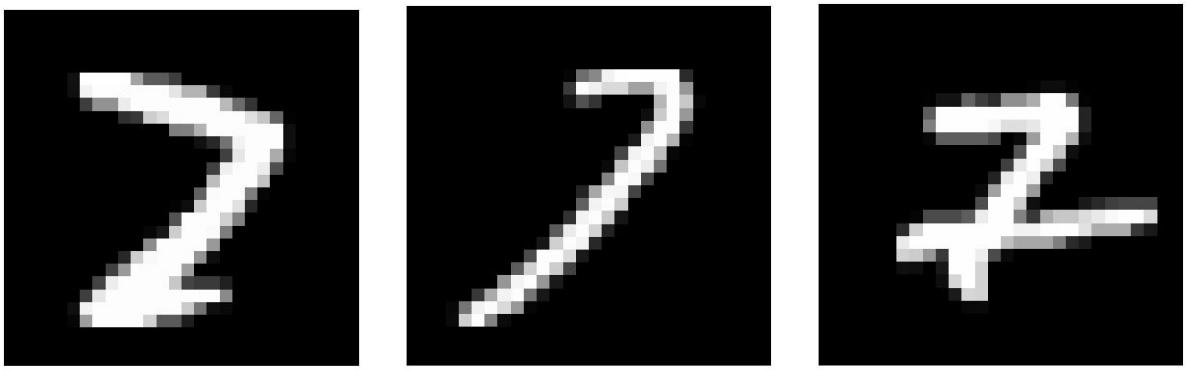
For both C = 5, 10 same confusion matrix is observed showing that increasing the value of C doesn't have much effect.

The misclassification rate of each class is:

0	1	2	3	4	5	6	7	8	9
1.12	1.15	3.1	2.48	2.04	2.91	1.88	4.09	3.29	5.15

Since class 9 has highest miss rate, it is the most difficult to classify.

We see that the highest non-diagonal entry in the above confusion matrix is between 7 & 2. So, numbers that are actually 7 are being labeled as 2. Some examples of such numbers:



The first example may be difficult to classify for a human too.

Similarly, some numbers that are actually 9 are being labeled as 8: