Assignment 6

Pedram Agand (301400607)
pagand@sfu.ca

March 14, 2022

Note: This assignment has been written with LaTeX.
Note: Please change the file name < *lasso.py* > to < *lasso1.py* > to avoid overshadowing the sklearn. It was renamed to < *lasso.py* > due to submission constraint.

Exercise 1

Instruction: Please run < *nb.py* > and read the result under Q1

```
10-fold cross validation total test error 0.0819
```

Exercise 2

Instruction: Please run < *lasso1.py* > and read the result under Q2
Note: I changed the file name so it does not shadow the lasso function in sklearn

```
10CV Error rate 0.030172413793103345
```

Comparing the two classifier, the lasso turns out to be more accurate in regard the test error.

Exercise 3

Instruction: to get the lasso train and test error, please run < *lasso1.py* > and look the result under "Q3". For the train and test error of the naive bayes and the plot comparing all of them, please run < *nb.py* >

```
<lasso>
train error for different data size
[0.         0.         0.         0.         0.002      0.015
 0.01571429 0.0225     0.02       0.018     ]
test error for different data size
[0.15086207 0.15086207 0.15517241 0.14224138 0.11206897 0.05603448
 0.03448276 0.03017241 0.03017241 0.03017241]

 <naive bayes>
 train error for different data size
[0.01       0.06       0.04       0.0725     0.092      0.09666667
 0.09714286 0.09375    0.11555556 0.104     ]
test error for different data size
[0.09051724 0.09051724 0.09482759 0.09051724 0.09482759 0.09913793
 0.09913793 0.09913793 0.0862069  0.0862069 ]

```

As we can see in Fig. 1, as the data increases, both train of NB and lasso increases. The reason is that the structure is the same, but we have more data points with uncertainty. So it

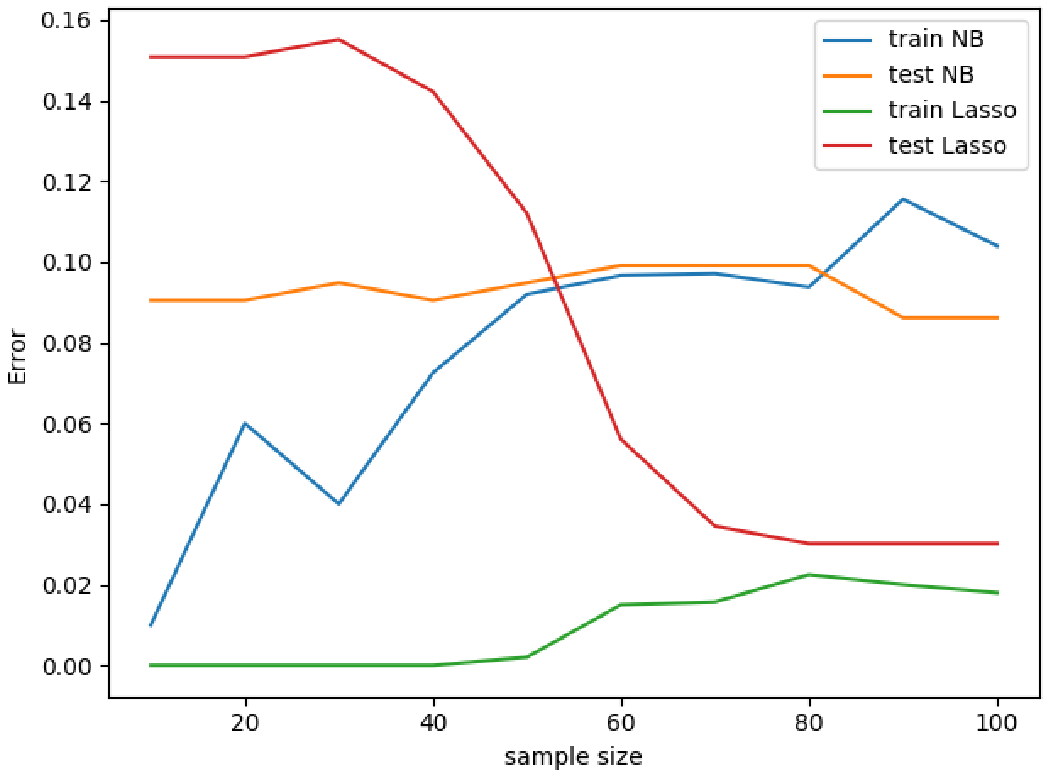

Figure 1: NB training error, NB test error, LASSO test error, and LASSO training error against sample size

make sense to have more training error. On the other hand, the test error of lasso decreases, as the classifier have more information and can judge the data more precisely, however, the test error of the NB is almost flat. This shows that the generative models like Bayes approach can have fairly good result even in few number of samples. But this is not the case for discriminate approaches like lasso, cause they are in danger of over fitting. This also show that for data less than 50, naive Bayes is a better approach, but with data more than 50, lasso will surpass this.

