Francisco Hu
CSE 380
Assignment 3

UB_BB.py Output

NB,UB,0.9212370653692343,0.896079894421603,0.9031376237260946
NB,BB,0.8918244506172566,0.876764428201614,0.8811474759175979
LR,UB,0.9040365729133806,0.8973843459722857,0.9000113480911696
LR,BB,0.8693641875625819,0.8454452058271155,0.8519678193233128
SVM,UB,0.877458495043551 8,0.862375666209837,0.8663072067757873
SVM,BB,0.868746933580165,0.8611511090807573,0.8636227346111975
RF,UB,0.8988977732666389,0.8435609867519416,0.848525072259511
RF,BB,0.8545435008394634,0.8021952692756713,0.8086294450832845

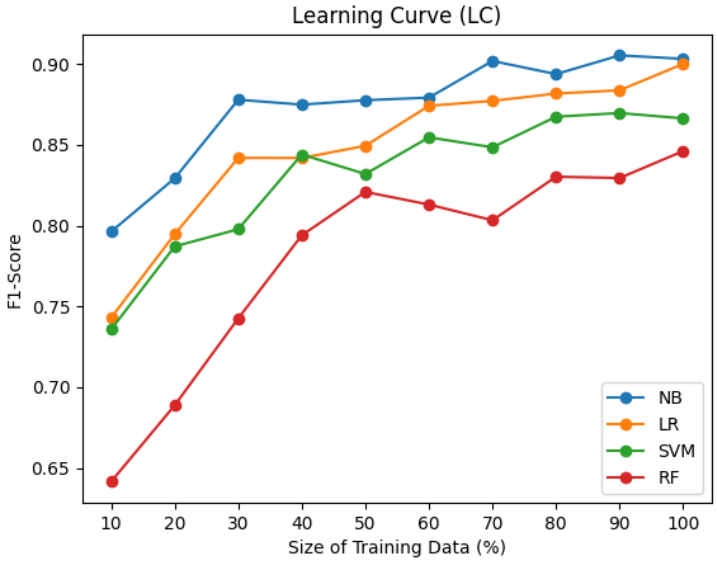
MBC_exploration.py Output

NB,UBBB,0.9230284329668158,0.9127289477691487,0.9165939673879522
LR,UB,0.9214221973253098,0.9149410689812699,0.9174733422581456
SVM,UBBB,0.9538514602208329,0.9465797675244911,0.9496500922331201
RF,UB,0.938145710740995,0.921323486117456,0.9272578600182354

MBC_final.py Output

SVM,UBBB,0.9538514602208329,0.9465797675244911,0.9496500922331201

UB_BB.py Part C



In the figure above, there is a clear difference between each algorithm's performances across all sets of sizes of training data. The performances are ranked in the order as follows:

Naive Bayes, Logistic Regression, SVM, and RandomForest. There is also a clear upward trend for all of these classification methods which makes sense because the more data that an algorithm trains with, the higher their performance.

I think the reason why Naive Bayes performed so well despite it being less complex compared to the other algorithms is because data sets used are relatively small in size. Although Naive Bayes makes independence assumptions even if there are dependencies between data, it does not need to learn all of the possible feature dimensions and instead uses just one dimension to estimate the likelihoods. Considering too many dimensions makes it difficult for the algorithm to form meaningful connections between each data point. Naive Bayes avoids this issue when presented with small data sizes.

Logistic Regression follows a close second with it reaching to nearly the same F1-score as that of Naive Bayes when using 100% of the training data. On lower percentages of the training data however, it performs worse than Naive Bayes. I believe that the reason why we see this happening is because Logistic Regression has a lower asymptotic error than that of Naive Bayes so Naive Bayes tends to do better in smaller samples. However, if there was more data to train from, Logistic Regression will most likely perform better in the long run.

SVM trailed behind both Logistic Regression and Naive Bayes across each sample size. I think the reason why SVM was outperformed was because of the fact that I did not tune the model when it came to regularization. This seems to be especially true in the cases where the number of features were much greater than the number of samples.

RandomForest performed the worst out of all the algorithms tested, especially in the cases with low training data sizes. This might be due to the fact that the sample sizes are too low and/or because I did not manipulate the parameters and left them on default.

MBC Part C

The SVM classification method yielded the best results for me with an F1-score of 0.9494965. For SVM, I used word stemming, TFIDFVectorizer, ngrams of 1 and 2, SelectPercentile for feature selection, and hyperparameter tuning in order to optimize my results. Word stemming played a large role in making SVM a better training model since it was able to reduce the number of total features. The choice of using TFIDVectorizer over CountVectorizer also improved the performance of the model since it assigned weights to each feature so that SVM can use the most relevant terms in its decision making process.

Ngrams of both 1 and 2 made the model better since oftentimes in English, analyzing the 2 words linked together will give better meaning than if you were just to look at 1 word at a time. My implementation of feature selection was using SelectPercentile and I think this helped because it was able to pick out the most important features of the data. This reduced the need for SVM to use unimportant data in its analysis which would have lowered the accuracy. Finally, tuning the parameters on my SVM implementation significantly improved the accuracy of the model since it minimized the loss function and allowed the model to have better convergences as well as helping the model to optimize its learning.

References

https://scikit-learn.org/stable/modules/naive_bayes.html

https://sebastianraschka.com/Articles/2014_naive_bayes_1.html

https://stats.stackexchange.com/questions/379383/why-does-naive-bayes-work-better-when-the-number-of-features-sample-size-comp#:~:text=Because%20of%20the%20class%20independence,such%20as%20images%20or%20texts.

https://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn06-notes-nup.pdf

https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e

http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

https://stats.stackexchange.com/questions/73139/why-does-naive-bayes-outperform-support-vector-machines

https://scikit-learn.org/stable/modules/svm.html

https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

https://medium.com/turo-engineering/how-not-to-use-random-forest-265a19a68576

https://towardsdatascience.com/why-you-should-do-feature-engineering-first-hyperparameter-tuning-second-as-a-data-scientist-334be5eb276c#:~:text=What%20is%20the%20importance%20of%20hyperparameter%20tuning%3F,function%20to%20give%20better%20results.

Friend used: Eddie Xu