



► PULSEWATCH: Real Time AIOps Anomaly Detection

Project X

Mentees -

Anushree Upasham
Sakshi Kalunge

▶ TABLE OF CONTENTS

- 01 INTRODUCTION
- 02 JOURNEY SO FAR
- 03 MINI PROJECT
- 04 CHALLENGES FACED
- 05 UPCOMING WORKFLOW



01

► INTRODUCTION

WHAT IS ▶ PULSEWATCH ABOUT?

PulseWatch is a cloud-deployed AIOps solution that uses machine learning to detect anomalies in real-time system metrics like CPU, memory, and latency—allowing teams to prevent outages, not just react to them.

- Proactive issue detection
- ML analyses time-series data

Tech companies often face several crashes, the goal is to react before impact.



► APPLICATIONS OF PULSEWATCH



PROACTIVE ISSUE DETECTION

Enable operations teams to identify and resolve performance issues before they impact users.



AUTOMATION

Eliminate manual dashboard monitoring by automating anomaly detection and alerting.



SCALABLE OPERATIONS

Support large scale systems with a cloud native, containerized architecture.

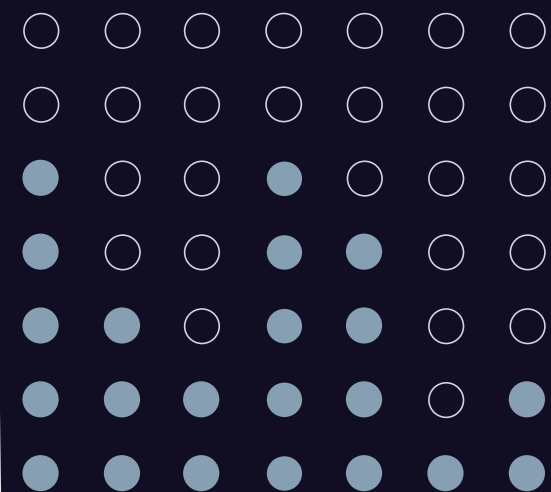


DevOps EFFICIENCY

Streamline development and deployment with CI/CD pipelines, reducing operational overhead.


02

► JOURNEY SO FAR





► INTRODUCTION TO NEURAL NETWORKS

- Here we learnt basic concepts of neural networks like perceptrons, error functions, activation functions, Logistic Regression, and Gradient Descent.
 - We learnt FeedForward and Backpropagation to train the model.
 - Lastly in this module we learnt concepts like Overfitting and Dropout.
- 



