Part#2:

Code file: Assignment2\_Part2.py

1. Processing data using z-score (line 51), which basically scale the given data by mean and standard deviation. It is useful to use z-score in the given data as there is no missing or unknown data point (line 42 in code is detecting whether there is nan value). Also, it seems there is no outliers, and there are four categories related to each feature. The visualization of features is shown in figures 1 and 2 below, and each histogram represent a feature with the respected data points.

For using min-max, we can define the range of scaling (as shown in line 51) and thus get positive scale only. One problem of min-max is that the probability of having out-of-bound error if the new data falls outside the range of the previous data.

In general, we need to do normalization, so we can have same range for all given data thus we seek relation. This is important especially before doing any classification, so we will not cause any bias or variance between features of different range.

Finally, as it shown in figure 3, the distribution of labels (-1, +1) has the ratio of 1063:1137 or 0.93.

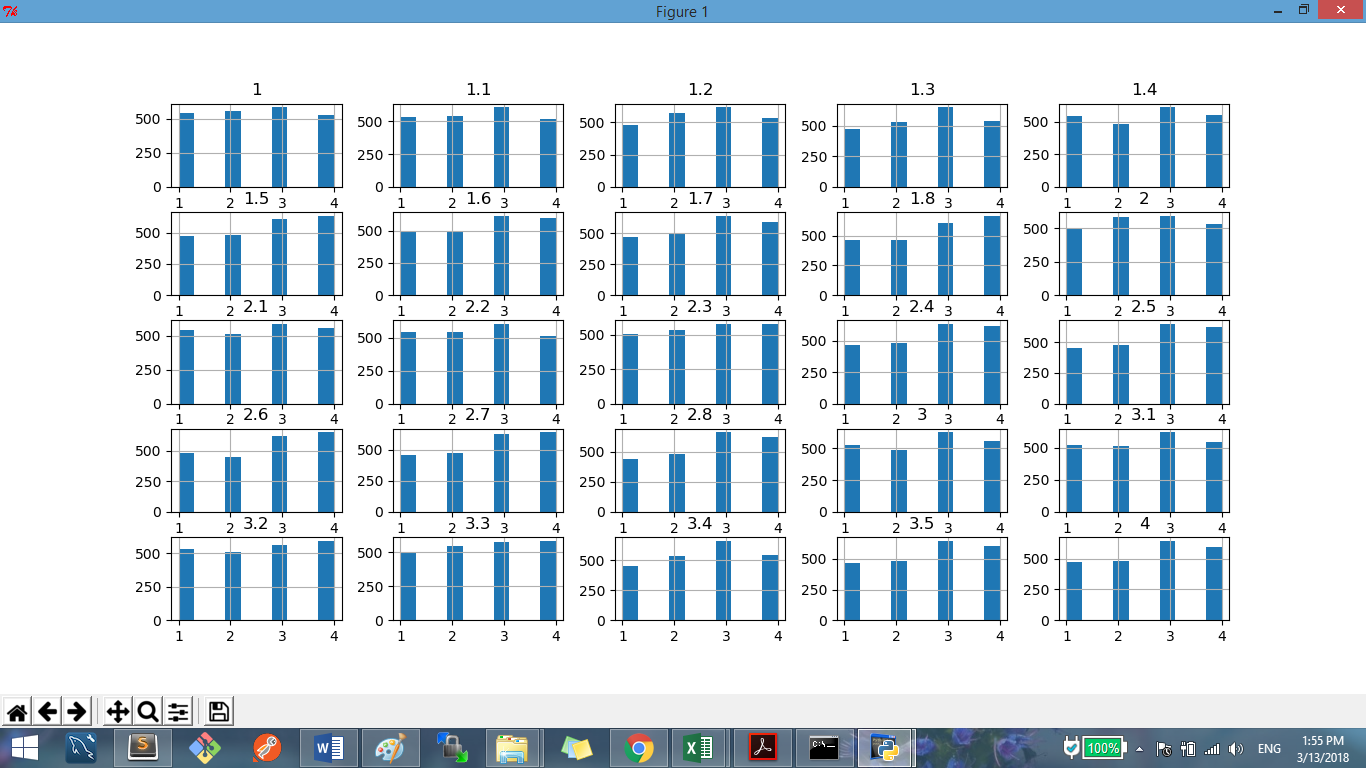


Figure 1. Visualizing features from 1 to 26.

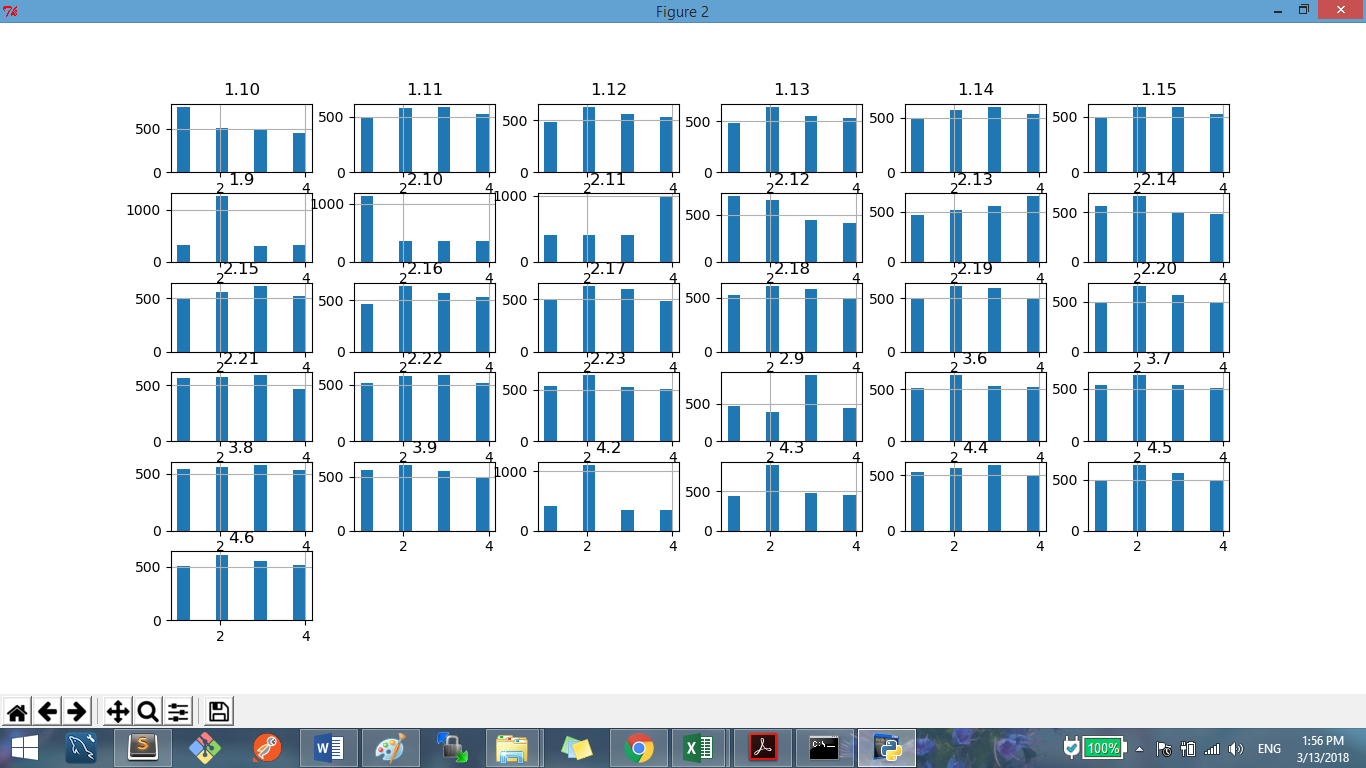


Figure 2. Visualizing features from 27 to 57.

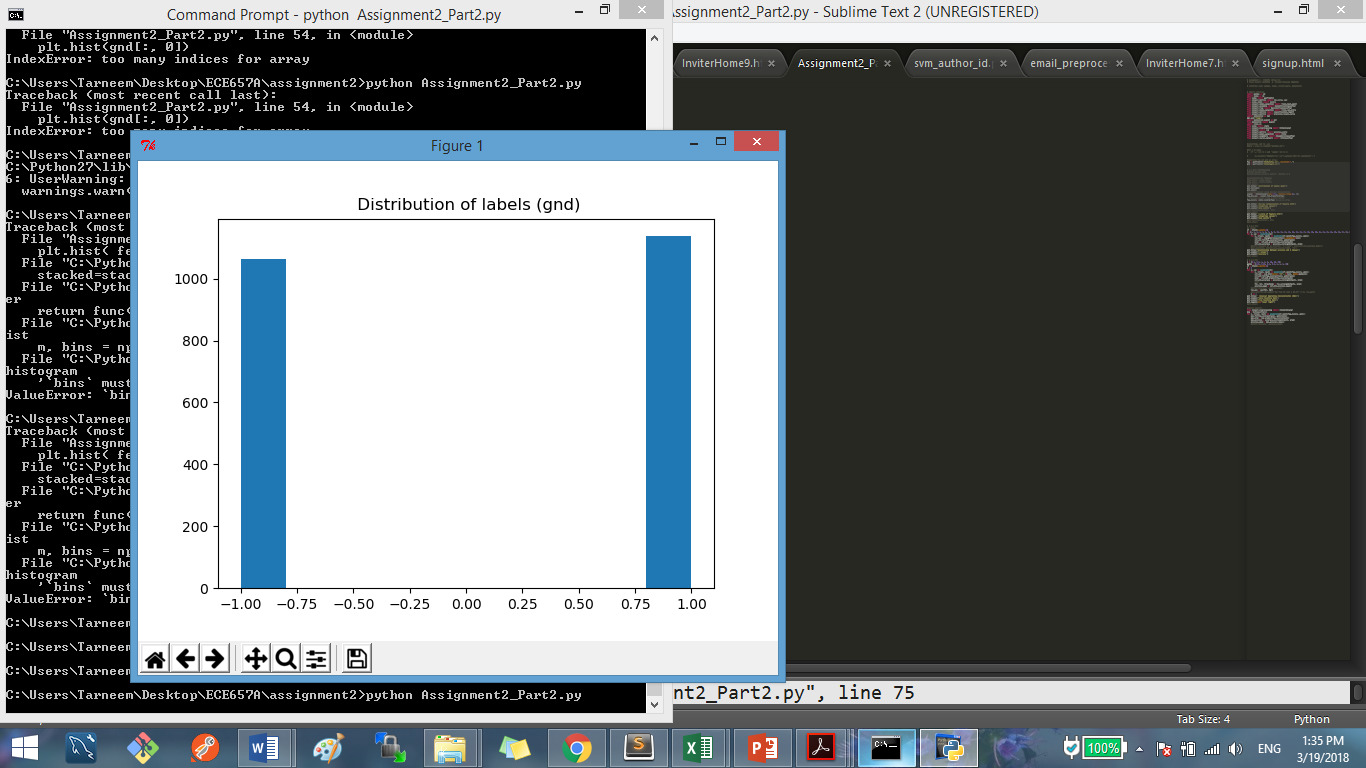


Figure 3. Distribution of labels (gnd).

