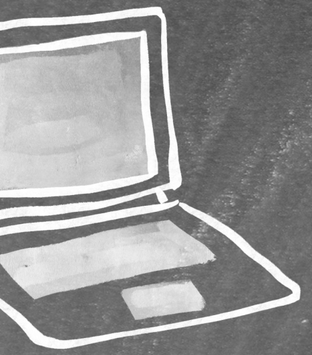
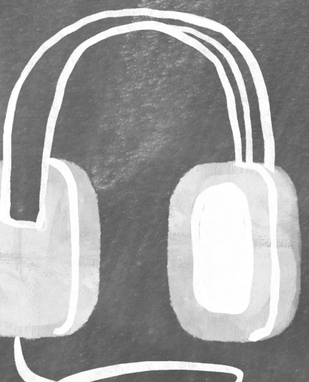
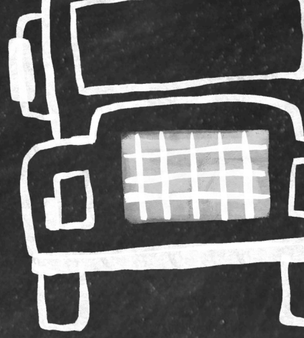
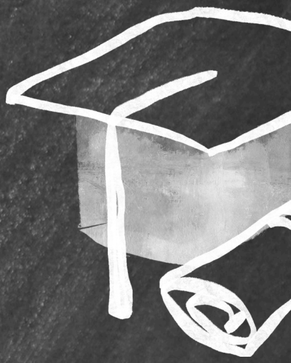





