Assignment 5: Classification

Q1 Classification

This assignment needs train.csv and test.csv. train.csv is for training and test.csv is for test. Both of them have samples in the following format:

label	text
2	I must admit that I'm addicted to "Version 2.0...
1	I think it's such a shame that an enormous tal...
2	The Sunsout No Room at The Inn Puzzle has oddl...
...	...

Write a function **classify** to conduct a classification experiement as follows:

1. Take the training and testing file names (strings) as inputs, e.g. classify(training_file, testing_file).
2. Classify text samples in the training file using **Multinomial Naive Bayes model** as follows:
    - a. First apply grid search with **5-fold cross validation** to find the best values for parameters **min_df**, **stop_words**, and **alpha** of Naive Bayes model that are used the modeling pipeline. Use f1-macro as the scoring metric to select the best parameter values. Potential values for these parameters are:
        - min_df' : [1,2,3]
        - stop_words' : [None,"english"]
        - alpha: [0.5,1,2]
    - b. Using the best parameter values, train a Multinomial Naive Bayes classifier with all samples in the training file
3. Test the classifier created in Step 2.b using the test file. Report the testing performance as:
    - Precision, recall, and f1-score of each label
    - Treat label 2 as the positive class, plot precision-recall curve and ROC curve, and calculate AUC.
4. Your function "classify" has no return. However, when this function is called, the best parameter values from grid search is printed and the testing performance from Step 3 is printed.

Q2. How many samples are enough? Show the impact of sample size on classifier performance

This question will use train_large.csv dataset.

Write a function "impact_of_sample_size" as follows:

- Take the full file name path string for a dataset inputs, e.g. impact_of_sample_size(dataset_file).
- Starting with 800 samples from the dataset, **in each round you build a classifier with 400 more samples**. i.e. in round 1, you use samples from 0:800, and in round 2, you use samples from 0:1200, …, until you use all samples.
- In each round, do the following:
  1. create tf-idf matrix using TfidfVectorizer with **stop words removed**
  2. train a classifier using **multinomial Naive Bayes** model with 5-fold cross validation
  3. train a classifier using **linear support vector machine** model with 5-fold cross validation
  4. for each classifier, collect the following average metrics across 5 folds:
     - average F1 macro
     - average AUC: treat label 2 as the positive class, and set "roc_auc" along with "f1_macro" as metrics
- Plot a line chart (two lines, one for each classifier) show **the relationship between sample size and F1-score**. Similarly, plot another line chart to show **the relationship between sample size and AUC**
- Write your analysis in a **separate pdf file (not in code)** on the following: (1 point)
  - How does the sample size affect each classifier's performance?
  - How many samples do you think would be needed for each model for good performance?
  - How is performance of SVM classifier compared with Naïve Bayes classifier, as the sample size increases?
- There is no return for this function, but the charts should be plotted.

Q3 (Bonus): Predict duplicate questions by classification

You have tired to predict duplicate questions using the dataset 'quora_duplicate_question_500.csv' by similarity. This time, try to use a classification model to predict if a question pair (q_1, q_2) are indeed duplicate.

q1	**q2**	**is_duplicate**
How do you take a screenshot on a Mac laptop?	How do I take a screenshot on my MacBook Pro? ...	1
Is the US election rigged?	Was the US election rigged?	1
How scary is it to drive on the road to Hana g...	Do I need a four-wheel-drive car to drive all ...	0
...	...	...

In your Assignment 4, with cosine similarity, the AUC is about 74%. In this assignment, define a function **classify_duplicate** to achieve the following:

- Take the full name of the dataset file (i.e.'quora_duplicate_question_500.csv') as the input
- **do feature engineering to extract a number of good features**. A few possible options for feature engineering can be:
  - Unigram, bigram, trigram etc.
  - Keep or remove stop words
  - Different metrics, e.g. cosine similarity, BM25 score (https://en.wikipedia.org/wiki/Okapi_BM25 (https://en.wikipedia.org/wiki/Okapi_BM25)), etc.
- build a **classification model (e.g. SVM) using these features to predict if a pair questions are duplicate or not**.
- Your target is to **improve the average AUC of the positive class through 5-fold cross validation by at least 1%, reaching 75% or higher**.
- return the average AUC