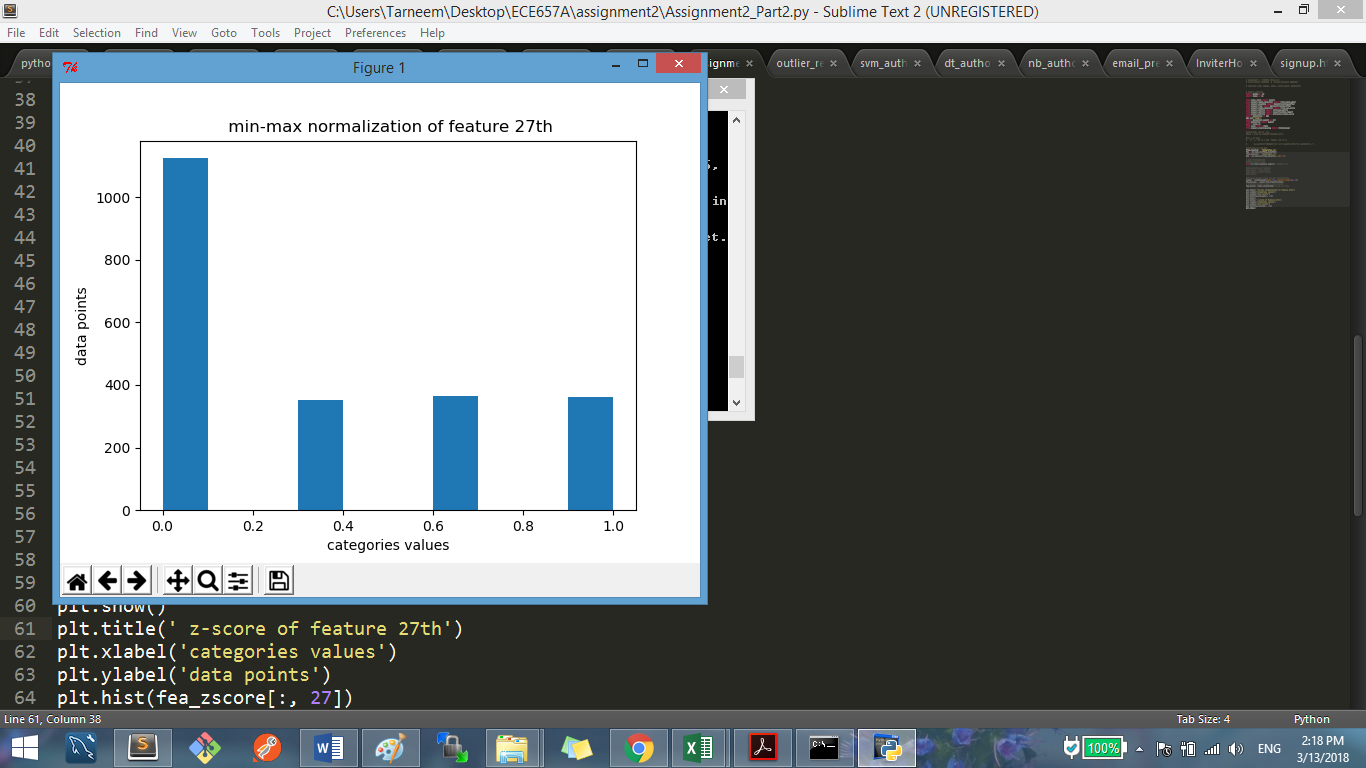
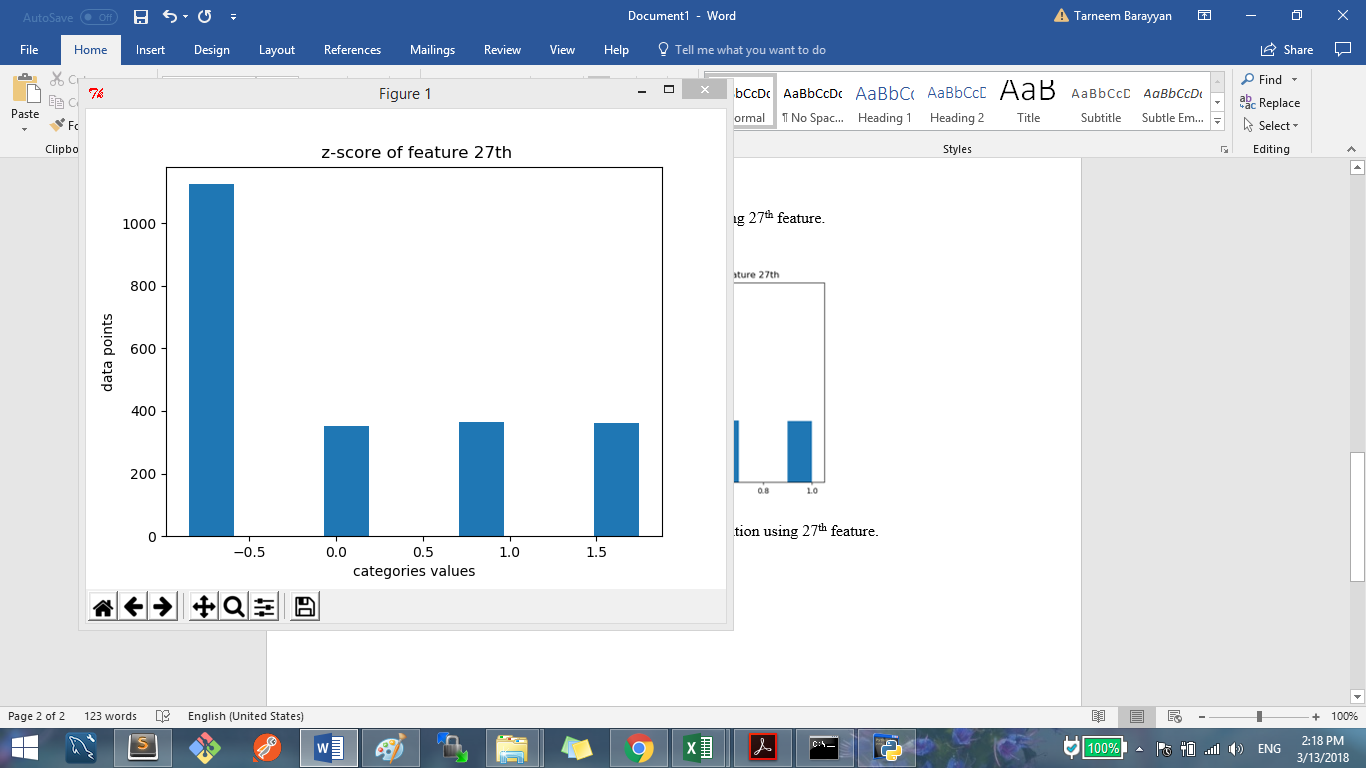


Figure 4. example of z-score and min-max normalization for 27th feature.

2) At first, we split data into test and train sets, using 50% used for testing dataset. Then we will be using training sets for part a and b and using k\_fold= 5. Code lines (79-80). The reason we choose 50% split is because we will apply cross\_validation with kfold on training set, which will divided to sub set of train and test sets. The splitting test set will be used for final evaluation in part 3-c.

a) code lines (71 -86) represents knn algorithm. The relationship between k value chosen in knn and the accuracy obtained is shown in figure 5. Accuracy for k from 1 to 31 are shown in figure 5. As it shown, it seems k with odd value is doing better than even value. For example, when k= 1 the accuracy is 76.13% while k=2 has accuracy 69.77%. k of values 29, 25, 7, and 5 have same accuracy of 76.5%. The highest accuracy achieved is 77.7 where k= 13.

The reason we use test value to do the evaluation on training set is because k means number of nearest point to the given data point to determine its labels. Thus, in training we are given the labels of the neighbors and hence we can make the voting of k to classify given data points, so we want to find the best parameter that gave us best results. Test dataset is used after we find the parameter that worked the best so now we can analysis and compare the performance of this algorithm to another one.

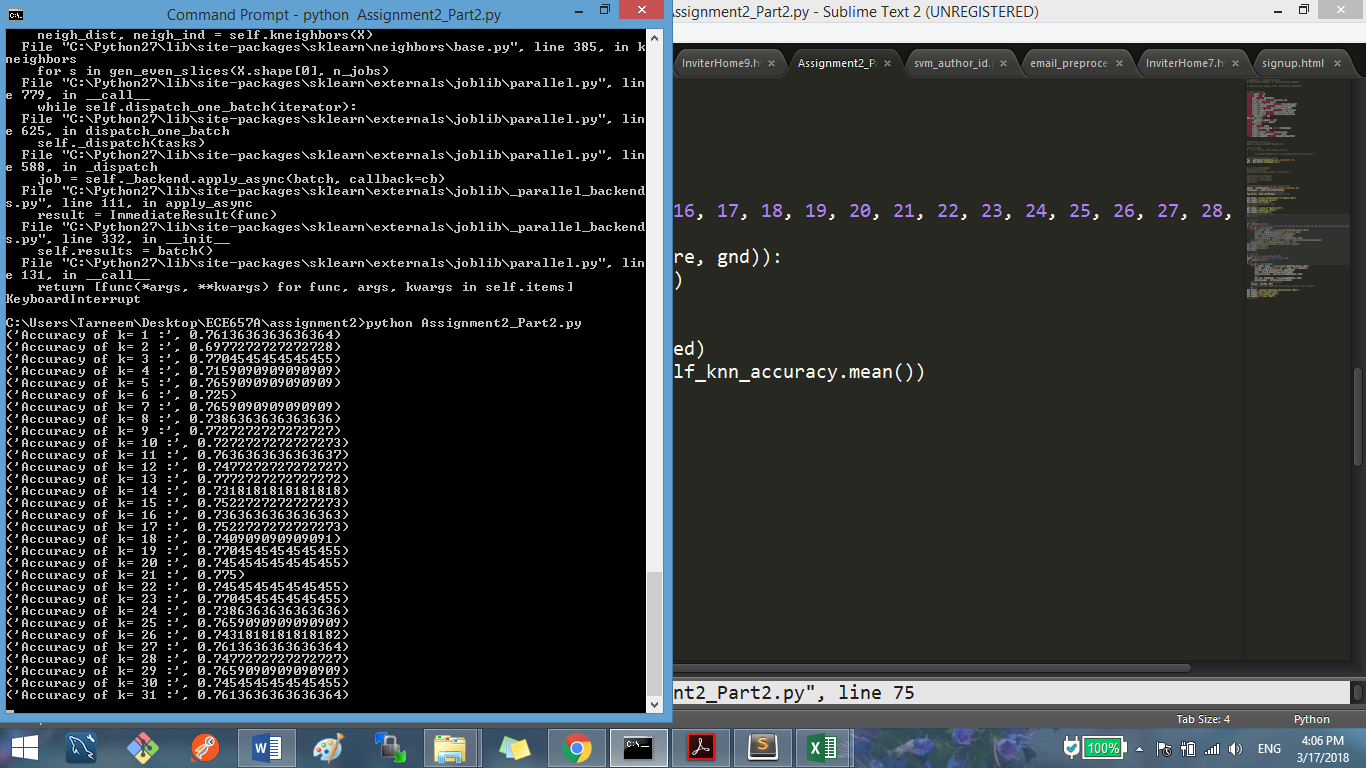


Figure 5. Accuracy of knn with k range from 1 to 31.