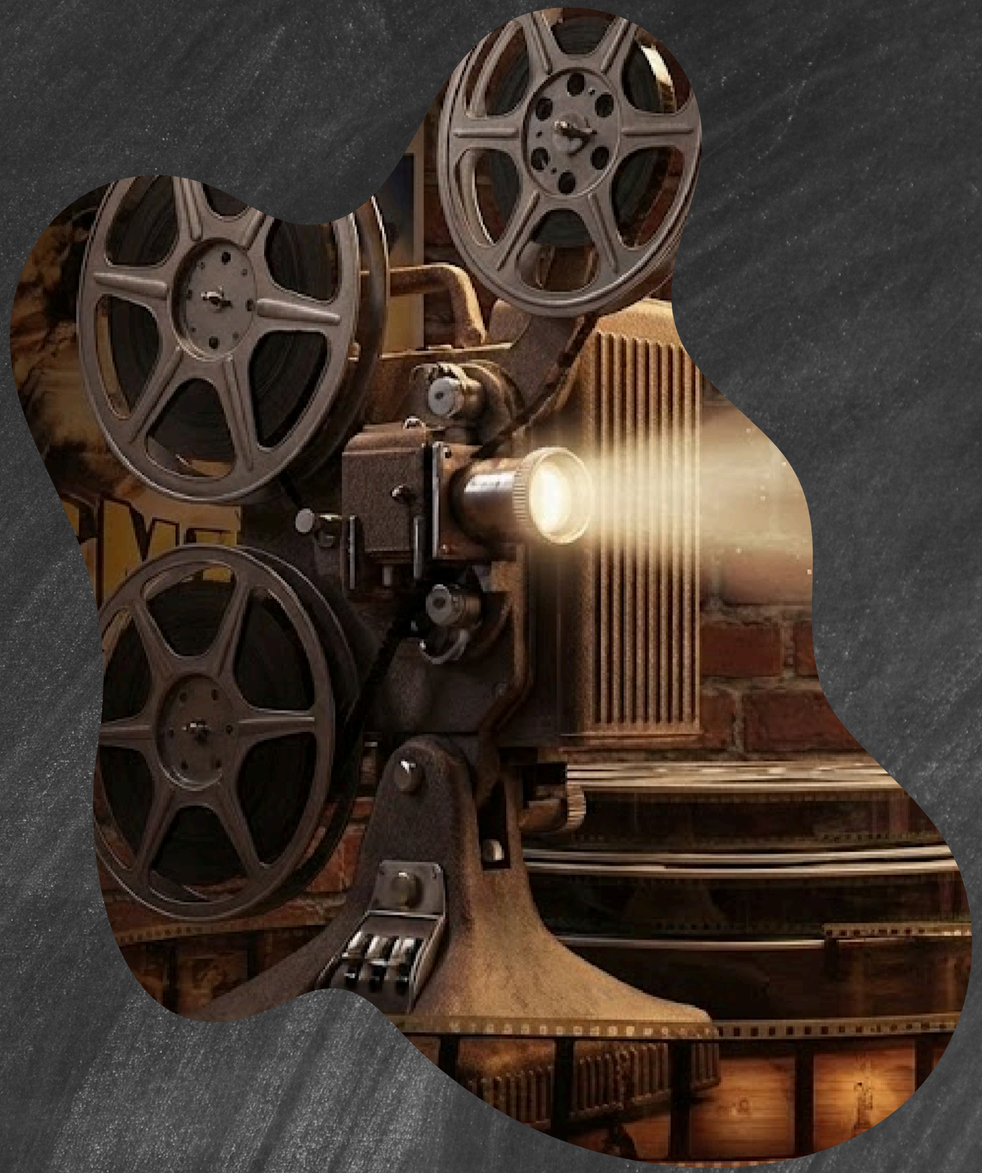

DEEP LEARNING - BONUS PROJECT

SUBMIT BY:

ITAMAR BABAI , DOTAN DANINO

PROJECT DESCRIPTION

- Dataset - "IMDB Movie Reviews Dataset" that contain reviews text and a label of negative / positive review.
- Goal of The project: build a model that classifies whether a given movie review is positive or negative.
- we split the data 80% for train and 20% test.



DUMMY MODEL

- for the dummy model we build a model that completely ignores the content of the reviews and always predicts the majority class in the data

- To implement we used the DummyClassifier from scikit-learn Library with strategy = most_frequent and of course train him on the train and test on the test

RESULTS

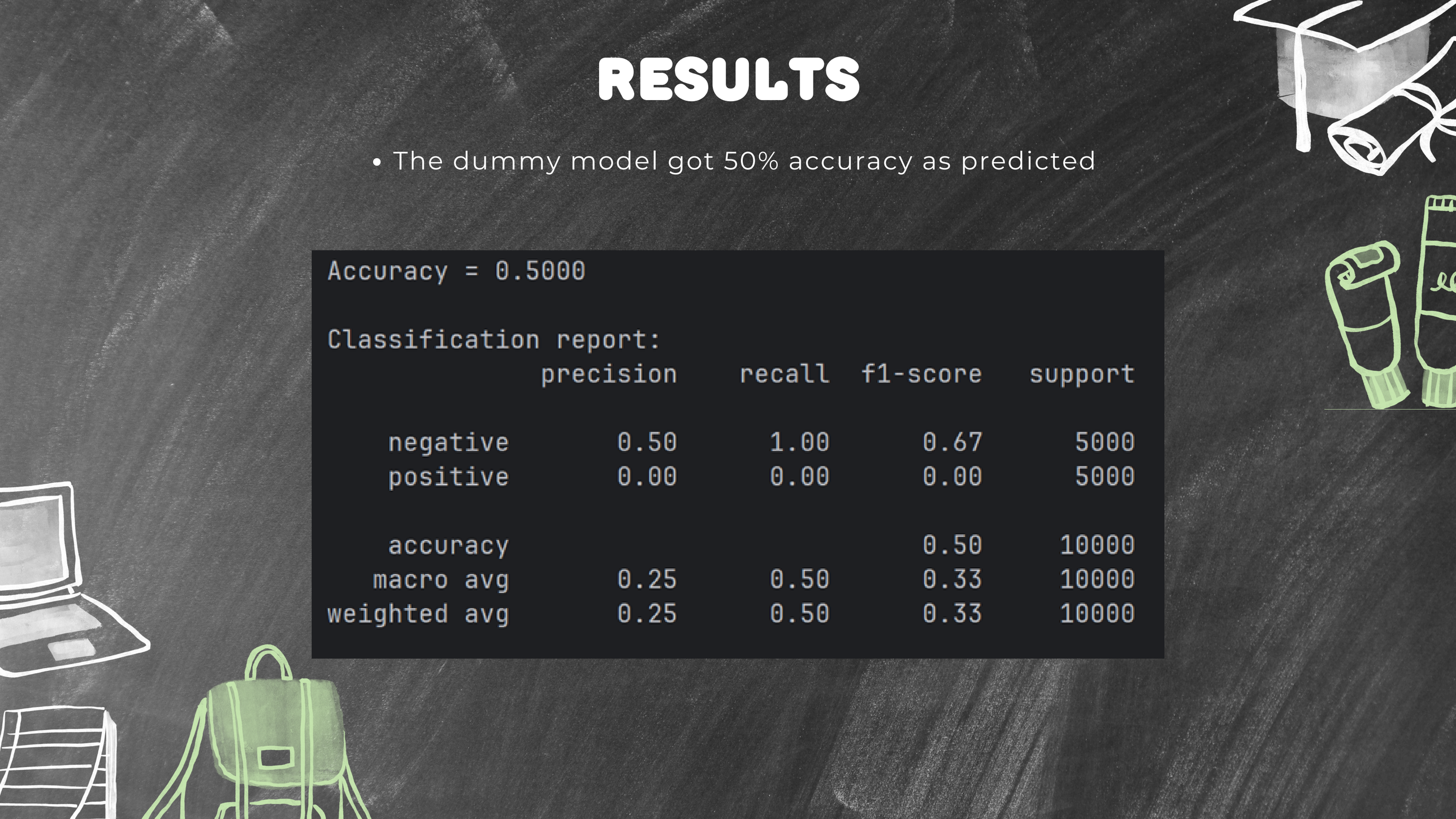
- The dummy model got 50% accuracy as predicted

```
Accuracy = 0.5000

Classification report:
              precision    recall  f1-score   support

    negative       0.50      1.00      0.67      5000
    positive       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
```



