AML PA1

ajchavan

April 2017

Run program like => e.g. - python pa2_svm.py

1 Data preprocessing

1. Data preprocessing is executed in a method called preprocess_data()
2. Appended train and test file together and converted it to adult.csv
3. Read csv file using pandas by speicifying na_values=['?'] to consider for missing values and later on drop those columns using dropna() method in pandas.
4. From the data I removed work-class race and native-country as they didn't form a substantial logic for the income prediction. After reading the data description I still didn't understand what fnlwgt meant so I looked up online and I don't think it will have any credibility for income prediction so dropped that too
5. From the data available I found online how to convert ordinal and nominal variables to convert to boolean and feed it to sklearn estimation. For continous values too e.g. if data is in the range say different numbers from 0 to 1000 then we do the same procedure as ordinal variables i.e. Go through entire dataset if a value corresponding to the current value is found then make it true else False. Repeat this for every unique value. Apply this same procedure for each algorithm.

   I have explained this in the code too for better understanding.
6. Since we are predicting whether income is greater than \$50K or not I use > \$50K as the parameter and drop <= \$50K

2 Training the data

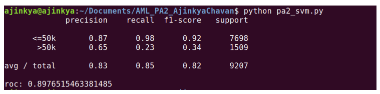
1. After the procedure for preprocessing data we train the data for the 4 Machine Learning algorithms
2. In training I split train/test to 0.8/0.2.
3. Using sklearn fit the X and y from the training data from preprocessing data and then predict using Xtest and ytest
4. Using classification_report from sklearn.metrics we get precision recall and f1 score and support.
5. After that using predict_proba for the given estimator to get the yscore and getting roc value using it.

3 Choice of parameter = ROC

I chose ROC as a better performance measure as opposed to F1 score Precision Recall because of 2 reasons:

1. Data is not askewed. Every attribute of a given category isn't equally distributed and so accuracy cannot be considered as a reliable measure

2. Our main aim is to maximize our prediction(TPR) or minimize the error. This can be best measured by using ROC curve by finding the ratio of True Positive rate vs False Positive Rate. So even though precision and recall are good measures but for current case ROC works the best.



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_svm.py
             precision    recall  f1-score   support

      <=50k       0.87      0.98      0.92      7698
       >50k       0.65      0.23      0.34      1509

avg / total       0.83      0.85      0.82      9207

roc: 0.8976515463381485
```

Figure 1: svm C=0.1

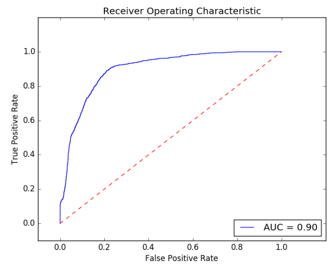
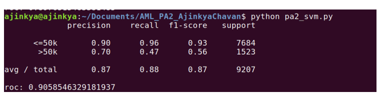


Figure 2: svm roc C=0.1



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_svm.py
             precision    recall  f1-score   support

      <=50k       0.90      0.96      0.93      7684
       >50k       0.70      0.47      0.56      1523

avg / total       0.87      0.88      0.87      9207

roc: 0.9058546329181937
```

Figure 3: svm C=0.5

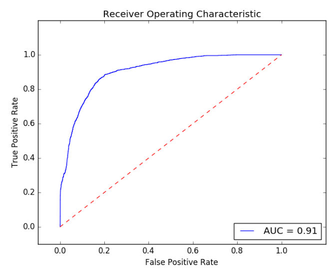


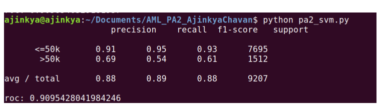
Figure 4: svm roc C=0.5

4 Effect of Hyperparameters

4.1 SVM

For SVM more the value of C better the result.

Below is output for C=0.1,0.5,1.0:



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_svm.py
                   precision    recall  f1-score   support

      <=50k       0.91      0.95      0.93      7695
       >50k       0.69      0.54      0.61      1512

avg / total       0.88      0.89      0.88      9207

roc: 0.9095428041984246
```

Figure 5: svm C=1.0

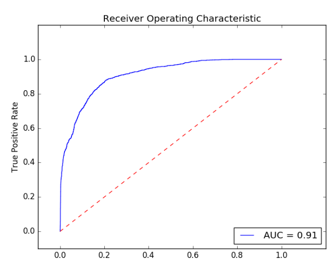
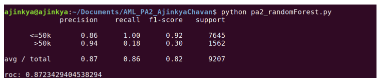


Figure 6: svm roc C=1.0