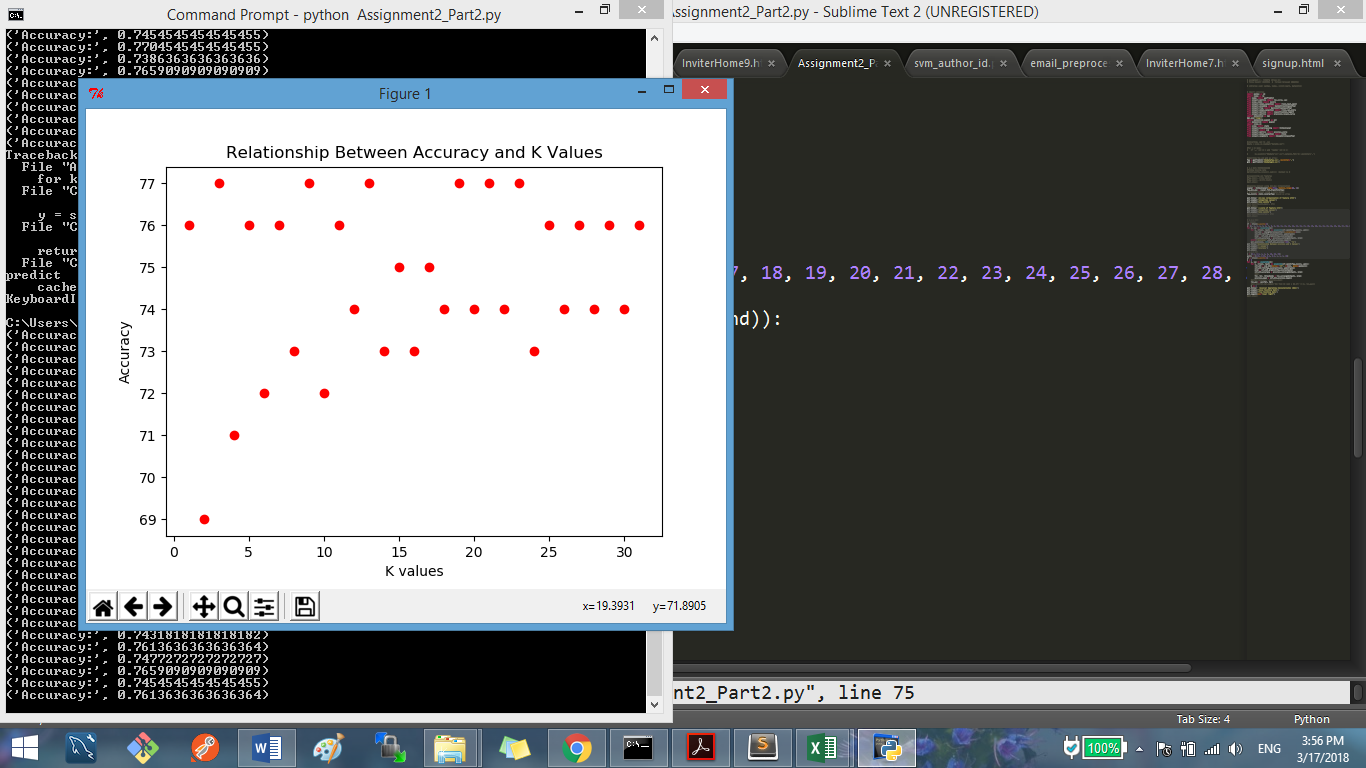
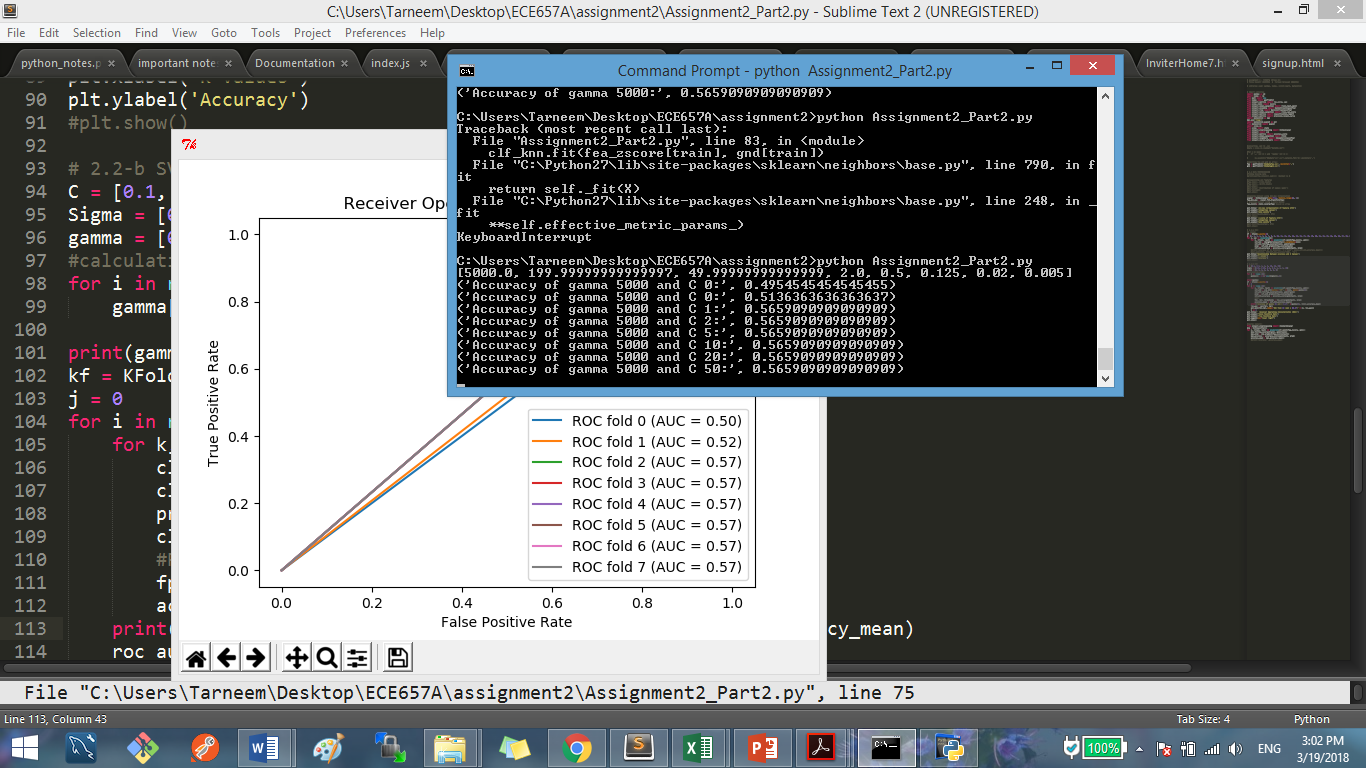


Figure 6. k values and accuracy in knn

b) code lines (93-121) represents gamma calculation, SVM setting and ROC curves. As obtained from code, gamma = [5000.0, 199.99, 49.99, 2.0, 0.5, 0.125, 0.02, 0.005]. Also, the code basically uses k\_fold = 5 and its iterates 8 times for C per each gamma value. Figures 7 – 14 shows accuracy and ROC curvesobtained per each gamma value.



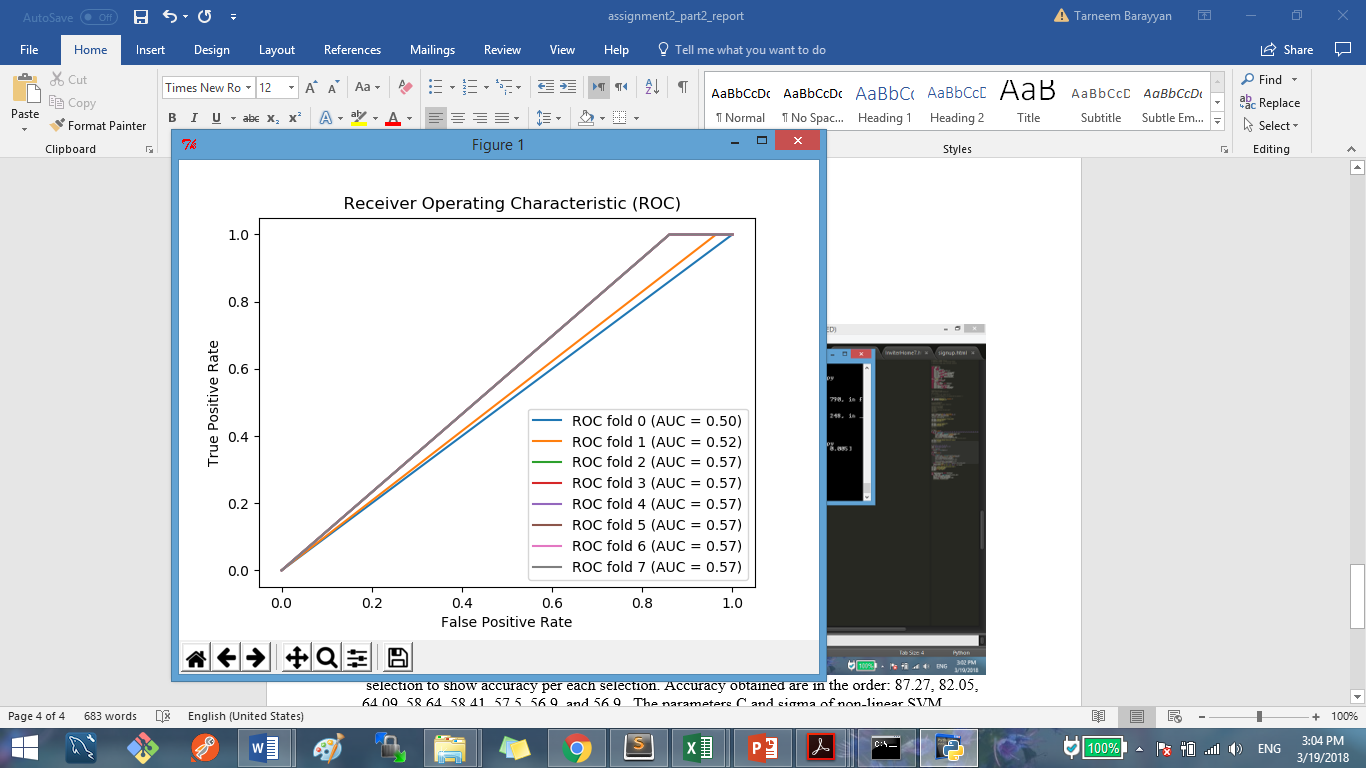
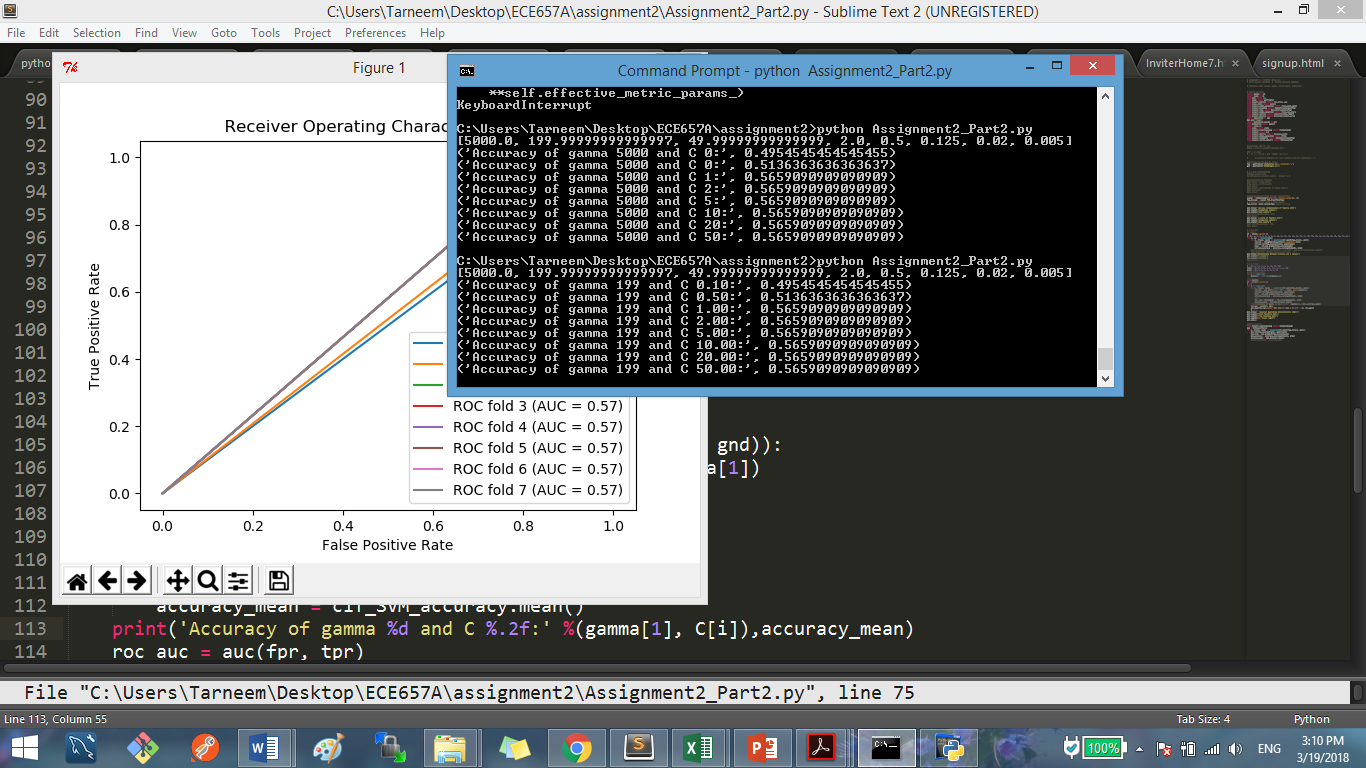


Figure 7. Accuracy and ROC curve of Sigma: 0.01, gamma:5000 and C:0.1-50.



