CS57300: Assignment 3

Name: Tunazzina Islam, PUID: 0031294421

1. Preprocessing:

   python preprocess-assg3.py dating-full.csv trainingSet.csv testSet.csv
   Outputs of preprocess-assg3.py
   Mapped vector for female in column gender: [1]
   Mapped vector for Black/African American in column race: [0, 1, 0, 0]
   Mapped vector for Other in column race_o: [0, 0, 0, 0]
   Mapped vector for economics in column field: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0]

2. Implement Logistic Regression and Linear SVM:

(i) python lr_svm.py trainingSet.csv testSet.csv 1
    Output:
    Training Accuracy LR: 0.66
    Testing Accuracy LR: 0.66

ii) python lr_svm.py trainingSet.csv testSet.csv 2
    Output:
    Training Accuracy for svm: 0.74
    Testing Accuracy for svm: 0.75

3. Learning Curves and Performance Comparison:

   python cv.py trainingSet.csv
   Output:
   avg_accuracy_test_nbc (validation accuracy) [0.5640000000000001, 0.562, 0.569, 0.569, 0.5750000000000001, 0.571]
   std_accuracy_test_nbc [0.020591260281973976, 0.023579652245103173, 0.018138357147217034, 0.019723082923315996, 0.020124611797498082, 0.015132745950421531]
   sterr_test_nbc [0.0065115282384398745, 0.0074565407529228935, 0.005735852159879988, 0.006236986451805062, 0.006363961030678919, 0.004785394445602152]
   avg_accuracy_test_LR (validation accuracy) [0.6619999999999999, 0.652, 0.6719999999999999, 0.673, 0.6610000000000001, 0.665]
   std_accuracy_test_LR [0.06368673331236262, 0.028565713714171402, 0.03280243893371343, 0.027946377224964212, 0.026248809496813366, 0.022022715545545222]

sterr_test_LR [0.02013951340027856, 0.009033271832508972, 0.010373041983911944, 0.008837420438114271, 0.00830060238777885, 0.006964194138592054]
avg_accuracy_test_SVM (validation accuracy) [0.6589999999999999, 0.6500000000000001, 0.644, 0.593, 0.584, 0.595]
std_accuracy_test_SVM [0.04988987873306569, 0.048989794855663564, 0.042237424163885755, 0.04583666654546335, 0.016248076809271934, 0.020124611797498124]
sterr_test_SVM [0.015776564898608316, 0.015491933384829668, 0.013356646285612, 0.014494826663330608, 0.005138093031466055, 0.006363961030678933]

iii) Plot the learning curve with error bars:

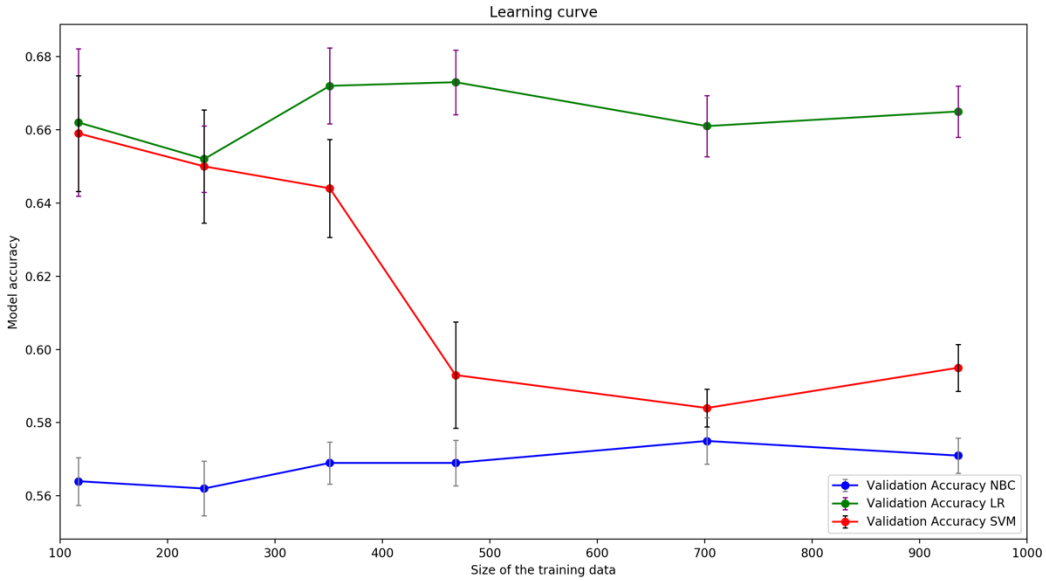

Figure 1: Learning curves for 3 models (NBC, LR, Linear SVM) training data size vs. accuracy including error bars.

iv)

In the plot, the green line represents 10-fold cross validation accuracy for logistic regression (LR) and the red line represents 10-fold cross validation accuracy for Linear SVM (Fig. 1).

When the training data size is small (t_frac = 0.025, $|S_t|$ = 4680 so size of training dataset = t_frac * $|S_t|$ = 117) and size of validation set = 520 (same size with different values of attribute), validation accuracy for LR is 66% with higher standard error approximately 0.02. The purple error bar on learning curve of LR (green line) represents standard error. Size of training data has been increased by keeping the same size of validation set with different attribute values. This

means K-fold cross validation requires to use each fold as the testing fold once. The validation accuracy dropped at 65% with less standard error (0.009) for t_frac = 0.05, then accuracy increased to 67% with the increment of training data size. For size of training set= 936, accuracy was 67% with standard error 0.007. We can say from the plot that, for LR accuracy has been increased with the increasing size of training data set and standard error has been decreased with the increasing size of training data set.

