Que1:-
(a)

Naive Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

We have two classes, where **Negative (-ve) = 0** and **Positive (+ve) = 4**.

Here, we need to find out which class has a bigger probability for the new tweet. i.e., we need to find which of the below is bigger.

P(Positive/New tweet) = P(New Tweet/Positive)*P(Positive)
(or)
P(Negative/New tweet) = P(New Tweet/Negative)*P(Negative)

As there are negligible chances that a whole tweet will be repeated hence we calculate probability on words rather than the whole tweet.
Following points have been considered during the procedure:

1. Log has been taken to avoid underflow which might be caused due to the multiplication of very small - small probabilities values.
2. Laplace Smoothing is done to avoid 0 - probability problem
3. Any kind of processing is not done on the data.
4. Split of a tweet is done on space, comma, semi-colon, and full stop

Following are the accuracy results:

1. Accuracy on **Test Data** : 0.8161559888579387
2. Accuracy on **Training Data** : 0.862645625

Que1:-
(b)

1. The accuracy with random value of 0 and 4, the accuracy varies but it has been observed that it remains between 50 and 55 or near by these values. Following is the data for one execution:
    a. Number of examples for testing : 359
    Number of value match : 191
    Accuracy on random data : 0.532033426183844

2. As number of positive and negative classes are equal in the training data, hence I have calculated the accuracy for 0 and 4 both as shown the below:
   a. When **0(-ve)** is considered as majority class:
      Number examples for testing : 359
      Number of value match : 177
      Accuracy on 0 as majority value : 0.49303621169916434

   b. When **4(+ve)** is considered as majority class:
      Number examples for testing : 359
      Number of value match : 182
      Accuracy on 4 as majority value : 0.5069637883008357

3. **The improvement (on test data) of the algorithm over the random/majority baseline:**
   a. Accuracy of the algorithm is **1.534031414** time of the accuracy with **random prediction**, please note that it may vary as the accuracy may change in different rounds based on the random generated values.
   b. Accuracy of the algorithm is **1.655367232** time of the accuracy with **majority prediction** with 0 as majority value.
   c. Accuracy of the algorithm is **1.60989011** time of the accuracy with **majority prediction** with 4 as majority value.

Que1:-
(c)
Following is the confusion matrix on the test data.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	147 (TN)	30 (FP)
	Positive	36 (FN)	146 (TP)

Accuracy = (TP + TN) / (TP + TN + FP + FN)= (146+147) /(146+147+30+36)= 0.816155989
Recall: Recall gives us an idea about when it's actually yes, how often does it predict yes.
Recall=TP / (TP + FN)=146/(146+36)=0.802197802
Precision: Precision tells us about when it predicts yes, how often is it correct.
Precision = TP / (TP + FP)=146/ (146+30)=0.829545455

   a. True Positive has the highest value among all.

b. A **true positive** is an outcome where the model correctly predicts the **positive** class.
c. A **false positive** is an outcome where the model incorrectly predicts the **positive** class.
d. A **true negative** is an outcome where the model correctly predicts the **negative** class.
e. A **false negative** is an outcome where the model incorrectly predicts the **negative** class.

The accuracy of our model is 0.816155989 (on test data), It means, these many times the model has predicted correctly.
It can be noticed that, there are **177 negative** values in the data set and 146 values are predicted correctly, while there are **182 positive** values and 147 are predicted correctly.

Que1:-
(d)

1. Without doing stemming but after removal of stop words and punctuations:
    a. Accuracy on **test data**: 0.8189415041782729
    b. Accuracy on Training Data : 0.792615625

Note that, accuracy on test data remains same while accuracy on training data has been **reduced**.

2. If we do stemming using the **SnowballStemmer** then followings are the accuracy:
    a. Accuracy on Test Data : 0.7688022284122563
    b. Accuracy on Training Data : 0.783856875
3. If we do stemming using the **WordNetLemmatizer** then followings are the accuracy:
    a. Accuracy on Test Data : 0.7855153203342619
    b. Accuracy on Training Data : 0.790635625

Note:- Lemmatizer performs better than stemming and both perform a little less than without performing any pre-processing of data.

