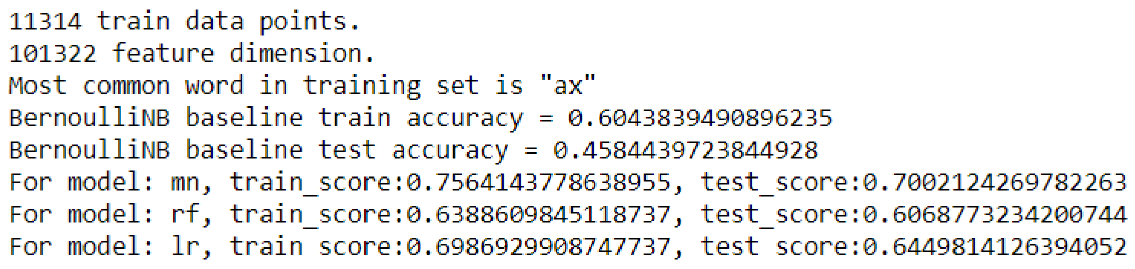
Question 1.

Note: The runtime of the code is appox. 45mins, due to gridsearchCV.

Screenshot of code output: Part 1



```
11314 train data points.
101322 feature dimension.
Most common word in training set is "ax"
BernoulliNB baseline train accuracy = 0.6043839490896235
BernoulliNB baseline test accuracy = 0.4584439723844928
For model: mn, train_score:0.7564143778638955, test_score:0.7002124269782263
For model: rf, train_score:0.6388609845118737, test_score:0.6068773234200744
For model: lr, train_score:0.6986929908747737, test_score:0.6449814126394052
```

1.0 Description of the model selection criteria

The zero-one loss is used here as the loss function to be compared through all 4 models. After normalization, the zero-one loss is equivalent to (1-accuracy). Thus, during the model selection, to minimize the zero-one loss is equivalent to maximizing the accuracy.

I used a nested cross validation approach to tune the hyperparameter and then report the cross validation scores on the train set.

Hyperparameters of the models were tuned with sklearn.GridSearchCV, where each combination of GridSearch parameters were evaluated for 3 iterates of inner cross validation. The parameters with the highest average inner CV scores were used as the optimal hyperparameter.

Then in the outer loop, the optimal hyperparameters were adopted in the model, and then 3 outer cross validation were executed. The accuracy score of each outer CV is reported. Then the mean of outer CV iterations was taken as the final score for the train set. And the train loss is calculated from *loss = 1 – accuracy score.*

1.1 Description of chosen algorithms:

Multiple algorithms are tested, such as decision tree, random forest, SVM with linear kernel, multinomialNB and logistic regression.

Based on the testing result, three algorithms are chosen:

Multinomial Naïve Bayes, Logistic Regression and Random Forest.

1.2 Evaluation of train and test loss:

With zero-one loss metric used, the normalized loss can be calculated from:

Loss = 1-accuracy

	Baseline	**Random Forest**	**Logistic Regression**	**Multinomial Naïve Bayes**
Train Accuracy	0.604	0.639	0.699	0.756
Train Loss	0.396	0.361	0.301	0.244
Test Accuracy	0.458	0.607	0.645	0.700
Test Loss	0.542	0.393	0.355	0.3

1.3 Hyperparameter selection

The train set is divided in to 3 folds using sklearn.StraitifiedKFold. So that to make sure each class is evenly distributed in the train and test set. Then I used GridSearchCV approach to select the optimal hyperparameters. Firstly, the potential hyperparameters for each model was listed in a dictionary:

Logistic Regression: param_dist = {'C': [0.001, 0.01, 0.1, 1]}

MultinomialNB: param_dist = {'alpha':[0.001, 0.01, 0.1, 1]}

RandomForest: param_dist = {'max_depth':[10,30,50]}

** Note: For Random Forest, n_estimators=200, max_features ='sqrt'. Due to the lack of computational resource, only max_depth were tuned.*

Then, each combination of potential hyperparmeters were fit to the train folds, and then validated on the cross-validation fold. In the end, the hyperparameter with highest CV score were chosen as the optimal hyperparameter.

