HW7 Report

1. Libraries:

I used numpy and pandas as the essential tools for data organization/manipulation, and scikit-learn to implement the ML methods we learned in class.

1. Data Organization
   1. Class Ratios:
      1. We see that the ratio of y=1’s are only 15% of the dataset. Therefore, I need to use a stratified folding algorithm for cross-validation.
   2. Removing low variant binary & continuous features:
      1. I observed that there are quite few features that differ very little. I picked a threshold of p=0.01 to eliminate some of the invariant binary features using Bernoulli density’s variance formula p\*(1-p).
      2. I then normalized the continuous features, which are already a little in number, and calculated variances. I discarded the ones with variance that is less than or equal to 0.01.
   3. Feature Selection
      1. For the rest, I still had 79 features, which would be quite time-consuming throughout iterations.
      2. I used RandomForestClassifier’s feature\_importances\_ attribute to specify the contributions of each feature and derived
2. Sparing Test Data
   1. I have the X\_test.csv for submission, but not the labels. So, I thought I should test the model beforehand, as I already have 300K data points which is more than enough to spare some of. I used sklearn’s train\_test\_split function to separate 20% of the data with stratifying y.
3. Model Trials

In each trial, I used GridSearchCV, which is a method in sklearn, to try combinations of given parameter set. Another advantage of GridSearchCV is that it can separate the data into training and validation sets and obtain scores of each. I used RepeatedStratifiedKFold for each algorithm where n\_splits = 3 and n\_repeat = 2, to validate the models with 6 shuffled, diverse sets. I also used the following parameters for the algorithms:

* 1. Random Forest Classifier:

n\_estimators=[1,10,50,200,600,1000]

criterion=['gini','entropy']

max\_depth=[4,6,8,12]

min\_samples\_split=[50,200,1000]

min\_samples\_leaf=[20,80,400]

max\_features=[10,20,'auto']

class\_weight=[{1:1,0:1},

{1:1.5,0:0.5}]

* 1. Adaptive Boosting Classifier: I used Logistic Regression as the weak learner and repeated it n\_estimators times.

base\_estimator: [LogisticRegression(penalty='l2',

dual=False,

tol=0.01,

C=1.0,

class\_weight=None,

random\_state=421,

max\_iter=4)],

n\_estimators:[5,50,200,1000]

learning\_rate:[0.8,0.6]

random\_state:[421]

* 1. Gradient Boosting Classifier

loss:['deviance'],

learning\_rate:[0.01,0.001,0.0001],

n\_estimators:[50,200,400],

criterion:['friedman\_mse'],

min\_samples\_split:[100],

min\_samples\_leaf:[40],

max\_depth:[7],

random\_state:[421],

max\_features:[None,40,60]

1. Results

Grid search process is very time consuming, so I could not try every parameter set that I aimed to. So, I get the probabilities with one of the models I found via RandomForestClassifier with the following parameters:

'clf\_\_n\_estimators':[400]

'clf\_\_criterion':['entropy']

'clf\_\_max\_depth':[13]

'clf\_\_min\_samples\_split':[50]

'clf\_\_min\_samples\_leaf':[20]

| **0** |
| --- |
| **mean\_fit\_time** | 10.9543 |
| **std\_fit\_time** | 0.706547 |
| **mean\_score\_time** | 1.48762 |
| **std\_score\_time** | 0.23725 |
| **param\_clf\_\_class\_weight** | {1: 1, 0: 1} |
| **param\_clf\_\_criterion** | entropy |
| **param\_clf\_\_max\_depth** | 12 |
| **param\_clf\_\_min\_samples\_leaf** | 20 |
| **param\_clf\_\_min\_samples\_split** | 50 |
| **param\_clf\_\_n\_estimators** | 100 |
| **param\_clf\_\_random\_state** | 421 |
| **params** | {'clf\_\_class\_weight': {1: 1, 0: 1}, 'clf\_\_crit... |
| **split0\_test\_score** | 0.7638 |
| **split1\_test\_score** | 0.761719 |
| **mean\_test\_score** | 0.76276 |
| **std\_test\_score** | 0.00104043 |
| **rank\_test\_score** | 1 |
| **split0\_train\_score** | 0.794854 |
| **split1\_train\_score** | 0.797119 |
| **mean\_train\_score** | 0.795986 |
| **std\_train\_score** | 0.0011326 |