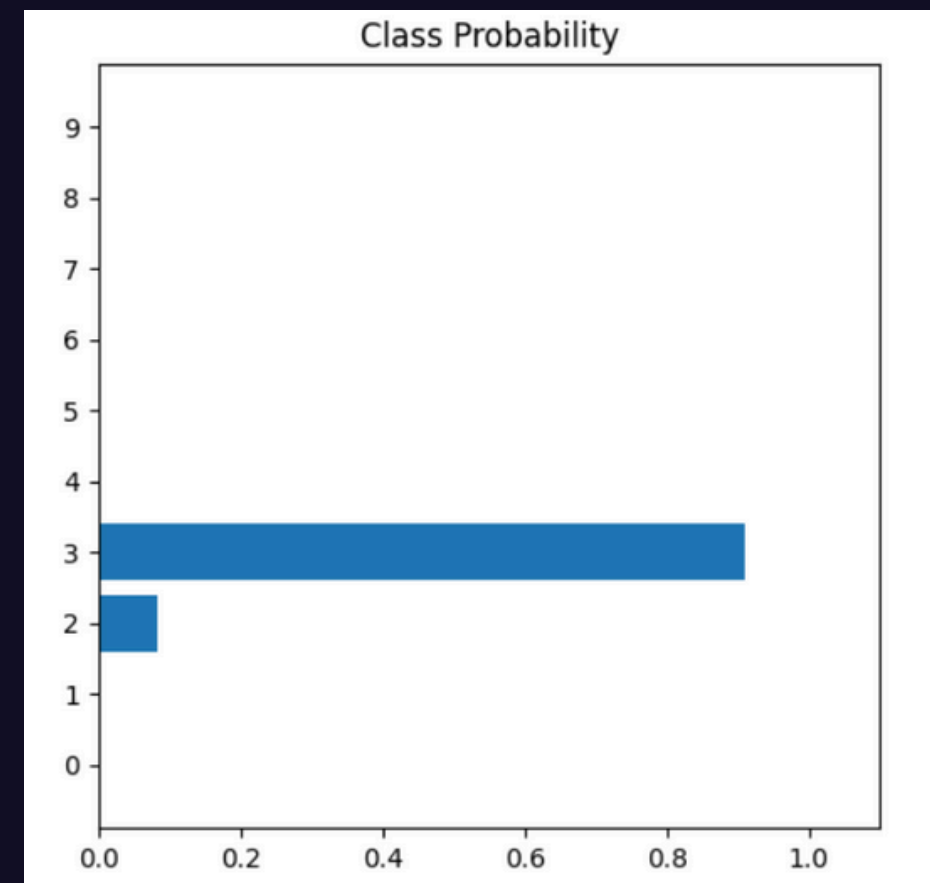
▶ INTRODUCTION TO PYTORCH

- We learnt to create a single layer neural network architecture and then proceeded to multi-layer.
- Learnt how to train models.
- Studied on various Multi-Class MNIST datasets.
- Implementation of Dropout and Validation for our model's test accuracy.



