ADALET VEYİS TURGUT
CMPE 493 PROJECT 3 REPORT

1)Data preprocessing

I used separate preprocessing for Naive Bayes and KNN algorithms.

1.1) Naive Bayes preprocessing

For each of the articles, I first found whether the article belongs to test or training.I found topics, concatenated title and body, lowercased it, removed stopwords, removed punctuation. I also eliminated numeric strings. Finally, I store these articles in csv files named "NB_training.csv" and "NB_test.csv". Those csv files consist of document id, preprocessed article content separated with semicolon and topics separated with semicolon.

There are 6494 articles in the training set and 2547 articles in the test set. Size of the vocabulary of the training set is 26683.

1.2) KNN preprocessing

This part is almost the same as previous projects. For each of the articles, I first eliminated articles with no topics, then I found topics, concatenated title and body, lowercased it, removed stopwords, removed punctuation. I also eliminated numeric strings. Finally I created two indexes: inverted index and document frequency index with training data, and created a preprocessed test data file with test data. Since an article may contain more than one topic, I represented them as different files, i.e points overlapping in a coordinate system.

There are 6491 articles in the training set. But since I represented those articles as if they were different for each topic they contain, there are 7194 articles in the training set. There are also 2545 articles in the test set. Size of the vocabulary of the training set is 26678. It differs from KNN vocabulary by 5, I don't know the reason why.

2)Data information

Top 10 classes are: *grain, wheat, corn, earn, acq, ship, trade, crude, money-fx and interest*.

The number of documents each topic has in the training set: *{'grain': 434, 'wheat': 212, 'corn': 183, 'earn': 2877, 'acq': 1650, 'ship': 198, 'trade': 369, 'crude': 391, 'money-fx': 539, 'interest': 347}*

The number of documents each topic has in the test set: *{'grain': 149, 'wheat': 71, 'corn': 56, 'earn': 1088, 'acq': 719, 'ship': 89, 'trade': 117, 'crude': 189, 'money-fx': 180, 'interest': 133}*

There are 6494 articles in the training set and 2547 articles in the test set. Size of the vocabulary of the training set is 26683.

There are 622 articles in the training set labeled with more than one topic.

3)Parameter Tuning

3.1)Naive Bayes

Original naive bayes model works with single label output. I modified it so that my model can support multi label output. I calculated each class probability using log normalization. I took max of those as a topic naturally. For the rest of the topics, I calculated their difference between the first topic and the next closest topic. If that ratio is less than a threshold, I included that topic too. To determine that threshold, I ran the algorithm several times. Best result was when the parameter equals 5.

I used the development-validation split approach. First 80 percent of the training data was for development, remaining 20 percent was for validation to configure the parameter.

3.2) KNN

I ran the algorithm for four different k values: 1,3,5,7. Best result was when k equals 7. I looked at the evaluation result when determining the best k: micro-macro averages of precision-recall-f1 scores were higher when k equals 7. You can observe the result by running evaluation.py script in the src folder.

I looked at topic probabilities to determine which topics to include. First I included topics with probabilities higher than 40 percent which corresponds to 2 out of 3, 2 out of 5, 3 out of 7. Topics with probabilities higher than this threshold, will be the majority out of the k automatically. If none of the probabilities were higher than 40 percent, I took topics with probabilities higher than 32 percent which makes 1 out of 3, 2 out 5, 2 out 7. Again, topics with probabilities higher than this threshold, will be the majority out of the k automatically. I decided these 40 percent and 32 percent by running the algorithm several times and evaluating the results.

4)Evaluation results

First of all, you can reach the evaluation results of all of the models (NB, KNN with different k values) by running the evaluation.py script. I will just discuss F1 scores of NB and KNN with K=7 here.

NB F1:

earn	0.9816
acq	0.9682
money-fx	0.8508
crude	0.9096
grain	0.9084
trade	0.8583
interest	0.7934
wheat	0.2824
ship	0.7898
corn	0.1587
MACRO_AVERAGE:	0.7501
MICRO_AVERAGE:	0.9199

KNN K=7 F1:

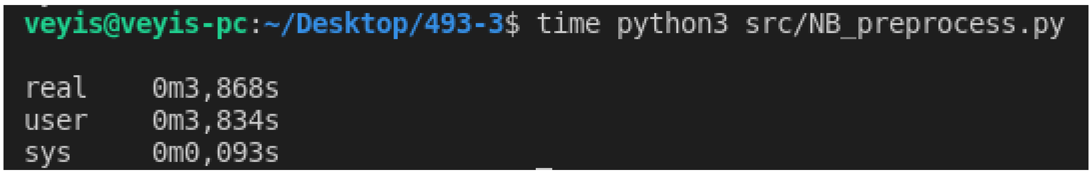
earn	0.9061
acq	0.9417
money-fx	0.8566
crude	0.8911
grain	0.8176
trade	0.8245
interest	0.7752
wheat	0.6103
ship	0.7839
corn	0.5385
MACRO_AVERAGE:	0.7946
MICRO_AVERAGE:	0.8771

We can observe that for topics with less data such as ship and wheat KNN worked better than NB, while topics with more data such as earn and grain NB worked better.
Overall, models have similar results. But considering the runtime, I would go for Naive Bayes.

Result of the randomization test was 1/1001 which makes 0 if we subtract 1 from the numerator and denominator. p value is the smallest it can get. So the null hypothesis is wrong which implies NB and KNN with K=7 are indeed different from each other and the result did not occur by chance.

5)Screenshots

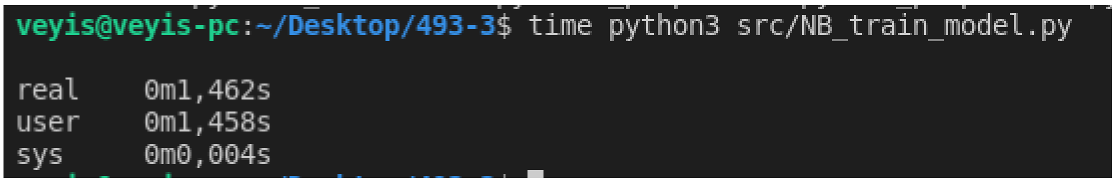
- Preprocessing the articles for Naive Bayes algorithm.



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/NB_preprocess.py

real	0m3,868s
user	0m3,834s
sys	0m0,093s
```

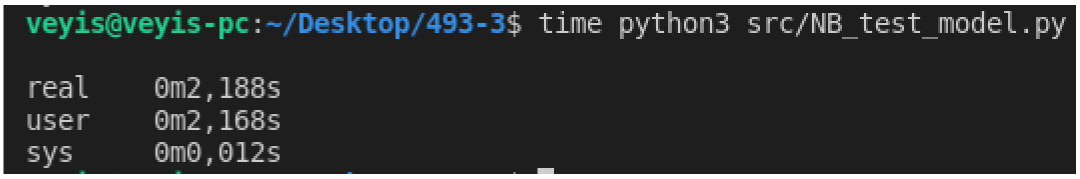
- Training the Naive Bayes algorithm.



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/NB_train_model.py

real	0m1,462s
user	0m1,458s
sys	0m0,004s
```

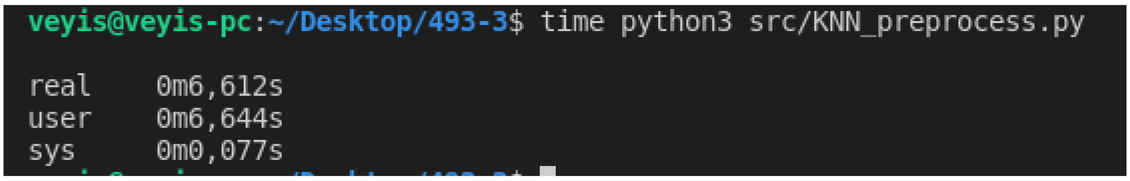
- Running the Naive Bayes algorithm on test data



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/NB_test_model.py

real	0m2,188s
user	0m2,168s
sys	0m0,012s
```

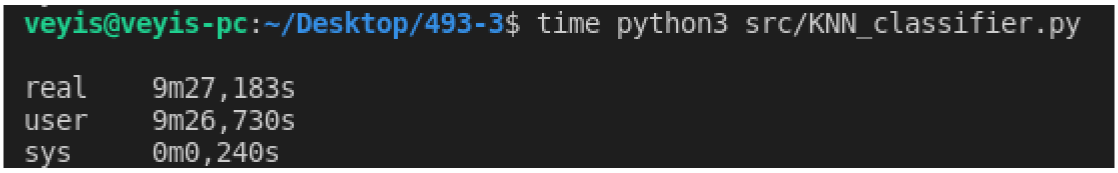
- Preprocessing the articles for KNN algorithm.



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/KNN_preprocess.py

real	0m6,612s
user	0m6,644s
sys	0m0,077s
```

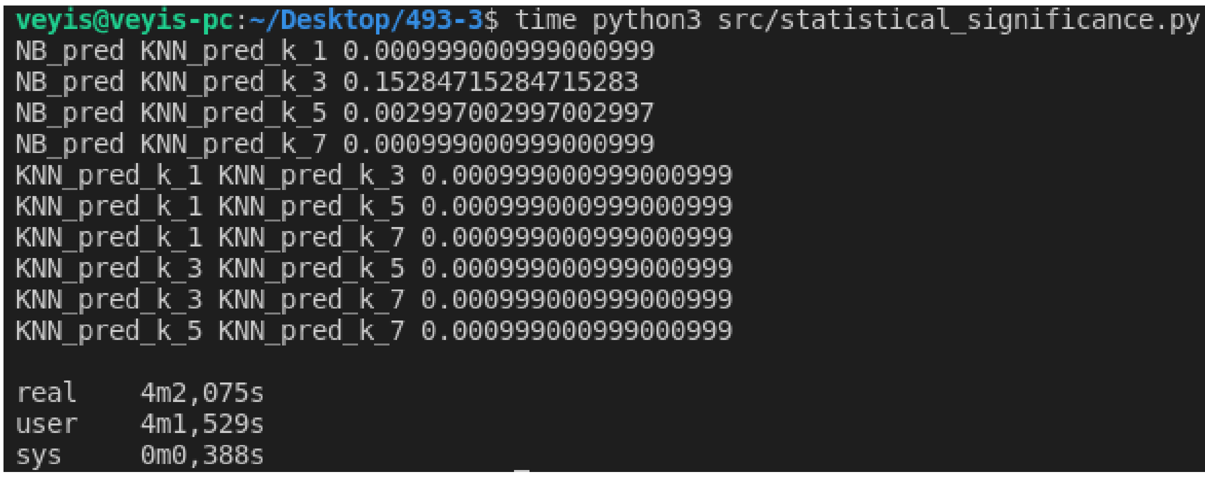
- Running the KNN algorithm on test data



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/KNN_classifier.py

real	9m27,183s
user	9m26,730s
sys	0m0,240s
```

- Running randomization test



```
veyis@veyis-pc:~/Desktop/493-3$ time python3 src/statistical_significance.py
NB_pred KNN_pred_k_1 0.000999000999000999
NB_pred KNN_pred_k_3 0.15284715284715283
NB_pred KNN_pred_k_5 0.002997002997002997
NB_pred KNN_pred_k_7 0.000999000999000999
KNN_pred_k_1 KNN_pred_k_3 0.000999000999000999
KNN_pred_k_1 KNN_pred_k_5 0.000999000999000999
KNN_pred_k_1 KNN_pred_k_7 0.000999000999000999
KNN_pred_k_3 KNN_pred_k_5 0.000999000999000999
KNN_pred_k_3 KNN_pred_k_7 0.000999000999000999
KNN_pred_k_5 KNN_pred_k_7 0.000999000999000999

real	4m2,075s
user	4m1,529s
sys	0m0,388s
```