Exercise 4

Instructions: For the naive Bayes, please run $< nb.py >$ under Q4 and for lasso please run $< lasso1.py >$ under Q4.
Note that since the data is changing in this section, and back to normal in the last one, I did this in the last part in the code (after Q5).

For the naive Bayes, I first call the $< generate\_q4\_data >$ function, generate synthetics data and then load data with $< load\_simulate\_data >$. After that I called the $< evaluate >$ function. In order to access to the $P(Yes|Democrat)$ I create function $< get\_bill\_importance >$. To call this function, I need to change the $< evaluate >$ function to return the rate of yes for both Democrat and Republican to the total in the train set. You see the implementation in follows:

```
generate_q4_data(4000, "./data/synthetic.csv")
 global A_data, C_data
 A_data, C_data = load_simulate_data("./data/synthetic.csv")
 train_error = np.zeros(10)
 test_error = np.zeros(10)
```

```
  pdmr_list = np.zeros(10)
  for x in range(10):
    accuracy, train_accuracy, pdmr = evaluate(NBClassifier, train_subset=True,
     ↪  subset_size=(x + 1) * 400, get_bills = 1)
    pdmr_list[x] = pdmr
    train_error[x] = 1 - train_accuracy
    test_error[x] = 1 - accuracy
    print('  10-fold cross validation for synthetic data total test error {:2.4f}
     ↪  total train error {:2.4f} on {} ''examples'.format(
      1 - accuracy, 1 - train_accuracy, (x + 1) * 400))
  print(train_error)
  print(test_error)
  sample_size = range(400, 4400, 400)
  plt.figure()
  plt.plot(sample_size, pdmr_list)

```

```
def get_bill_importance(classifier, A, C):
  results = []
  dem = np.zeros(16)
  rep = np.zeros(16)
  for entry, c in zip(A, C):
    c_pred, unused = classifier.classify(entry)
    if(c_pred): # Democrat
      dem = dem + entry
    else:
      rep = rep+ entry
    results.append(c_pred)
    # print('logprobs', np.array(pp))

  pdem = dem/sum(results)
  prep = rep / (len(results)-sum(results))
  return sum(abs(pdem-prep)<0.1)/12

```

```
   def evaluate(classifier_cls, train_subset=False, subset_size = 0,get_bills =
    ↪  0):
   ...
    if (get_bills):
       pdmr = get_bill_importance(classifier, A_train, C_train)

    ...
    if get_bills:
        return 1.*tot_correct/tot_test, 1.*train_correct/train_test, pdmr
    else:
        return 1.*tot_correct/tot_test, 1.*train_correct/train_test

```

As we can see in the Fig. 2, the fraction of nonpartisan bills that are ignored are constant in the training size. This will show that even in a small data, the naive Bayes is capable of correctly identify the effective features. Here are the result:

```
 Q4
  10-fold cross validation for synthetic data total test error 0.0100 total train
   ↪  error 0.0078 on 400 examples
  10-fold cross validation for synthetic data total test error 0.0090 total train
   ↪  error 0.0101 on 800 examples
```