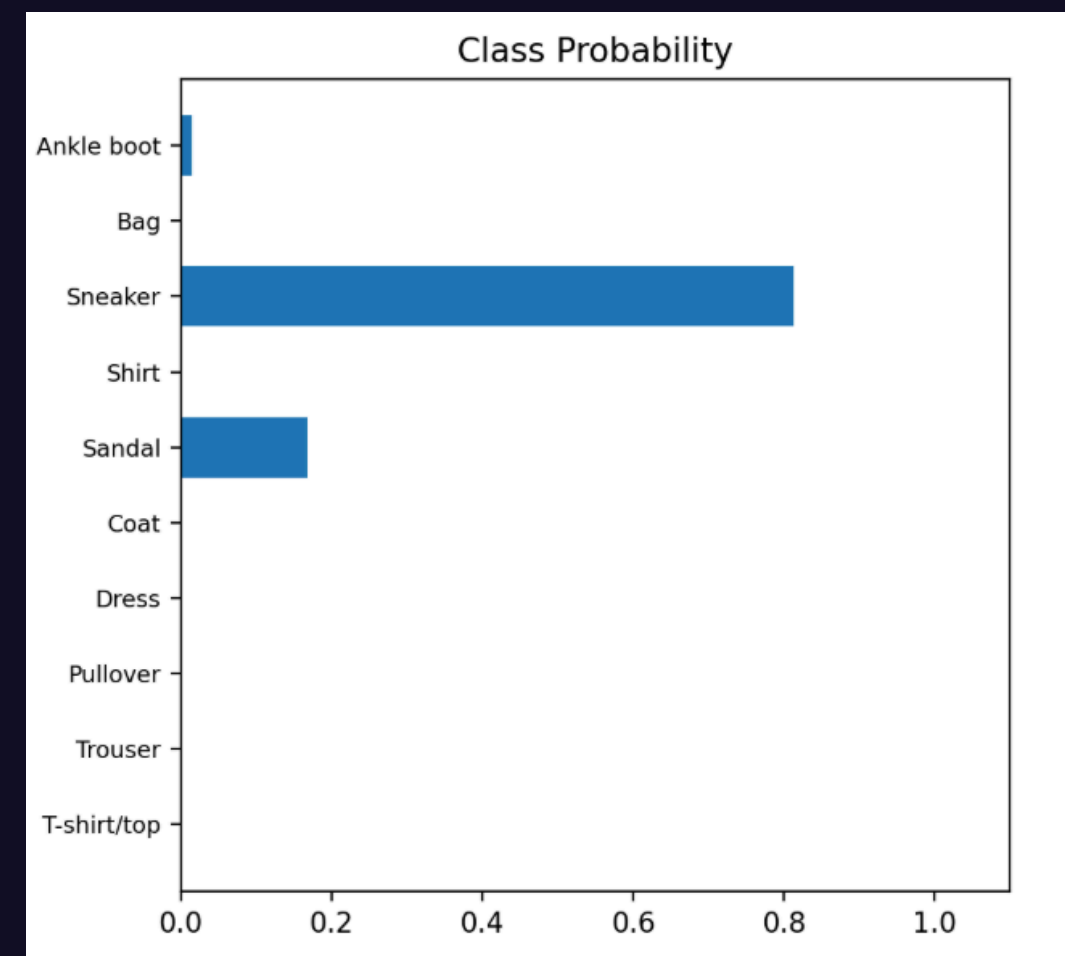
► MNIST Digit Classifier

We implemented an MNIST digit classifier as part of a course exercise, using PyTorch to train a neural network on the handwritten digit dataset.



► Fashion MNIST Classifier

We also implemented a Fashion MNIST image classifier to train a neural network on the clothing image dataset."





03

► MINI PROJECT

Step 1: Dataset and anomalies

We found a small dataset of 1000 timestamps with around 30% anomalies

```
timestamp      value  is_anomaly
0  2025-08-07 00:00:00  53.528105      0
1  2025-08-07 00:00:01  50.800314      0
2  2025-08-07 00:00:02  51.957476      0
3  2025-08-07 00:00:03  54.481786      0
4  2025-08-07 00:00:04  53.735116      0
..  ...
995 2025-08-07 00:16:35  50.825742      0
996 2025-08-07 00:16:36  49.603202      0
997 2025-08-07 00:16:37  50.188385      0
998 2025-08-07 00:16:38  47.704778      0
999 2025-08-07 00:16:39  67.554410      1
[1000 rows x 3 columns]
```

Step 2: Preprocessing the data

- We split the data in the ratio of 80:20 for training and testing respectively.
- Then we created tensors for values and labels in training and testing datasets.
- Lastly we created a trainloader and testloader with appropriate batch size