```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.86      1.00      0.92      7645
       >50k       0.94      0.18      0.30      1562

avg / total       0.87      0.86      0.82      9207

roc: 0.8723429404538294
```

Figure 7: random forest estimators=5 depth=5

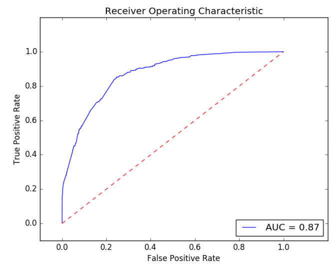
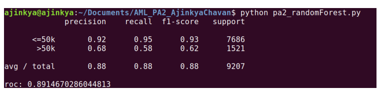


Figure 8: random forest roc estimators=5 depth=5


```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.92      0.95      0.93      7686
       >50k       0.68      0.58      0.62      1521

avg / total       0.88      0.88      0.88      9207

roc: 0.8914670286044813
```

Figure 9: random forest estimators=5 depth=50

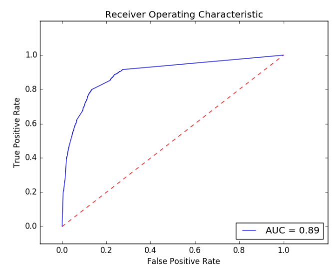
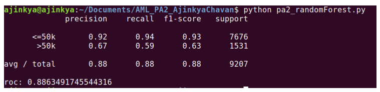


Figure 10: random forest roc estimators=5 depth=50

4.2 Random Forest

In random forest as the number of estimators grow the accuracy increases till a point where we find a perfect combination of number of estimators and max_depth. After that point the accuracy decreases again however the decrease in accuracy is not as steep for a given number of estimators with higher depths.

Below is the output for number of estimators and depth combinations of : (5,5)(5,50)(5,100)(50,5)(50,50)(50,100)(100,5)(100,50)(100,100)



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.92      0.94      0.93      7676
       >50k       0.67      0.59      0.63      1531

avg / total       0.88      0.88      0.88      9207

roc: 0.8863491745544316
```

Figure 11: random forest estimators=5 depth=100

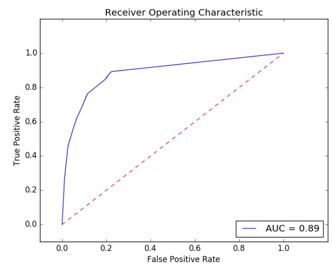
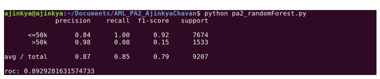


Figure 12: roc random forest estimators=5 depth=100



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.84      1.00      0.92      7674
       >50k       0.98      0.08      0.15      1533

avg / total       0.87      0.85      0.79      9207

roc: 0.8929281631574733
```

Figure 13: random forest estimators=50 depth=5

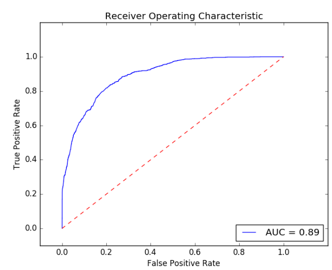
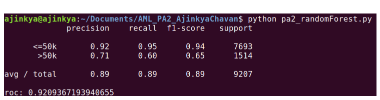


Figure 14: roc random forest estimators=50 depth=5



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.92      0.95      0.94      7693
       >50k       0.71      0.60      0.65      1514

avg / total       0.89      0.89      0.89      9207

roc: 0.9209367193940655
```

Figure 15: random forest estimators=50 depth=50

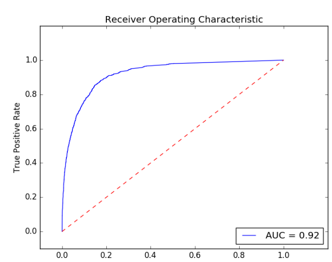
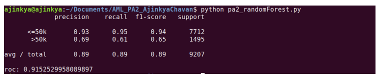


Figure 16: roc random forest estimators=50 depth=50



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.93      0.95      0.94      7712
       >50k       0.69      0.61      0.65      1495

avg / total       0.89      0.89      0.89      9207

roc: 0.9152529958089897
```

Figure 17: random forest estimators=50 depth=100

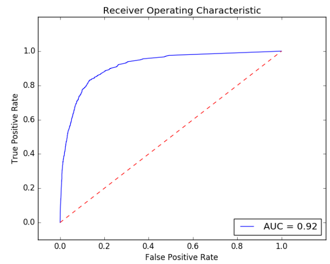
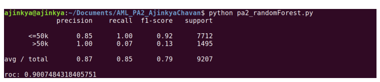


Figure 18: roc random forest estimators=50 depth=100



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.85      1.00      0.92      7712
       >50k       1.00      0.07      0.13      1495

avg / total       0.87      0.85      0.79      9207

roc: 0.9007484318405751
```

