Assignment 2 Report
Due on November 02, 2017 (23:59:59)

Student Name: Berk Gülay

Student ID: 21527013

Algorithm of Naive Bayes and BoW representations:

****** If bi-gram is 'True' than words will be modelled/used as pairs with their adjacents, else they will be modelled/used as unique words without any change.***

- "BAG" class implements BoW model using CountVectorizer and dictionaries of Python.
  a) When it's constructor was initialized, attributes which is dict keeping word numbers and int keeping total word num in bag will be initialized as well.
  b) Word num in dict will be calculated for each word addition by add_tweet() method and by using this method each given tweet will be added to the dict with their frequencies.

- "Naive Bayes" class implements standart Naive Bayes algorithm with log probabilities usage.
  a) It has 6 type of attributes. 1- list to keep training tweets which is added , 2- flag to keep if bi-gram or uni-gram model will be used , 3- int var. to keep total vocabulary num , 4- BAG models for each class to keep word freq. of that class , 5- int var. to keep number of tweets for each class , 6- dict to keep calculated probabilities for specific class and word(to prevent re-calculation)
  b) Class has all essential method implementations like train, test or find_prob(to calculate likelihood)
  c) Train() method adds given tweets as train data into specific BAG model and also arranges vocabulary&class tweet numbers.
  d) Test() method sorts possibilities and predicts label(class) of given tweet as test data. It uses log possibilities while calculations.
  e) Find_prob() method calculates likelihood of a tweet for a specific class which is given as parameter as well.

- "Extended Naive Bayes" class just uses Naive Bayes but manages bi-gram and uni-gram usages and their combinations. It's only implementation is when bi-gram and uni-gram models used together. Else it does not implement anything just uses Naive Bayes class above and it's methods as they are. If uni-gram and bi-gram models will be used together so class calculates each model's possibility separatedly first and then re-calculates these possibilities together also using log probabilities feature again. Returns latest probabilities and predictions according to uni-gram/bi-gram attribute's value coming from user.

Tried features for test classification:

1) Uni-Gram **(~ 0.57 accuracy**)
2) Bi-Gram **(~ 0.46 accuracy**)
3) Uni-Gram & Bi-Gram together **(~ 0.54 accuracy**)

Optimum feature for test classification with our data:

- Uni-Gram **(latest test classification accuracy was %57.2, highest**)
- Keeping vocabulary(words) as they are gives better results. (Eliminating unnecessary words from vocabulary gives worser results.) **(approximately %2 accuracy lost**)

Best ML model for this problem and data-set:

- Because of insufficient data for bi-gram, it's accuracy found low by comparison uni-gram. Uni-gram and bi-gram composition may be useful when data is sufficient but in our case and with our current data only uni-gram usage gives best result with accuracy **%57.2 (highest)**.
- Also deleting some unnecessary English words like 'up', 'of', 'yet' gave worser results. (Some taken results as image are given below in "Observations" section)
- **Latest test classification model for our data (in our case) is only uni-gram usage without deleting standart english words (like 'up' , 'more', 'of', 'yet').**

Observations – Fitting the ML model:

