CS 334: Homework #5
Ruohan Zhang

1. (5+10+10+10=35 pts) PCA & NMF For this problem, we will try to quantify the impact of dimensionality reduction on logistic regression.

(b) How many components were needed to capture at least 95% of the variance in the original data. Discuss what characterizes the first 3 principal components (i.e., which original features are important)

6 components capture at least 95% (`0.9558504315533898`) of the variance.

```
[[3.72055716e-02 1.03944572e-02 2.46263594e-04 1.71311658e-01
  2.25983938e-02 6.77262318e-01 7.13279523e-01 1.53329911e-04
  1.44135376e-05 2.78458549e-02 2.05408976e-02]
```

free sulfur dioxide and *total sulfur dioxide* are important for PC1

```
[2.11210278e-01 2.83702216e-01 8.60044481e-01 1.91074627e-01
 2.60236857e-01 6.17111400e-02 1.35045140e-02 1.24217659e-03
 4.03405933e-02 1.58725076e-01 1.34474307e-02]
```

citric acid is the most important and *fixed acidity, volatile acidity, chlorides* are important for *PC2*

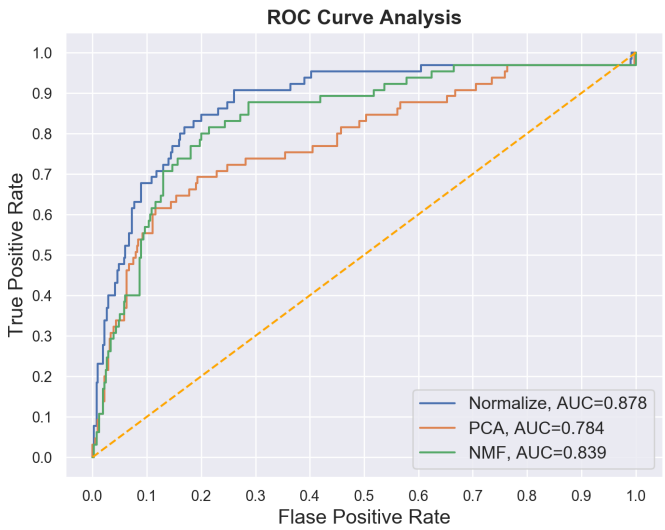
```
[5.56406681e-02 2.05159118e-01 1.68500669e-01 2.69842373e-01
 8.99377426e-01 6.05135438e-02 7.93387899e-02 1.00860347e-04
 8.45624802e-03 1.76609307e-01 5.92268309e-02]
```

chlorides is the most important and *volatile acidity, residual sugar* are important for PC3

(c) Justify your selection of R for NMF.

R = 6, because 6 principle component can capture 95% variance, thus transformed data with a dimension of 6 should be a good approximation of the original data.

(d) Plot the ROC curves for all 3 models (normalized dataset, PCA dataset, NMF dataset) on the same graph. Discuss your findings from the ROC plot.



Dimensionality reduction reduce the complexity of data
but **reduces** the **accuracy** of the model.

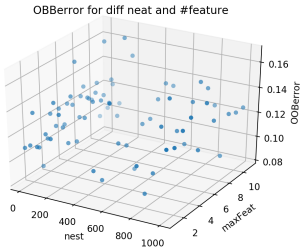
2. (30+10+5=45 pts) Almost Random Forest

(b) Find the best parameters on the wine quality training dataset based on classification error. Justify your selection with a few plots or tables.

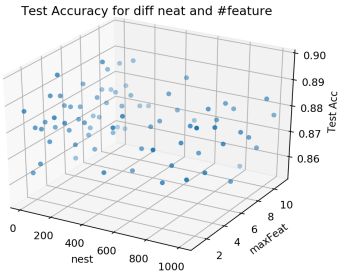
In HW2's question 3, we implement Decision Tree on the wine data and find the optimal parameter are *the Gini criterion, max depth of 3, and minimum number of samples is 5*.
Thus I use these value directly and tune on nest and maxFeatures.

maxFeat [1,2,3,4,5,6,7,8,9,10,11]
nest[10,50,100,200,500,800,1000]

	nest	maxFeat	OOBerror	Test Acc
0	10	1	0.110119	0.887500
1	10	2	0.133929	0.879167
2	10	3	0.086310	0.879167
3	10	4	0.125000	0.875000
4	10	5	0.125000	0.881250
..	...	...	...	...
72	1000	7	0.130952	0.879167
73	1000	8	0.098214	0.875000
74	1000	9	0.127976	0.868750
75	1000	10	0.125000	0.885417
76	1000	11	0.125000	0.877083




```
print(param.iloc[param['OOBerror'].argmin()])
nest        500.000000
maxFeat       6.000000
OOBerror      0.083333
Test Acc      0.877083
```




```
print(param.iloc[param['Test Acc'].argmax()])
nest        100.000000
maxFeat       3.000000
OOBerror      0.139881
Test Acc      0.897917
```

Optimal parameter minimize OOBerror on traning data, best parameters are:
Nest =500, maxfeat = 6, criterion = gini, maxDepth = 3, minLeaf =5
(Nest =100, maxfeat = 3, criterion = gini, maxDepth = 3, minLeaf =5 generates better test accuracy)

(c) Using your optimal parameters, how well does your version of the random forest perform on the test data? How does this compare to the estimated OOB error?

Nest =500, maxfeat = 6, criterion = gini, maxDepth = 3, minLeaf =5
Test accuracy: 0.877083
OOB accuracy: 1-0.083333 = 0.916667
This random-forest model performs better on OOB samples than on teat samples.

3. (15+5=20 points) Gradient Boosting

(a) Find the best parameters on the wine quality training dataset for xgboot. The parameters you'll want to consider tuning are num rounds,learning rate,max depth.

n_round [100,200,500,800,1000]
learning_rate[0.1, 0.5, 1, 2]
max_depth[3,4,5,6]

I use GridSearchCV with KFold (k=5) to find out optimal (learning rate, max_depth) combination for different number of rounds.

n_round	max_depth	learning_rate	Best cv train accuracy
100	5	0.1	0.89723
200	5	0.1	0.892761
500	6	0.1	0.894549
800	6	0.5	0.892761
1000	6	0.5	0.892761

Optimal param: n_round = 100, max_depth = 5, learning_rate = 0.1 because this combination generates best traning accuracy.

(b) Using the optimal parameters, how well does the xgboost model perform on the test data? How does this compare to your results from 2(c)?

`(100,5,0.1)Test Accuracy: 0.912`
(n_round = 500, max_depth = 6, learning_rate = 0.1 generates a better accuracy (0.917) on test data but in (a) I choose optimal parameter only base on best accuracy from kfold cross validation preformed on training data.)

This xgboost model (test accuracy: 0.912) performs better than the random-forest model in 2(c) (test accuracy: 0.877.