# Artificial Intelligence

## Assignment #4


### Nour El-Din Hazem - 6261 ~~ Amr Mohamad Salah - 6287


---

## Part 1: Data Preprocessing

The following preprocessing methods are used:

- Removing Stop word
- Removing XML/HTML Tags
- Removing punctuation
- Lowercase all characters
- Lemmatization of words

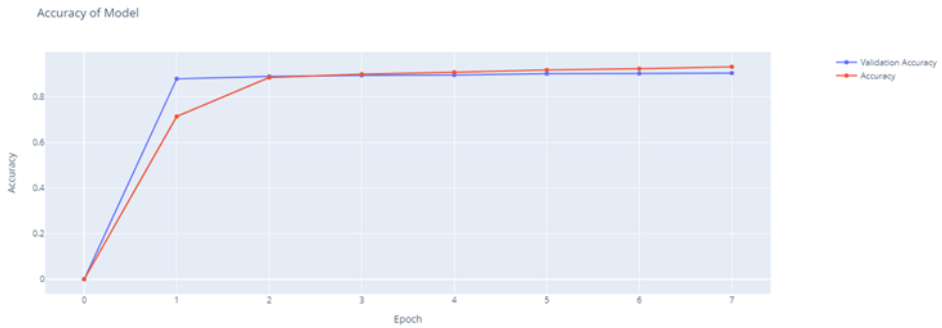
---

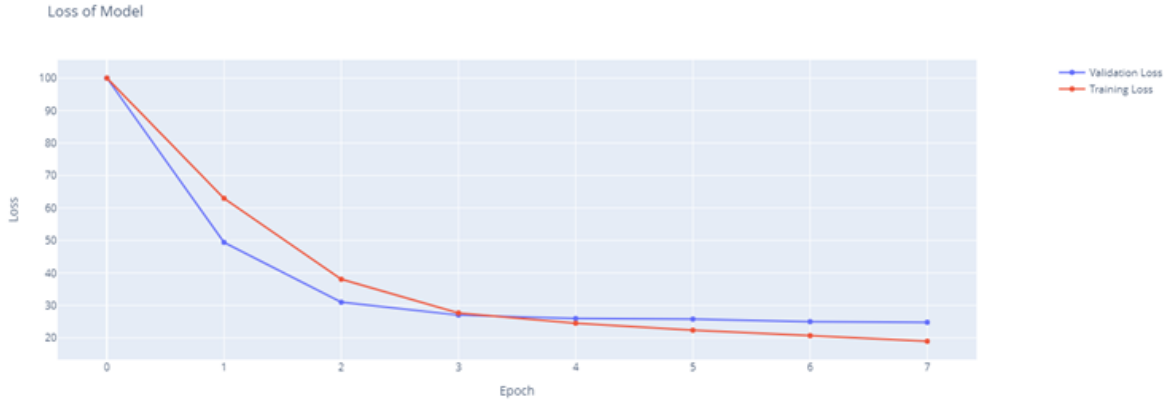
## Part 2: Results On the preprocessed Data

Different learning rates are used to get the best results:

Learning rates : $10^{-6}$, $10^{-5}$, $10^{-4}$, $10^{-2}$ and $10^{-8}$

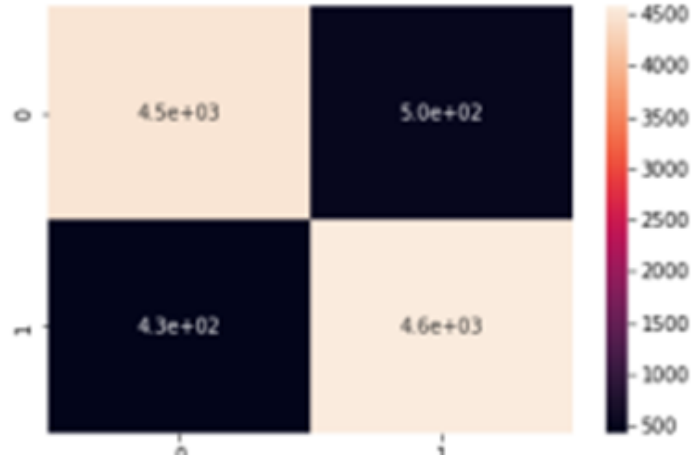
1. $10^{-6}$



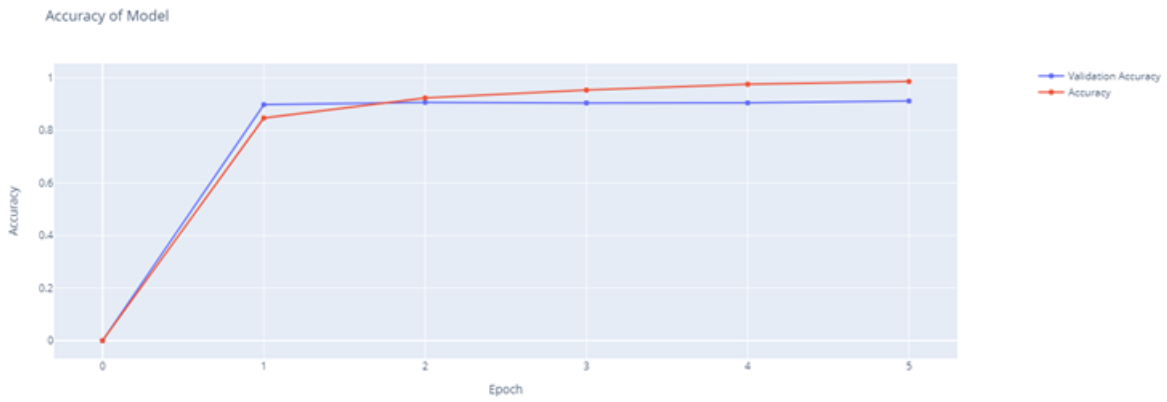


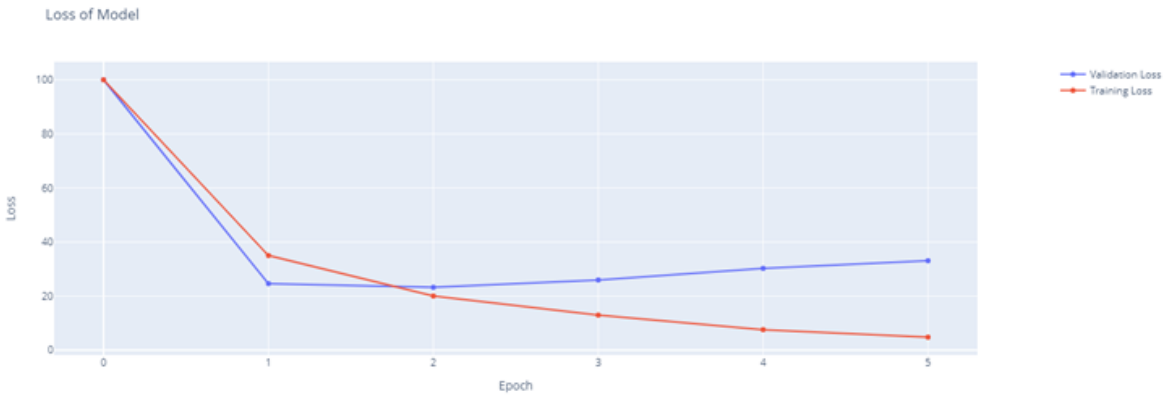

Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.91 | 0.90 | 0.91 | 5000 |
| 1.0 | 0.90 | 0.91 | 0.91 | 5000 |
| accuracy | | | 0.91 | 10000 |
| macro avg | 0.91 | 0.91 | 0.91 | 10000 |
| weighted avg | 0.91 | 0.91 | 0.91 | 10000 |