RULE BASED MODEL

This time we created 2 lists of “positive” and “negative” words and with 2 counters we checked how many “positive” words and “negative” words we have and return answer based on the majority.

In this model we tried 3 attempt of lists until we got satisfied results , of course we took lists of the same length for “positive” and “negative” words to make it equal

FIRST & SECOND ATTEMPTS

First

positive_words = ["good", "great", "best"
, "love", "liked","perfect", "well", "nice",
“positive"]
negative_words = ["negative", "bad", "worst",
"boring","awful", "waste","hard”, “hate"]

```
=== RuleClassifier Results ===
Accuracy = 0.6771

Classification report:
              precision    recall  f1-score   support

    negative       0.72      0.58      0.64      5000
    positive       0.65      0.77      0.70      5000

    accuracy                           0.68     10000
   macro avg       0.68      0.68      0.67     10000
weighted avg       0.68      0.68      0.67     10000
```

Second

positive_words = ["good", "great", "best", "love",
"liked", "perfect", "well", "nice", "positive",
"wonderful", "excellent", "amazing", "enjoyed"]
negative_words = ["negative", "bad", "worst",
"boring", "awful", "waste" ,"hard" ,"hate", "pointless"
,"terrible", "annoying", "poor", "disappointing"]

```
=== RuleClassifier Results ===
Accuracy = 0.7134

Classification report:
              precision    recall  f1-score   support

    negative       0.76      0.62      0.68      5000
    positive       0.68      0.81      0.74      5000

    accuracy                           0.71     10000
   macro avg       0.72      0.71      0.71     10000
weighted avg       0.72      0.71      0.71     10000
```

THIRD ATTEMPT

In this final attempt we removed all the inculclusive words from the second attempt list and got better results

```
Accuracy = 0.7376

Classification report:
              precision    recall  f1-score   support

    negative       0.74      0.72      0.73      5000
    positive       0.73      0.75      0.74      5000

    accuracy                           0.74     10000
   macro avg       0.74      0.74      0.74     10000
weighted avg       0.74      0.74      0.74     10000
```

LOGISTIC REGRESSION

- init: We first of all used bag of words because the model can receive only numerical feature. To save memory and time we set a limit of 20,000 words in the vocabulary (we took only the most populars)
- presentation: we presented each sample as numerical vector each number tell how many time each word appears in the sentence.
- model: we used logistic regression and converted the labels to 1,0

FIRST & SECOND ATTEMPTS

First

Learning rate = 0.001
num of epochs = 2000

```
Loss: {0.6086071133613586}
Loss: {0.6050587892532349}
Loss: {0.6016834378242493}
Loss: {0.5984599590301514}
Loss: {0.5953720211982727}
Loss: {0.5924065709114075}
Loss: {0.589552640914917}
Loss: {0.5868014693260193}
Loss: {0.5841453075408936}

=== logisticModelClassifier Results ===
Accuracy = 0.7238

Classification report:
              precision    recall  f1-score   support

         0.0       0.71      0.75      0.73      5000
         1.0       0.74      0.70      0.72      5000

    accuracy                           0.72     10000
   macro avg       0.72      0.72      0.72     10000
weighted avg       0.72      0.72      0.72     10000
```

