

CS6375.003 FALL 2021
MACHINE LEARNING ASSIGNMENT - I

SUBMITTED BY:
NAME: AYUSH MISHRA

Detailed write-up with reports containing Accuracy, Precision, Recall and F1 Score.

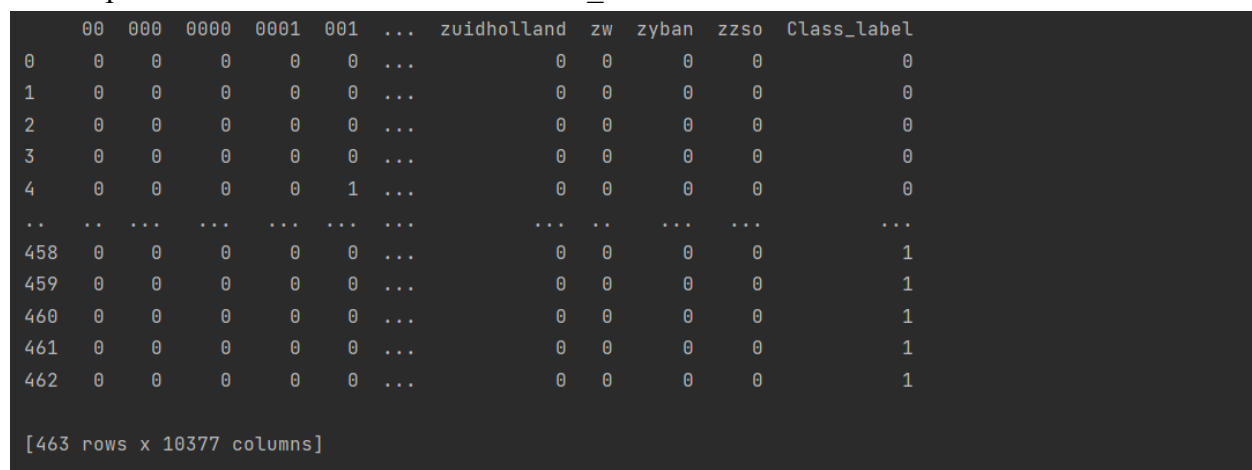
Multinomial Naive Bayes on the bag of words model:

Bag of words model:

To create this representation, all unique words in all the documents in the dataset were considered as features(bag of words). To generate a bag of words, I used CountVectorizer() from sklearn.feature_extraction.text.CountVectorizer.

To represent each email as a vector of word frequencies, I used vectorizer.fit_transform().

The snapshot below shows the feature x class_label matrix for dataset 1



	00	000	0000	0001	001	...	zuidholland	zw	zyban	zzso	Class_label
0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	...	0	0	0	0	0
2	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	1	...	0	0	0	0	0
..
458	0	0	0	0	0	...	0	0	0	0	1
459	0	0	0	0	0	...	0	0	0	0	1
460	0	0	0	0	0	...	0	0	0	0	1
461	0	0	0	0	0	...	0	0	0	0	1
462	0	0	0	0	0	...	0	0	0	0	1

[463 rows x 10377 columns]

Multinomial Naive Bayes:

The algorithm was coded from scratch. I used <http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf> (Figure 13.2) for algorithm reference. Also, 1-laplace smoothing was implemented. To prevent underflow, all the calculations were done in log-scale.

A dictionary was used to store conditional probabilities with dictionary key in the syntax of key="word|document_no" where
word = any word in the vocabulary and
document_no = index of document in the feature matrix

The following formulae were used to calculate precision, recall, accuracy and f1 score:

Precision = True Positive/(False Positive+True Positive)

Recall = True Positive/(False negative + True Positive)

Accuracy = (True Positive + True Negative)/(True Positive+False Negative+False Positive+True Negative)

The table below gives Precision, Accuracy, Recall and F1 score values for different datasets