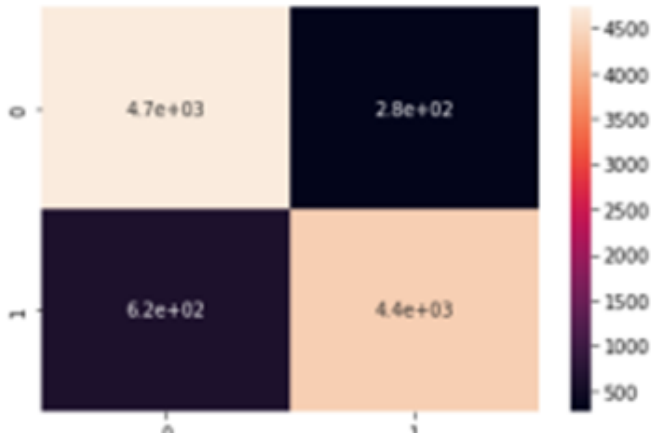

2. $10^{-5}$


Accuracy of Model
Validation Accuracy
Accuracy
Accuracy
Epoch
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5

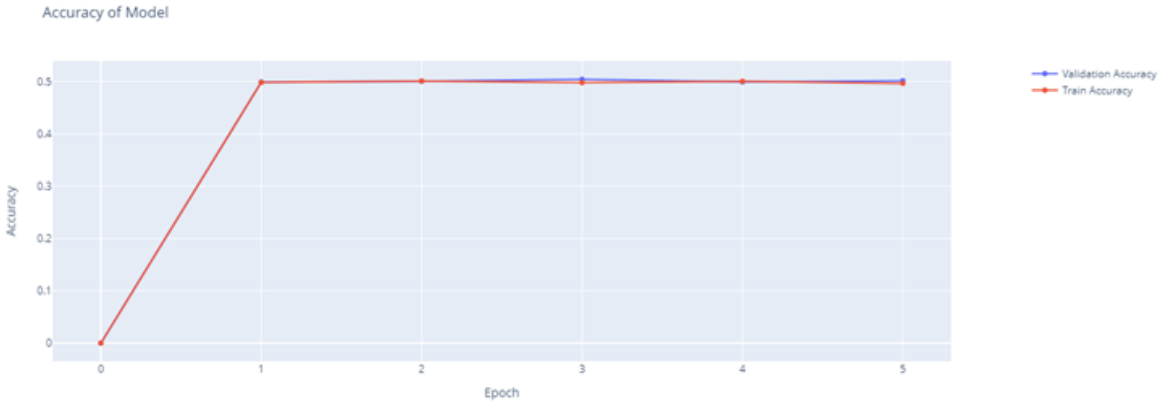

Loss of Model
Validation Loss
Training Loss
Loss
Epoch
0
20
40
60
80
100
0
1
2
3
4
5

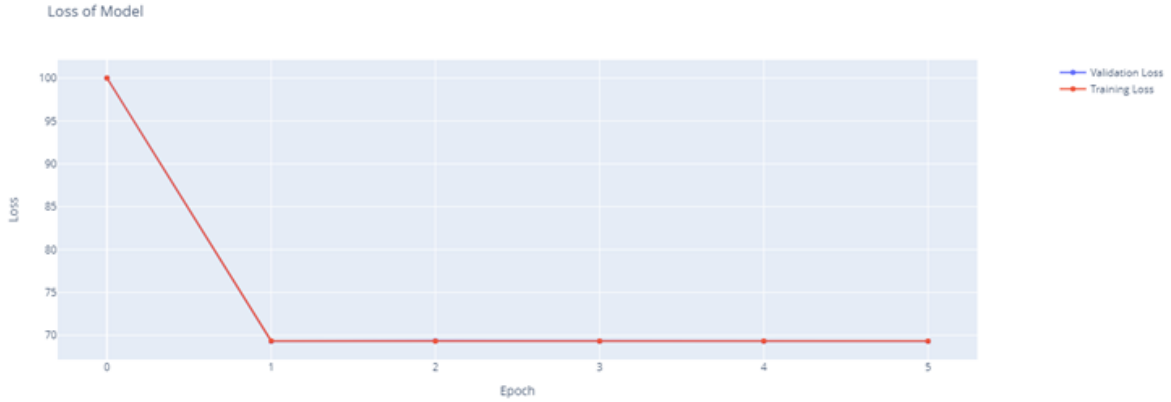
Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.88 | 0.94 | 0.91 | 5000 |
| 1.0 | 0.94 | 0.88 | 0.91 | 5000 |
| accuracy | | | 0.91 | 10000 |
| macro avg | 0.91 | 0.91 | 0.91 | 10000 |
| weighted avg | 0.91 | 0.91 | 0.91 | 10000 |



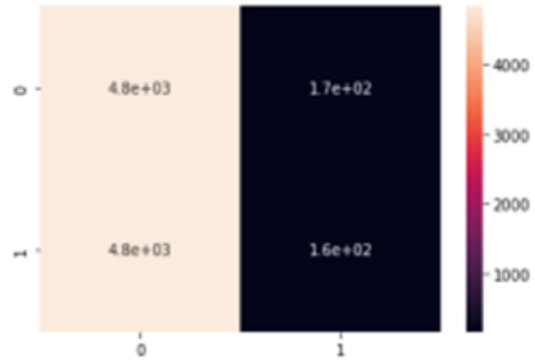

3. $10^{-4}$



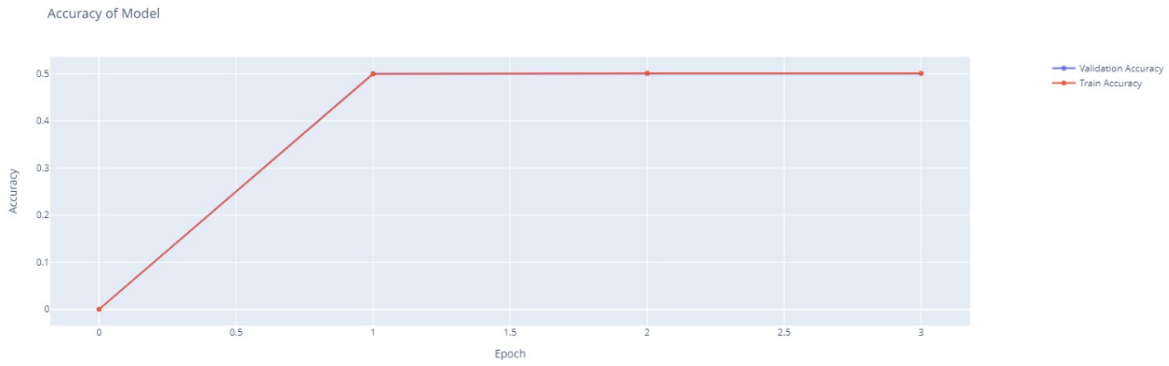


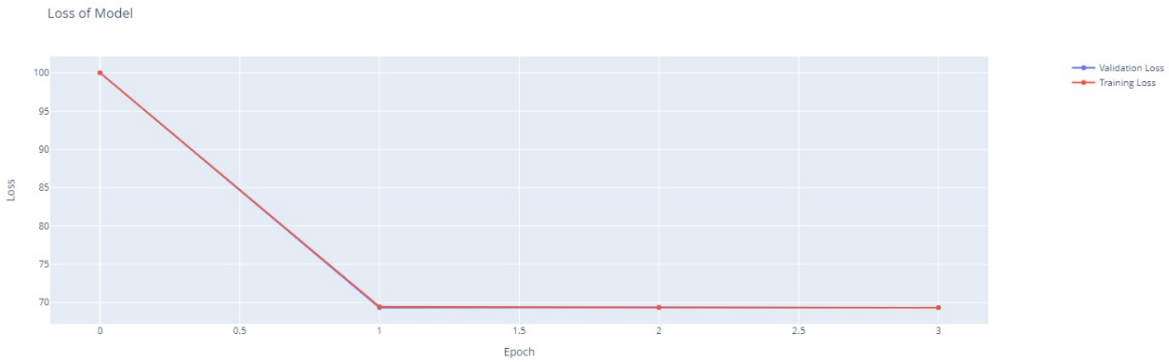

Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.50 | 0.97 | 0.66 | 5000 |
| 1.0 | 0.48 | 0.03 | 0.06 | 5000 |
| accuracy | | | 0.50 | 10000 |
| macro avg | 0.49 | 0.50 | 0.36 | 10000 |
| weighted avg | 0.49 | 0.50 | 0.36 | 10000 |