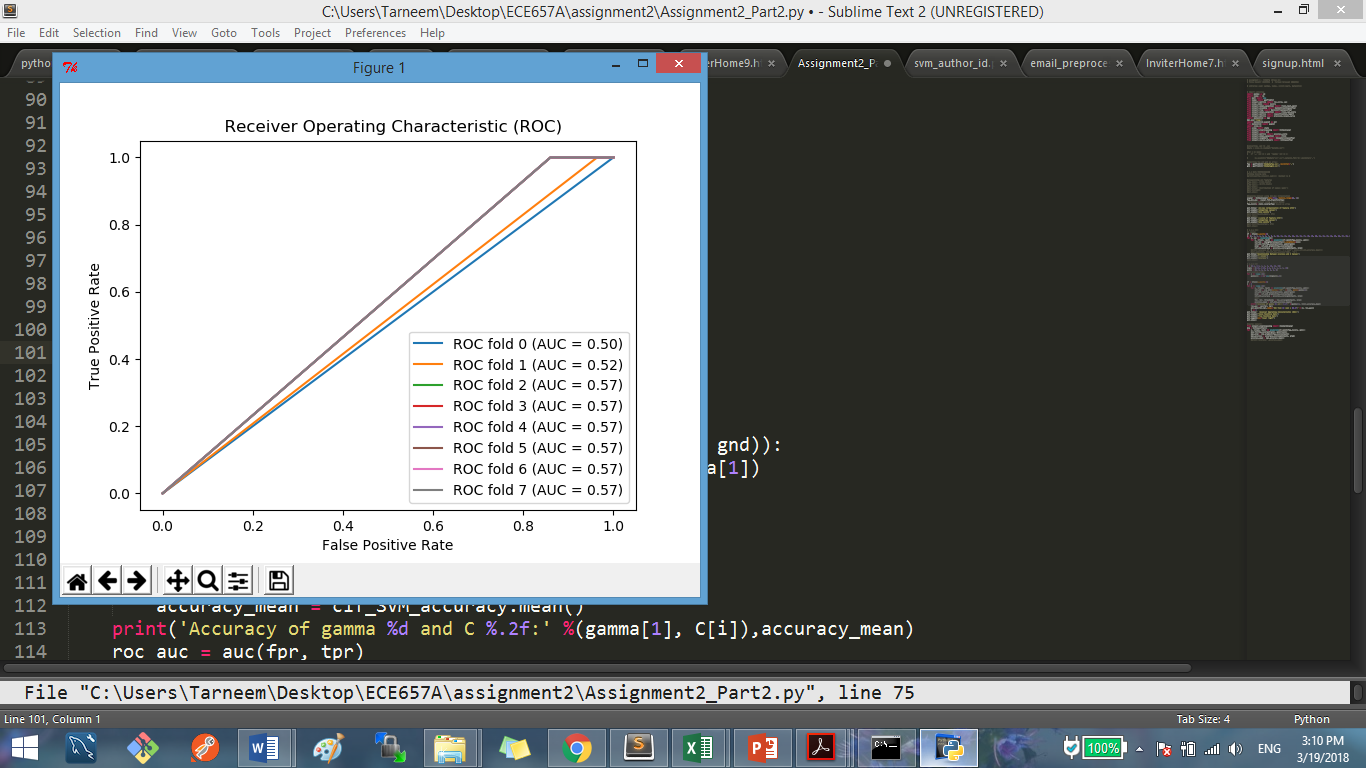
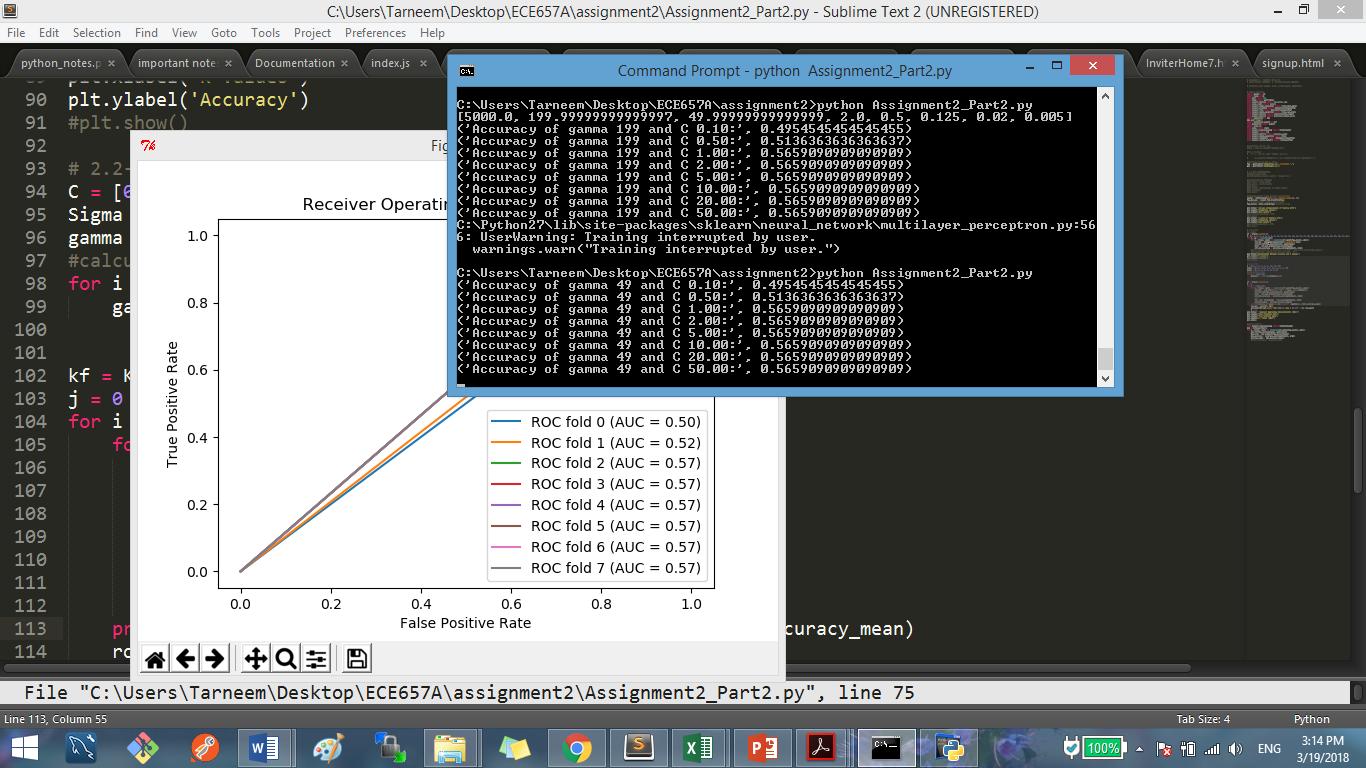


Figure 8. Accuracy and ROC curve of Sigma:0.05, gamma:199.99 and C:0.1-50.



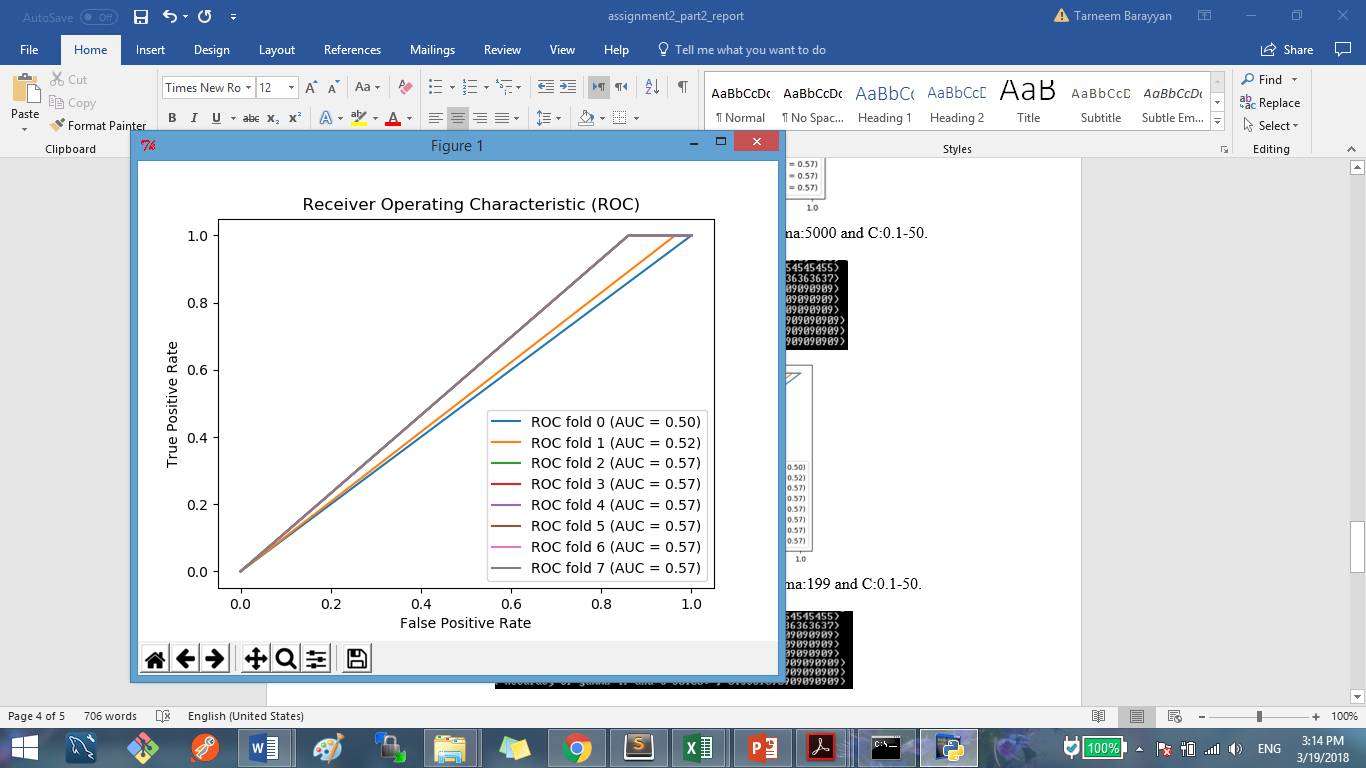
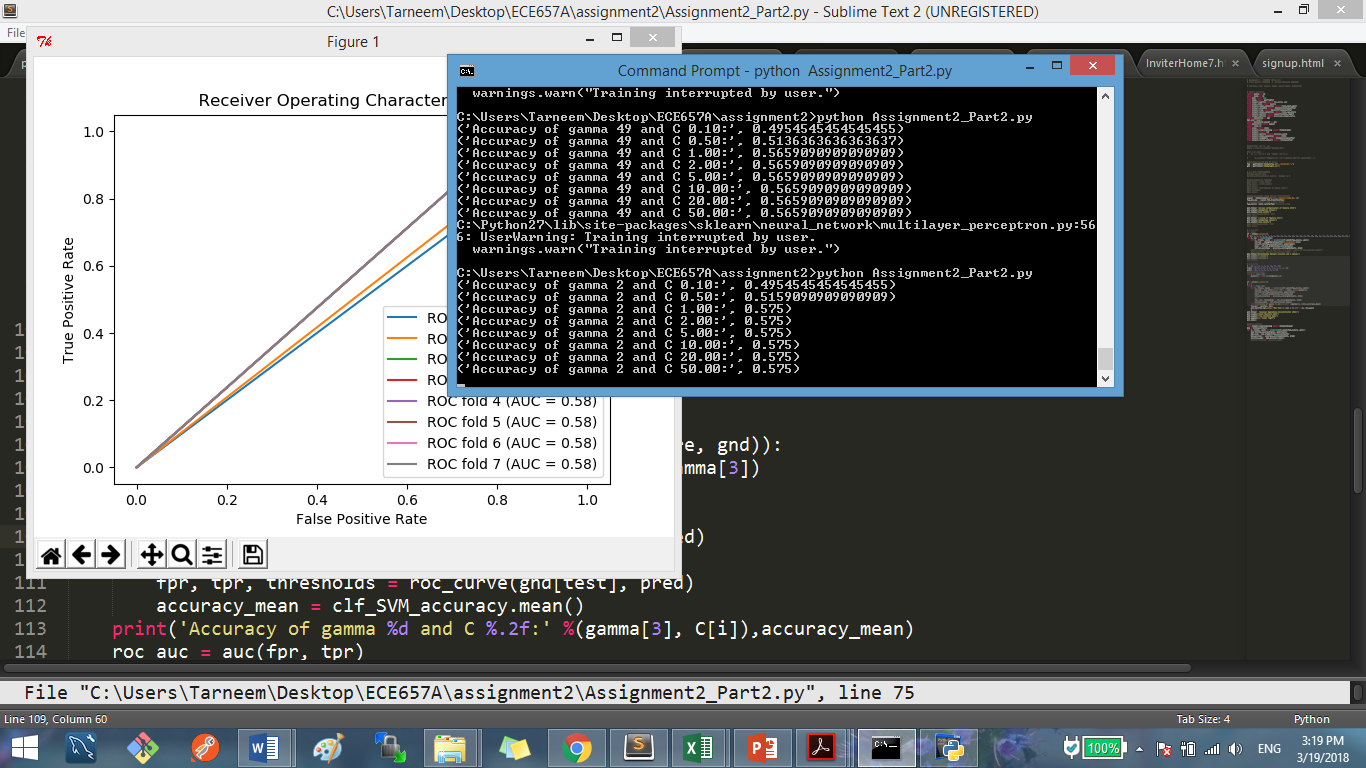


