**Report**

 First I have preprocessed the tweets by applying removing non alphanumeric chars, integers, remove the stopwords and did tokenization.

Each tweet is tokenized, then tokenizer is called to make the each document of equal length. Each tweet is accounted for max padded length 64.

Now for each padded tokenized tweet or doc having index of words in the vocab created by calling tokenizer.

We will use pre trained model glove for extracting the embedding of each token word in each tweet.

Each padded tokenized doc is processed if the index of the word is not zero and exist in vocab, its embedding is extracted from pertained model using embedding\_index dictionary, I am going to add each embedding of corresponding word present in a single tweet cumulatively and will store that in document embedding matrix for each tweet.

This will be done for each tweet or document which will be feed in the multilayer perceptron.

I have tried to run the multilayer perceptron two, three and four hidden layers respectively. I have also tried to implement the neural network using keras where multilayer prepceptron offered by keras api have been used and I have tested it on tested data. I have added embedding layer as input for it. Then one hidden layer , followed by dense layer.

Results are as follows it was done in order to compare with sklearn multilayer prepceptron:

Results for train test split:

Model: "sequential\_4"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_5 (Embedding) (None, 64, 100) 1843700

flatten\_5 (Flatten) (None, 6400) 0

dense\_5 (Dense) (None, 1) 6401

=================================================================

Total params: 1,850,101

Trainable params: 6,401

Non-trainable params: 1,843,700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Accuracy: 82.824773

Results for test file:

Loaded 400000 word vectors.

Model: "sequential\_4"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_5 (Embedding) (None, 64, 100) 1843700

flatten\_5 (Flatten) (None, 6400) 0

dense\_5 (Dense) (None, 1) 6401

=================================================================

Total params: 1,850,101

Trainable params: 6,401

Non-trainable params: 1,843,700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

Accuracy: 0.116279

**Result for Two layer:**

**Predicted probabilities for test data out of training data:**

[[0.58463613 0.41536387]

[0.94721926 0.05278074]

[0.53609484 0.46390516]

...

[0.86768698 0.13231302]

[0.95292953 0.04707047]

[0.07924145 0.92075855]]

[[1792 418]

[ 473 627]]

precision recall f1-score support

0 0.79 0.81 0.80 2210

1 0.60 0.57 0.58 1100

accuracy 0.73 3310

macro avg 0.70 0.69 0.69 3310

weighted avg 0.73 0.73 0.73 3310

For Test data

Probabilities:

[[0.0694259 0.9305741 ]

[0.70226752 0.29773248]

[0.74382285 0.25617715]

...

[0.35968539 0.64031461]

[0.54252447 0.45747553]

[0.72939341 0.27060659]]

Confusion Matrix

[[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

...

[0 1 0 ... 0 0 0]

[0 1 0 ... 0 0 0]

[1 0 0 ... 0 0 0]]

Getting Accuracy zero for test file:

accuracy 0.00 860

macro avg 0.00 0.00 0.00 860

weighted avg 0.00 0.00 0.00 860

**Result for one layer:**

For the test data out of trained data

[[0.71784282 0.28215718]

[0.98654325 0.01345675]

[0.6572887 0.3427113 ]

...

[0.7342784 0.2657216 ]

[0.89708054 0.10291946]

[0.16990044 0.83009956]]

[[1830 380]

[ 540 560]]

precision recall f1-score support

0 0.77 0.83 0.80 2210

1 0.60 0.51 0.55 1100

accuracy 0.72 3310

macro avg 0.68 0.67 0.67 3310

weighted avg 0.71 0.72 0.72 3310

For the actual test data:

[[0.0694259 0.9305741 ]

[0.70226752 0.29773248]

[0.74382285 0.25617715]

...

[0.35968539 0.64031461]

[0.54252447 0.45747553]

[0.72939341 0.27060659]]

[[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

...

[0 1 0 ... 0 0 0]

[0 1 0 ... 0 0 0]

[1 0 0 ... 0 0 0]]

Getting Accuracy zero for test file:

accuracy 0.00 860

macro avg 0.00 0.00 0.00 860

weighted avg 0.00 0.00 0.00 860

**Result for three layer**

For the test data out of trained data

[[0.80316921 0.19683079]

[0.86733713 0.13266287]

[0.78142511 0.21857489]

...

[0.91709002 0.08290998]

[0.94046621 0.05953379]

[0.05938537 0.94061463]]

[[1848 362]

[ 474 626]]

precision recall f1-score support

0 0.80 0.84 0.82 2210

1 0.63 0.57 0.60 1100

accuracy 0.75 3310

macro avg 0.71 0.70 0.71 3310

weighted avg 0.74 0.75 0.74 3310

**For the actual test data:**

[[0.27093277 0.72906723]

[0.72509668 0.27490332]

[0.52624476 0.47375524]

...

[0.88612517 0.11387483]

[0.89638668 0.10361332]

[0.68711385 0.31288615]]

[[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

...

[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]

[1 0 0 ... 0 0 0]]

Getting Accuracy zero for test file:

accuracy 0.00 860

macro avg 0.00 0.00 0.00 860

weighted avg 0.00 0.00 0.00 860

With increasing layers accuracy has been resulted 75 percentage using three hidden layer on train test split on entire data but when tested on actual test file given I am getting zero accuracy perhaps model has gone overfitted. I have used pretrained model glove for embeddings.