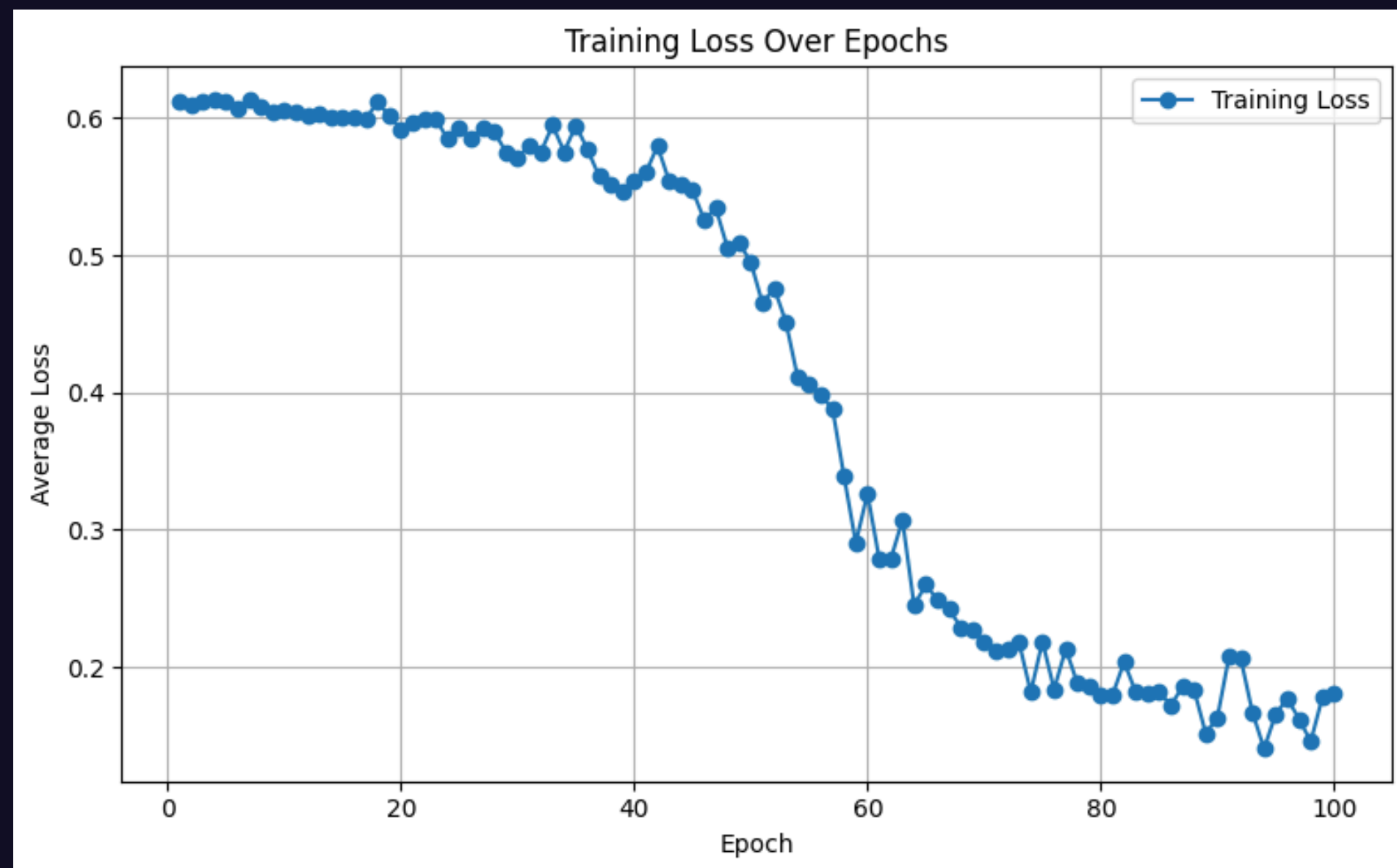
Step 3: Model Architecture

- We created a model with 3 hidden layers. The activation function used is ReLU and the dropout rate is 0.3.
- The loss function used is Binary Cross Entropy Loss
- We defined a learning rate of 0.003.

Step 4: Training the model

The model trained itself over 150 epochs.

```
Epoch 135/150, Loss: 0.0353
Epoch 136/150, Loss: 0.0247
Epoch 137/150, Loss: 0.0353
Epoch 138/150, Loss: 0.0291
Epoch 139/150, Loss: 0.0282
Epoch 140/150, Loss: 0.0338
Epoch 141/150, Loss: 0.0699
Epoch 142/150, Loss: 0.0364
Epoch 143/150, Loss: 0.0344
Epoch 144/150, Loss: 0.0245
Epoch 145/150, Loss: 0.0306
Epoch 146/150, Loss: 0.0275
Epoch 147/150, Loss: 0.0449
Epoch 148/150, Loss: 0.0396
Epoch 149/150, Loss: 0.0344
Epoch 150/150, Loss: 0.0239
```



Step 5: Testing the model with Real-Time Data

- Now we feed the model our test data over the time period of 1 second and get real time predictions.

The model achieved an accuracy of 99.50% on the test data.

```
tensor([0.])  
tensor([0.])  
tensor([0.])  
tensor([1.])  
tensor([0.])  
tensor([0.])  
tensor([0.])  
tensor([0.])  
tensor([0.])  
tensor([0.])  
tensor([1.])  
tensor([1.])  
tensor([0.])  
tensor([0.])  
tensor([1.])  
tensor([0.])  
tensor([0.])  
tensor([0.])  
Test Accuracy: 99.50%
```


Step 6: Evaluating the Model

The model was evaluated based on certain metrics such as Confusion Matrix, Precision, Recall and F1 score.

```
Confusion Matrix:
```