Figure 9. Accuracy and ROC curve of Sigma:0.1, gamma:49.99 and C:0.1-50.



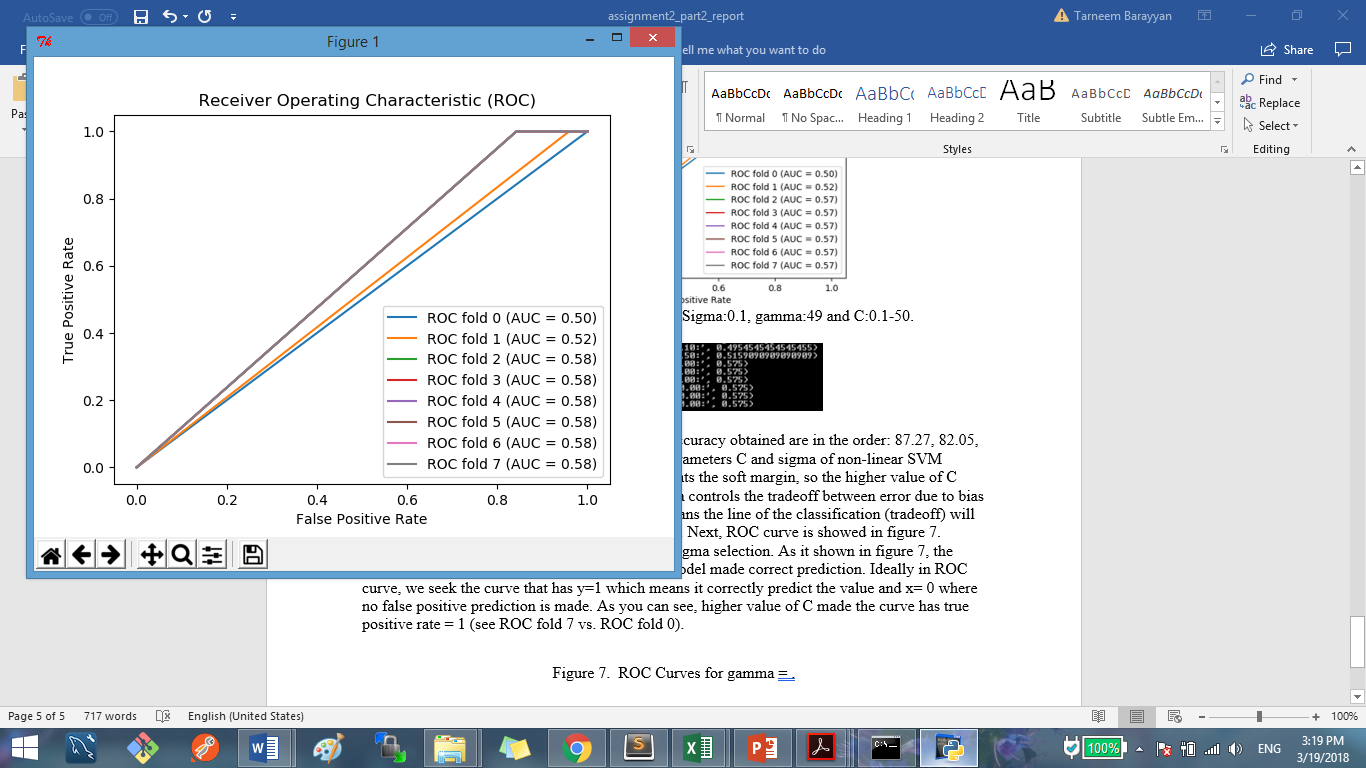


Figure 10. Accuracy and ROC curves of Sigma:0.5, gamma:2 and C:0.1-50.

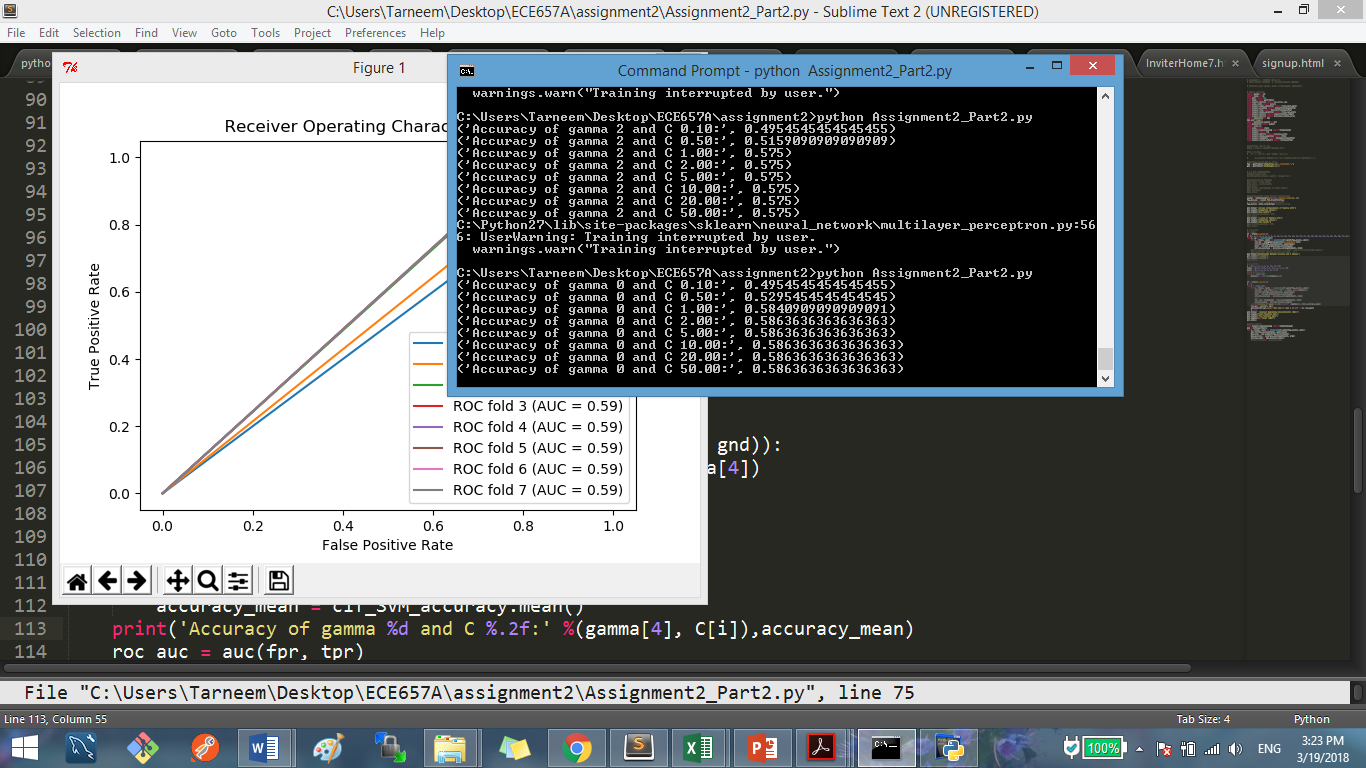
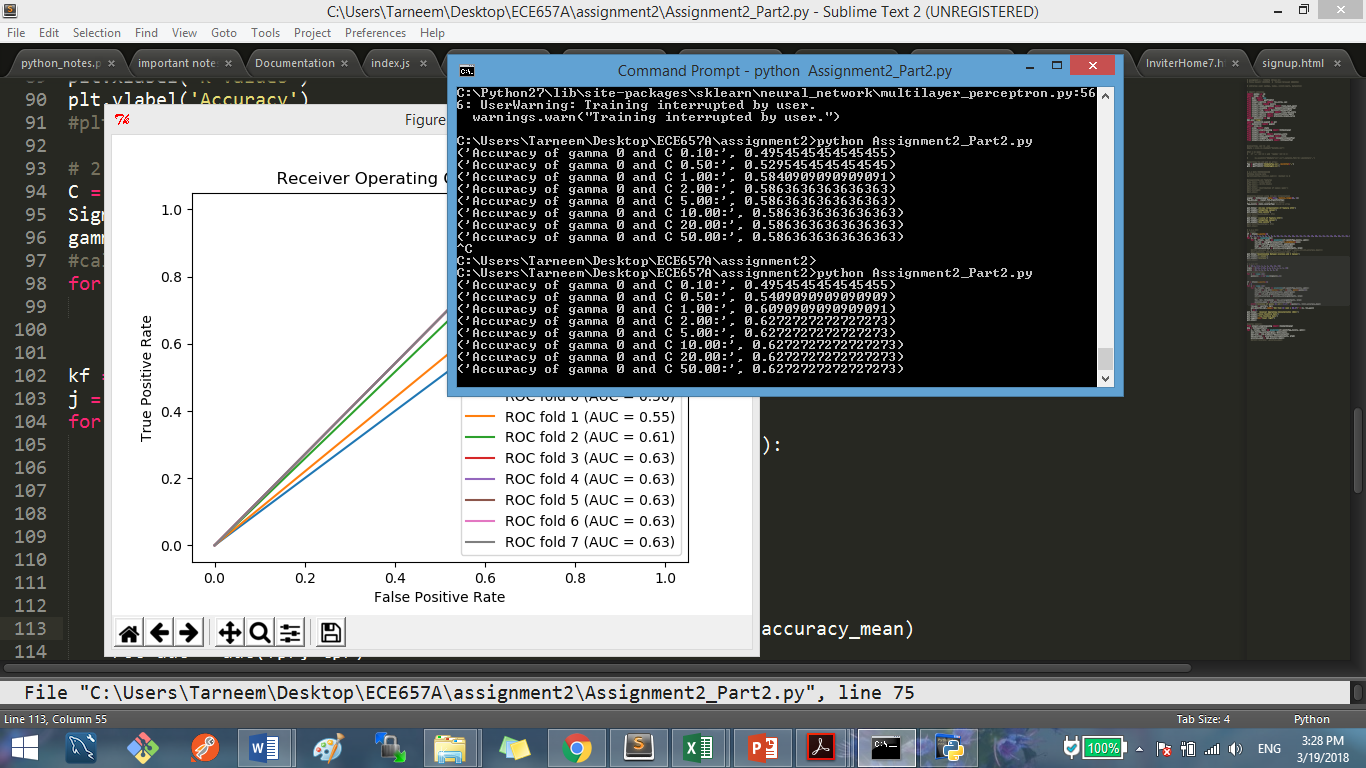




Figure 11. Accuracy and ROC curves of Sigma:1, gamma:0.5 and C:0.1-50.