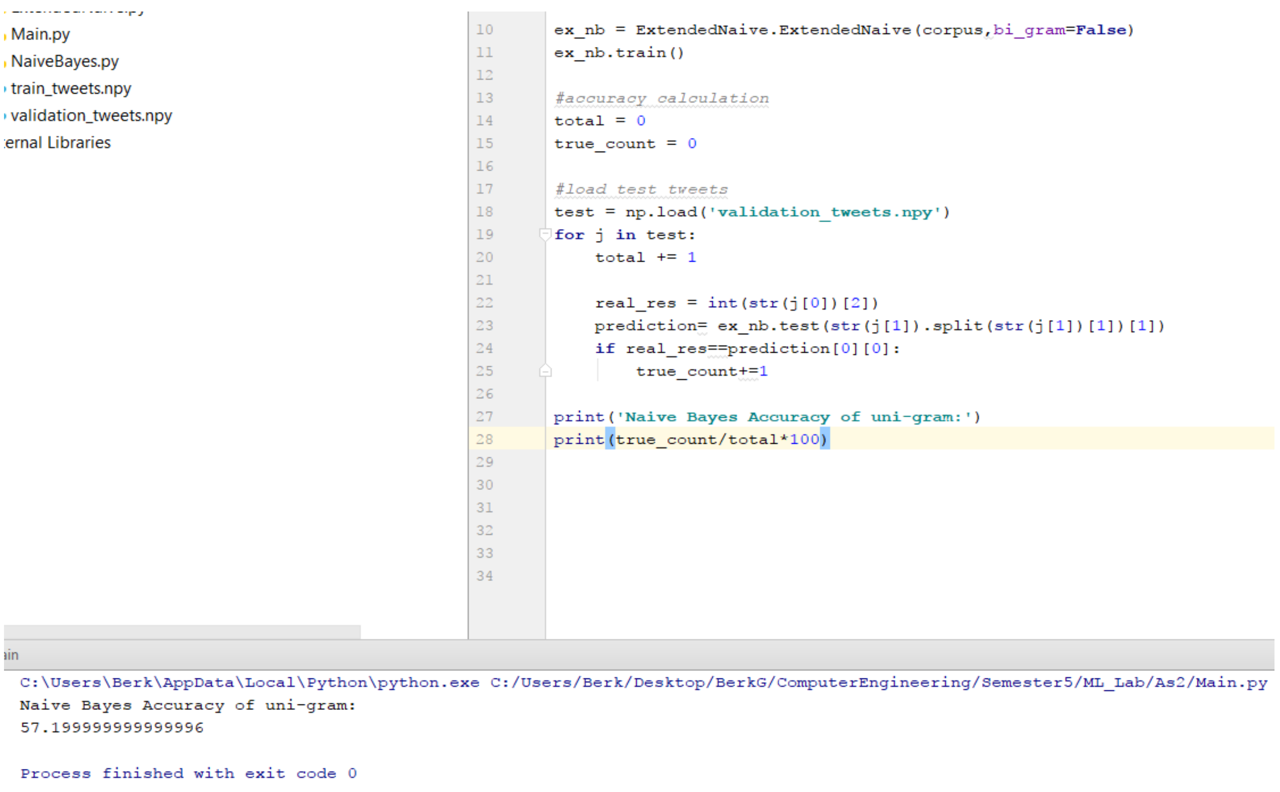
All of the cases give accuracy between 0.46 and 0.58 according to chosen features and combinations. Most accurate case is written above in "Best ML model" part with it's explanation.

Below I will show results for different cases(which I mentioned above) and different models with their accuracy scores.

Output type in console is;

```
Description:

Accuracy Score (%)
```

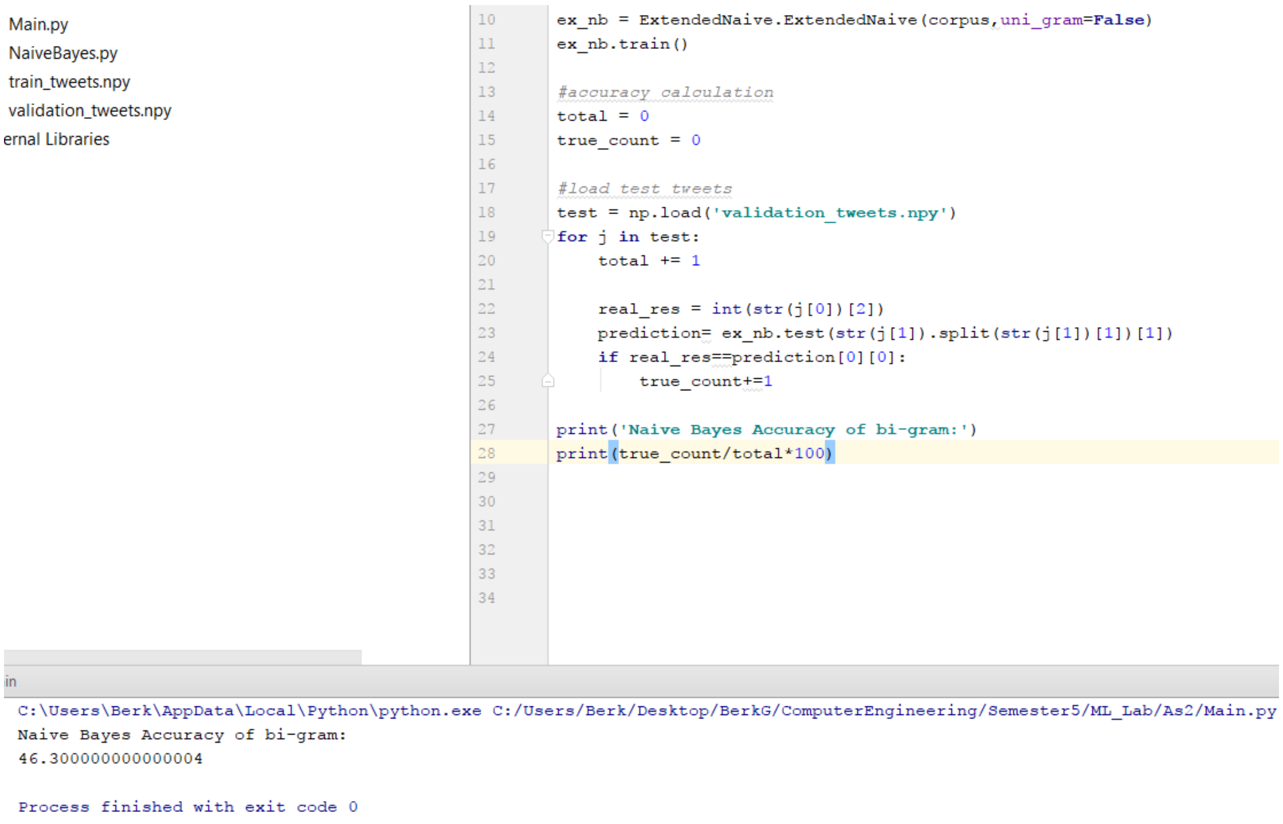
(Naive Bayes Accuracy of Uni-Gram by itself, words like 'of' , 'yet' are not eliminated)



```
ex_nb = ExtendedNaive.ExtendedNaive(corpus,bi_gram=False)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes Accuracy of uni-gram:')
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes Accuracy of uni-gram:
57.199999999999996

Process finished with exit code 0
```