After choosing the hyperparameter in the inner loop of nested CV, train score is then calculated again in the outer loop CV.

1.4 Why picking 3 methods:

Logistic Regression:

- Logistic regression can give a probabilistic view of class prediction

- It's easy to train, and has convex loss function

- It's robust and less sensitive to incorrect modeling assumptions

- It works better with non-Gaussian data

Multinomial Naïve Bayes:

- Fast to train

- Works well with classification with discrete features, suitable for the fractional counts such as tf-idf in this case.

- Compared with the baseline, NultinomialNB takes into consideration not only the Boolean appearance of the term, but also how many time the term occurs. The richer information it collects means it is likely to outperform the BernoulliNB baseline.

Random Forest:

- Using bagging, the variance can be greatly reduced.

- It can determine the importance of the features in the news files

- Robust to outliers

- Easy to train

- Works well on classification problems with either 'Gini' or 'Information Gain' criterion.

1.5 Did they work as you thought?

They all worked as I expected, that is they all beat the baseline model.

However, the only point I didn't expect is Multinomial Naïve Bayes outperforms the Logistic Regression. As the Naïve Bayes assumption doesn't hold.

My explanation to this scenario is that, even the Naïve Bayes assumption doesn't hold perfectly here. But in the sparse feature matrix we have here, the correlation among features are diluted, and are weak. So even the assumption is not perfectly matched, it still can predict with a high accuracy.

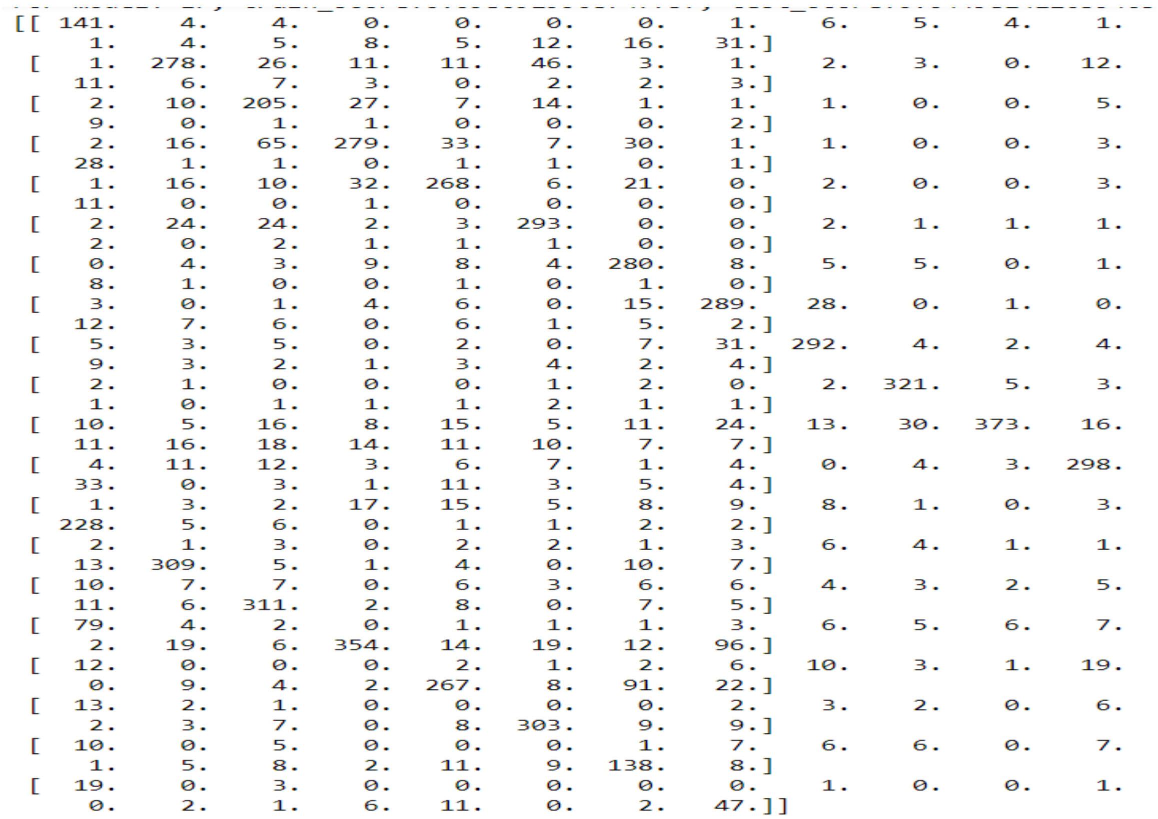
1.6 Confusion Matrix & Heatmap

As shown in the following page, the confusion matrix and the corresponding heat map are presented. With Cij in the matrix meaning number of text examples belonging to class j that were classified as class i.