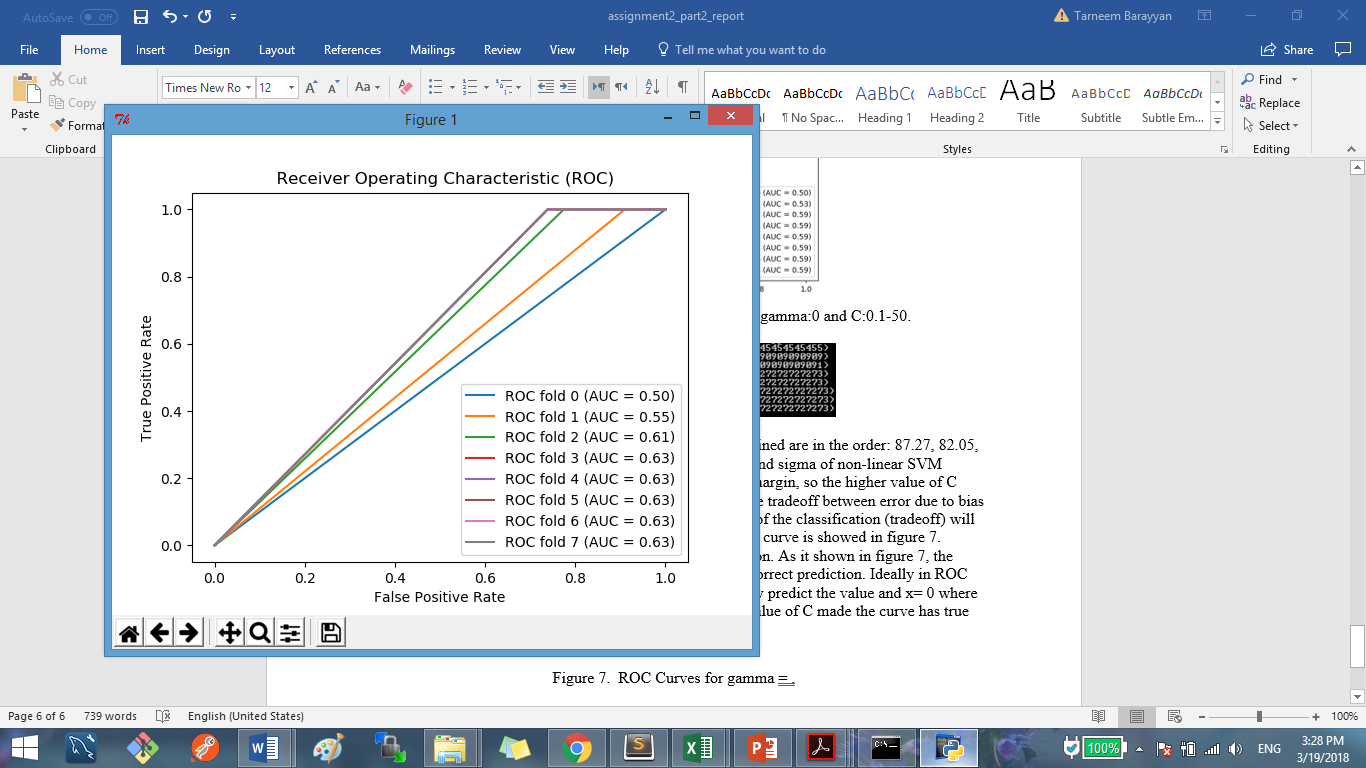
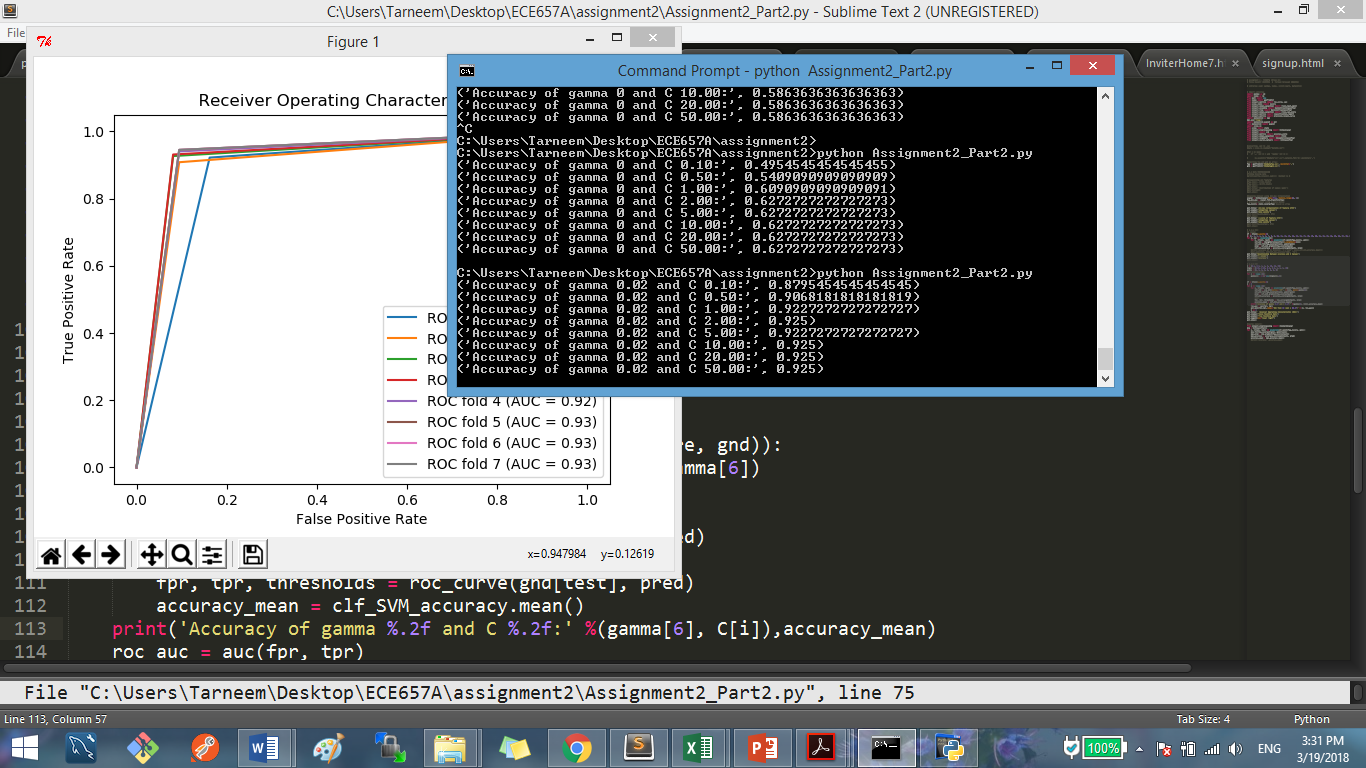


Figure 12. Accuracy and ROC curves of Sigma:2, gamma:0.125 and C:0.1-50.



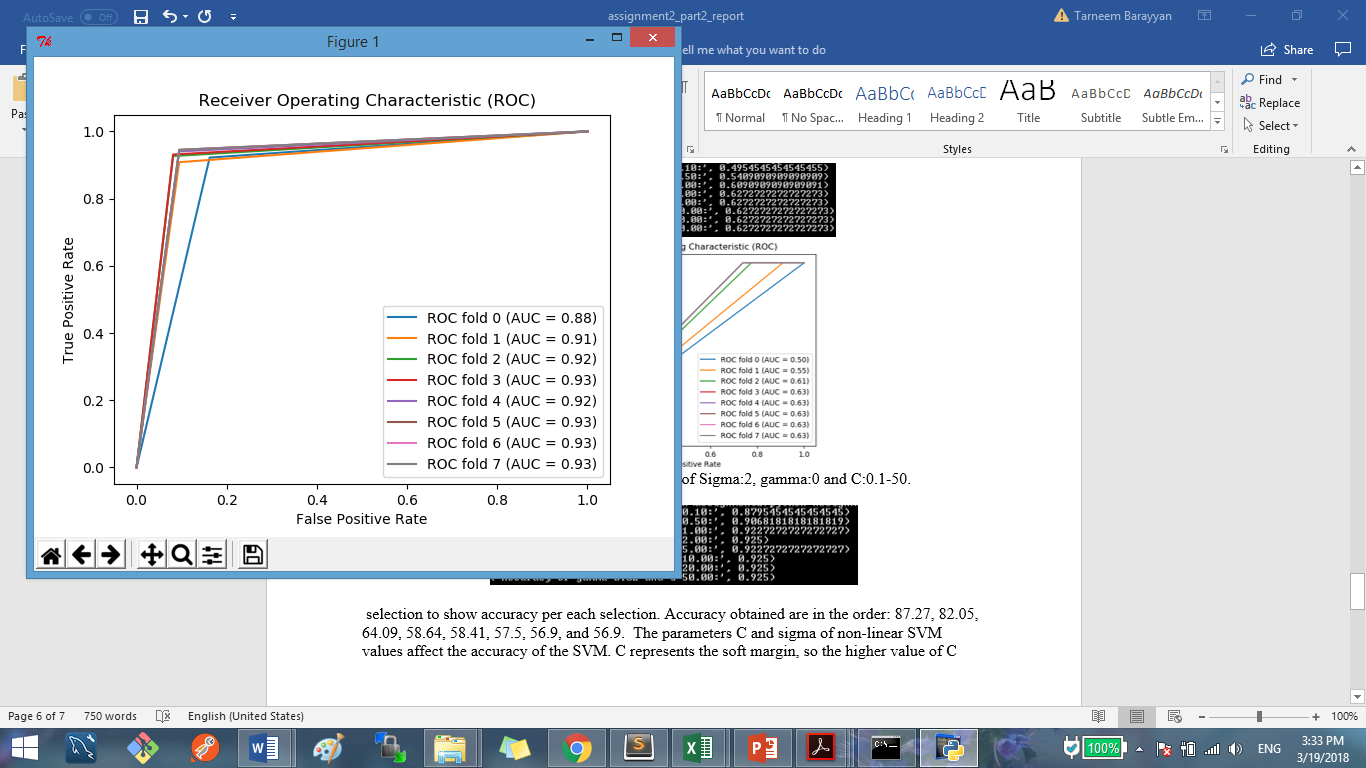
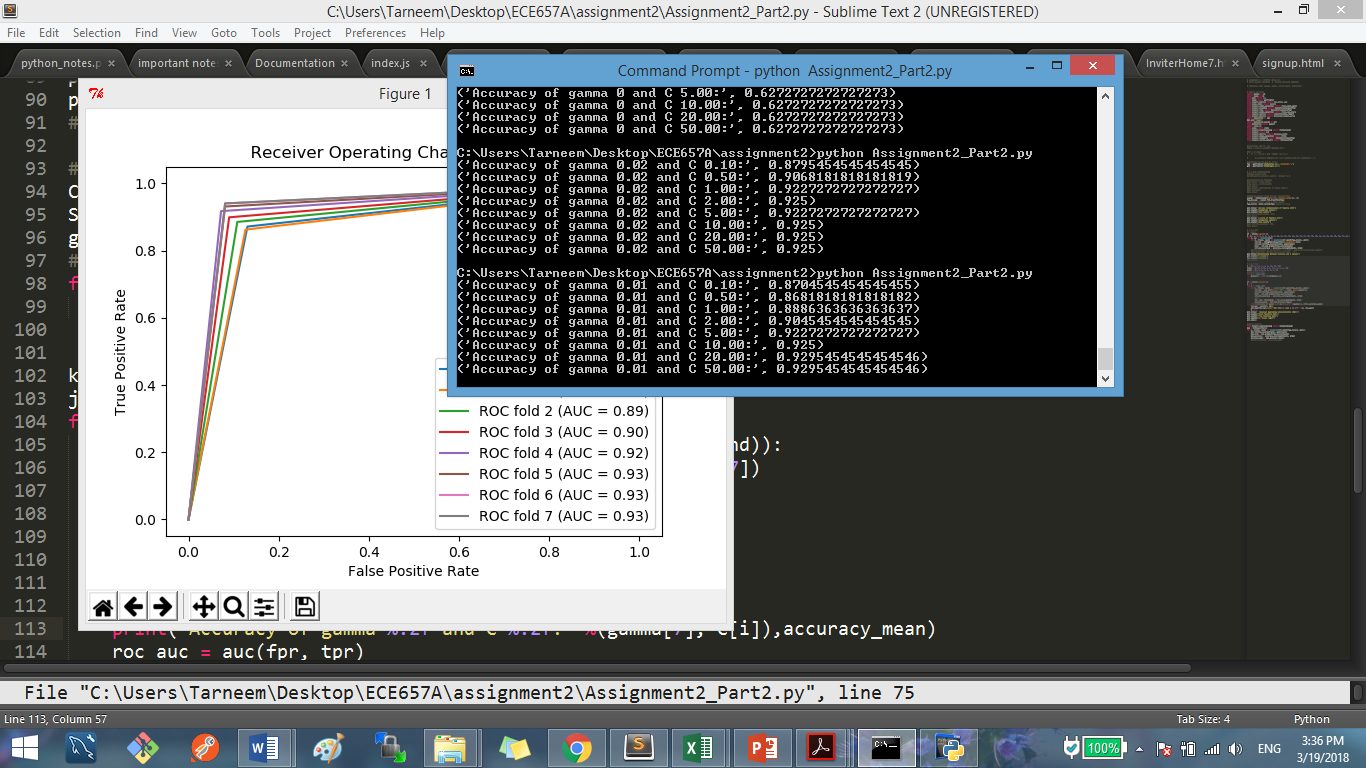


