DML Group No: 18

Group Members Names:

1. PEYALA SAMARASIMHA REDDY - 2023AA05072
2. ANIRUDDHA DILIP MADURWAR - 2023AA05982
3. PEGALLAPATI SAI MAHARSHI - 2023AA05924
4. TUSHAR DEEP - 2023AA05885

Assignment - 2 Report

Real-Time Prediction with Apache Kafka and Neural Networks

Neural Network Model (Training Separately)

1. Problem Statement

We took a common machine learning problem for this assignment, i.e to **predict California house prices** using a **trained neural network model** and implement a **real-time streaming system using Apache Kafka**. The system consists of three main components:

- **A Producer** that streams input data to Kafka.
- **A Consumer** that listens to the Kafka topic, makes predictions using a trained neural network, and publishes results.
- **Evaluation & Monitoring** to assess system performance based on Mean Squared Error (MSE), Mean Absolute Error (MAE), and latency metrics.

2. Dataset: California Housing Prices

We used the **California housing dataset (`california_housing.csv`)**, which contains **20,640 rows** and **9 columns**:

- **Features (8):** `MedInc`, `HouseAge`, `AveRooms`, `AveBedrms`, `Population`, `AveOccup`, `Latitude`, `Longitude`
- **Target (1):** `Target` (actual house price)

2.1. Data Preprocessing & Splitting

- We **split** the dataset into **80% training (16,512 rows) and 20% test data (4,128 rows)**.
- We **standardized the feature values** using **`StandardScaler`** for better model performance.
- We **saved the scalar object (`scaler.pkl`)** for use in the consumer process.

3. Training the Neural Network

3.1. Model Architecture

We trained a **feedforward neural network** using **TensorFlow/Keras** with the following structure:

- **Input Layer:** 8 neurons (one for each feature)
- **Hidden Layers:** 2 layers with **64 neurons each**, activation function: **ReLU**
- **Output Layer:** 1 neuron (predicting the house price), activation function: **linear**

3.2. Training Process

- **Optimizer:** Adam
- **Loss Function:** Mean Squared Error (MSE)
- **Epochs:** 50
- **Batch Size:** 32
- **Evaluation Metric:** Mean Absolute Error (MAE)

After training, we **saved the model (`california_housing_model.h5`)** for real-time predictions.

Tasks and Requirements

1. Environment Setup

To facilitate **real-time streaming and processing**, we used **Apache Kafka**. We setup the Apache Kafka locally into our system. After configuring and installing all the requirements and dependencies, we ran Kafka. For this, the setup included:

1.1. Installing and Running Kafka

Step 1: Start Zookeeper (required for Kafka)

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

Step 2: Start Kafka Broker

```
bin/kafka-server-start.sh config/server.properties
```

1.2. Creating Kafka Topics

We created two Kafka topics:

1. **input-data** → Receives input feature data from the producer.
2. **predictions** → Stores the predicted house prices from the consumer.

```
bin/kafka-topics.sh --create --topic input-data --bootstrap-server
localhost:9092

bin/kafka-topics.sh --create --topic predictions --bootstrap-server
localhost:9092
```

---

2. Producer Implementation

The **Kafka Producer** is responsible for sending **real-time house feature data** to the **input-data** Kafka topic.

2.1. Data Generation & Preprocessing

- The producer loads the **last 100 rows** of the `california_housing.csv` dataset for real-time simulation.
- Each row represents **house features** such as `MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude`.
- The **Target value** (house price) is also included for evaluation.

2.2. Kafka Message Structure

Each message contains:

- **A unique ID**

- **Timestamp of the message**
- **Feature values (list format)**
- **Actual house price (for later comparison with predictions)**

2.3. Streaming to Kafka

- The producer **sends data row by row** to the `input-data` topic every second to simulate real-time data.
- **Logs** are maintained in `producer_logs.log` for monitoring.

2.4. Monitoring Producer Messages

To ensure that messages were sent, we monitored the producer topic using:

```
bin/kafka-console-consumer.sh --topic input-data --from-beginning
--bootstrap-server localhost:9092
```

---

3. Consumer Implementation with Neural Network Prediction

The **Kafka Consumer** reads messages from `input-data`, processes them, makes predictions using the neural network model, and sends the results to `predictions`.

