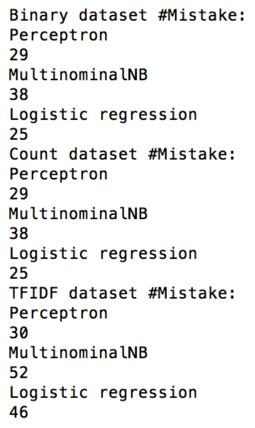
CS 334: Homework #4
Ruohan Zhang

1. Feature Extraction + Model Selection

(a) Define your model assessment strategy that you will use for this dataset. You must justify your methodology. Implement your methodology in the function model assessment.

Use same dataset with same train-test split for three models and compare the prediction accuracy of different models. If the number of mistake is smaller, the accuracy is better.
I use holdout method for this assignment but for final project I probably gonna use k-fold or mc cross validation.



```
Binary dataset #Mistake:
Perceptron
29
MultinominalNB
38
Logistic regression
25
Count dataset #Mistake:
Perceptron
29
MultinominalNB
38
Logistic regression
25
TFIDF dataset #Mistake:
Perceptron
30
MultinominalNB
52
Logistic regression
46
```

Thus, for Binary and Count I choose Linear Regression model and for TFIDF I choose Perceptron model.

2. (30+9+5 = 44 pts) Spam Detection via Perceptron

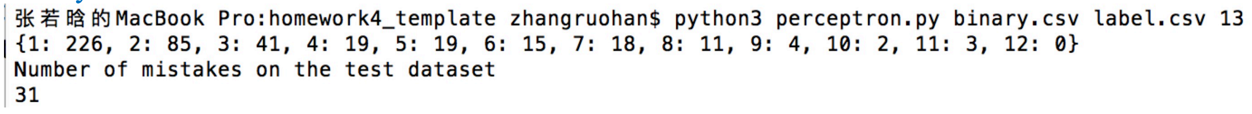
(a) Implement the perceptron algorithm in perceptron.py

Command format:
% python perceptron.py [feature csv] [label csv] mepoch
feature csv: *binary.csv* or *count.csv*, *tfidf.csv*
label csv: *label.csv*

(b) For each of the three datasets above, train the perceptron using your training set. How many mistakes are made before the algorithm terminates? What is your estimated predictive error? What should be the optimal number of epochs?

- Split data into 70% train and 30%test
- {#epoch : #mistake}

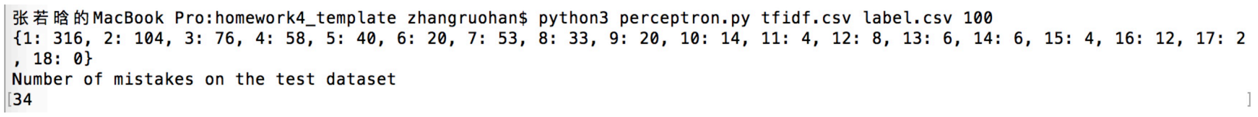
Binary dataset:



```
张若晗的MacBook Pro:homework4_template zhangruohan$ python3 perceptron.py binary.csv label.csv 13
{1: 226, 2: 85, 3: 41, 4: 19, 5: 19, 6: 15, 7: 18, 8: 11, 9: 4, 10: 2, 11: 3, 12: 0}
Number of mistakes on the test dataset
31
```

test error(w at terminate): 31
Estimated predictive error: 31/1800 = 0.017
Optimal epoch: 12 (#mistake= 0)

TDIDF dataset:



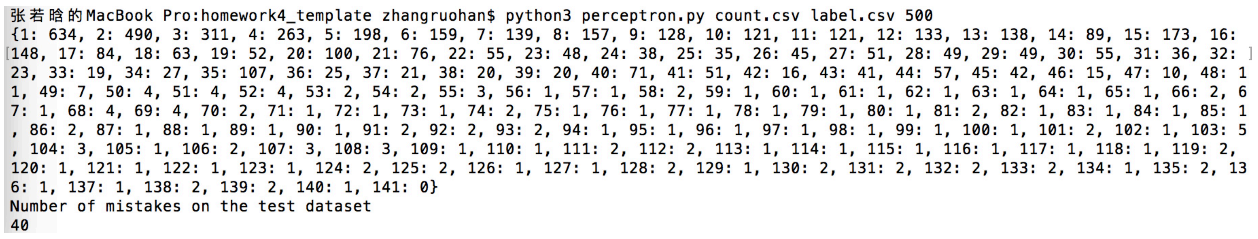
```
张若晗的MacBook Pro:homework4_template zhangruohan$ python3 perceptron.py tfidf.csv label.csv 100
{1: 316, 2: 104, 3: 76, 4: 58, 5: 40, 6: 20, 7: 53, 8: 33, 9: 20, 10: 14, 11: 4, 12: 8, 13: 6, 14: 6, 15: 4, 16: 12, 17: 2
, 18: 0}
Number of mistakes on the test dataset
34
```

test error(w at terminate): 34

Estimated predictive error: 34/1800 = 0.019

Optimal epoch: 18.