(Naive Bayes Accuracy of Bi-Gram by itself, words like 'of' , 'yet' are not eliminated)

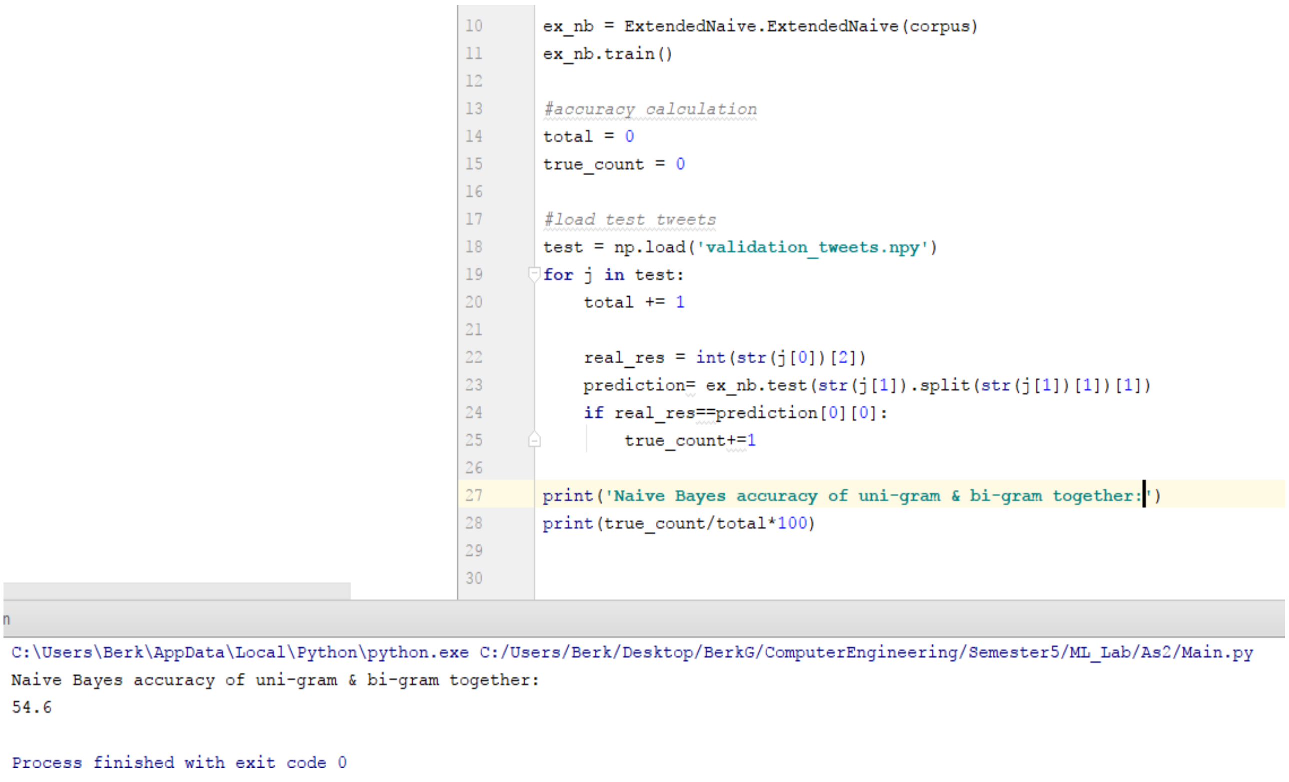


```
ex_nb = ExtendedNaive.ExtendedNaive(corpus,uni_gram=False)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes Accuracy of bi-gram:')
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes Accuracy of bi-gram:
46.300000000000004

Process finished with exit code 0
```