Que1:-
(e)
Features selections are done over the model built in the d part.I have considered the following 4 models.
Features selection

a. Unigram + Bigram + Lemmatization
    1. Accuracy on Test Data : 0.8022284122562674
    2. Accuracy on Training Data : 0.89836

    Note:- We can note here that accuracy on both **test data**(by 2 more percent) and **training data** (by 10 more percent) have been improved. Improvement in the accuracy of training data is as huge as 10% than part d with preprocessing of data.

b. Unigram + Bigram + Stemming
    1. Accuracy on Test Data : 0.7827298050139275
    2. Accuracy on Training Data : 0.88901375

    Note:- The accuracy is better than the part d without any preprocessing of data but it's a little

less than then the above model with lemmatization.

c. Unigram + Trigram + Stemming

1. Accuracy on Test Data : 0.7632311977715878
2. Accuracy on Training Data : 0.94555125

Note:- The accuracy on test data has been decreased but it has been increased hugely on training data to **94.56%**.

d. Unigram + Trigram + Lemmatization

1. Accuracy on Test Data : 0.8189415041782729
2. Accuracy on Training Data : 0.94787

Note:- Performance has slightly been improved.

Que1:-
(f)

1. Model trained on the **row data without any pre-processing**
   Without features selection (**training data**)
   a. Time to train the model without features selection on training data : approx 3 hours
   b. **Without features selection** accuracy on test data: 0.5069637
   c. **With features selection** accuracy on test data : 0.50417827

2. Model trained on the **processed data(SW removal + lemmatization)**
   a. Time to train the model without features selection on training data : approx 2 hours
   b. **Without features selection** accuracy on test data: 0.48746518105849584
   c. **With features selection** accuracy on test data : 0.5013927576601671

Que1:-
(g)
ROC Curve:-
An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

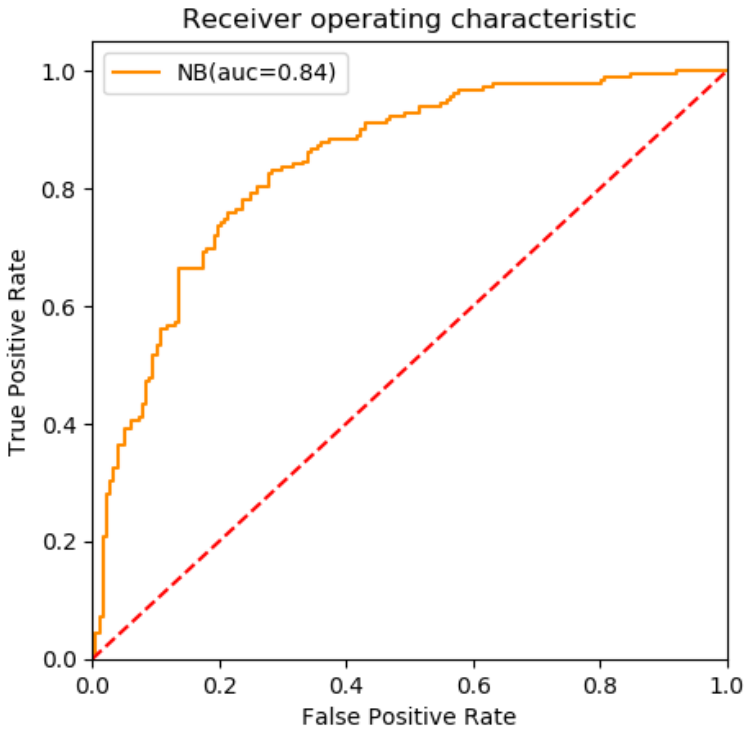
- True Positive Rate
- False Positive Rate

True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

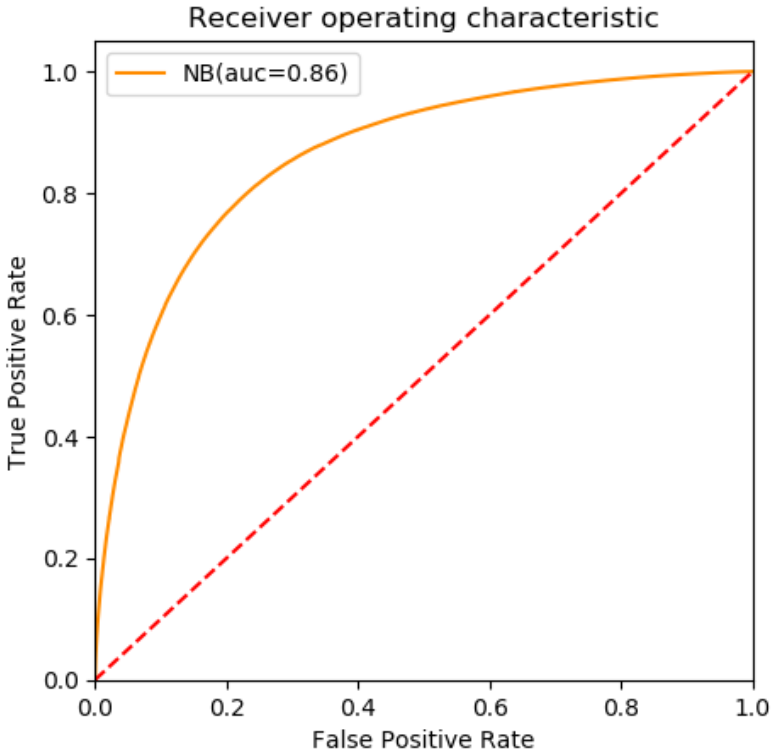
$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (**FPR**) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$