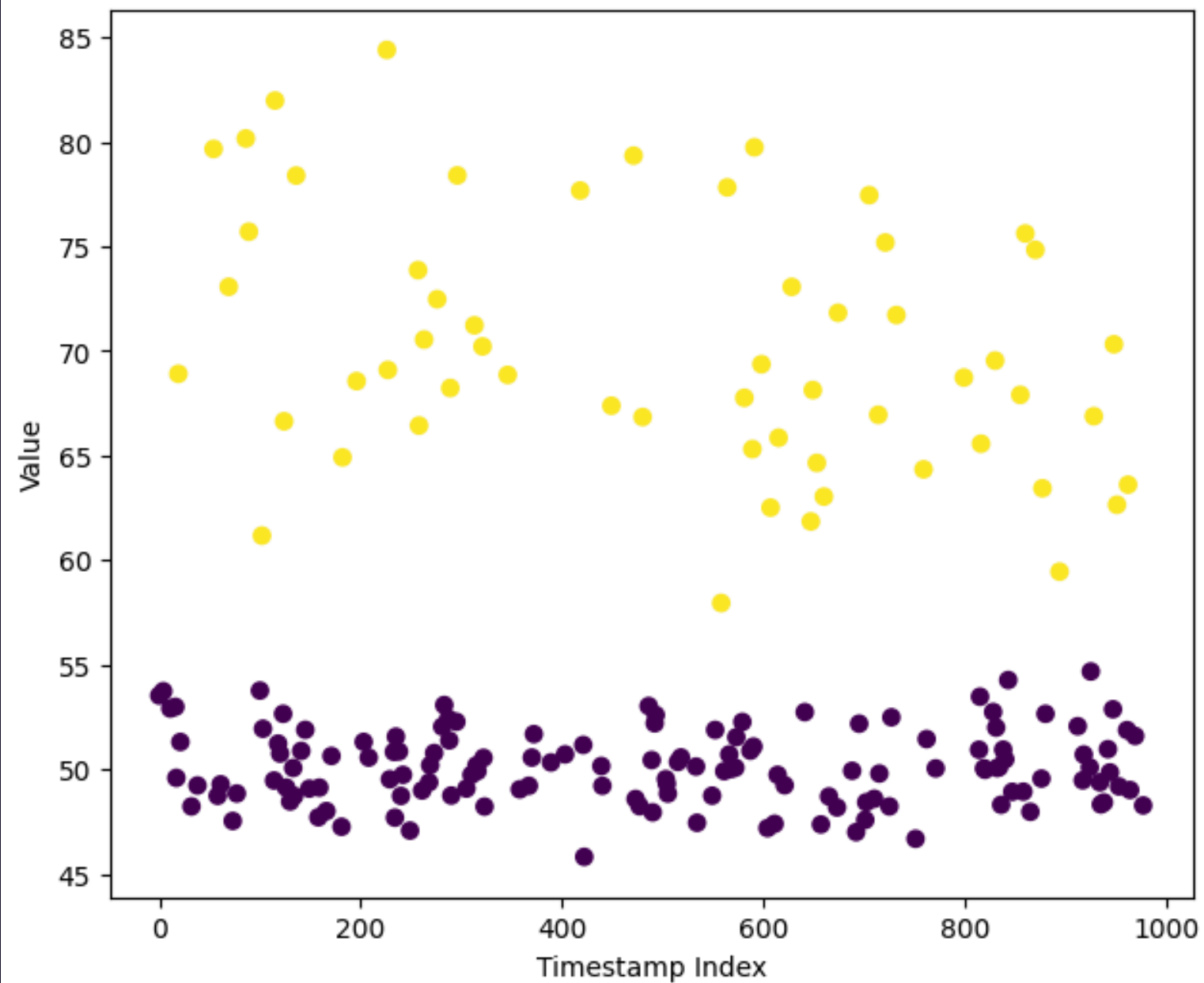
```
[[143  0]
 [  1 56]]
```

```
Classification Report:
```