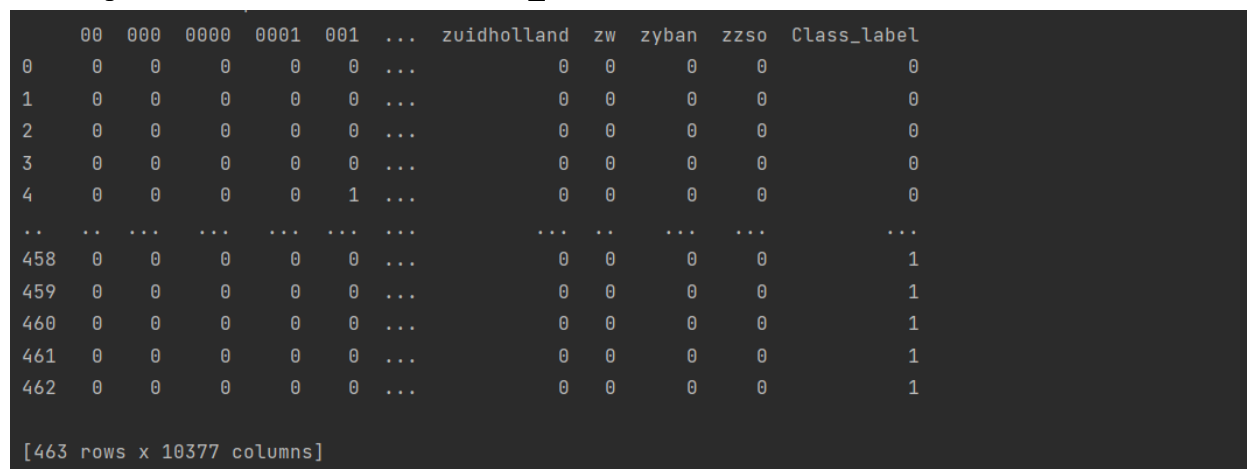
	Precision	Recall	Accuracy	F1 score
Dataset 1	0.9401	0.8461	0.9435	0.8906
Dataset 2 enron1	0.9485	0.8657	0.9407	0.9052
Dataset 3 enron4	0.9527	0.9795	0.9502	0.9659

Discrete Naive Bayes on the Bernoulli model:

Bernoulli model:

To create this representation, I used CountVectorizer(binary=True) from sklearn.feature_extraction.text.CountVectorizer. To create email vectors with 1 signifying occurrence of the term and 0 signifying absence of the term, I used vectorizer.fit_transform from sklearn.feature_extraction.text.CountVectorizer

The snapshot below is the feature x class_label matrix in Bernoulli model



```
      00 000 0000 0001 001  ... zuidholland zw zyban zzso Class_label
0      0  0    0    0    0  0  ...           0  0    0    0           0
1      0  0    0    0    0  0  ...           0  0    0    0           0
2      0  0    0    0    0  0  ...           0  0    0    0           0
3      0  0    0    0    0  0  ...           0  0    0    0           0
4      0  0    0    0    0  1  ...           0  0    0    0           0
..    ..  ...  ...  ...  ...  ...           ... ..  ...  ...           ...
458    0  0    0    0    0  0  ...           0  0    0    0           1
459    0  0    0    0    0  0  ...           0  0    0    0           1
460    0  0    0    0    0  0  ...           0  0    0    0           1
461    0  0    0    0    0  0  ...           0  0    0    0           1
462    0  0    0    0    0  0  ...           0  0    0    0           1

[463 rows x 10377 columns]
```

Discrete Naive Bayes:

The algorithm was coded from scratch. Algorithm reference used:

<http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf>

Add 1-laplace smoothing was also implemented. All calculations were computed in log-scale to avoid underflow.

A dictionary was used to store values of different combinations of conditional probabilities. The table below shows Precision, Recall, Accuracy, F1 score for Discrete Naive Bayes on Bernoulli model:

	Precision	Recall	Accuracy	F1 score
Dataset 1	0.7839	0.9769	0.9205	0.8698
Dataset 2 enron1	0.8674	0.9664	0.9407	0.9142
Dataset 3 enron4	0.9715	0.9590	0.9502	0.9652

Logistic Regression:

It was implemented from scratch. The following steps were followed to implement logistic regression:

1. Reading the training dataset and converting each email into vectors based on bag of words or bernoulli model
2. For each example/email vector, conditional probability was calculated using

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

3. For each example, error was calculated and stored in a list using the following formula

$$(Y^l - \hat{P}(Y^l = 1|X^l, W))$$

4. Partial derivative for each weight was calculated using

$$\sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

5. Bias term and all weights were updated using the formula below

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \eta \lambda w_i$$

$$w_0 \leftarrow w_0 + \eta \sum_{l=1}^d \left(Y^l - \hat{P}(Y^l = 1|X^l, W) \right)$$

6. Step 2 to step 6 were repeated for upper_limit iteration number of times.

The training dataset was divided into a 70/30 split. 70% of the training data was used to train the model. The trained model was validated against 30% of remaining training data. The following precision, recall, accuracy and f1 score was obtained by changing the parameters lambda, learning rate and number of iterations.