In [231]:
```
import pandas as pd
import nltk
....
```

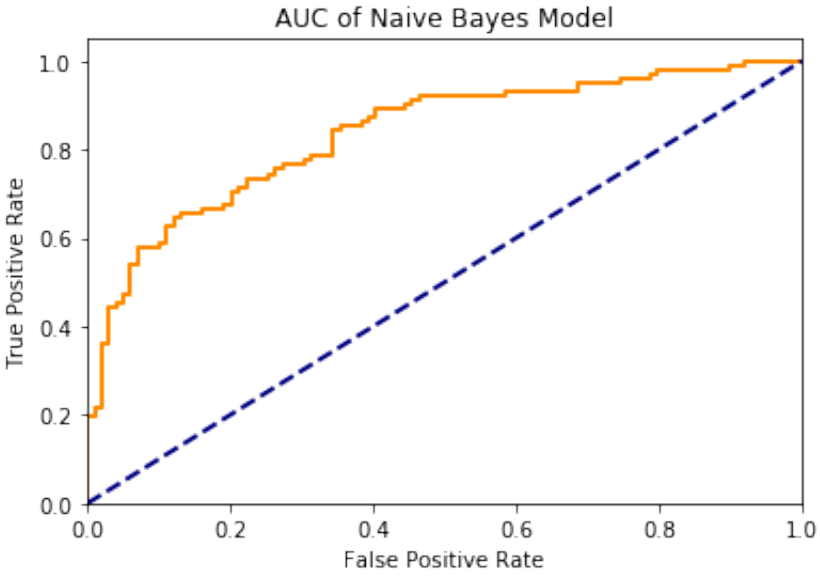
In [238]:
```
# Q1

def classify(train_file, test_file):

    # ADD YOUR CODE HERE
```

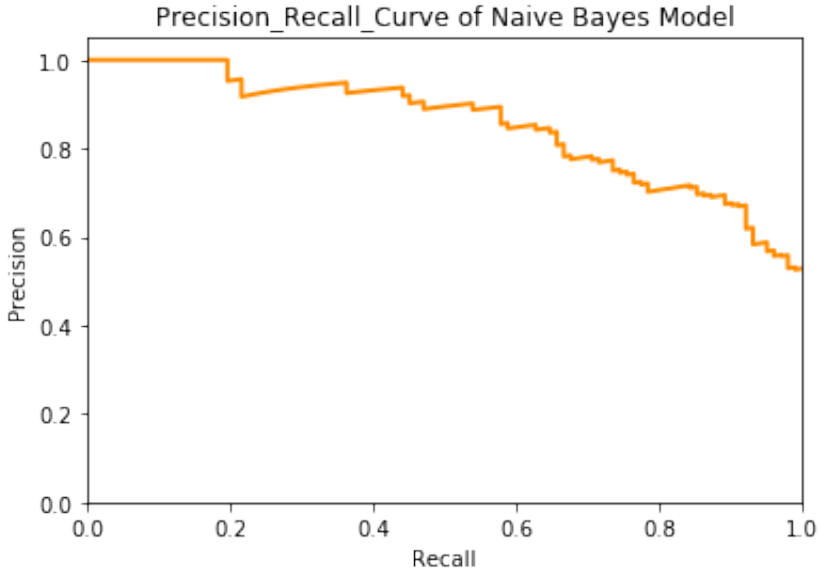
In [239]:
```
# Q2

def impact_of_sample_size(train_file):

    # Add YOUR CODE HERE
```