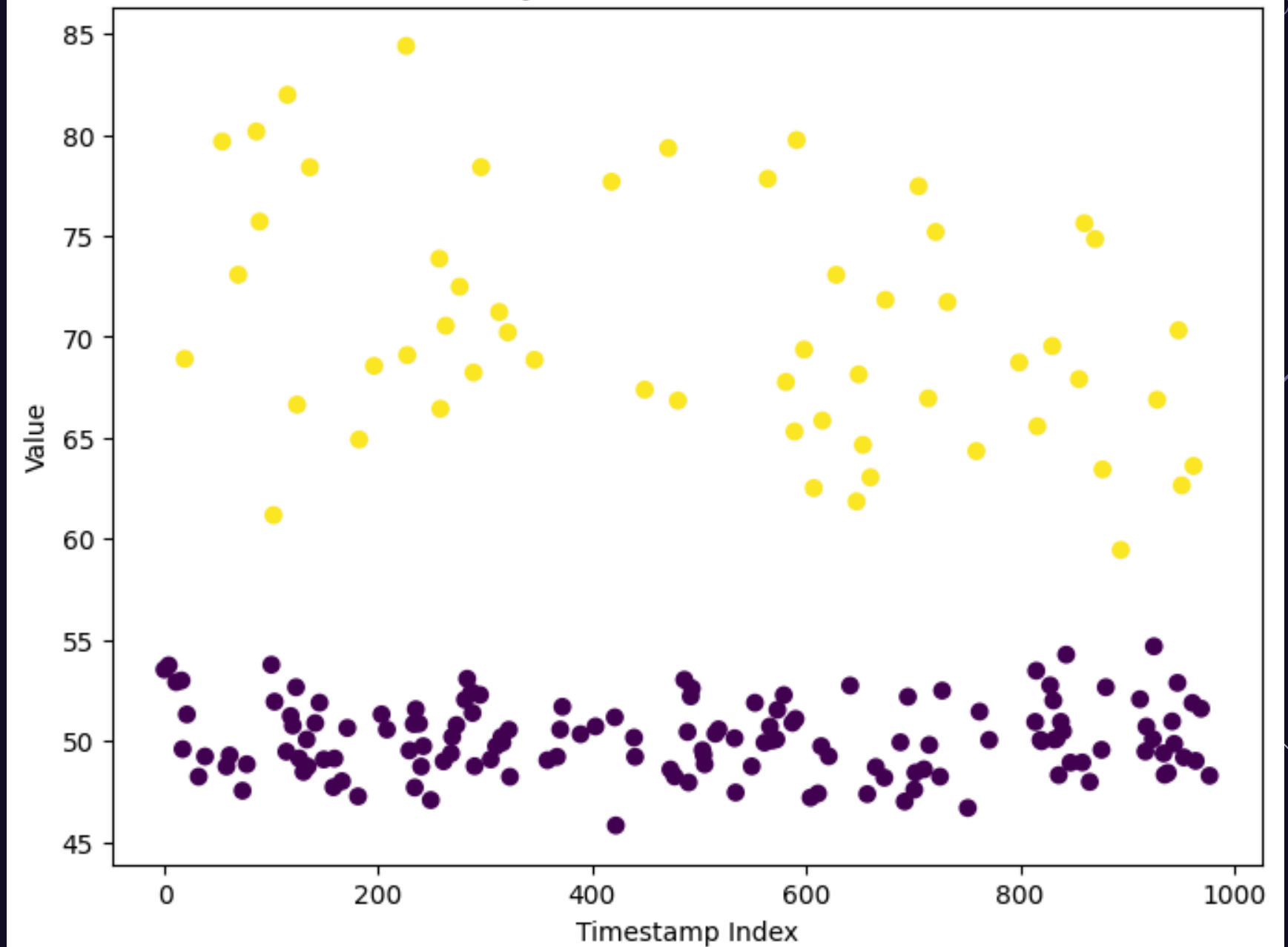
	precision	recall	f1-score
0.0	0.9931	1.0000	0.9965
1.0	1.0000	0.9825	0.9912