	Precision	Recall	Accuracy	F1 score
lambda=1 Learning rate = 0.005	0.9032	0.7	0.8920	0.7887
lambda=1 eta=0.001	0.9117	0.775	0.9136	0.8378
lambda=2 eta=0.005	0.9310	0.675	0.8920	0.7826
lambda=2 eta=0.01	0.9285	0.65	0.8848	0.7647
lambda=2 eta=0.05	0.90625	0.725	0.8992	0.8055
lambda=3 eta=0.05	0.9375	0.75	0.9136	0.8333

Based on the above data, the following values were hard-coded before complete training and testing without dataset split

- Lambda = 3
- Learning rate = 0.05
- Total_iterations = 15

The table below shows Precision, Recall, Accuracy and F1 score with Logistic Regression on bag of words model:

	Precision	Recall	Accuracy	F1 score
Dataset 1	0.8782	0.7769	0.9100	0.8244
Dataset 2	0.9115	0.6912	0.8771	0.7862
Dataset 3	0.8426	1.0	0.8655	0.9146

The table below shows Precision, Recall, Accuracy and F1 score with Logistic Regression on bernoulli model:

	Precision	Recall	Accuracy	F1 score
Dataset 1	0.928	0.8923	0.9518	0.9098
Dataset 2	0.9038	0.9463	0.9495	0.9245
Dataset 3	0.9676	0.9948	0.9723	0.9810

SGD Classifier:

I ran SGD classifier with GridSearchCV using the modules `sklearn.linear_model.SGDClassifier` and `sklearn.model_selection.GridSearchCV`

The following parameters were given in GridSearchCV:

```
params = {'loss': ["log", "hinge", "squared_hinge"], "penalty": ["l2", "l1", "none"], 'max_iter': [20, 40, 60, 80]}
```

The best estimator according to GridSearchCV with hyper-parameters `loss=log`, `max_iter=60`, `penalty="none"` with best score 0.9350

Therefore, the hyper-parameters were set to the above values and the SGD classifier was run on the dataset

The table below shows Precision, Recall, Accuracy and F1 score with Logistic Regression on bag of words model:

	Precision	Recall	Accuracy	F1 score
Dataset 1	0.8560	0.8692	0.9246	0.8625
Dataset 2	0.8823	0.9060	0.9298	0.8940
Dataset 3	0.9494	0.9616	0.9355	0.9555

The table below shows Precision, Recall, Accuracy and F1 score with SGD Classifier on bernoulli model:

	Precision	Recall	Accuracy	F1 score
Dataset 1	0.8705	0.9307	0.9435	0.8996
Dataset 2	0.8679	0.9261	0.9298	0.8961
Dataset 3	0.9580	0.9923	0.9631	0.9748

Question1:

Which data representation and algorithm combination yields the best performance (measured in terms of the accuracy, precision, recall and F1 score) and why?

Answer:

- Bernoulli Representation with Logistic Regression and
- Bernoulli Representation with SGD Classifier

were the combinations that gave the best performance. The performance of Logistic Regression on bernoulli model was slightly lesser than the performance of SGD Classifier on bernoulli model

The above combination is better than other combinations of bag-of-words/bernoulli and multinomial/discrete Naive bayes because logistic regression/SGD has no assumptions whereas Naive bayes makes assumptions on probability

Bernoulli Representation is better than bag of words representation in case of Logistic Regression and SGD Classifier because the values in bernoulli representation are either 1 or 0. On the other hand, bag-of-words representation has frequencies of words which is unnecessary for Logistic regression/SGD Classifier. Furthermore, the frequencies of words affect the weight of corresponding feature which reduces the chance of convergence.

Question 2:

Does Multinomial Naive Bayes perform better (again performance is measured in terms of the accuracy, precision, recall and F1 score) than LR and SGDClassifier on the Bag of words representation? Explain your yes/no answer.

Answer:

Yes, The performance(accuracy, precision, recall and f1 score) of multinomial naive bayes is better than LR and SGDClassifier on the bag of words representation.

Question 3:

Does Discrete Naive Bayes perform better (again performance is measured in terms of the accuracy, precision, recall and F1 score) than LR and SGDClassifier on the Bernoulli representation? Explain your yes/no answer.

Answer:

No, LR and SGDClassifier perform better on the Bernoulli representation than Discrete Naive Bayes.

Question 4:

Does your LR implementation outperform the SGDClassifier (again performance is measured in terms of the accuracy, precision, recall and F1 score) or is the difference in performance minor? Explain your yes/no answer.

Answer:

The difference in performance of LR and SGDClassifier is minor.