(Naive Bayes Accuracy of Uni-Gram and Bi-Gram together,

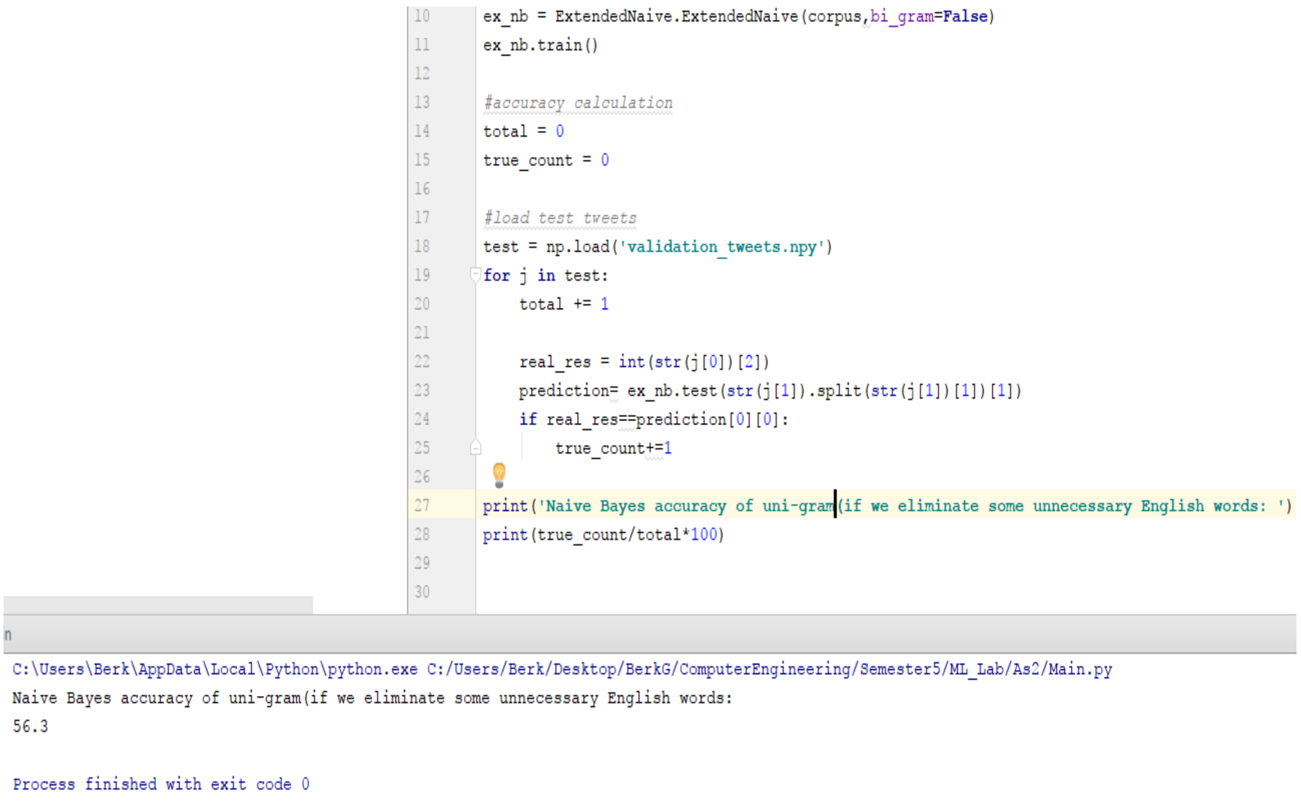
words like ‘of’ , ‘yet’ are not eliminated)



```python
ex_nb = ExtendedNaive.ExtendedNaive(corpus)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes accuracy of uni-gram & bi-gram together:')
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes accuracy of uni-gram & bi-gram together:
54.6

Process finished with exit code 0
```