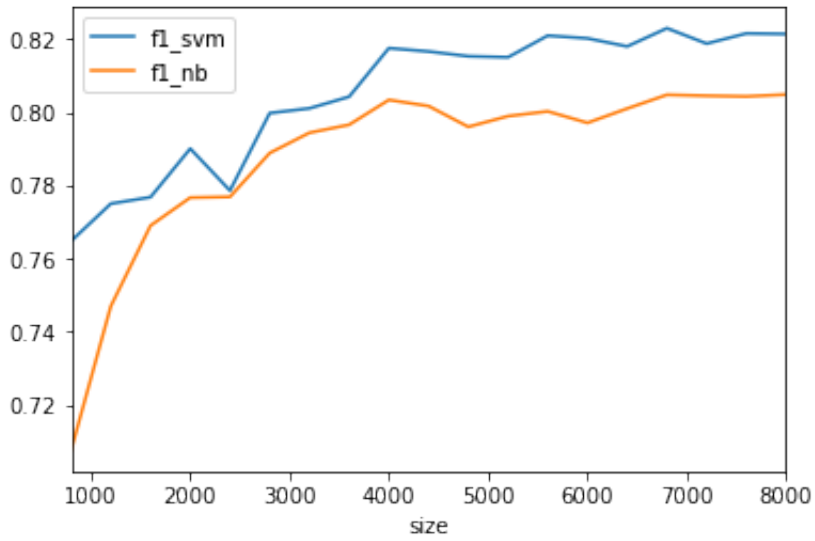
In [240]:
```
#Q3
def classify_duplicate(filename):

    auc = None

    # ADD YOUR CODE HERE

    return auc
```

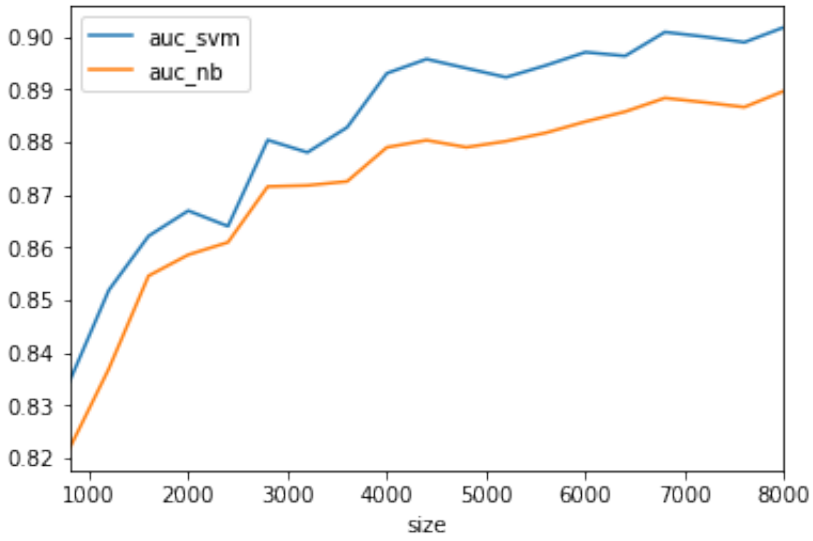
In [242]:
```
if __name__ == "__main__":
    # Question 1
    # Test Q1
    classify("../../dataset/amazon_review_500.csv",\
         "../../dataset/sent_test.csv")

    # Test Q2
    impact_of_sample_size("../../dataset/sent_train_large.csv")

    # Test Q3
    result = classify_duplicate("../../dataset/quora_duplicate_questio
n_500.csv")
    print("Q3: ", result)
```

```
clf__alpha: 2
tfidf__min_df: 1
tfidf__stop_words: None
best f1_macro: 0.7134380001639543
              precision    recall  f1-score   support

           1       0.74      0.76      0.75        99
           2       0.76      0.74      0.75       102

   micro avg       0.75      0.75      0.75       201
   macro avg       0.75      0.75      0.75       201
weighted avg       0.75      0.75      0.75       201

0.835016835016835
```


AUC of Naive Bayes Model
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Precision_Recall_Curve of Naive Bayes Model
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


f1_svm
f1_nb
0.72
0.74
0.76
0.78
0.80
0.82
1000
2000
3000
4000
5000
6000
7000
8000
size




```
Q3:  0.760092681967682
```

In [ ]: