**Final** Folder contains dataset as well as code files

**Code Files:**

**1)question1.py:** This file shows text data into a matrix of features × examples for all training Datasets Hw1, Enron1, Enron4. Execute the file by writing python question1.py on terminal

**2)final\_mnb\_bow.py** This file contains the whole implementation of MultiomialNB. I have trained the algorithm on 3 datasets (hw1, enron4, enron1) and tested on all of them. Execute the file by python final\_mnb\_bow.py on terminal.

**Testing accuracies (enron1: 99.77%, enron4: 91.02%, hw1: 98.2%)**

Classification Report on enron1:

precision recall f1-score support

0 1.00 1.00 1.00 319

1 0.99 1.00 1.00 131

Classification Report on enron4:

precision recall f1-score support

0 0.73 1.00 0.85 133

1 1.00 0.88 0.94 402

Classification Report on hw1:

precision recall f1-score support

0 0.99 0.98 0.99 340

1 0.95 0.98 0.97 123

**3)discrete\_naive\_bayes.py** This file contains the whole implementation of DiscreteNB. I have trained the algorithm on 3 datasets (hw1, enron4, enron1) and tested on all of them. Execute the file by writing python discrete\_naive\_bayes.py on terminal.

**Testing accuracies (enron1: 66.7%, enron4: 41.3%, hw1: 74.3%)**

**4-1) bow\_mcap\_lr.py:** This file contains the whole implementation of MCAP Bag of Words. I have trained the algorithm on 3 datasets (hw1, enron4, enron1) and tested on all of them. Execute the file by writing python bow\_mcap\_lr.py on terminal.

**Testing accuracies (enron1: 94.3%, enron4: 97.6%, hw1: 94.35%) for Lamda(L2 Regularizer)=1**

**Learning Rate=0.0045**

**Epochs: 75**

**Why these Hyperparameters?**

**Lamda:** I took a list of lamda[1, 2, 3, 4] values and ran the algorithm on every lambda and found one with highest validation accuracy.

**Epochs:** I plotted the graph of all the epochs vs log likelihood values of training and validation set and found the best suitable value of epochs as 75(This is the hard limit that I set on my algorithm)

**Learning Rate:** Found the perfect balance of consistent increase on gradient ascent when used 0.0045 learning rate.

Enron1:

precision recall f1-score support

0 0.97 0.94 0.96 307

1 0.89 0.94 0.92 149

Enron4:

precision recall f1-score support

0 0.99 0.92 0.96 152

1 0.97 1.00 0.98 391

Hw1:

precision recall f1-score support

0 0.97 0.96 0.96 348

1 0.89 0.91 0.90 130

**4-2) bernoulli\_mcap\_lr.py:** This file contains the whole implementation of MCAP Bernoulli. I have trained the algorithm on 3 datasets (hw1, enron4, enron1) and tested on all of them. Execute the file by writing python bernoulli\_mcap\_lr.py

**Testing accuracies (enron1: 94.9%, enron4: 99.8%, hw1: 94.5%) for Lamda(L2 Regularizer)=1**

**Learning Rate=0.0045**

**Epochs: 75**

**Why these Hyperparameters?**

**Lamda:** I took a list of lamda[1, 2, 3, 4] values and ran the algorithm on every lambda and found one with highest validation accuracy.

**Epochs:** I plotted the graph of all the epochs vs log likelihood values of training and validation set and found the best suitable value of epochs as 75(This is the hard limit that I set on my algorithm)

**Learning Rate:** Found the perfect balance of consistent increase on gradient ascent when used 0.0045 learning rate.

On enron1:

precision recall f1-score support

0 0.97 0.96 0.96 307

1 0.91 0.93 0.92 149

On enron4:

precision recall f1-score support

0 1.00 0.87 0.93 152

1 0.95 1.00 0.98 391

On Hw1:

precision recall f1-score support

0 0.95 0.97 0.96 348

1 0.93 0.87 0.90 130

**6) sgd.py :** This file contains the whole implementation of SGDClassifier implementation of sklearn. I have trained the algorithm on 3 datasets (hw1) and tested on all of them. Execute the file by writing python sgd.py

* I achieved higher accuracy on Logistic Regression as compared to SGDClassifier.

**Validation accuracies BOW (enron1: 94.66%, enron4: 95.7%, hw1: 93.3%)**

**Validation accuracies Bernoulli (enron1: 95.8%, enron4: 95%, hw1: 94.5%)**