CSE 472
Machine Learning Sessional
Assignment 2 Report
Logistic Regression and AdaBoost for Classification

Name: A. H. M. Osama Haque
ID: 1805002
Section: A
Date of Submission: 09.12.2023

Script Running Instructions

1. **Directory Tree:** 3 datasets downloaded from
   a. https://www.kaggle.com/blastchar/telco-customer-churn
   b. https://archive.ics.uci.edu/ml/datasets/adult
   c. https://www.kaggle.com/mlg-ulb/creditcardfraud

- Then in the root directory, create a directory named "datasets"
- Inside the "datasets" directory, create subdirectories for each downloaded dataset, using the numbering scheme ("1," "2," "3")
- The directory tree looks like below

```
datasets
  1
    WA_Fn-UseC_-...
  2
    adult.data
    adult.names
    adult.test
    Index
    old.adult.names
  3
    creditcard.csv
```

2. **Dataset Type:** Change dataset_type in line 10 accordingly

```
9
10  dataset_type = 1
11
12  missing_columns = ['
```

3. **Regressor signature:** change hyperparameters according to signature (in lines 375, 456)

```
class modified_regressor:
    def __init__(self, X_train, Y_train,
                 data_point_weights, threshold=-1, max_feature_count=-1, learning_rate=0.1):
        self.X = X_train
```

```
374         # initialize regressor
375         regressor = L_weak(resampled_examples_X, resampled_examples_Y, data_point_weights, 0.5, num_features)
376         regressor.train()
377         y_hat = regressor.predict(examples_X)
```

```
454
455   data_point_weights = np.ones(len(trunc_X_train)) / len(trunc_X_train)
456   regressor = modified_regressor(trunc_X_train, Y_train, data_point_weights)
457   regressor.train()
458
459   print(f"\nWithout Boosting:\n")
460   print(f"Training data")
461
```

4. **No. of epochs:** change in line 319 and 330 accordingly

```
319                 steps = 1000
320                 for i in range(steps):
321                     self.gradient_descent()
322                     iteration += 1
323                     z = np.dot(self.X, self
324                     y_hat = self.sigmoid(z)
325                     sum_error = np.sum(self
326                     error = sum_error / len
327                     if error <= self.thresh
328                         break
329             else:
330                 steps = 1000
331                 for i in range(steps):
```

5. **Adaboost signature:** change hyperparameters according to signature

```
350  def adaboost(examples_X, examples_Y, L_weak, K, num_features=-1):
351      """
352      Parameters
```

6. **No. of hypotheses:** Change K accordingly to vary number of hypothesis in line 474

```
473
474  for K in range(5, 30, 5):
475      print(f"K: {K}")
476      hypothesises, hypo_weights = adaboost(trunc_X_train, Y_train,
477                                            modified_regressor, K)
478      ## predictions
```

7. **Plots:** Uncomment lines from 495 to generate plot. Uncomment any one from lines 495(dataset1), 496(dataset2), 497(dataset3)

```
492
493  # #plot accuracy, F1 score of train and test set vs max_feature_co
494  # # for dataset1,2,3
495  # feature_counts = [5, 10, 15, len(trunc_X_train.columns)]
496  # feature_counts = [20, 40, 60, len(trunc_X_train.columns)]
497  # feature_counts = [5, 10, 20, len(trunc_X_train.columns)]
498
```

8. **Run:** Run 1805002.py

Dataset: TelcoCustomerChurn

Epochs = 1000

Performance Measure	Training	Test
Accuracy	80.3 %	79.91 %
True positive rate (sensitivity, recall, hit rate)	54.36 %	53 %
True negative rate (specificity)	89.72 %	89.43 %
Positive predictive value (precision)	65.75 %	63.94 %
False discovery rate	34.25 %	36 %
F1 score	59.52 %	57.95 %

Number of boosting rounds	Training	Test
5	78.36 %	77.36%
10	77.8 %	76.93 %
15	78.29 %	76.65 %
20	78.56 %	77.57 %
25	78.36 %	77.35 %

Dataset: Adult

Epochs = 1000

Performance Measure	Training	Test
Accuracy	82.61 %	77.63 %
True positive rate (sensitivity, recall, hit rate)	48.85 %	16.43 %
True negative rate (specificity)	93.32 %	96.55 %
Positive predictive value (precision)	69.88 %	59.62 %
False discovery rate	30.12 %	40.38 %
F1 score	57.5 %	25.76 %

Number of boosting rounds	Training	Test
5	82.46 %	76.6 %
10	83 %	76.9 %
15	83.24 %	76.7 %
20	83.07 %	76.94 %
25	83.09 %	77.33 %

Dataset: CreditCardFraud

Epochs = 1000

(randomly selected 20000 negative samples + all positive samples)

Performance Measure	Training	Test
Accuracy	97.61 %	97.53 %
True positive rate (sensitivity, recall, hit rate)	0 %	0 %
True negative rate (specificity)	99.99 %	99.99 %
Positive predictive value (precision)	0 %	0 %
False discovery rate	0 %	0 %
F1 score	0 %	0 %

Number of boosting rounds	Training	Test
5	99.35 %	99.14 %
10	99.31 %	99.27 %
15	99.37 %	99.29 %
20	99.34 %	99.2 %
25	99.3 %	99 %

Observation:

Preprocessing: It is important to correctly preprocess dataset, sometimes it is observed that missing values may influence mean value, standardization. One-hot-encoding is applied on the whole dataset. However, min-max normalization is applied on the train and test set separately as we do not want to be biased towards the train dataset. We want to keep test data as unbiased as possible. Therefore splitting is done before normalization and feature selection.

No bias: In the defined logistic regressor, no bias term is used alongside the weights. This may have impacted the overall result. Especially in the CreditCardFraud dataset, the F1-score and precisions were very low while having very high accuracy. Reason is that because of very high numbers of negative labels, models tend to label every test data to negative. So it can perfectly score high specificity, because the negative data samples are high in number, accuracy is increased. However, applying adaboost solved this problem. With an increasing number of hypotheses, model precision and F1-score increases as well, meaning the model is now able to predict positive labels as well.

Low Accuracy in Adaboost: In some cases, the performance with boosting does not significantly be better than normal regressor.

Variable number of features: It can be observed that with increasing number of features selected through information gain, model performance improves. An example plot of dataset 1 is given below where both accuracy and F1-score increases with the increasing number of features selected by information gain