# House Price Prediction Application

## ➢ Kafka and Docker Setup with Producer and Consumer.

This repository provides a simple setup for running Kafka and ZooKeeper using Docker, along with example Python scripts for a Kafka producer and consumer to demonstrate streaming data and model predictions. The producer sends data from a CSV file to a Kafka topic, and the consumer receives the data, makes predictions using a pre-trained machine learning model, and prints the results.

- **Prerequisites**:
  Make sure you have the following installed:
  Docker (https://docs.docker.com/get-docker/)
  Docker Compose (https://docs.docker.com/compose/install/)
- **Setup:**
  Put your pre-trained machine learning model (model.pkl) and the dataset file (test.csv) in the same directory as the consumer.py script.
  Adjust the required columns in the consumer.py script to match the features used during model training.
  Create a Dockerfile with the required dependencies for running the producer and consumer Python scripts.
- **Running Kafka and ZooKeeper**
  To start Kafka and ZooKeeper using Docker Compose, run the following command:
  ➔ docker-compose up -d
  This will start the Kafka and ZooKeeper containers in the background.
- **Creating Kafka Topic:**

  After starting Kafka and ZooKeeper, you can create a Kafka topic using the kafka-topics.sh script. To create a topic named "my_topic" with one partition and a replication factor of one, run the following command:
  ➔ docker exec -it kafka /bin/bash
  ➔ cd /opt/kafka/bin

➔ ./kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic my_topic

## ➢ Running the Producer

To send data from the test.csv file to the Kafka topic, run the following command:

➔ python producer.py

This will read data from the CSV file and send each row as a message to the Kafka topic**.**

## ➢ Running the Consumer

To consume messages from the Kafka topic, make predictions using the pre-trained model, and print the results, run the following command:

➔ python consumer.py

The consumer script will listen for messages from the Kafka topic and print the predicted sale prices based on the received data.

## ➢ Stopping Kafka and ZooKeeper

To stop the Kafka and ZooKeeper containers, run the following command:

➔ docker-compose down

This will stop and remove the containers created by Docker Compose.

**Conclusion:** With this setup, you can stream data to Kafka, process it with a consumer application, and make predictions using a pre-trained machine learning model. It's a basic example of how to integrate Kafka, Docker, and Python for data streaming and real-time processing.