3.1. Data Ingestion & Preprocessing

- The consumer **listens** for new messages from `input-data`.
- It extracts **features** from incoming messages.
- The **scaler.pkl** file is used to **standardize features** before prediction.

3.2. Neural Network Model Integration

- The **pre-trained neural network** model (`california_housing_model.h5`) is loaded.
- It takes the **preprocessed input features** and predicts the **house price**.

3.3. Output Handling

- The predicted price, along with the actual price and timestamp, is sent to the `predictions` topic.

3.4. Monitoring Consumer Messages

We verified that the consumer was correctly processing messages using:

```
bin/kafka-console-consumer.sh --topic predictions --from-beginning
--bootstrap-server localhost:9092
```

---

4. Testing and Evaluation

To ensure the **accuracy and efficiency** of our system, we evaluated:

4.1. Performance Metrics

Since house price prediction is a **regression task**, we used:

(A) Mean Squared Error (MSE)

- Measures the **average squared difference** between actual and predicted values.
- **Lower MSE = better performance.**

(B) Mean Absolute Error (MAE)

- Measures the **absolute difference** between actual and predicted values.
- **Lower MAE = more precise predictions.**

(C) Latency Measurement

- We tracked **how long** it took for each message to be processed.
- **Lower latency = faster predictions.**

4.2. Evaluation Process

- Every **10 messages**, we **computed and logged** MSE, MAE, and average latency.
- These were stored in `consumer_logs.log` for **real-time monitoring**.

4.3. Evaluation Results

Since for easy to understand and simplicity, out of 20% of the test data, we only took the last 100 rows of data to send to the consumer. For each row i.e for each message we computed the predicted price and for every 10 messages or 10 rows, we computed the MSE, MAE and average latency values. All the results are preserved in the consumer, producer jupyter notebook files.

5. Real-Time Monitoring & Logging

To track **system performance over time**, we implemented **logging** for both the **Producer and Consumer**.

5.1. Producer Logging

Stored in `producer_logs.log`, containing:

- **Message ID**
- **Timestamp**
- **Feature values**
- **Actual house price**

Example Log Entry (Producer)

```
2025-03-16 18:07:02,627 - INFO - Sent Message ID: 1, Timestamp:
2025-03-16 18:07:02, Features: [4.6225, 13.0, 6.115695067264574,
1.0385650224215246, 2828.0, 2.536322869955157, 38.54, -121.7], Actual
Price: 2.2650
```

5.2. Consumer Logging

Stored in `consumer_logs.log`, containing:

- **Message ID**
- **Actual price vs. Predicted price**
- **Latency**

Example Log Entry (Consumer)

```
2025-03-16 18:07:02,670 - Processed Message: ID: 1, Timestamp:
2025-03-16 18:07:02, Features: {'MedInc': 4.6225, 'HouseAge': 13.0,
'AveRooms': 6.115695067264574, 'AveBedrms': 1.0385650224215246,
'Population': 2828.0, 'AveOccup': 2.536322869955157, 'Latitude':
38.54, 'Longitude': -121.7}, Actual Price: 2.265, Predicted Price:
1.80, Latency: 0.0409 sec
```

5.3. Performance Metrics Logging

In the same consumer_logs.log file, we also logged the performance metrics values too for every 10 messages (for every ten rows) and overall data too.

6. Conclusion

- Successfully implemented a Kafka-based real-time house price prediction system.
- Used a pre-trained neural network to process and predict house prices.
- Evaluated system performance using MSE, MAE, and latency.
- Implemented real-time logging for monitoring producer and consumer performance.
- All files are attached

7. Total File List

- california_housing.csv
- california_housing_model.h5
- scalar.pkl
- DML_Assignment2_Group18_producer.ipynb
- DML_Assignment2_Group18_consumer.ipynb
- DML_Assignment2_Group18_Report.pdf
- DML_Assignment2_Group18_Model_Training.ipynb
- consumer_logs.log
- producer_logs.log