Figure 13. Accuracy and ROC curves of Sigma:5, gamma:0.02 and C:0.1-50.



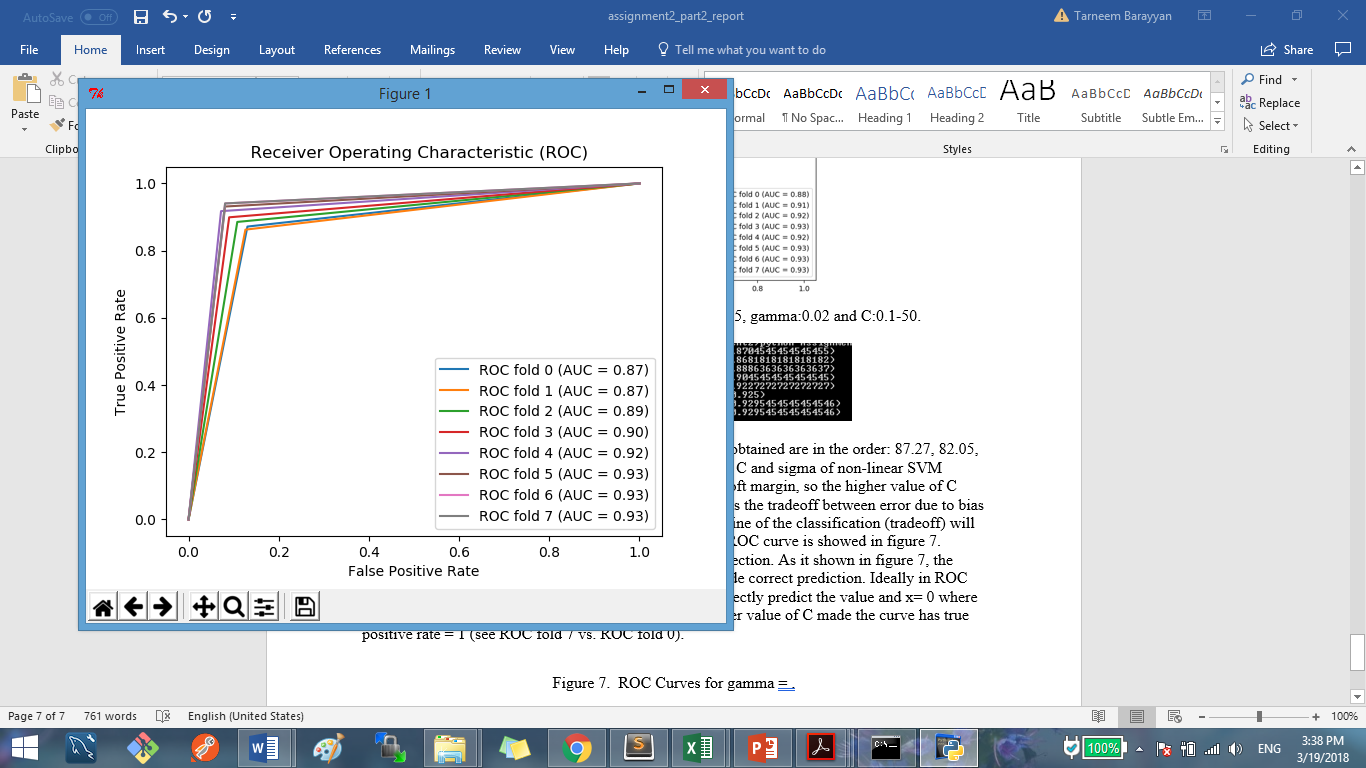


Figure 14. Accuracy and ROC curves of Sigma:10, gamma:0.01 and C:0.1-50.

As seen from previous figures, the parameters C and sigma of non-linear SVM values affect the accuracy of the SVM. C represents the soft margin, so the higher value of C means more accurate of training data point. Sigma controls the tradeoff between error due to bias and variance in the model. Thus, higher value means the line of the classification (tradeoff) will be too narrowed to the data points, and vice versa. For ROC curves, there are 8 curves per each figure (from 0 to 7) for C and Sigma (or gamma) has fixed value. The ideal ROC curve climes quickly toward top left, meaning the model made correct prediction. Thus, in ROC curve, we seek the curve that has y=1 which means it correctly predict the value and x= 0 where no false positive prediction is made. We concluded that the sigma values: 0.01, 0.05, 0.1, 0.5, and 1 has poor accuracy, and C does not scientifically improve it. When sigma has values 5 and 10, accuracy got much better and it seems that C of value 1 to 50 has similar effect (same accuracy). The difference between Sigma value of 5 and 10 are of 0.925 and 0.929. Thus, we will be selecting the following value as best value for sigma and C. Sigma:10 (gamma:0.01), and C:20.

Question 2 conclusion: it seems SVM is doing better than Knn in classifying the dataset. Overall accuracy obtained is higher in SVM (92.29% vs. 77.7 in knn).

3)

a) In this part, we are using test set that is part from the training set we did splitting from question 2. We used best parameters found for knn and SVM, and the default parameters form MPL and RT. The accuracy obtained are:

knn Accuracy: 64.5%

SVM Accuracy: 83.6%

MPL Accuracy: 87.1%

RF Accuracy: 87.1%

b) code lines (175-185) For Random Forest Classifier, when trying different values for parameters, it seems the following value are the best to use (in term of accuracy as well as classification report, where precision changed for -1 from 0.45 to 56, and for +1 from 0.60 to 68): n\_estimators=10, max\_depth=3, max\_features='sqrt', random\_state=42

for MPL, it improved in term of accuracy, and precision as well. The following parameters where modified:

hidden\_layer\_sizes= (16, 8), solver='lbfgs', learning\_rate\_init=0.01

c) Now, in this part, we are going to evaluate all classifier using cross\_validation where k\_fold = 20. The following are output figures of each classifier and evaluation (accuracy, classification report and time):

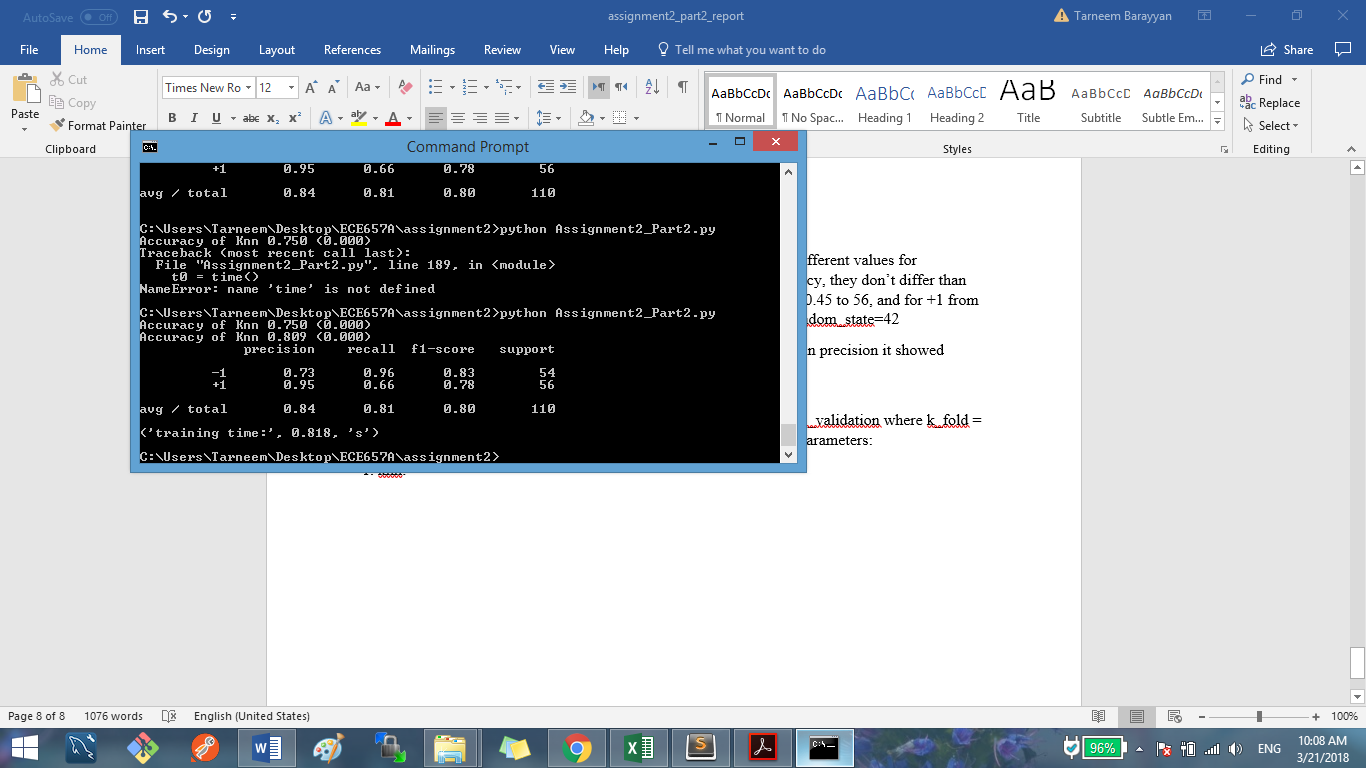


Figure 15. knn (k=13) Classifier Results (code lines 185-197).