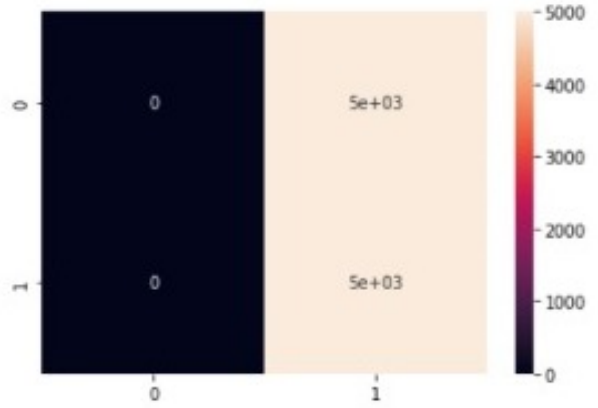

4. $10^{-2}$


Accuracy of Model
Validation Accuracy
Train Accuracy
Accuracy
0
0.1
0.2
0.3
0.4
0.5
0
0.5
1
1.5
2
2.5
3
Epoch

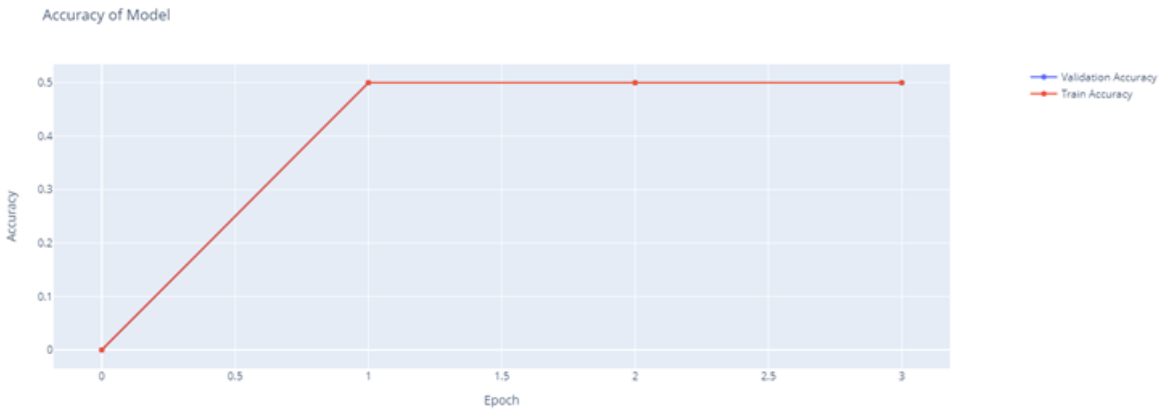

Loss of Model
Validation Loss
Training Loss
Loss
70
75
80
85
90
95
100
0
0.5
1
1.5
2
2.5
3
Epoch

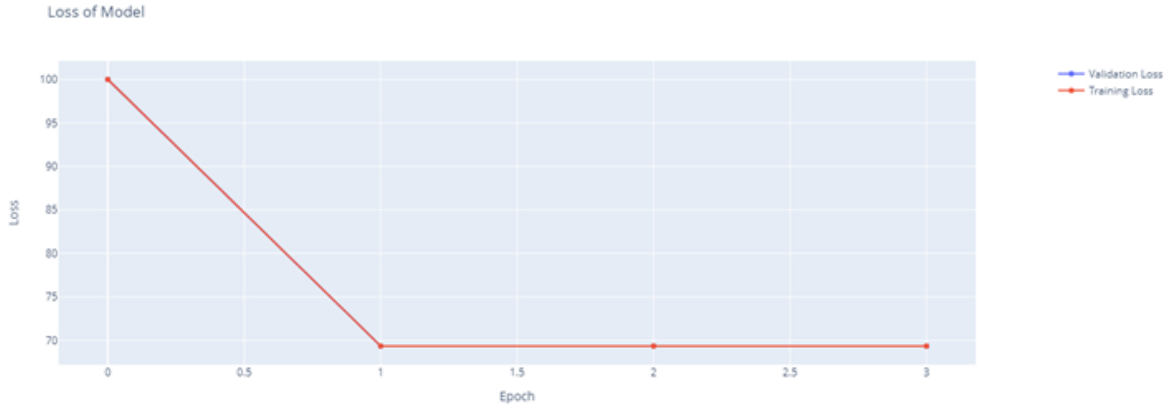
Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 5000 |
| 1.0 | 0.50 | 1.00 | 0.67 | 5000 |
| accuracy | | | 0.50 | 10000 |
| macro avg | 0.25 | 0.50 | 0.33 | 10000 |
| weighted avg | 0.25 | 0.50 | 0.33 | 10000 |