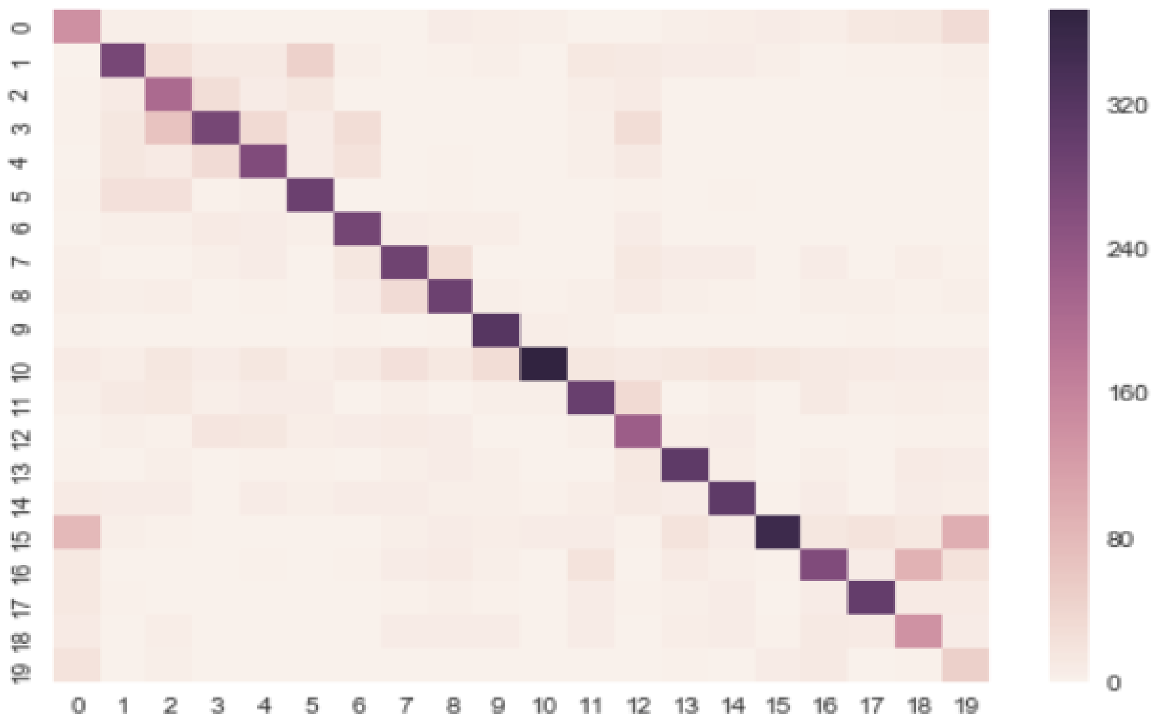
1.7 Two classes the classifier was most confused about

From the matrix and heatmap, we can see, the most misclassification happens in C[15][19]=96, that means the classifier confused most about classes with index 15 and 19.