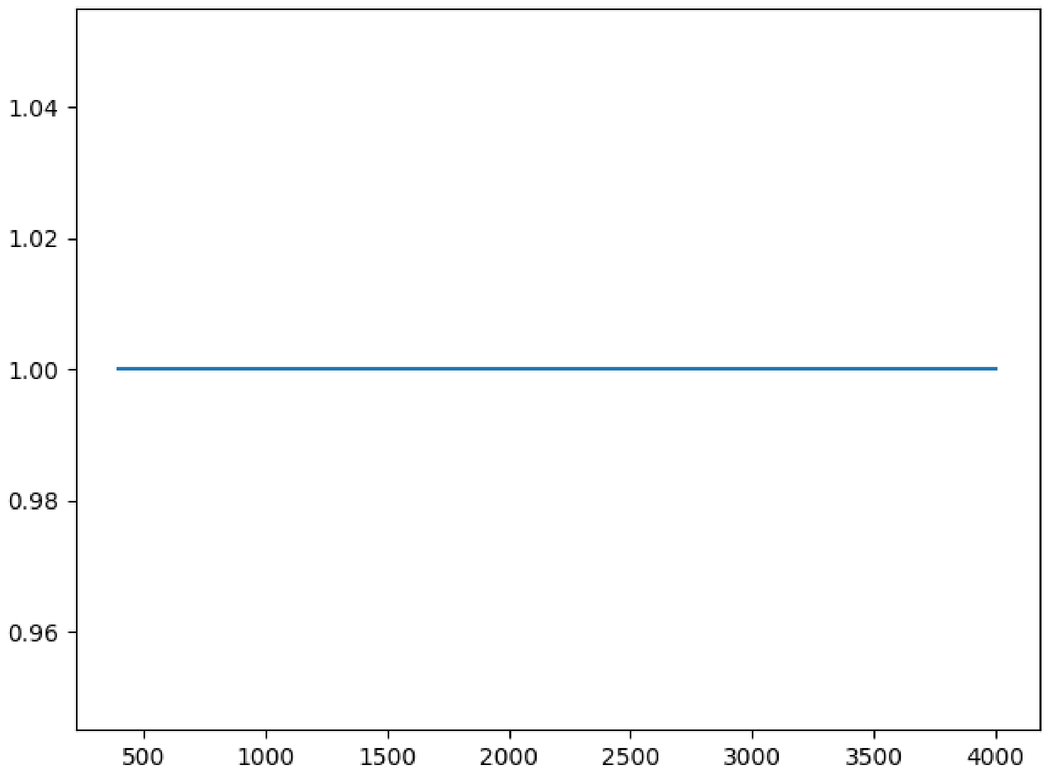

Figure 2: fraction of nonpartisan bills that are ignored as a function of the training set size for NB classifier

```
10-fold cross validation for synthetic data total test error 0.0092 total train
 ↪  error 0.0120 on 1200 examples
10-fold cross validation for synthetic data total test error 0.0092 total train
 ↪  error 0.0095 on 1600 examples
10-fold cross validation for synthetic data total test error 0.0082 total train
 ↪  error 0.0082 on 2000 examples
10-fold cross validation for synthetic data total test error 0.0092 total train
 ↪  error 0.0080 on 2400 examples
10-fold cross validation for synthetic data total test error 0.0085 total train
 ↪  error 0.0079 on 2800 examples
10-fold cross validation for synthetic data total test error 0.0090 total train
 ↪  error 0.0078 on 3200 examples
10-fold cross validation for synthetic data total test error 0.0090 total train
 ↪  error 0.0081 on 3600 examples
10-fold cross validation for synthetic data total test error 0.0090 total train
 ↪  error 0.0081 on 4000 examples

```

For the Lasso, I only changed the $< lasso\_evaluate >$ function to access the coefficient and find number of the ones that are zero. So, I read the synthetic data, and then make the (pay,paX) global variable and the call the $< lasso\_evaluate >$ for smaller subset as follows:

```python
dpa = pd.read_csv('./data/synthetic.csv')
dpa['Class'] =dpa['Class'].map({'republican': 0, 'democrat': 1})
for i in range(16):
        index = 'A' + str(i + 1)
        dpa[index] = dpa[index].map({'y': 1, 'n': 0})
# dpa.info()
global pay, paX
pay = dpa.Class
paX = dpa.drop('Class', axis=1)

train_error = np.zeros(10)
test_error = np.zeros(10)
pdmr_list = np.zeros(10)
for i in range(10):
        x, y,pdmr = lasso_evaluate(train_subset=True, subset_size=i * 400 + 400, get_coeff = 1)
        pdmr_list[i] = pdmr
        test_error[i] = y
        train_error[i] = x
        print('  10-fold cross validation for synthetic data total test error {:2.4f} total train error {:2.4f} on {} ''examples'.format(
                        y, x, (i + 1) * 400))
print(train_error)
print(test_error)
sample_size = range(400, 4400, 400)
plt.plot(sample_size,pdmr_list)
plt.show()
```

```
def lasso_evaluate(train_subset=False, subset_size = 0,get_coeff = 0):
    global pay, paX
    ...
    if get_coeff:
                coeff = lasso.coef_
                pdmr = sum(abs(coeff) ==0)/12
        ...
    if get_coeff:
                return 1.0 * tot_incorrect / tot_test, 1.0 * tot_train_incorrect / tot_train, pdmr
    else:
                return 1.0*tot_incorrect/tot_test, 1.0*tot_train_incorrect/tot_train
```