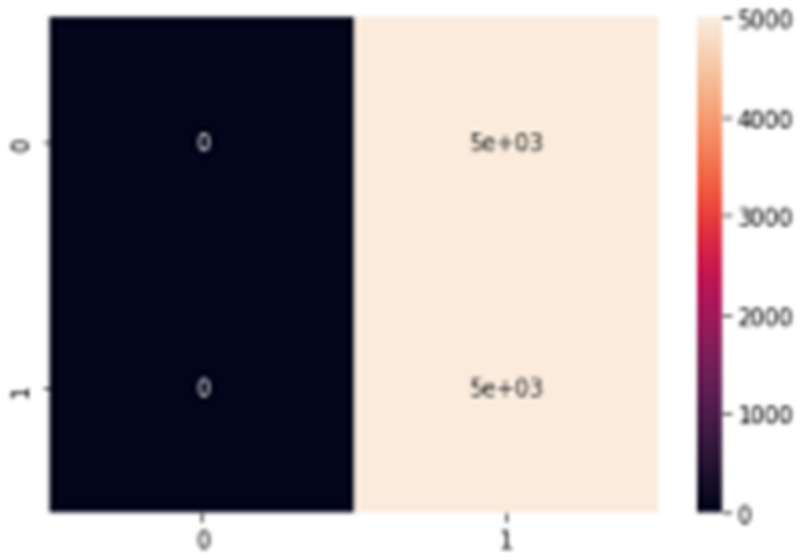

5. $10^{-8}$



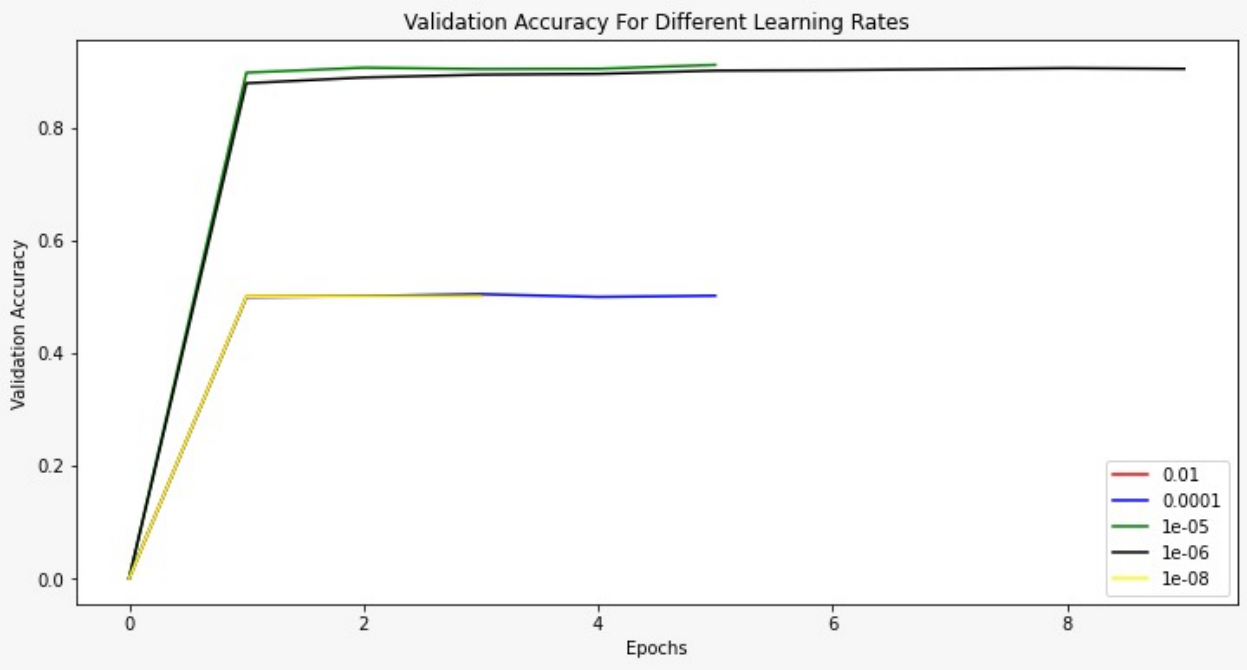



Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 5000 |
| 1.0 | 0.50 | 1.00 | 0.67 | 5000 |
| accuracy | | | 0.50 | 10000 |
| macro avg | 0.25 | 0.50 | 0.33 | 10000 |
| weighted avg | 0.25 | 0.50 | 0.33 | 10000 |




Plotting Various Learning rates with the validation Accuracies:

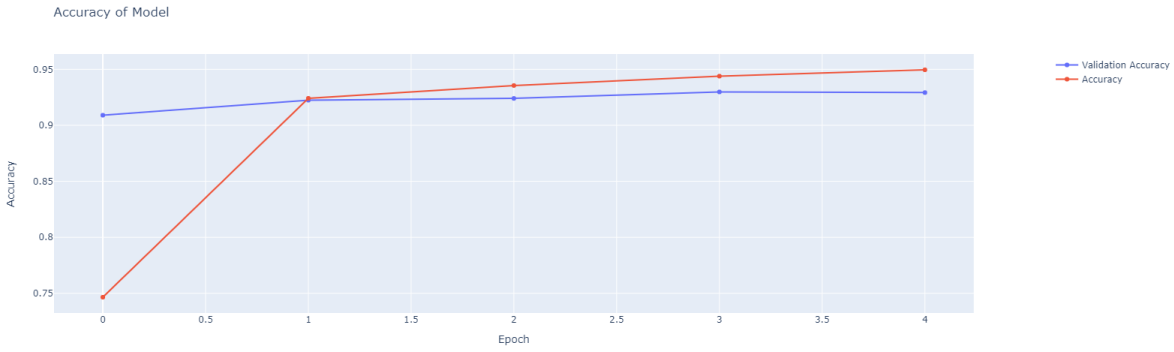


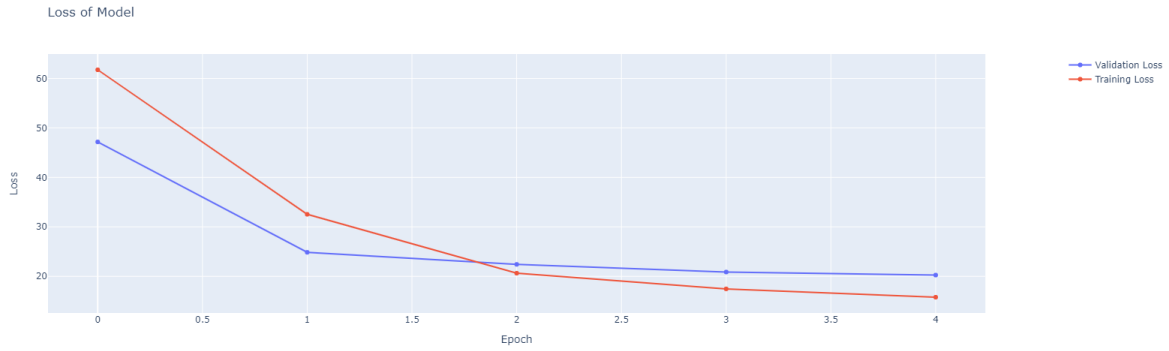

## Results of Original Data

Different learning rates are used to get the best results:

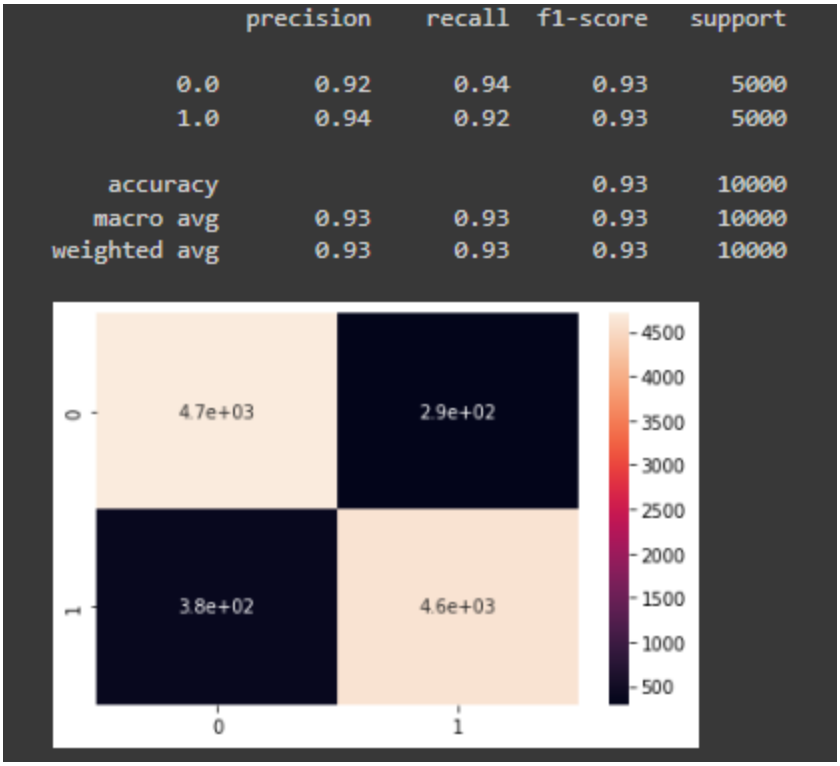
Learning rates : $10^{-6}$, $10^{-5}$, $10^{-4}$ and $10^{-2}$