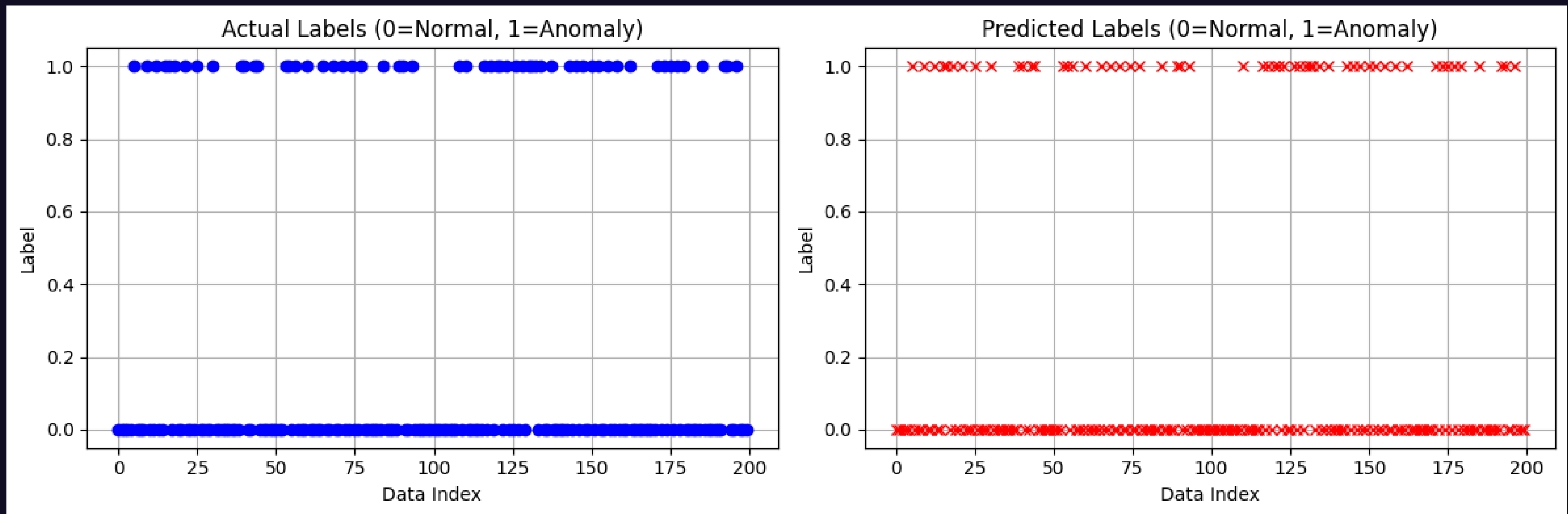
Step 7: Visualising the data

Actual Anomalies (Test Data)



Correctly Predicted Points (Test Data)