ROC/AUC calculated for test data



ROC/AUC calculated for Training Data

----------Part B ----------

Que 1(a)

SVM dual problem with linear kernel:

a. Number of Support vectors : 120
b. alpha values are considered 0 where alpha is < 1e-5
c. Training time in seconds = 17.92 seconds
d. Mean of W = 0.019834981507734047
e. Value of b : -0.9348673
f. Accuracy on test data : 99.40%
g. Accuracy on validation data : 99.20 %
h. Accuracy on training data : 100.0 %

Que1(b)

SVM dual problem with Gaussian kernel:

a. Number of Support vectors : No. of Support vectors : 1063
b. alpha values are considered 0 where alpha is < 1e-5
c. Training time in seconds = 51.26 seconds
d. Value of b : 0.7050011089297443
e. Accuracy on test data : 99.5 %
f. Accuracy on validation data : 98.6 %
g. Accuracy on training data : 100.0 %

We generally get better accuracies with Gaussian kernel than linear kernel as it maps data in higher dimensions. Usually linear kernels are less time consuming and provide less accuracy than the Gaussian kernels.
Here we can notice that accuracies are almost the same as there are large enough features in the data set so not much improvement if mapping done in higher dimensions. Moreover, SVM with a linear kernel is already performing very well which is nearly 100%.

Que 1 (C) -

a. **SVM Linear Model :**
    a. Mean of W of the linear SVM Model **:** 0.019834433262008663
    b. b value of the linear SVM Model : -0.9348381492157227
    c. No. of support vectors of the linear SVM model : 120
    d. Accuracy of Test data on the linear SVM model : 99.4 %
    e. Accuracy of Validation data on the linear SVM model : 99.2 %
    f. Time to train the linear SVM model : 503 ms ± 23.2 ms

b. SVM Gaussian Model:
    a. b value of the gaussian SVM Model : 0.7056494588279296
    b. No. of support vectors of the gaussian SVM Model : 338
    c. Accuracy of Test data on the gaussian SVM Model : 99.5 %
    d. Accuracy of Validation data on the gaussian SVM Model : 98.6
    e. Time to train the gaussian SVM model : 3.92 s ± 8.3 ms

Time to train a model in part a and part b is way more than the time taken in part c.

Que2(Multi-class Classification)

Que2(a):

a. Accuracy on test data : 0.8608
b. Accuracy on validation data : 0.8592

Que2(b): Learning with Gaussian Kernel using LIBSVM

a. Training time in seconds = 209.3273320198059
b. Accuracy on test data : 0.8808
c. Accuracy on validation data : 0.8792

Que2(c)

```
Confusion matrix for test data(part a

[[212   1   0  21   0   1  15   0   0   0]
 [  0 232   4  12   0   0   2   0   0   0]
 [  3   0 182   2  38   0  22   0   3   0]
 [ 15   1   0 217   7   0   8   0   2   0]
 [  0   1  28  13 182   0  25   0   1   0]
 [  0   0   0   1   0 225   0  14   1   9]
 [ 59   0  33   9  21   0 127   0   1   0]
 [  0   0   0   0   0  16   0 218   1  15]
 [  0   1   1   1   2   1   1   2 241   0]
 [  0   0   0   0   0   5   0  12   0 233]]
```

Non-diagonal values are very less in comparison to dogonal values which shows that accuracy is very good. 8th class has the highest diagonal value 241, it performed better than others.

```
Confusion matrix for test data


[[433   0   5  11   3   0  38   0  10   0]
 [  1 482   4   9   0   0   4   0   0   0]
 [  5   0 411   7  37   0  32   0   8   0]
 [ 12   0   3 457   9   0  14   0   5   0]
 [  3   1  41  13 399   0  38   0   5   0]
 [  0   0   0   0   0 473   0  16   5   6]
 [ 80   0  55   9  34   0 315   0   7   0]
 [  0   0   0   0   0  14   0 471   1  14]
 [  1   0   1   1   2   2   2   2 489   0]
 [  0   0   0   0   0  11   0  14   1 474]]
```

Confusion matrix for test data(part b)

Non-diagonal values are very less in comparison to dogonal values which shows that accuracy is very good. 8th class has the highest diagonal value 489, it performed better than others.