Screenshot of code output: Part 2

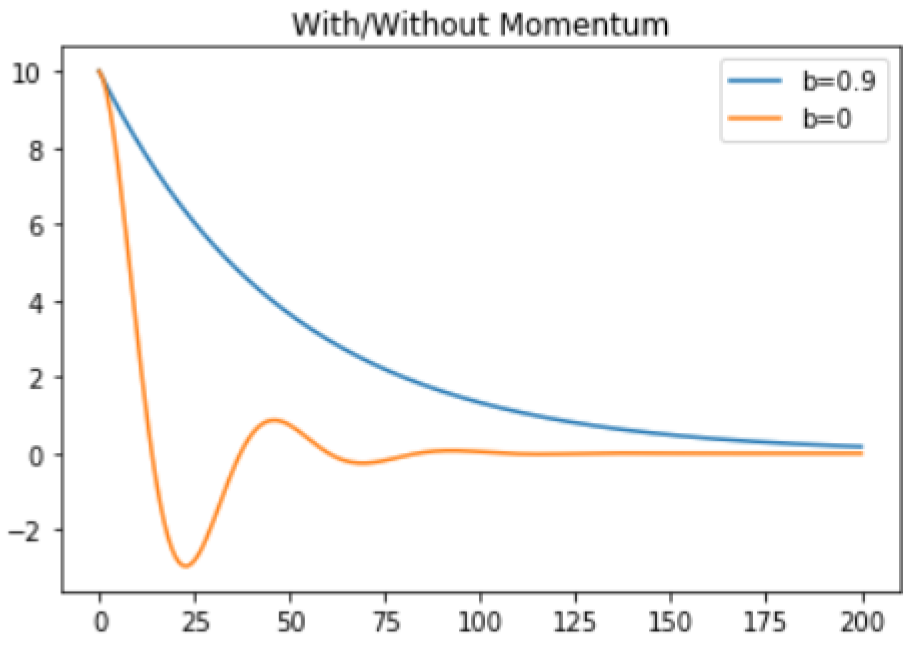


```
[[ 141.    4.    4.    0.    0.    0.    0.    1.    6.    5.    4.    1.
     1.    4.    5.    8.    5.   12.   16.   31.]
 [   1.  278.   26.   11.   11.   46.    3.    1.    2.    3.    0.   12.
    11.    6.    7.    3.    0.    2.    2.    3.]
 [   2.   10.  205.   27.    7.   14.    1.    1.    1.    0.    0.    5.
     9.    0.    1.    1.    0.    0.    0.    2.]
 [   2.   16.   65.  279.   33.    7.   30.    1.    1.    0.    0.    3.
    28.    1.    1.    0.    1.    1.    0.    1.]
 [   1.   16.   10.   32.  268.    6.   21.    0.    2.    0.    0.    3.
    11.    0.    0.    1.    0.    0.    0.    0.]
 [   2.   24.   24.    2.    3.  293.    0.    0.    2.    1.    1.    1.
     2.    0.    2.    1.    1.    1.    0.    0.]
 [   0.    4.    3.    9.    8.    4.  280.    8.    5.    5.    0.    1.
     8.    1.    0.    0.    1.    0.    1.    0.]
 [   3.    0.    1.    4.    6.    0.   15.  289.   28.    0.    1.    0.
    12.    7.    6.    0.    6.    1.    5.    2.]
 [   5.    3.    5.    0.    2.    0.    7.   31.  292.    4.    2.    4.
     9.    3.    2.    1.    3.    4.    2.    4.]
 [   2.    1.    0.    0.    0.    1.    2.    0.    2.  321.    5.    3.
     1.    0.    1.    1.    1.    2.    1.    1.]
 [  10.    5.   16.    8.   15.    5.   11.   24.   13.   30.  373.   16.
    11.   16.   18.   14.   11.   10.    7.    7.]
 [   4.   11.   12.    3.    6.    7.    1.    4.    0.    4.    3.  298.
    33.    0.    3.    1.   11.    3.    5.    4.]
 [   1.    3.    2.   17.   15.    5.    8.    9.    8.    1.    0.    3.
   228.    5.    6.    0.    1.    1.    2.    2.]
 [   2.    1.    3.    0.    2.    2.    1.    3.    6.    4.    1.    1.
    13.  309.    5.    1.    4.    0.   10.    7.]
 [  10.    7.    7.    0.    6.    3.    6.    6.    4.    3.    2.    5.
    11.    6.  311.    2.    8.    0.    7.    5.]
 [  79.    4.    2.    0.    1.    1.    1.    3.    6.    5.    6.    7.
     2.   19.    6.  354.   14.   19.   12.   96.]
 [  12.    0.    0.    0.    2.    1.    2.    6.   10.    3.    1.   19.
     0.    9.    4.    2.  267.    8.   91.   22.]
 [  13.    2.    1.    0.    0.    0.    0.    2.    3.    2.    0.    6.
     2.    3.    7.    0.    8.  303.    9.    9.]
 [  10.    0.    5.    0.    0.    0.    1.    7.    6.    6.    0.    7.
     1.    5.    8.    2.   11.    9.  138.    8.]
 [  19.    0.    3.    0.    0.    0.    0.    0.    1.    0.    0.    1.
     0.    2.    1.    6.   11.    0.    2.   47.]]
```