04



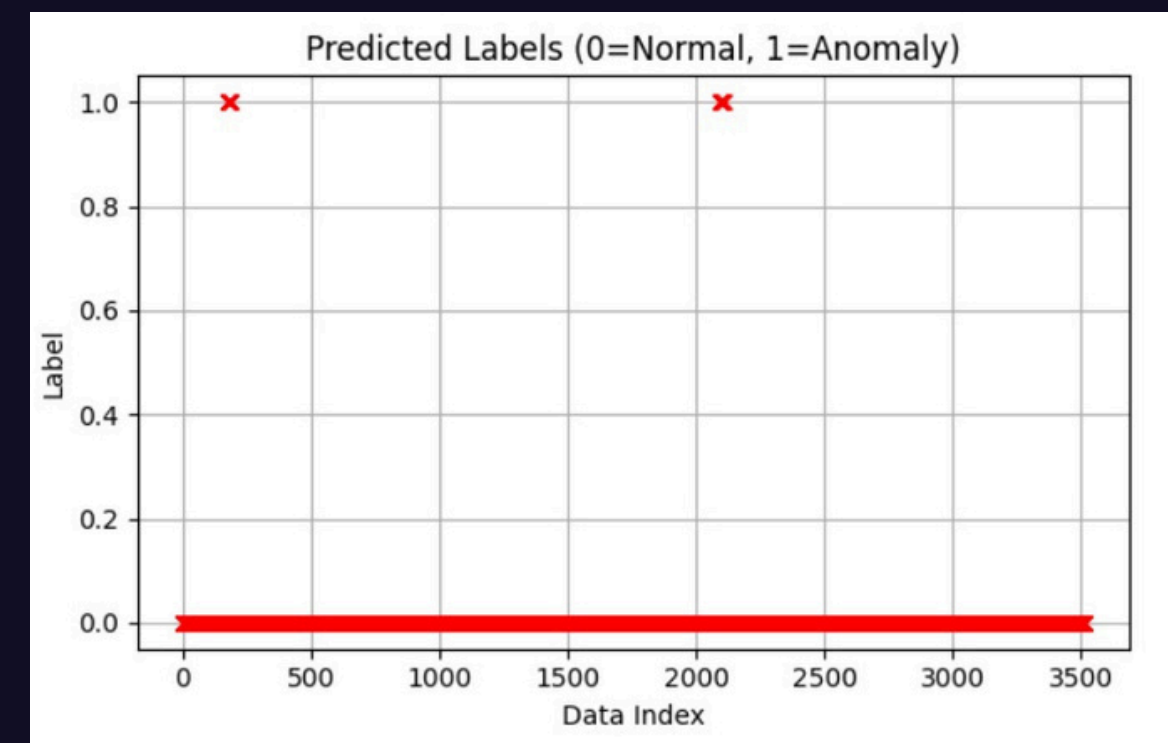
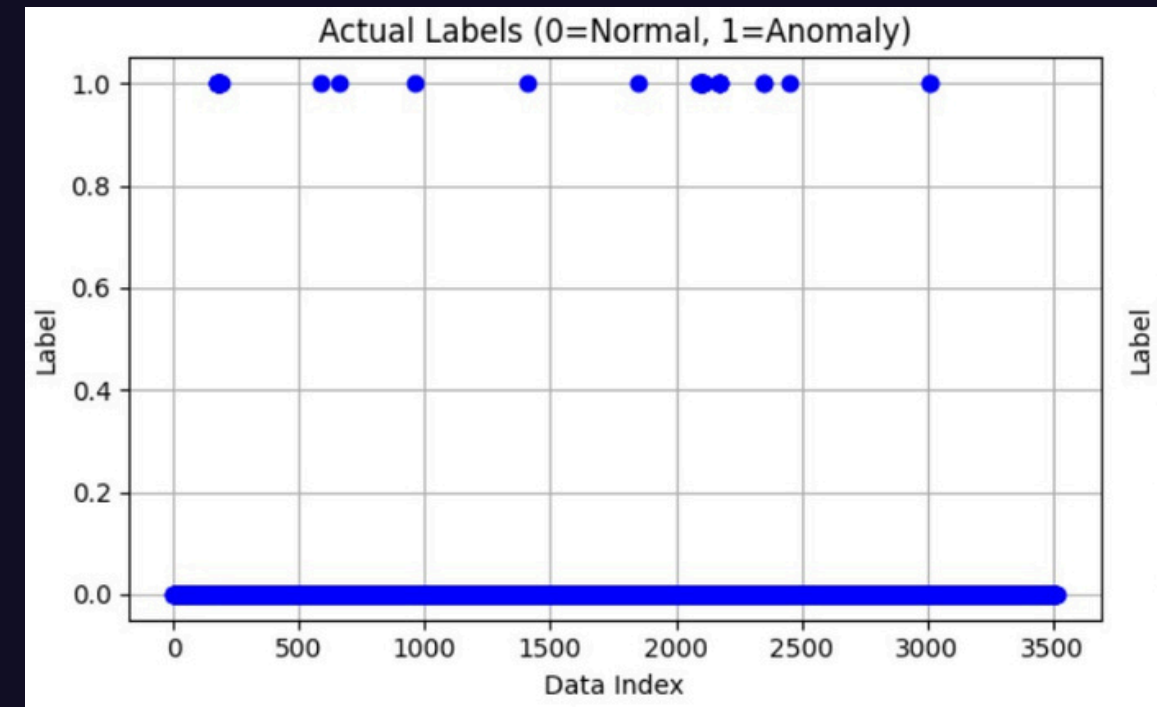
CHALLENGES FACED

► Challenges faced

1. Imbalanced Dataset

2. Visualizing the data

3. Illusion of Accuracy



05

► UPCOMING WORKFLOW

► Future Workflow :

1. Learn about Sequential models like RNNs, LSTMs, GRUs, etc to address time series input which improves our model for real world data
2. We will create a API endpoint by wrapping our model in FASTAPI
3. Cloud Deployment :
The final model might be too heavy and computationally expensive to infer locally, so we will go for a cloud-based deployment

THANK YOU....

Mentors:

1.Om Mukherjee

2.Khush Agrawal

3.Ninad Shegaokar

