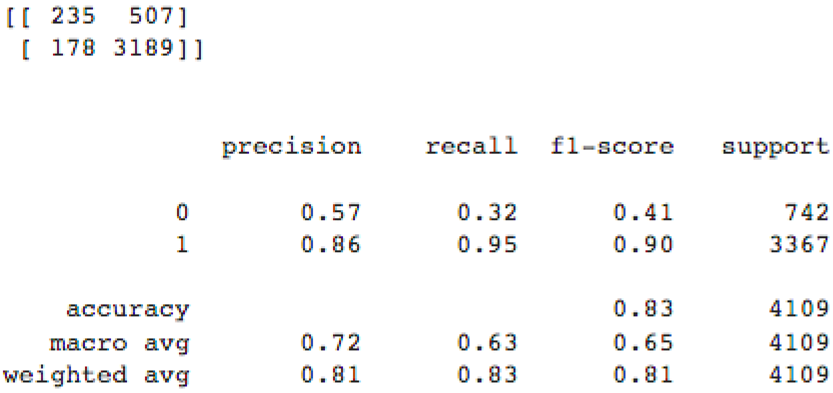
01/04/2022, Tuesday

502 Advanced Data Analysis with Python
Prof. D. Carlson
CSES Poputation Survey Prediction
Homework 3 Report
Işık Topçu

In the data exploration part, I have printed out the value counts for every feature to see which variables could be the best predictors for training and to see which ones have the least missing values. I have arrived at six features which also made theoretical sense to me. Then I imputed the missing values (in my case my features' missing values were 9s and 99s) by using the SimpleImputer method from ScikitLearn. After that, I have encoded my five discrete features (age is already continuous) , binarifying them using the Pandas get_dummies() method. I have then split my training and test datasets by using the train_test_split() method from ScikitLearn.

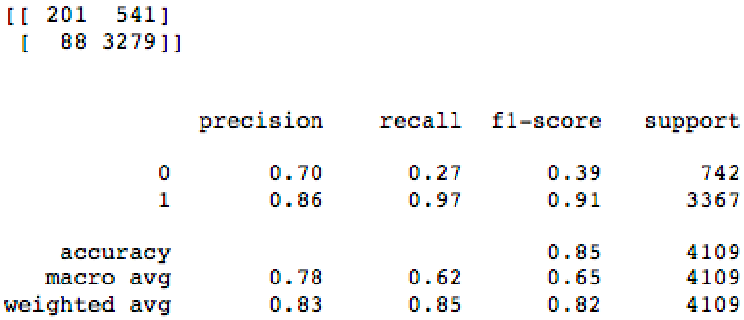
I have then tried GaussianNB, Logistic Regression, KNeighbors, Random Forest and SVC to find the best classifier fit for my data. One of the best performing algorithms, according to the classification reports, was the Random Forest Classifier with a classification report;



```
[[ 235  507]
 [ 178 3189]]


              precision    recall  f1-score   support

           0       0.57      0.32      0.41       742
           1       0.86      0.95      0.90      3367

    accuracy                           0.83      4109
   macro avg       0.72      0.63      0.65      4109
weighted avg       0.81      0.83      0.81      4109
```

Almost every algorithm had difficulty classifying the people who said they didn't vote. (Vote=False) I'm guessing that this might be the result of the dataset having disproportionately less false voted results. (True:10226, False:2225) The model had more data to train on the "voted" but much less data to train on the "not-voted" samples. This disproportion between 0's and 1's can maybe be explained by this. Still, for such simple algorithms, the scores don't seem that terrible.

I have also tuned the SVC, one of the algorithms most affected by this issue. I have used GridSearchCV to figure out hyperparameters to tune (C, gamma and kernel), which really helped improve the scores of the SVC algorithm;



```
[[ 201  541]
 [  88 3279]]


              precision    recall  f1-score   support

           0       0.70      0.27      0.39       742
           1       0.86      0.97      0.91      3367

    accuracy                           0.85      4109
   macro avg       0.78      0.62      0.65      4109
weighted avg       0.83      0.85      0.82      4109
```