# Next Number Prediction Model using LSTM

## MongoDB Schema

The project uses MongoDB as the primary database to store input historical data, model predictions, and evaluation metrics. The schema is structured as follows:

* **Database Name: Byner\_DB**

Collections used in this project:

* **Sequence\_Dataset (Historical Data Collection)**

This collection stores the extracted sequence of numbers and their corresponding timestamps.

* **Document Structure:**

{

"\_id": ObjectId("Unique\_ID"),

"date": "YYYY-MM-DD",

"time": "HH:MM:SS",

"feature": 7

}

* \_id: Auto-generated unique identifier for each document.
* date: The date when the number was scraped.
* time: The exact time of extraction.
* feature: The numerical value in the extracted sequence.

## predicted\_values (Predictions Collection)

This collection stores the model’s predicted outputs alongside their corresponding input sequences.

**Document Structure:**

{

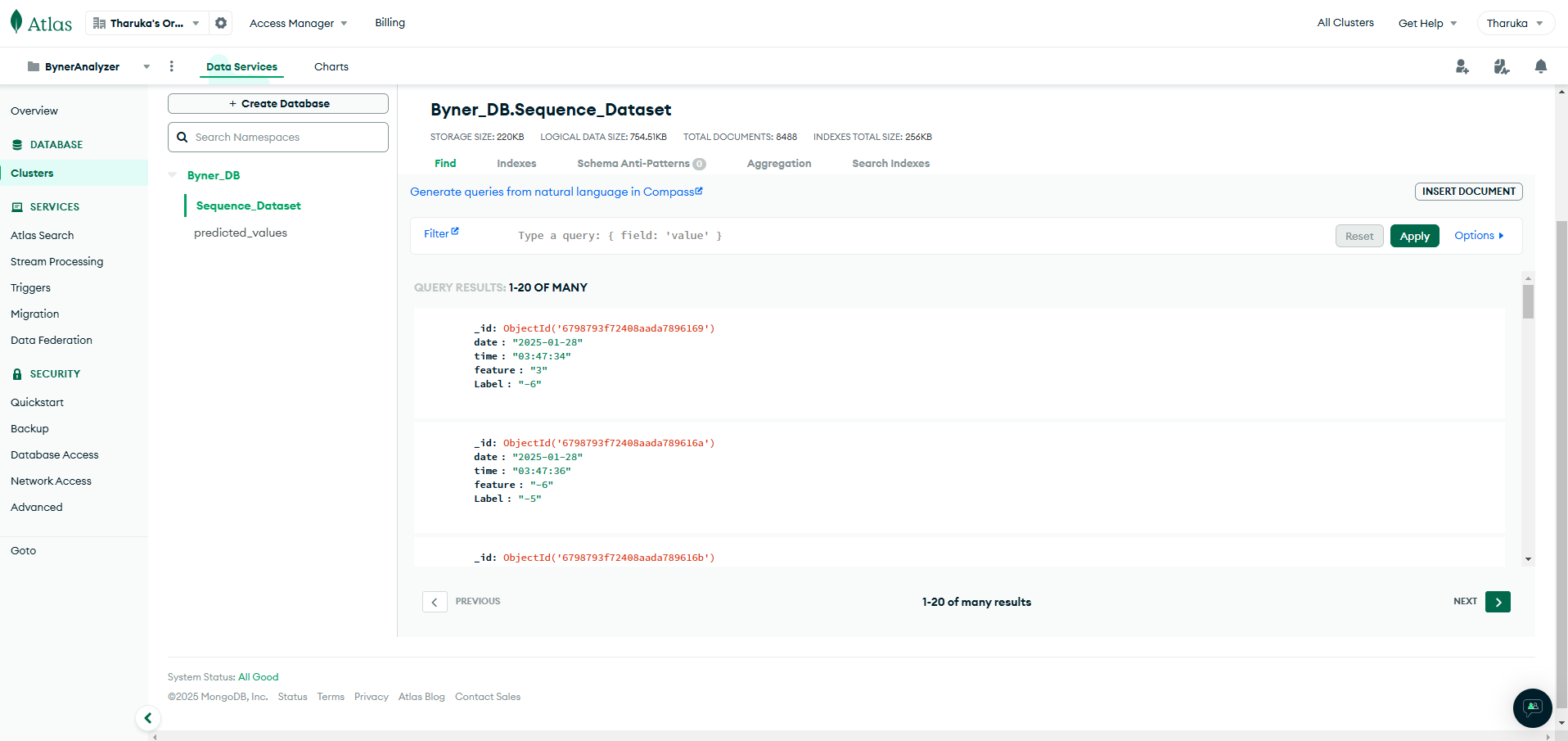
"\_id": ObjectId("Unique\_ID"),

"X\_test\_values": "[1, 2, 3, 4, 5, 6, 7]",

"Predicticted\_Value": 8

}

* \_id: Auto-generated unique identifier.
* X\_test\_values: The sequence of numbers used as input to the model for prediction.
* Predicticted\_Value: The predicted next number in the sequence.