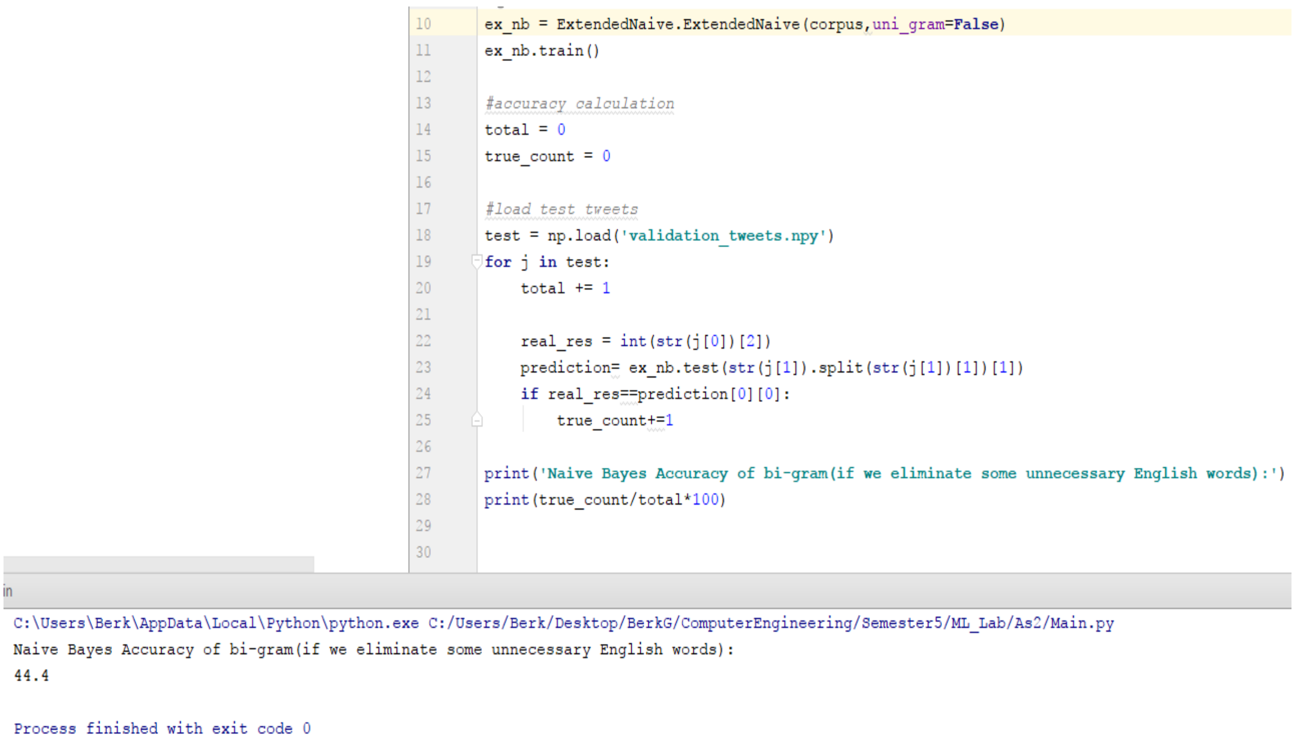
(Naive Bayes Accuracy of Uni-Gram itself, words like ‘of’ , ‘yet’ were <u>eliminated</u>)



```python
ex_nb = ExtendedNaive.ExtendedNaive(corpus,bi_gram=False)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes accuracy of uni-gram(if we eliminate some unnecessary English words: ')
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes accuracy of uni-gram(if we eliminate some unnecessary English words:
56.3

Process finished with exit code 0
```

(Naive Bayes Accuracy of Bi-Gram itself, words like ‘of’ , ‘yet’ were <u>eliminated</u>)