we can see with those parameters the model worser than the previous

Second

Learning rate = 0.001
num of epochs = 20000

```
Loss: {0.48566704988479614}
Loss: {0.47969764471054077}
Loss: {0.47427454590797424}
Loss: {0.46931153535842896}
Loss: {0.46474093198776245}
Loss: {0.4605082869529724}
Loss: {0.4565695822238922}
Loss: {0.4528886675834656}
Loss: {0.44943535327911377}
Loss: {0.4461844265460968}

=== logisticModelClassifier Results ===
Accuracy = 0.8164

Classification report:
              precision    recall  f1-score   support

         0.0       0.82      0.81      0.81      5000
         1.0       0.81      0.83      0.82      5000

    accuracy                           0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.82      0.82      0.82     10000
```

huge upgrade but still not enough

THIRD & FOURTH ATTEMPTS

Third

Learning rate = 0.001
num of epochs = 20000

```
Loss: {0.1938718855581012}
Loss: {0.189641073346138}
Loss: {0.18595729768276215}
Loss: {0.1827068030834198}
Loss: {0.1798073798418045}
Loss: {0.1771974414587021}
Loss: {0.17482970654964447}
Loss: {0.17266711592674255}
Loss: {0.17068006098270416}
Loss: {0.16884467005729675}
Loss: {0.1671413779258728}
Loss: {0.1655540019273758}
Loss: {0.1640690714120865}
Loss: {0.1626751571893692}
Loss: {0.16136261820793152}

=== logisticModelClassifier Results ===
Accuracy = 0.7983

Classification report:
              precision    recall  f1-score   support

         0.0       0.80      0.79      0.80      5000
         1.0       0.79      0.81      0.80      5000

    accuracy                           0.80     10000
   macro avg       0.80      0.80      0.80     10000
weighted avg       0.80      0.80      0.80     10000
```

we changed to MSE but the result was worser so we stay with BCE

Fourth

Learning rate = 0.001
num of epochs = 30000

```
Loss: {0.3233225643634796}
Loss: {0.32097572088241577}
Loss: {0.3187585473060608}
Loss: {0.3166578710079193}
Loss: {0.3146623969078064}
Loss: {0.31276240944862366}
Loss: {0.31094926595687866}
Loss: {0.3092155456542969}
Loss: {0.3075547218322754}
Loss: {0.3059609532356262}
Loss: {0.30442911386489887}

=== logisticModelClassifier Results ===
Accuracy = 0.8631

Classification report:
              precision    recall  f1-score   support

         0.0       0.87      0.86      0.86      5000
         1.0       0.86      0.87      0.86      5000

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000

[RuleClassifier] end
```

OUR Final Model

FULLY CONNECTED NEURAL NETWORK

- init: We first of all used bag of words because the model can receive only numerical feature. To save memory and time we set a limit of 20,000 words in the vocabulary (we took only the most populars)
- presentation: we presented each sample as numerical vector each number tell how many time each word appears in the sentence.
- model: we used MLP model like the example from the class

MODELS

First

self.linear1 = nn.Linear(input_dim, 128)
self.linear2 = nn.Linear(128, 1)
Optimizer = SGD , Num of epoches = 10000,

```
=== FullyConnectedClassifier Results ===
Final Accuracy: 85.45%
--------------------------------------------------------

              precision    recall  f1-score   support

     Class 0       0.81      0.92      0.86      5000
     Class 1       0.91      0.79      0.84      5000

    accuracy                           0.85     10000
   macro avg       0.86      0.85      0.85     10000
weighted avg       0.86      0.85      0.85     10000
```

we can see with those parameters the model worser than the previous

Second

self.linear1 = nn.Linear(input_dim, 128)
self.linear2 = nn.Linear(128, 32)
self.linear3 = nn.Linear(32, 1)
Optimizer = ADAM , num of epochs =20

```
Epoch [15/20] Loss: 0.2216
Epoch [16/20] Loss: 0.2129
Epoch [17/20] Loss: 0.2040
Epoch [18/20] Loss: 0.1973
Epoch [19/20] Loss: 0.1902
Epoch [20/20] Loss: 0.1833

=== FullyConnectedClassifier Results ===
Accuracy = 0.9066

Classification report:
              precision    recall  f1-score   support

         0.0       0.91      0.91      0.91      5000
         1.0       0.91      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
```

we can see with those parameters the model worser than the previous

CONCLUSIONS

	Dummy	Rule-based	Logistic regression	Fully connected Neural Network
ACC	50.00%	73.76%	86.31%	90.66%
Negative precision	50%	74%	87%	91%
Negative recall	100%	72%	86%	91%
Negative F1-score	67%	73%	86%	91%
Positive precision	0%	73%	86%	91%
Positive recall	0%	75%	87%	91%
Positive F1-score	0%	74%	86%	91%

THANK YOU VERY MUCH!