## Steps to Set Up the MongoDB Database

1. **Step 1: Install MongoDB**

MongoDB must be installed locally or can be used via a cloud-based service like MongoDB Atlas. To install MongoDB locally:

* Download MongoDB from MongoDB Official Website.
* Install it by following the instructions for your OS.

1. **Step 2: Create a MongoDB Atlas Cluster (Optional for Cloud Storage)**

* Go to MongoDB Atlas.
* Sign up and create a new free-tier cluster.
* In the database settings, create a new database named Byner\_DB.
* Add two collections: Sequence\_Dataset and predicted\_values.
* Obtain the MongoDB connection string

(e.g., mongodb+srv://username:password@cluster.mongodb.net/Byner\_DB)

1. **Step 3: Install the Required Python Library**

To interact with MongoDB, install the pymongo library:

**pip install pymongo**

1. **Step 4: Establish a Connection in Python**

In Python script, establish a connection using pymongo:

**import pymongo**

**def Get\_Connection():**

**myclient = pymongo.MongoClient("mongodb+srv://username:password@cluster.mongodb.net/?retryWrites=true&w=majority")**

**try:**

**myclient.admin.command('ping')**

**print("Successfully connected to MongoDB")**

**return myclient**

**except Exception as e:**

**print("Connection failed:", e)**

**# Connect to the database**

**myclient = Get\_Connection()**

**DB = myclient['Byner\_DB']**

1. **Step 5: Insert Data into MongoDB**

After scraping, insert data into MongoDB:

**def Insert\_Data(dataframe):**

**DB = myclient['Byner\_DB']**

**collection = DB.Sequence\_Dataset**

**records = dataframe.to\_dict(orient='records')**

**collection.insert\_many(records)**

**print("Data Inserted Successfully")**

## Model Architecture and Reasoning

1. **Model Architecture**

The model is designed using Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN) specialized for sequence prediction.

Layers Used:

* **LSTM (64 units, relu activation, return sequences=True)**

Captures long-term dependencies in the sequence.

* **Dropout (0.2)**

Prevents overfitting by randomly disabling neurons.

* **LSTM (32 units, relu activation, return sequences=True)**

Extracts deeper sequential patterns.

* **LSTM (16 units, relu activation, return\_sequences=False)**

Final processing of sequential dependencies.

* **Dense (1 unit, linear activation)**

Outputs the predicted next number.

1. **Model Summary:**

**model = Sequential([**

**LSTM(64, activation='relu', return\_sequences=True, input\_shape=(seq\_len, 1)),**

**Dropout(0.2),**

**LSTM(32, activation='relu', return\_sequences=True),**

**LSTM(16, activation='relu', return\_sequences=False),**

**Dense(1)**

**])**

**model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mae'])**

## Reasoning for Using LSTM

* LSTMs handle long-term dependencies better than traditional RNNs.
* They effectively learn patterns in numerical sequences, making them ideal for next-number prediction.
* Unlike simple regression models, LSTM networks adapt to changing trends over time.

## Model Training & Evaluation

The model is trained using Mean Squared Error (MSE) as the loss function and Mean Absolute Error (MAE) as an evaluation metric:

**history = model.fit(X\_train, y\_train, epochs=50, batch\_size=10, validation\_data=(X\_test, y\_test), verbose=1)**

## Visualization of Model Performance

To analyze performance, a loss comparison is plotted:

**import matplotlib.pyplot as plt**

**plt.plot(history.history['loss'], label='Training Loss')**

**plt.plot(history.history['val\_loss'], label='Validation Loss')**

**plt.title("Loss Over Epochs")**

**plt.xlabel("Epochs")**

**plt.ylabel("Loss")**

**plt.legend()**

**plt.show()**

## Conclusion

This project successfully implements an LSTM-based next-number prediction model. MongoDB is used to store both raw input sequences and model predictions. The model learns patterns from historical data and provides predictions that can be validated using visualization techniques.