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
320
240
160
80
0

Question 2.

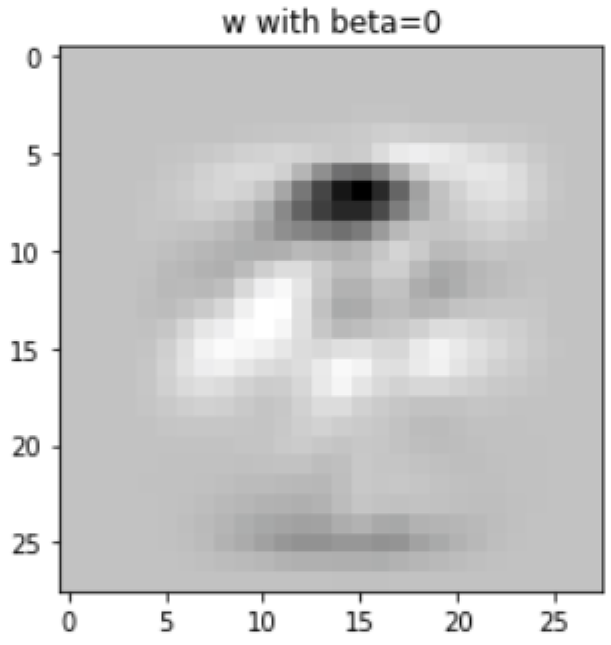
2.1 Plot w for 200 time steps using beta = 0 and beta =0.9 on the same graph



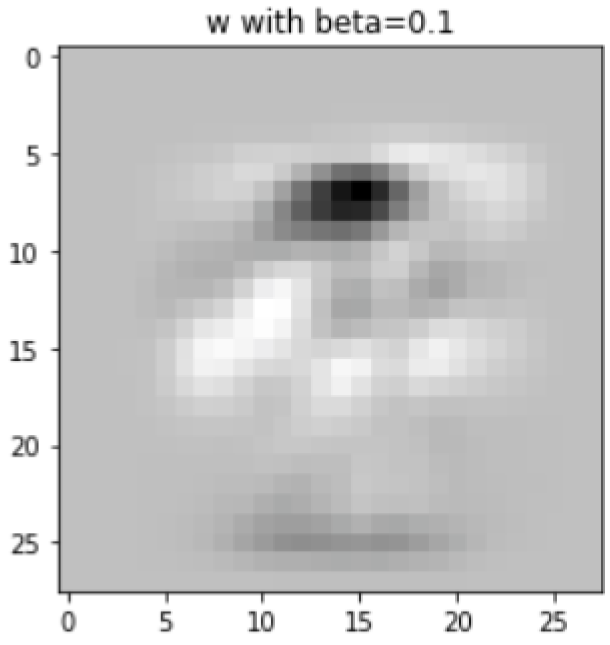
2.2 & 2.3

With the default hyperparameters given in the problem. Two SVM models were trained, with beta values 0 and 0.1 respectively.

```
For Model with beta=0:
           Train accuracy: 0.9287981859410431
           Test accuracy: 0.9281828073993471
           Avg train hinge loss: 0.33776093130000046
          Avg test hinge loss: 0.33257565493062063
```



```
For Model with beta=0.1:
          Train accuracy: 0.9287981859410431
          Test accuracy: 0.9289082335872325
          Avg train hinge loss: 0.3384608877475445
          Avg test hinge loss: 0.33359209525799677
```



Question 3:

3.1 To prove the claim, we have to prove both of the following

① If a symmetric matrix $K \in R^{d \times d}$ is positive semidefinite $\Rightarrow$ for all vector $x \in R^d$, $x^T K x \geq 0$

② If for all vectors $x \in R^d$, $x^T K x \geq 0 \Rightarrow$ The symmetric $K \in R^{d \times d}$ is positive semidefinite.