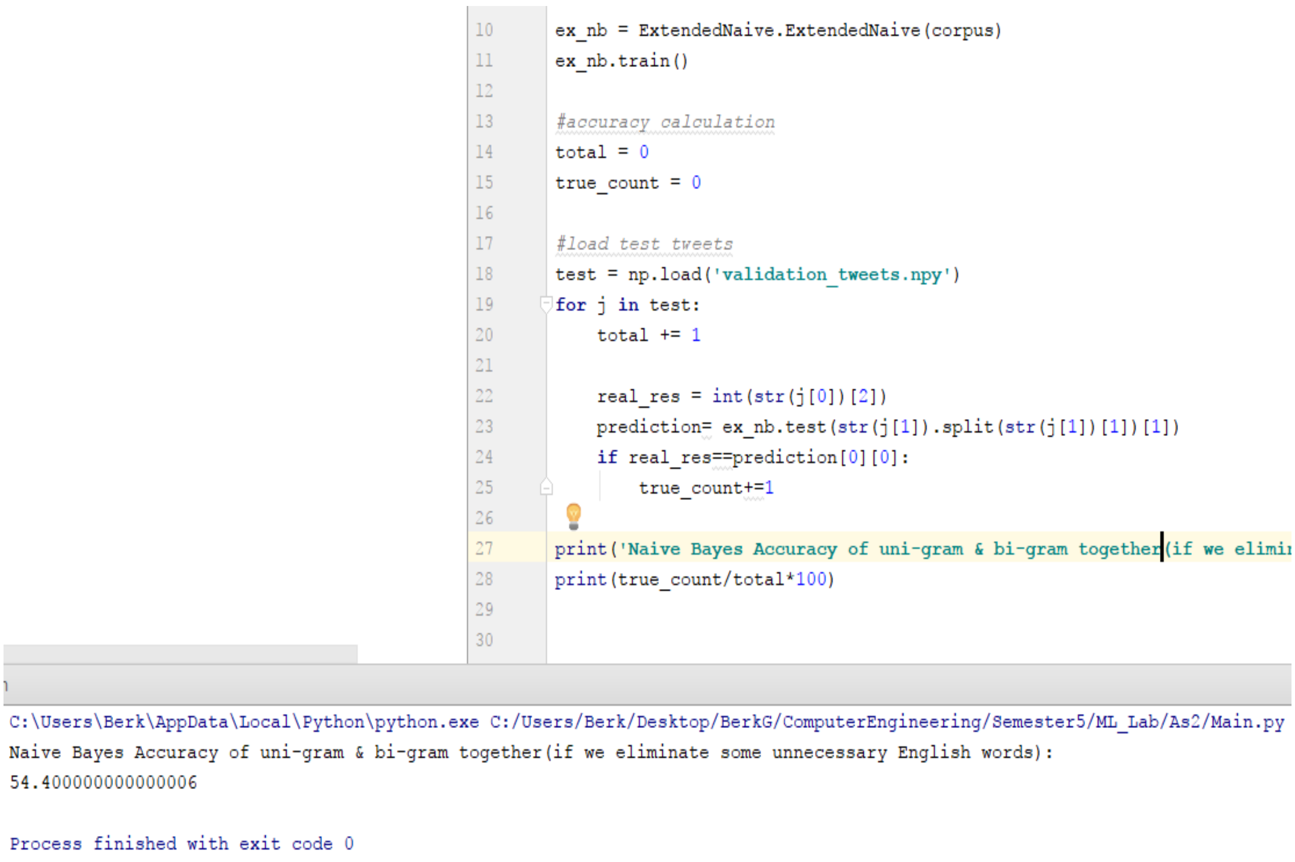


```python
ex_nb = ExtendedNaive.ExtendedNaive(corpus,uni_gram=False)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes Accuracy of bi-gram(if we eliminate some unnecessary English words):')
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes Accuracy of bi-gram(if we eliminate some unnecessary English words):
44.4

Process finished with exit code 0
```

(Naive Bayes Accuracy of Uni-Gram and Bi-Gram together,

words like ‘of’ , ‘yet’ were <u>eliminated</u>)



```python
ex_nb = ExtendedNaive.ExtendedNaive(corpus)
ex_nb.train()

#accuracy calculation
total = 0
true_count = 0

#load test tweets
test = np.load('validation_tweets.npy')
for j in test:
    total += 1

    real_res = int(str(j[0])[2])
    prediction= ex_nb.test(str(j[1]).split(str(j[1])[1])[1])
    if real_res==prediction[0][0]:
        true_count+=1

print('Naive Bayes Accuracy of uni-gram & bi-gram together(if we elimi
print(true_count/total*100)
```

```
C:\Users\Berk\AppData\Local\Python\python.exe C:/Users/Berk/Desktop/BerkG/ComputerEngineering/Semester5/ML_Lab/As2/Main.py
Naive Bayes Accuracy of uni-gram & bi-gram together(if we eliminate some unnecessary English words):
54.400000000000006

Process finished with exit code 0
```

Used libraries:

1- Numpy
2- Sklearn – CountVectorizer
3- math

Resources:

1- http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2- http://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
3- https://docs.python.org/2/library/math.html