For the lasso, however, we cannot say that it can find the non important features, by simply making their coefficient equal to zero. As we can see in Fig. 3, the fraction of nonpartisan bills that are ignored are increasing by adding more data. This is not always monotonically increasing, there are some ups and downs in it as well. But compare to naive Bayes, it can not detect all the non-partisan features.

```
Q4
  10-fold cross validation for synthetic data total test error 0.0097 total train error 0.0090 on 400 examples
  10-fold cross validation for synthetic data total test error 0.0095 total train error 0.0097 on 800 examples
  10-fold cross validation for synthetic data total test error 0.0088 total train error 0.0077 on 1200 examples
  10-fold cross validation for synthetic data total test error 0.0088 total train error 0.0080 on 1600 examples
  10-fold cross validation for synthetic data total test error 0.0092 total train error 0.0085 on 2000 examples
  10-fold cross validation for synthetic data total test error 0.0103 total train error 0.0082 on 2400 examples
```

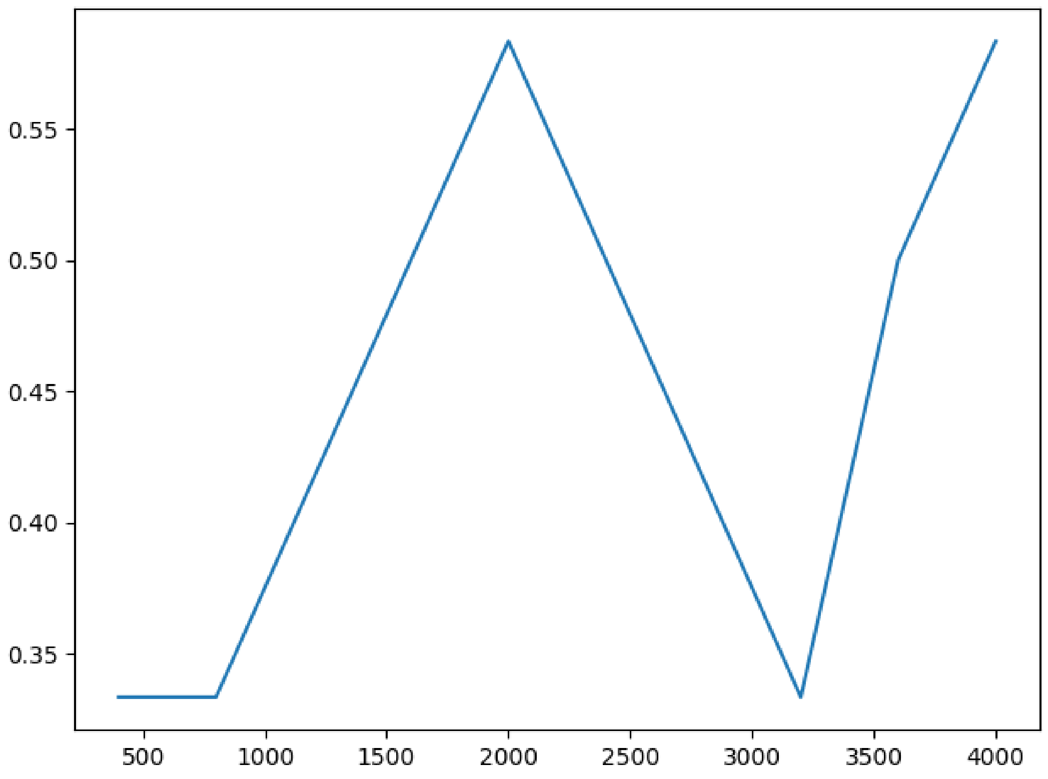

Figure 3: fraction of nonpartisan bills that are ignored as a function of the training set size for Lasso classifier

```
  10-fold cross validation for synthetic data total test error 0.0092 total train
    error 0.0083 on 2800 examples
  10-fold cross validation for synthetic data total test error 0.0092 total train
    error 0.0082 on 3200 examples
  10-fold cross validation for synthetic data total test error 0.0088 total train
    error 0.0084 on 3600 examples
  10-fold cross validation for synthetic data total test error 0.0088 total train
    error 0.0084 on 4000 examples

```

Exercise 5

Instruction: For this part please run $< nb.py >$ for naive Bayes, and $< lasso1.py >$ for lasso. The result are shown before Q4 in the code

```
Naive Bayes Classifier on missing data (Q5)
  P(C=1|A_observed) = 0.9826
Predicting vote of A12 using NBClassifier on missing data
  P(A12=1|A_observed) = 0.1416

```

```
  Q5
LASSO  P(C= 1|A_observed)
[0.93227933]

```

As we can see, we can evaluate on missing data, by marginalizing unknown variables. Furthermore, in order to obtain $A_{12}|A_{observed}$, we simply have:

$$P(A_{12}|A_{observed}) = \frac{\sum_{aobserved,c} p(A1, \cdots, A16, C)}{\sum_{aobserved,c,a12} p(A1, \cdots, A16, C)} \tag{1}$$