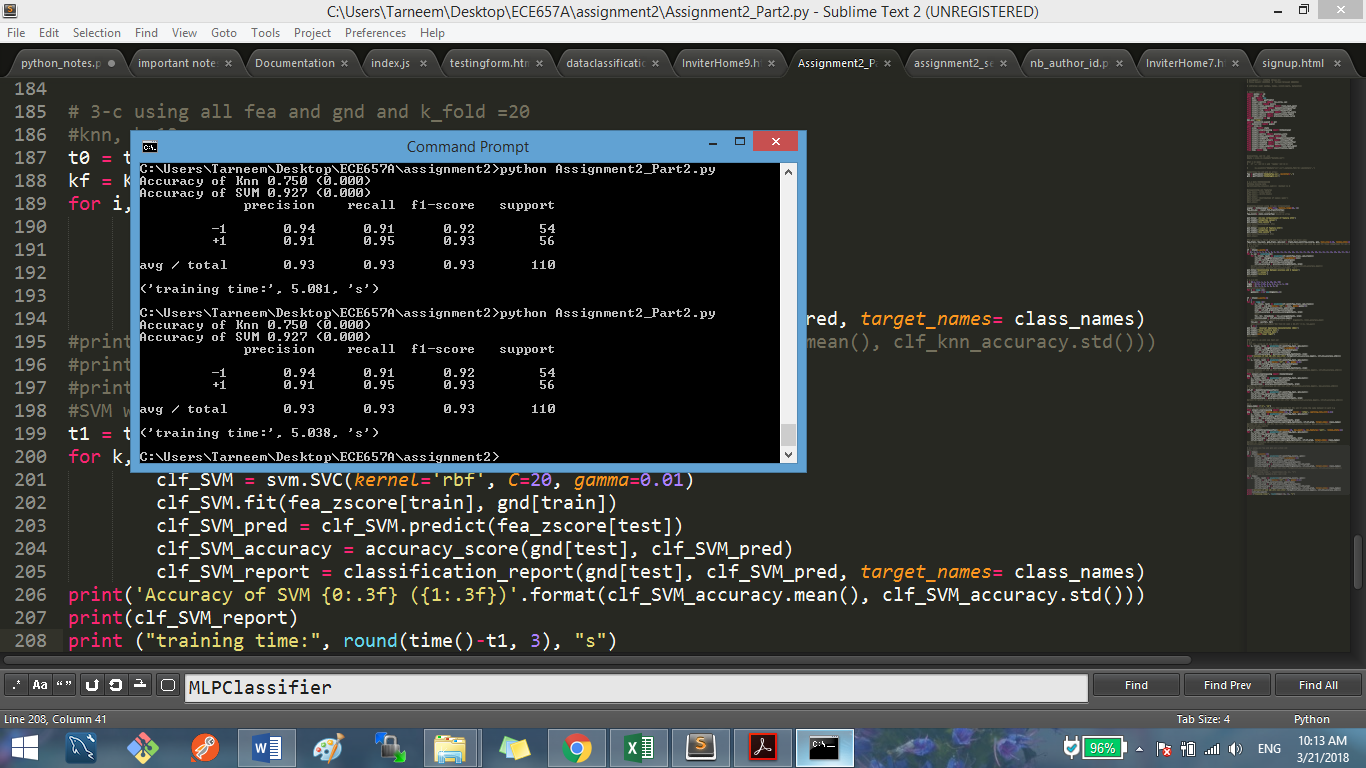


Figure 16. SVM Classifier (Sigma:10 (gamma:0.01), and C:20) Results (code lines 199-208).

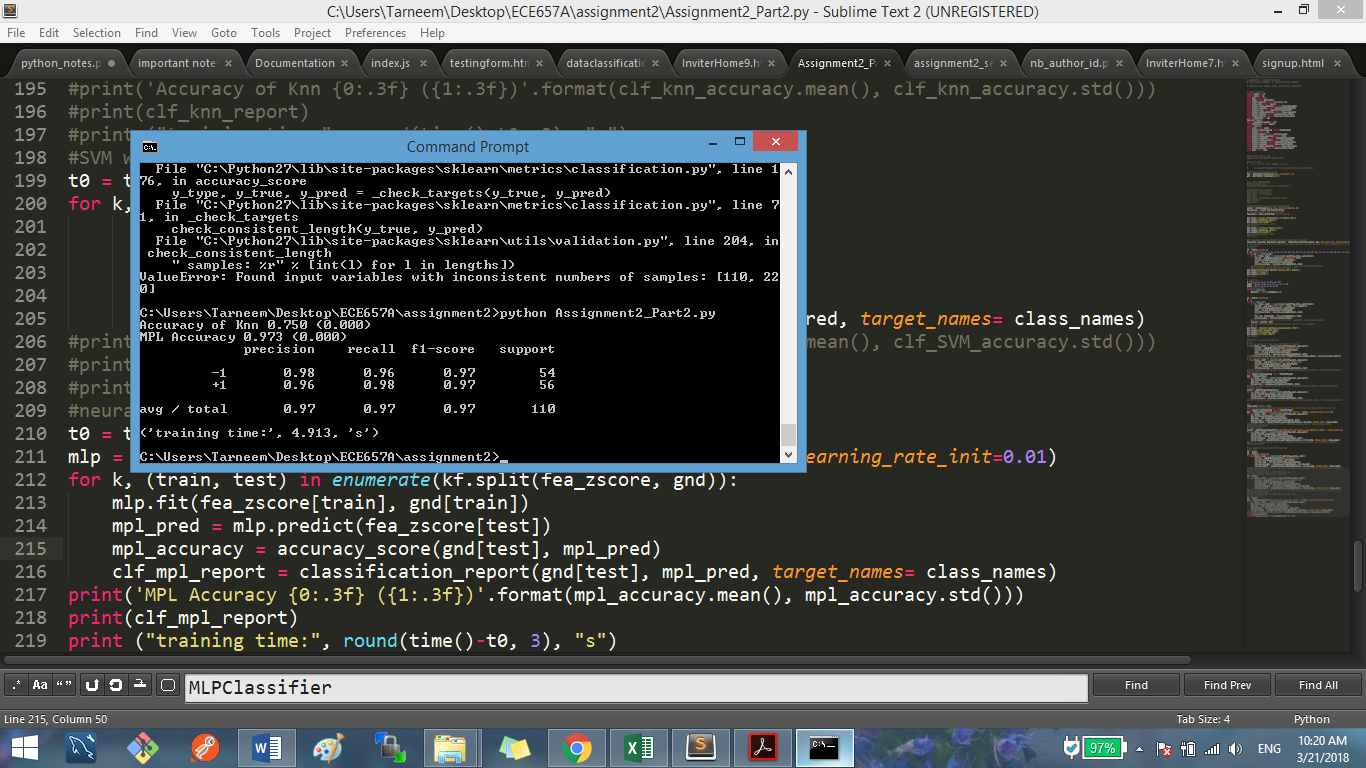


Figure 17. MPL Classifier (hidden\_layer\_sizes=(16, 8), solver = 'lbfgs', learning\_rate\_init=0.01) Results (code lines 210-219).

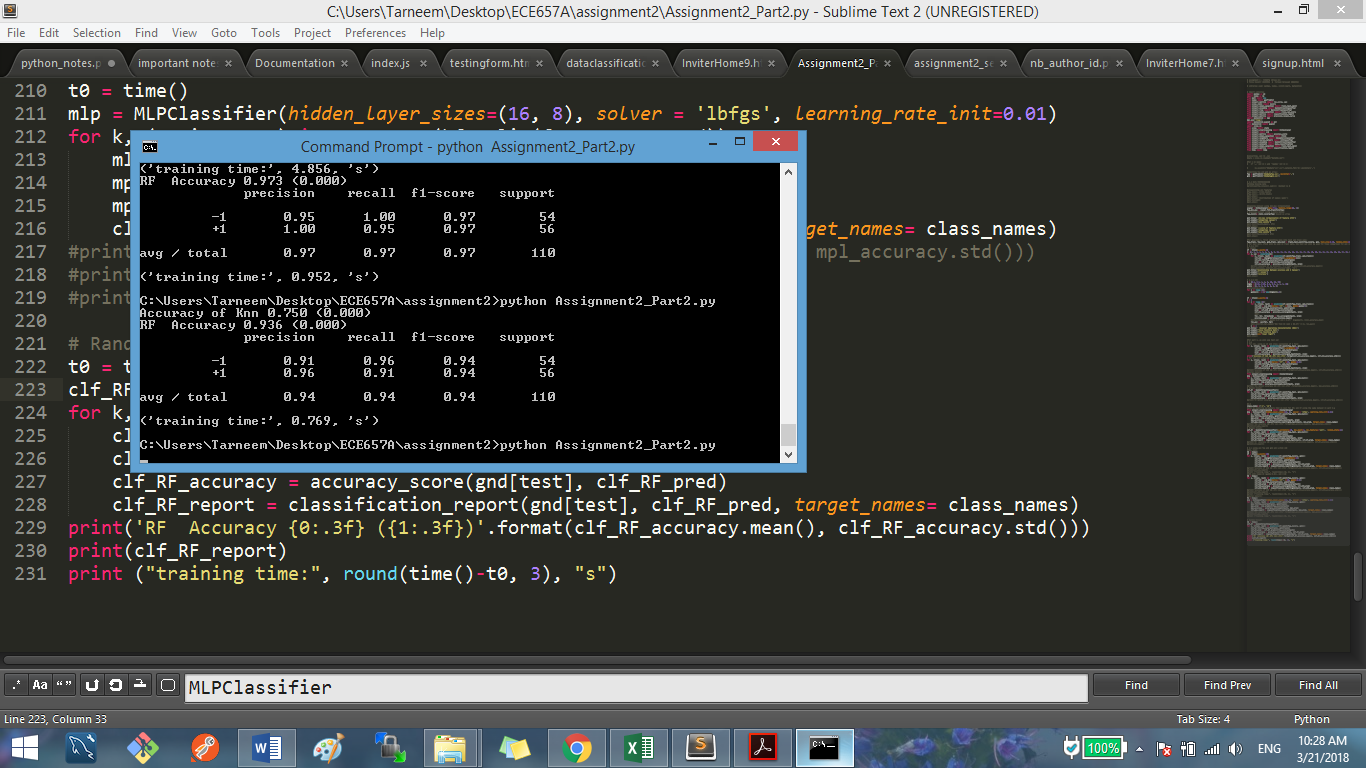


Figure 18. RF Classifier (n\_estimators=10, max\_depth=3, max\_features='sqrt', random\_state=42) Results (code lines 222- 231).

The classification was repeated 20 times due to choosing k\_fold= 20 as we are dividing test and train into 20 sets and randomly selecting them by iterating 20 times.

4) Overall, it seems MPL Classifier is doing the best in term of accuracy and classification report. However, considering the runtime we believe that Random forest classifier is doing well as its report results and accuracy is closer to MPL classifier.

Knn classifier is effective in term of large dataset, but the determination of k value is frustrating, and as it uses distance to do the classification, it is not clear which type of distance and attribute to use to obtain best results.

For SVM classifier, the flexibility to tune its parameters is helpful to avoid overfitting and increase its performance as well as the choice of kernel. However, SVM takes long time in case of having large dataset and the choice of kernel might be tricky.

For Random Forest classifier, as we saw, it has high accuracy and requires less time to run. It also can handle vast number of variables as well as its ability to produce internal unbiased estimate of the generalization error as the forest building progresses. The disadvantage is it might overfit data with noisy tasks.

MPL classifier is good in term of ability to learn non-linear and complex classification problem. It has the ability to generalize the relationship between features and labels and infer unseen or hidden relationship as well, so it will be able to predict it. On the other hand, the disadvantage of MPL classifier is that it requires tuning of hyperparameters, as well as its sensitivity to feature scaling (that’s why we need to do normalization, otherwise it would not function well).

5)