Count dataset:



```
张若晗的MacBook Pro:homework4_template zhangruohan$ python3 perceptron.py count.csv label.csv 500
{1: 634, 2: 490, 3: 311, 4: 263, 5: 198, 6: 159, 7: 139, 8: 157, 9: 128, 10: 121, 11: 121, 12: 133, 13: 138, 14: 89, 15: 173, 16:
148, 17: 84, 18: 63, 19: 52, 20: 100, 21: 76, 22: 55, 23: 48, 24: 38, 25: 35, 26: 45, 27: 51, 28: 49, 29: 49, 30: 55, 31: 36, 32:
23, 33: 19, 34: 27, 35: 107, 36: 25, 37: 21, 38: 20, 39: 20, 40: 71, 41: 51, 42: 16, 43: 41, 44: 57, 45: 42, 46: 15, 47: 10, 48: 1
1, 49: 7, 50: 4, 51: 4, 52: 4, 53: 2, 54: 2, 55: 3, 56: 1, 57: 1, 58: 2, 59: 1, 60: 1, 61: 1, 62: 1, 63: 1, 64: 1, 65: 1, 66: 2, 6
7: 1, 68: 4, 69: 4, 70: 2, 71: 1, 72: 1, 73: 1, 74: 2, 75: 1, 76: 1, 77: 1, 78: 1, 79: 1, 80: 1, 81: 2, 82: 1, 83: 1, 84: 1, 85: 1
, 86: 2, 87: 1, 88: 1, 89: 1, 90: 1, 91: 2, 92: 2, 93: 2, 94: 1, 95: 1, 96: 1, 97: 1, 98: 1, 99: 1, 100: 1, 101: 2, 102: 1, 103: 5
, 104: 3, 105: 1, 106: 2, 107: 3, 108: 3, 109: 1, 110: 1, 111: 2, 112: 2, 113: 1, 114: 1, 115: 1, 116: 1, 117: 1, 118: 1, 119: 2,
120: 1, 121: 1, 122: 1, 123: 1, 124: 2, 125: 2, 126: 1, 127: 1, 128: 2, 129: 1, 130: 2, 131: 2, 132: 2, 133: 2, 134: 1, 135: 2, 13
6: 1, 137: 1, 138: 2, 139: 2, 140: 1, 141: 0}
Number of mistakes on the test dataset
40
```

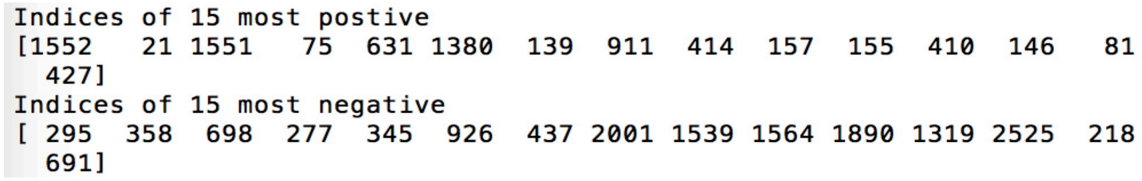
test error(w at terminate): 40

Estimated predictive error: 40/1800 = 0.022

Optimal epoch: 141.

(c) Using the vocabulary list together with the parameters learned in the previous question, output the 15 words with the most positive weights with your most promising perceptron. What are they? Which 15 words have the most negative weights?

Use binary dataset, epoch = 12. (code for indices in perceptron.py)

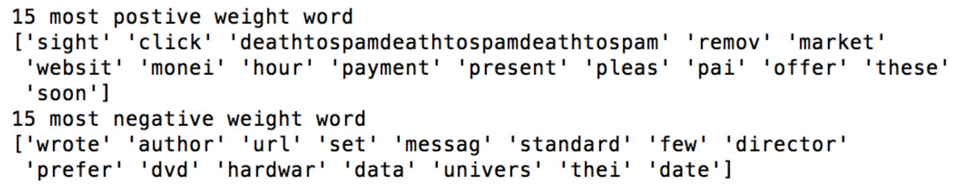


```
Indices of 15 most postive
[1552   21 1551   75  631 1380  139  911  414  157  155  410  146   81
  427]
Indices of 15 most negative
[ 295  358  698  277  345  926  437 2001 1539 1564 1890 1319 2525  218
  691]
```

My w[0] is bias, thus (index-1) is the correct index in vocalist

Use these indices with vocablist in q1.py

(code to print word in q1.py under model_assesment()):



```
15 most postive weight word
['sight' 'click' 'deathtospamdeathtospamdeathtospam' 'remov' 'market'
 'websit' 'monei' 'hour' 'payment' 'present' 'pleas' 'pai' 'offer' 'these'
 'soon']
15 most negative weight word
['wrote' 'author' 'url' 'set' 'messag' 'standard' 'few' 'director'
 'prefer' 'dvd' 'hardwar' 'data' 'univers' 'thei' 'date']
```

3. (15+9=24 pts) Spam Detection using Naive Bayes and Logistic Regression

Command format:

% python q3.py [feature csv]

feature csv: *binary.csv* or *count.csv*, *tfidf.csv*

(a) Train the appropriate Naive Bayes algorithm against each of the 3 datasets and assess the performance.

Naïve bayes:

GaussianNB is too general and our dataset is relatively small so I tried ComplementNB,BernoulliNB,and MultinominalNB on 3 dataset, code in q3.py.

Binary dataset:

#mistake:

ComplementNB: 38

BernoulliNB: 89

MultinominalNB: 38

TDIDF dataset:

#mistake:

ComplementNB: 66

BernoulliNB: 103

MultinominalNB: 52

Count dataset:

#mistake:

ComplementNB: 51

BernoulliNB: 89

MultinominalNB: 50

Thus, I choose MultinomialNB

Logistic Regression:

#mistake:

Binary dataset: 25

TDIDF dataset: 46

Count dataset: 38