For linear SVM, when the training data size is small (117) and size of validation set = 520 (same size with different values of attribute), validation accuracy for is approximately 66% with higher standard error approximately 0.015. The black error bar on learning curve of SVM (red line) represents standard error. Accuracy has been decreased with the increasing of training size. There is a sharp drop of accuracy from 65% to 59% with higher standard error approximately 0.01 during training size from 351 to 468. We can notice from the error bars that the standard error is higher (around 0.02 to 0.01) between size of training set 117 and 468. It has been decreased to 0.005 and 0.007 size of training set 702 and 936 correspondingly and accuracy ended up at 59% for linear SVM. We can say from the plot that, for linear SVM accuracy has been decreased with the increasing size of training data set and standard error was higher at the beginning and kept stable until a certain size of training data set and then started decreasing with the increasing size of training data set.

v)

For LR, from the output of code lr_svm.py (single train-test set split with high variance) we can see that the testing accuracy for LR was 66% but the 10-fold cross-validation (less variance than a single train-test set split) provides 67% accuracy. This did not change that much.

For linear SVM, from the output of code lr_svm.py (single train-test set split with high variance) we can see that the testing accuracy for SVM was 75% but the 10-fold cross-validation (less variance than a single train-test set split) provides a lower accuracy (59%) than the raw number. So, previously we had overfitting problem. When we use cross-validation, we expect to have less overfitting. We can get better accuracy by getting more data/better data, trying different combinations of features which seems to work best, generating better features, fine tuning hyperparameters.

10-fold cross validation is a more reliable estimate of the performance of the algorithm on new data given to the test data. It is more accurate because the algorithm is trained and evaluated multiple times on different data.

…… Bonus Points……..

Implement Logistic Regression and Linear SVM:

i) For LR, I changed the regularization parameter _lamda = 0.1, step size (learning rate) = 0.05 and number of iterations = 5000
Run the code:
python lr_svm_bonus.py trainingSet.csv testSet.csv 1
Output:
Training Accuracy LR: 0.78
Testing Accuracy LR: 0.78

ii) For linear SVM, I changed the regularization parameter _lamda = 0.07 and step size (learning rate) = 0.7
Run the code:
python lr_svm_bonus.py trainingSet.csv testSet.csv 2
Output:
Training Accuracy for svm: 0.75
Testing Accuracy for svm: 0.77

Learning Curves and Performance Comparison:

10-fold cross-validation:
Run the code:

python cv_bouns.py trainingSet.csv

Plot the learning curve with error bars:

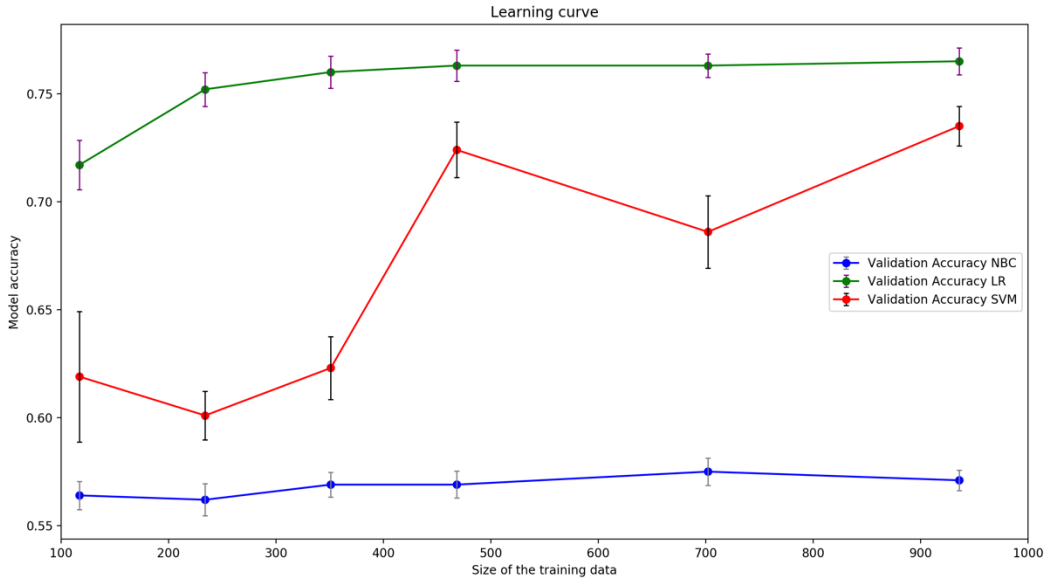

Figure 2: Learning curves for 3 models (NBC, LR, Linear SVM) training data size vs. accuracy including error bars. (bonus points)

At beginning when size of training data set is small (117), validation accuracy for LR is 72% with standard error 0.01 and SVM is 62% with standard error 0.03.
In case of size of training set= 936, validation accuracy of LR is 77% with standard error 0.006. For size of training set= 936, validation accuracy of linear SVM is 74% with standard error 0.009. Accuracy has been increased for both cases with the increasing of the size of training data set.