1. $10^{-6}$

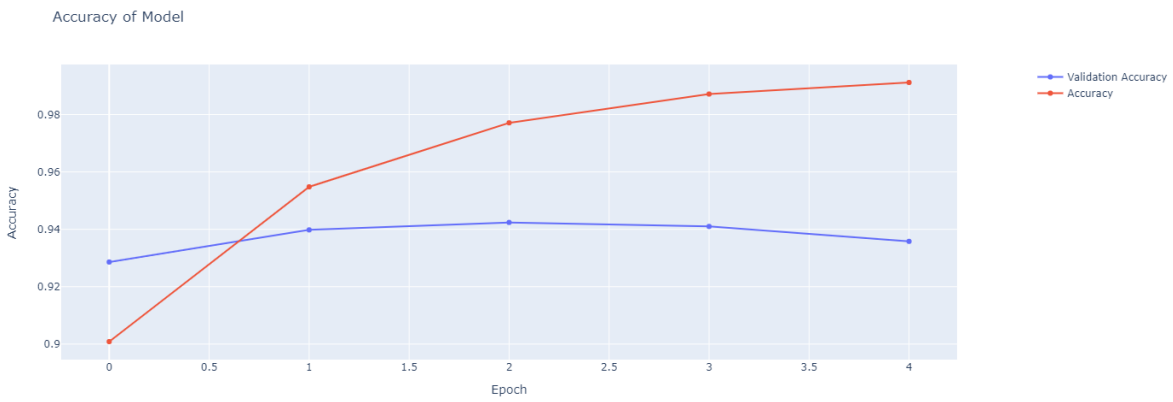


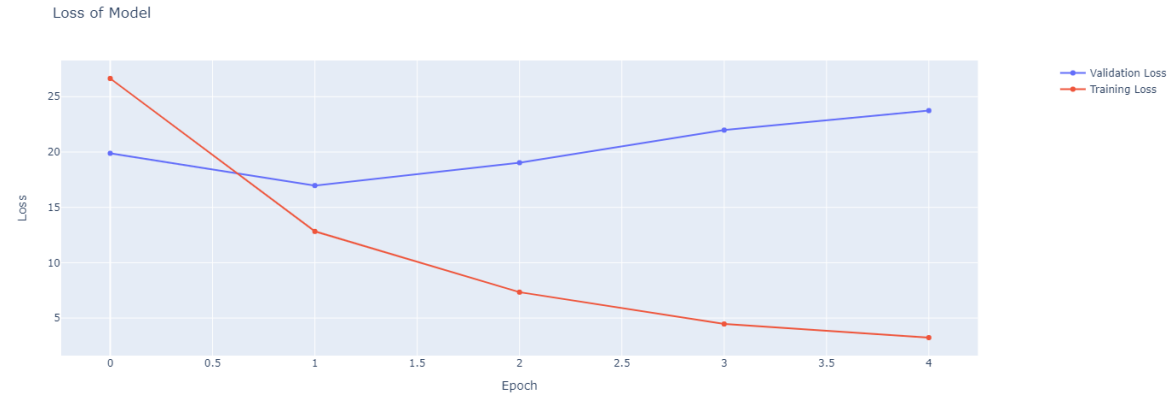



Results:

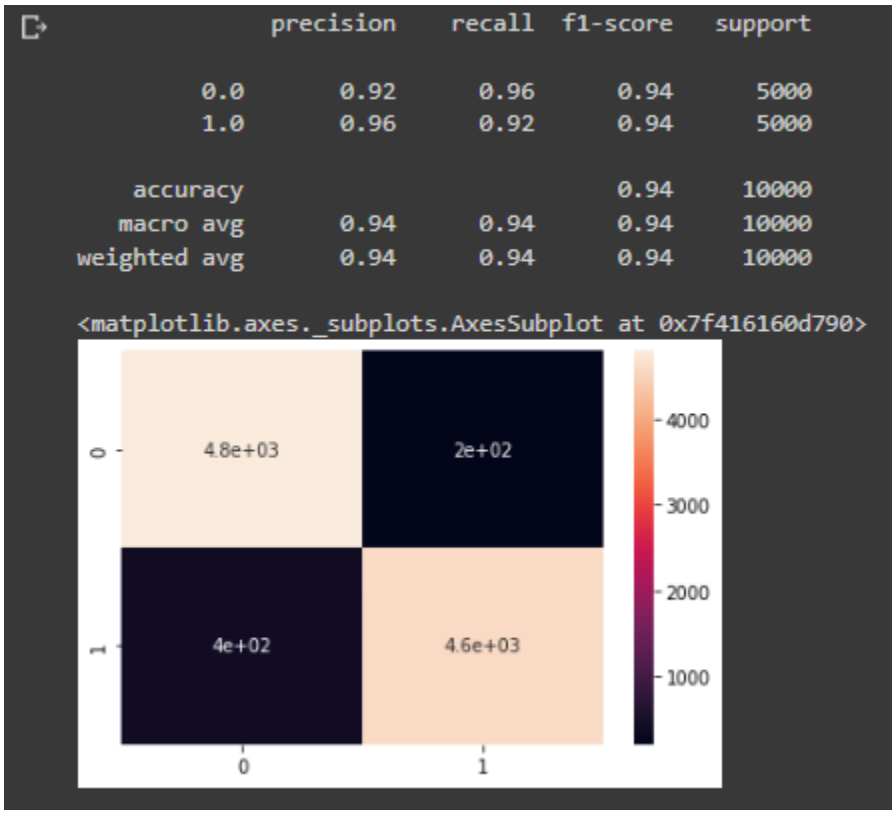



2. $10^{-5}$

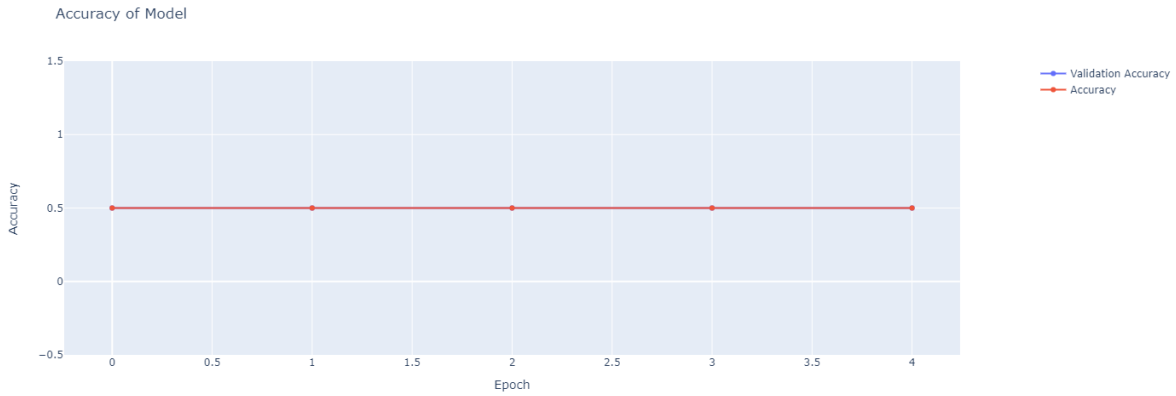

Accuracy of Model
Validation Accuracy
Accuracy
Accuracy
0.98
0.96
0.94
0.92
0.9
0
0.5
1
1.5
2
2.5
3
3.5
4
Epoch