Figure 19: random forest estimators=100 depth=5

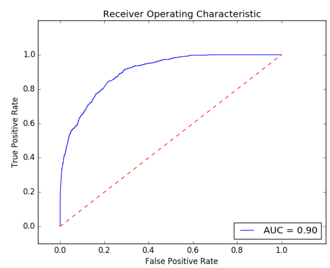
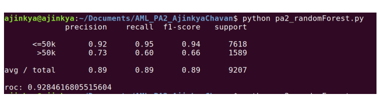


Figure 20: roc random forest estimators=100 depth=5



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.92      0.95      0.94      7618
       >50k       0.73      0.60      0.66      1589

avg / total       0.89      0.89      0.89      9207

roc: 0.9284616805515604
```

Figure 21: random forest estimators=100 depth=50

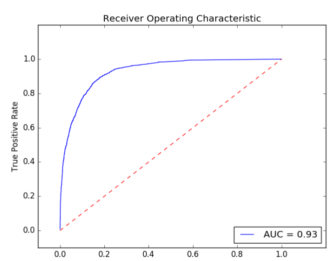
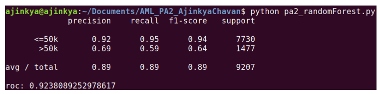


Figure 22: roc random forest estimators=100 depth=50



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_randomForest.py
             precision    recall  f1-score   support

      <=50k       0.92      0.95      0.94      7730
       >50k       0.69      0.59      0.64      1477

avg / total       0.89      0.89      0.89      9207

roc: 0.9238089252978617
```

Figure 23: random forest estimators=100 depth=100

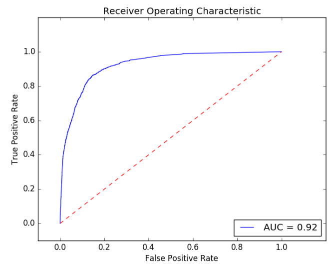
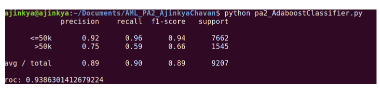


Figure 24: roc random forest estimators=100 depth=100



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.92      0.96      0.94      7662
       >50k       0.75      0.59      0.66      1545

avg / total       0.89      0.90      0.89      9207

roc: 0.9386301412679224
```

Figure 25: ada estimators=5 depth=5

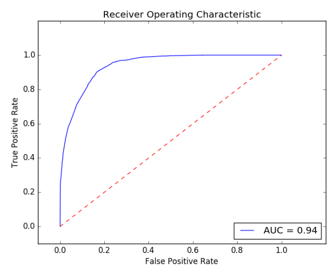
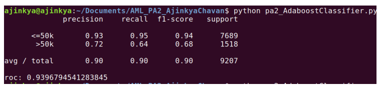


Figure 26: roc ada estimators=5 depth=5

4.3 Adaboost

I used Decision tree as the base estimator for Adaboost. From the combinations of number of estimators and depth of the tree I found that for number of estimators =5 50 100 -¿ lesser the depth of tree better the results. As depth increases the accuracy decreases beyond a certain point of combination of number of estimators and depth.

Below is the output for number of estimators and depth combinations of : for:(5,5)(5,50)(5,100)(50,5)(50,50)(50,100)(100,5)(100,50)(100,100)



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.93      0.95      0.94      7689
       >50k       0.72      0.64      0.68      1518

avg / total       0.90      0.90      0.90      9207

roc: 0.9396794541283845
```

Figure 27: ada estimators=5 depth=50

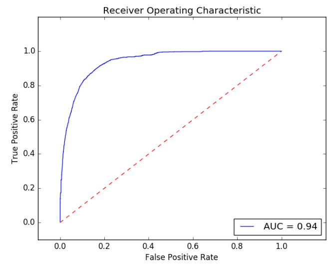
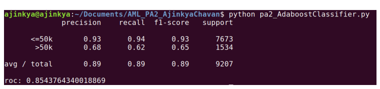


Figure 28: roc ada estimators=5 depth=50



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.93      0.94      0.93      7673
       >50k       0.68      0.62      0.65      1534

avg / total       0.89      0.89      0.89      9207

roc: 0.8543764340018869
```

Figure 29: ada estimators=5 depth=100

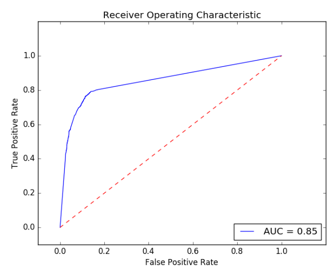
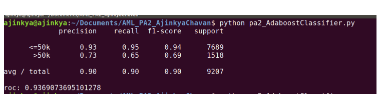


Figure 30: roc ada estimators=5 depth=100



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.93      0.95      0.94      7689
       >50k       0.73      0.65      0.69      1518

avg / total       0.90      0.90      0.90      9207

roc: 0.9369073695101278
```