To prove ①:
- we know K is positive semidefinite $\Rightarrow K \succeq 0$
- We can write $K = UDU^T$ where U is orthogonal and D is diagonal. $\Rightarrow$ Entries on the diagonal of D are eigenvalues of K, and are all non-negative.

$\Rightarrow$ Let $D = C^2$, where C is diagonal matrix.

$\Rightarrow K = UCCU^T = UC \cdot (UC)^T$

= For every $x \in R^d$, $x^T K x = x^T (UC)(UC)^T x = (x^T UC)^T \cdot (UCx) \geq 0$

$\Rightarrow$ proposition proved

To prove ②:

Known for all vectors $x \in R^d$, $x^T K x \geq 0$

Let v be an eigenvector of K with eigenvalue λ.

$\Rightarrow v^T K v = \lambda v^T v \geq 0$

$\Rightarrow \lambda \geq 0$

$\Rightarrow K \succeq 0$

$\Rightarrow$ The symmetric $K \in R^{d \times d}$ is positive semidefinite

$\Rightarrow$ proposition proved

From proof ① and ②
Both direction hold $\Rightarrow$ claim is proved

Q 3. Kernel

3.2.1
To prove $k(x,y) = a$ is a kernel,
we want to find an embedding $\phi(x)$ such that
$K(x,y) = \langle \phi(x), \phi(y) \rangle$

Let $\phi(x) = \sqrt{a}$, which is valid $\forall\ a \geq 0$

$K(x,y) = \langle \phi(x), \phi(y) \rangle = a$

$\Rightarrow$ exists an embedding, s.t $K(x,y) = a$ is kernel, $\forall\ a$

3.2.2. $\phi(x) = f(x)$, $f : \mathbb{R}^d \to \mathbb{R}$

$K(x,y) = \langle \phi(x), \phi(y) \rangle = f(x) \cdot f(y)$

$\Rightarrow$ exist an embedding, s.t. $K(x,y) = f(x) \cdot f(y)$
$\Rightarrow$ $K(x,y) = f(x) \cdot f(y)$ is a kernel for $f : \mathbb{R}^d \to \mathbb{R}$

3.2.3. k_1 has its feature map ϕ_1 and inner product $\langle\,\rangle_{k_1}$
k_2 has its feature map ϕ_2 and inner product $\langle\,\rangle_{k_2}$

$$
\begin{aligned}
K(x,y) &= a\, k_1(x,y) + B\, k_2(x,y) \\
&= \langle \sqrt{a}\, \phi_1(x), \sqrt{a}\, \phi_2(y) \rangle_{k_1} + \langle \sqrt{B}\, \phi_2(x), \sqrt{B}\, \phi_2(y) \rangle_{k_2} \\
&= \langle [\sqrt{a}\, \phi_1(x); \sqrt{B}\, \phi_2(x)], [\sqrt{a}\, \phi_1(y), \sqrt{B}\, \phi_2(y)] \rangle_{k_{new}}
\end{aligned}
$$

$\Rightarrow$ $k(x,z)$ can be expressed as inner product
$\Rightarrow$ its a valid kernel.

3.24. Define feature map = ϕ_1 as the feature map for k_1.

Define $\phi_{new} = \frac{\phi_1}{\|\phi_1\|}$

$$k(x,y) = \frac{k_1(x,y)}{\sqrt{k_1(x,x)}\sqrt{k_1(y,y)}} = \frac{\langle \phi_1(x), \phi_1(y) \rangle}{\sqrt{\langle \phi_1(x), \phi_1(x) \rangle}\sqrt{\langle \phi_1(y), \phi_1(y) \rangle}}$$

$$= \langle \phi_{new}(x), \phi_{new}(y) \rangle$$

$\Rightarrow$ $K(x,y) = \frac{k_1(x,y)}{\sqrt{k_1(x,x)}\sqrt{k_1(y,y)}}$ is a valid kernel.