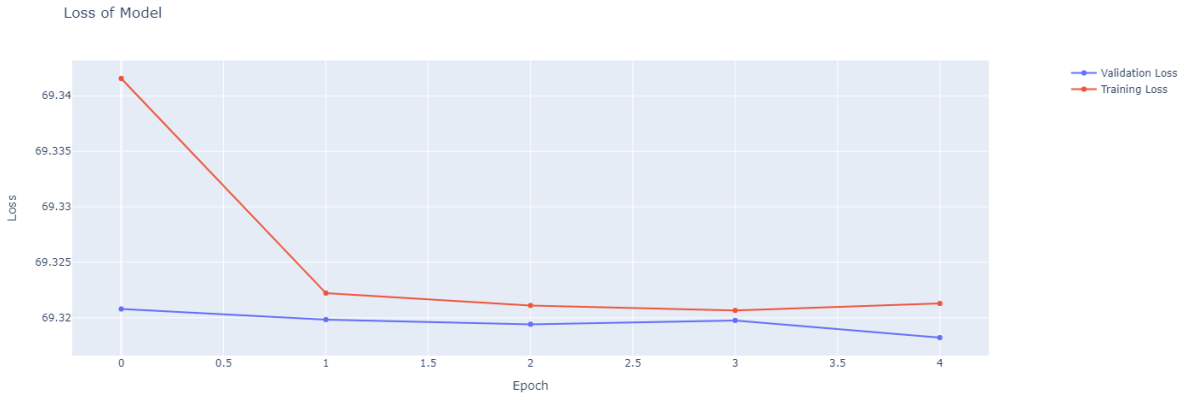

Loss of Model
Validation Loss
Training Loss
Loss
25
20
15
10
5
0
0.5
1
1.5
2
2.5
3
3.5
4
Epoch

Results:


precision recall f1-score support
0.0 0.92 0.96 0.94 5000
1.0 0.96 0.92 0.94 5000
accuracy 0.94 10000
macro avg 0.94 0.94 0.94 10000
weighted avg 0.94 0.94 0.94 10000
<matplotlib.axes._subplots.AxesSubplot at 0x7f416160d790>
4.8e+03
2e+02
4e+02
4.6e+03
0
1
0
1
4000
3000
2000
1000

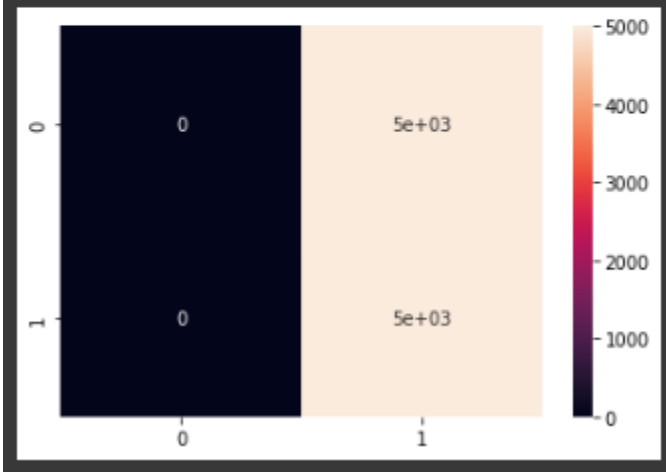
3. $10^{-4}$


Accuracy of Model
Validation Accuracy
Accuracy
Accuracy
Epoch
1.5
1
0.5
0
−0.5
0
0.5
1
1.5
2
2.5
3
3.5
4

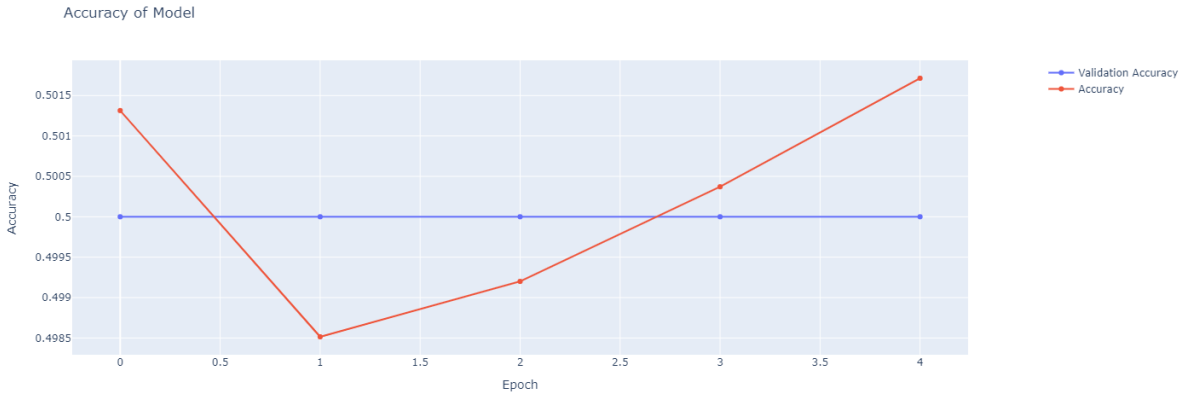


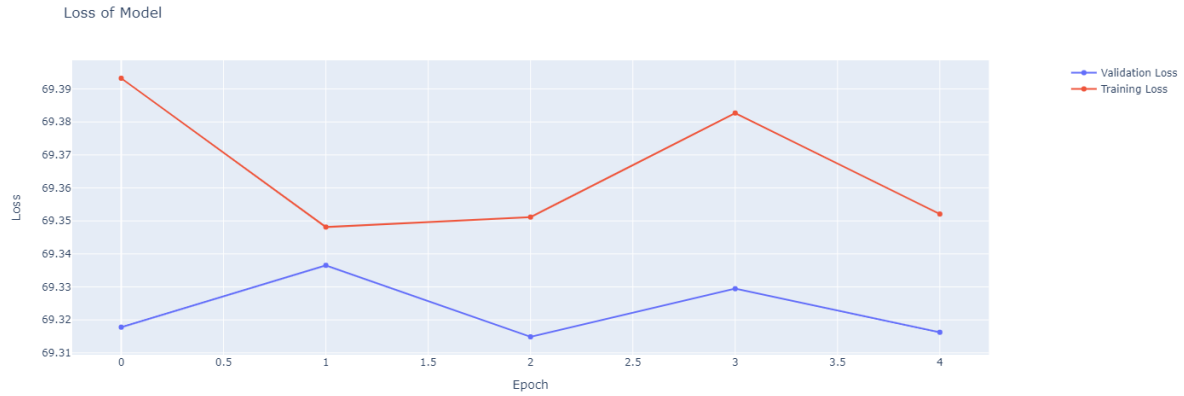

Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 5000 |
| 1.0 | 0.50 | 1.00 | 0.67 | 5000 |
| accuracy | | | 0.50 | 10000 |
| macro avg | 0.25 | 0.50 | 0.33 | 10000 |
| weighted avg | 0.25 | 0.50 | 0.33 | 10000 |



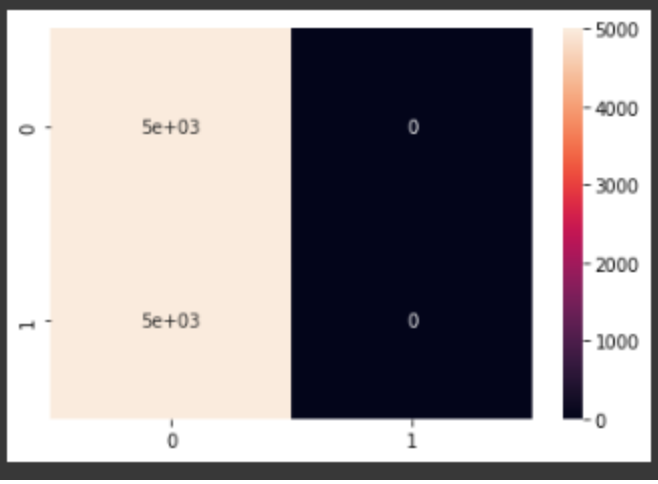

4. $10^{-2}$


Accuracy of Model
Validation Accuracy
Accuracy
0.5015
0.501
0.5005
0.5
0.4995
0.499
0.4985
Accuracy
0
0.5
1
1.5
2
2.5
3
3.5
4
Epoch