Figure 31: ada estimators=50 depth=5

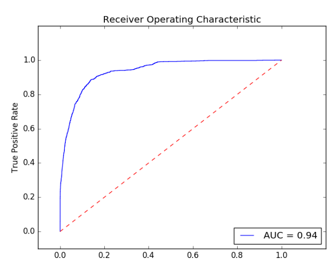
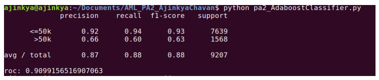


Figure 32: roc ada estimators=50 depth=5



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.92      0.94      0.93      7639
       >50k       0.66      0.60      0.63      1568

avg / total       0.87      0.88      0.88      9207

roc: 0.9099156516907063
```

Figure 33: ada estimators=50 depth=50

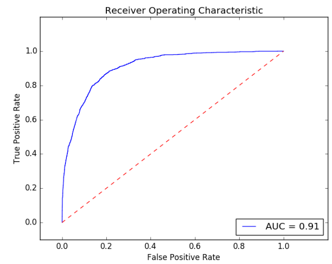
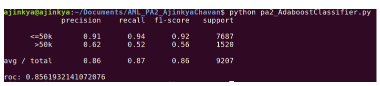


Figure 34: roc ada estimators=50 depth=50



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.91      0.94      0.92      7687
       >50k       0.62      0.52      0.56      1520

avg / total       0.86      0.87      0.86      9207

roc: 0.8561932141072076
```

Figure 35: ada estimators=50 depth=100

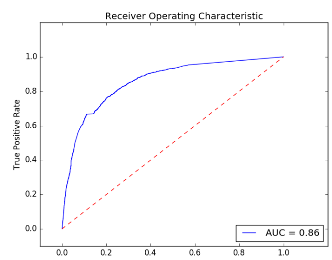
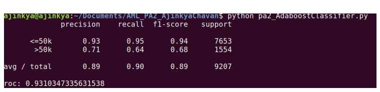


Figure 36: roc ada estimators=50 depth=100



```
ajinkya@ajinkya:~/Documents/AML_PA2_AjinkyaChavan$ python pa2_AdaboostClassifier.py
             precision    recall  f1-score   support

      <=50k       0.93      0.95      0.94      7653
       >50k       0.71      0.64      0.68      1554

avg / total       0.89      0.90      0.89      9207

roc: 0.9310347335631538
```

Figure 37: ada estimators=100 depth=5

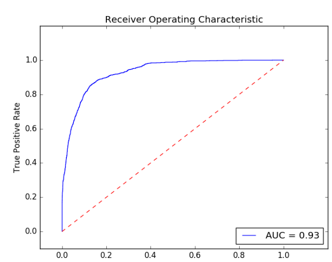


Figure 38: roc ada estimators=100 depth=5

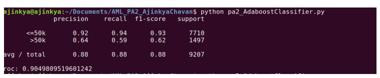


Figure 39: ada estimators=100 depth=50

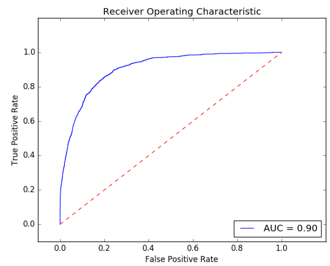


Figure 40: roc ada estimators=100 depth=50

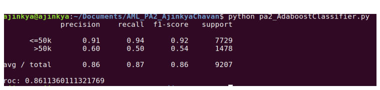


Figure 41: ada estimators=100 depth=100

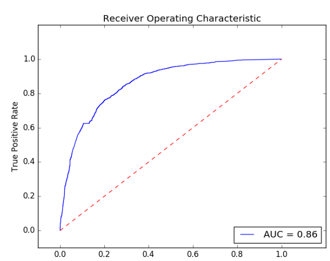


Figure 42: roc ada estimators=100 depth=100

5 Performance comparison

SVM is non-parametric model and hence training gets more expensive with larger datasets.

The complex and larger dataset leads to more number of support vectors and lots of tuning is required in SVM, where as in Random forest and Adaboost, its not adherent to a lot of tuning and is worry-free approach. So SVM's are not the best choice as compared to Random Forest and Adaboost.

Between Random forest and Adaboost, we could think of Random forest as more of a bagging technique. In random forests, there are parallel ensembles, i.e. each model is built independently and its aim is to decrease variance. So random forests are very good for high variance low bias models.

While Adaboost is as the name goes, a boosting technique. Its aim is to decrease bias and works great with high bias low variance models.

So if we have to speak about their performance comparison, the best answer would be I think a combination of Random forests and Adaboot i.e bagging and boosting to lower variance and bias to get a good result.

Random Forest $\approx Adaboaost > SVM$