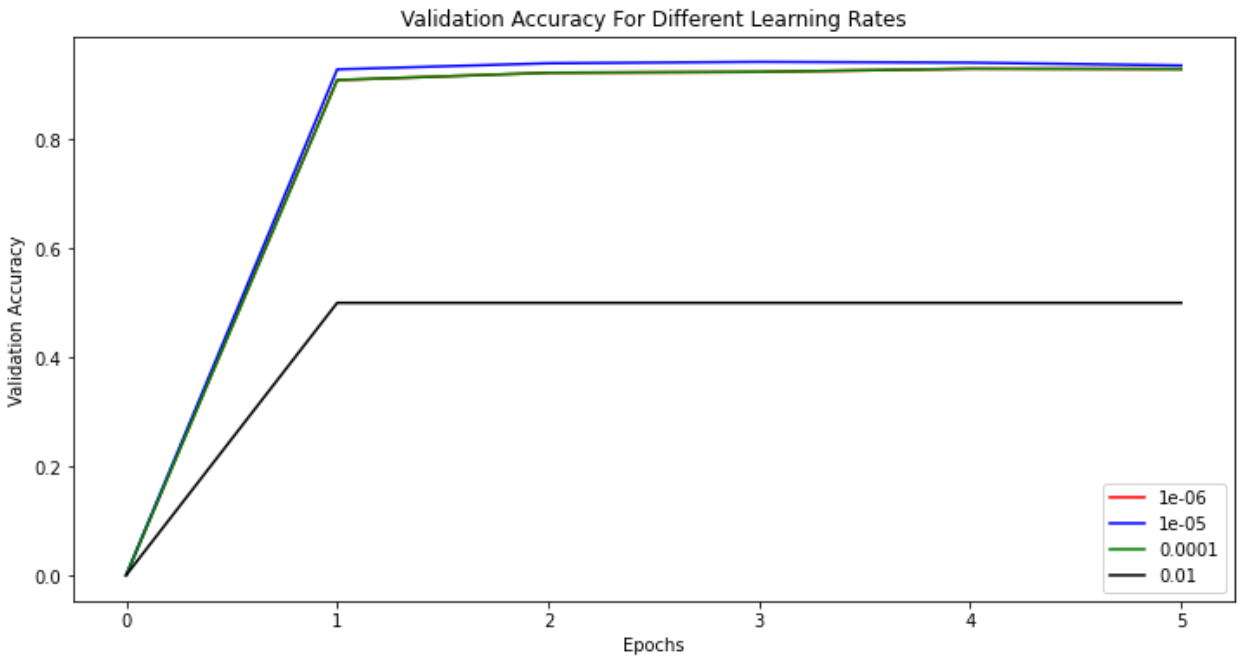

Loss of Model
Validation Loss
Training Loss
69.39
69.38
69.37
69.36
69.35
69.34
69.33
69.32
69.31
Loss
0
0.5
1
1.5
2
2.5
3
3.5
4
Epoch

Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.50 | 1.00 | 0.67 | 5000 |
| 1.0 | 0.00 | 0.00 | 0.00 | 5000 |
| accuracy | | | 0.50 | 10000 |
| macro avg | 0.25 | 0.50 | 0.33 | 10000 |
| weighted avg | 0.25 | 0.50 | 0.33 | 10000 |




Plotting Various Learning rates with the validation Accuracies:



# Comments

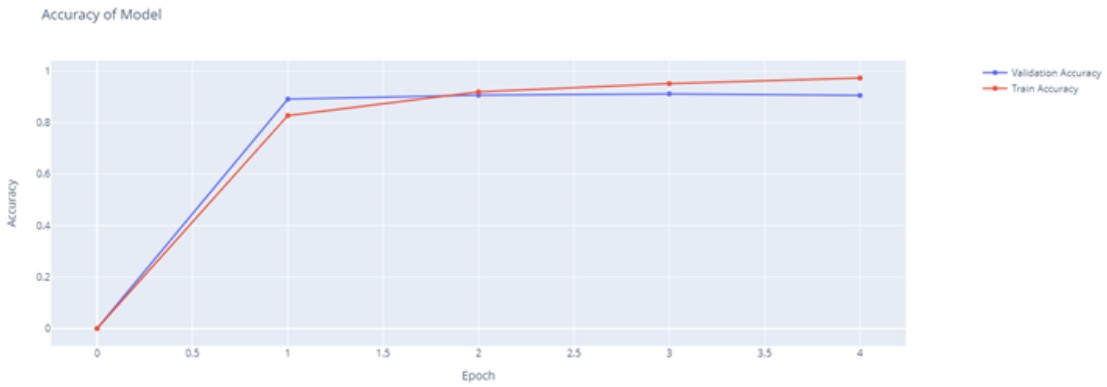
1. As Expected the results of the Model on the Original Data are better than The results acquired on the preprocessed data because by applying preprocessing some meaningful context was removed due to stopping words and also Lemmatization so the sentence lose its context.

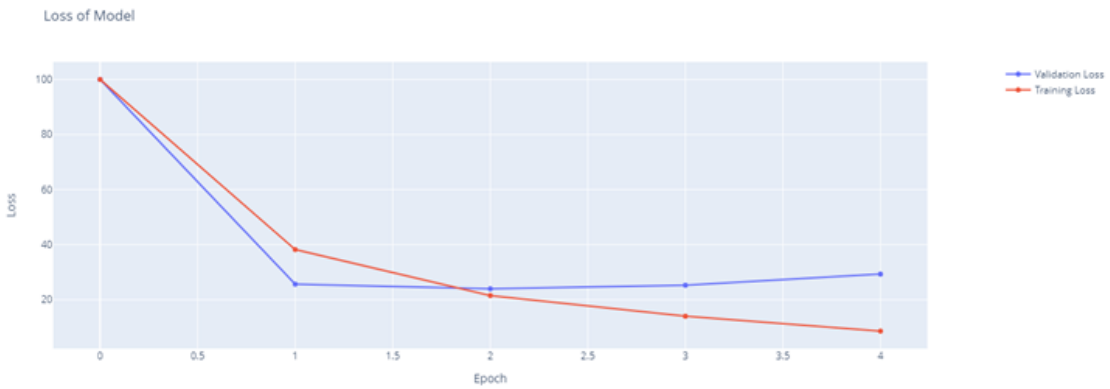
| Learning Rates | Original Data | Preprocessed |
|---|---|---|
| 10^-5 | 93% | 91% |
| 10^-6 | 92.8% | 90.3% |
| 10^-4 | 50% | 50% |
| 10^-2 | 50% | 50% |

   Depending on the validation Accuracies.

2. With Learning rate = 10^-5 → The Model Faced overfitting, to avoid overfitting dropout layers where added, the results are as shown

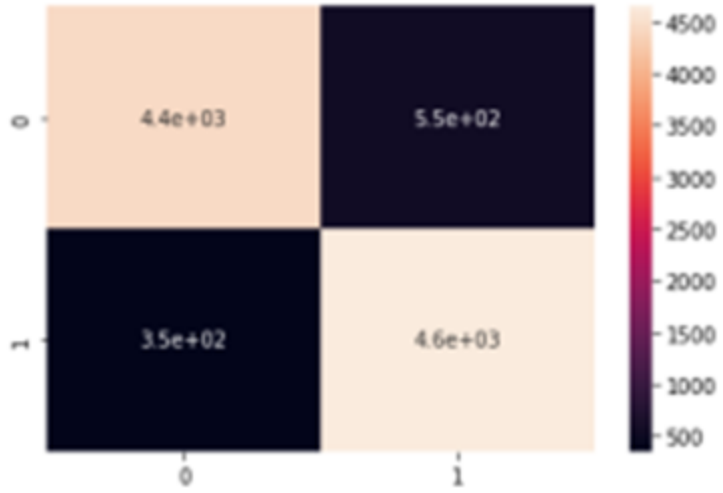
   a. Added 1 Dropout between the 2nd and 3rd layer → Total of 2 Dropout layers






Results

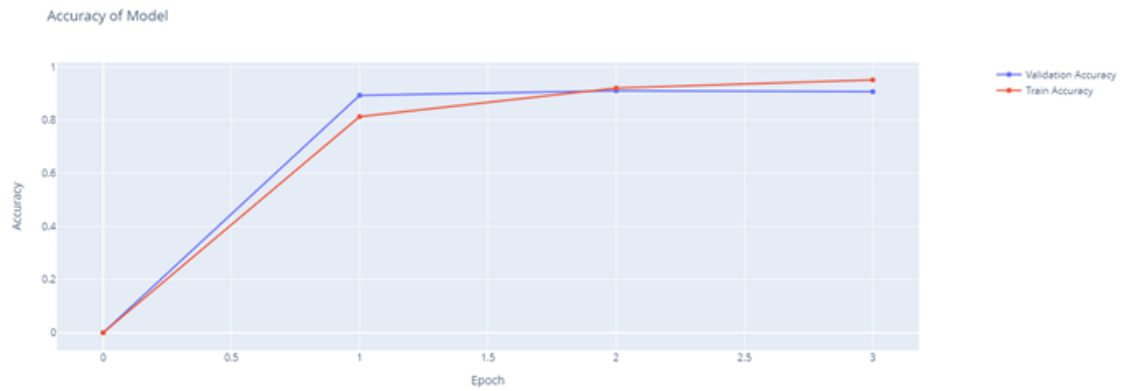
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.93 | 0.89 | 0.91 | 5000 |
| 1.0 | 0.89 | 0.93 | 0.91 | 5000 |
| accuracy | | | 0.91 | 10000 |
| macro avg | 0.91 | 0.91 | 0.91 | 10000 |
| weighted avg | 0.91 | 0.91 | 0.91 | 10000 |

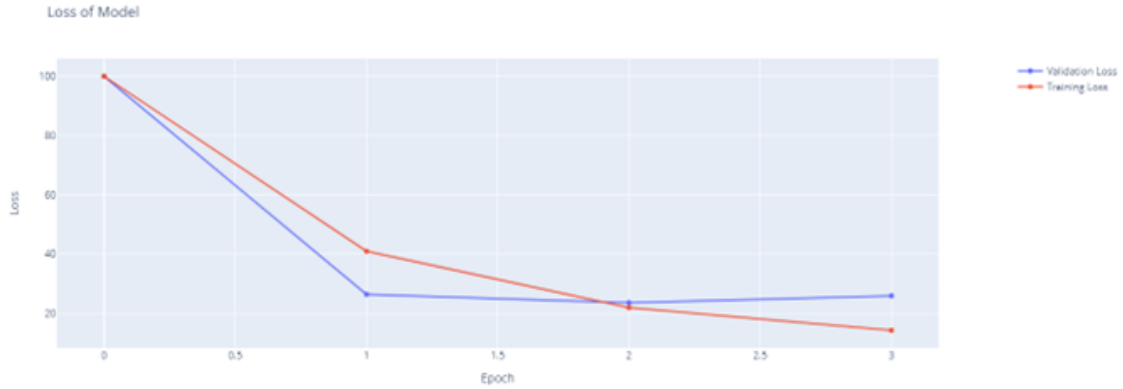



As shown in the results, the model still faced overfitting so another dropout layer was added.

---

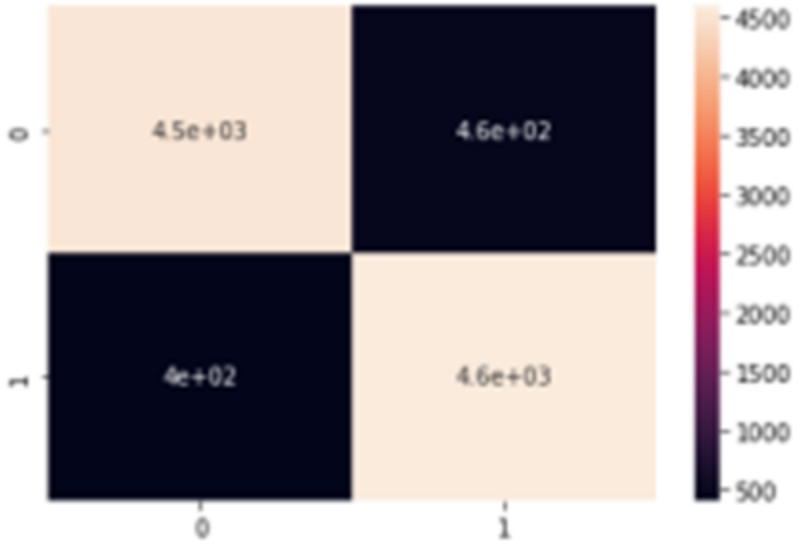
b. Added 1 Dropout between the 1nd and 2nd layer → Total of 3 Dropout layers


Accuracy of Model
Validation Accuracy
Train Accuracy
Accuracy
Epoch
0
0.2
0.4
0.6
0.8
1
0
0.5
1
1.5
2
2.5
3


Loss of Model
Validation Loss
Training Loss
Loss
Epoch
20
40
60
80
100
0
0.5
1
1.5
2
2.5
3

Results:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.92 | 0.91 | 0.91 | 5000 |
| 1.0 | 0.91 | 0.92 | 0.91 | 5000 |
| accuracy | | | 0.91 | 10000 |
| macro avg | 0.91 | 0.91 | 0.91 | 10000 |
| weighted avg | 0.91 | 0.91 | 0.91 | 10000 |




3. On Using Learning Rates = $10^{-2}, 10^{-4}\ and\ 10^{-8}$, the model faced underfitting where the training accuracy and validation accuracy remained constant at 50% for all epochs (the model always predicts the same class either positive or negative and since the data is balanced so the accuracy is always 50%).

   Our attempts to solve this problem:

   1. Changing Model Complexity by using 2 approaches:
      a. Adding Dropout layers.
      b. Adding/Removing layers to/from the model.
   2. Increasing the number of Epochs.
   3. Tried Different Weight initializations (Xavier, He).
   4. Using Scheduler.
   5. Tried Different Weight Decay values.

6. Added Batch Normalization.
7. Tried Using Different Activation function for the lase Dense layer (tanh or sigmoid).
8. Using **BCEWithLogitsLoss** which according to the documentation → “**combines a Sigmoid layer and the BCELoss in one single class. This version is more numerically stable than using a plain Sigmoid followed by a BCELoss**”.

Note: When the Model architecture was changed so that the last layer output is 2 classes instead of 1 the model accuracy improved significantly and we didn’t tackle upon the 1 class prediction problem while keeping all other variables constant.

Note: On trying different learning rates we made sure that all other variables regarding the process was constant and only the learning rate was changing i.e. model architecture, data, weight decay, tokenization aspects, loss function and scheduler options where all constant for the same learning rate. The same procedures were repeated while testing the effect of any other aspect on the accuracy of the model (Keeping all other aspects constant while changing only the aspect that is being tested)