Project Title: E-Commerce Product Recommendation and Review Prediction System

Problem Statement:

In the rapidly growing e-commerce industry, personalized recommendations and review predictions play a crucial role in enhancing customer experience. This project aims to develop a machine learning-based system that provides personalized product recommendations based on customer ID using KNN and predicts product review scores based on product ID. By leveraging historical data, the system helps improve customer engagement, boosts sales, and enhances decision-making for both customers and businesses.

Insights from the Data:

- Identified trends in customer purchasing behavior based on product categories and ratings.
- Detected the impact of review scores on customer purchase decisions.
- Discovered correlations between product demand and seasonal trends.
- Analyzed sentiment from customer reviews to enhance product recommendations.

📌 Workflow of the Project

1️⃣ Data Ingestion & Storage (Big Data Handling)

- **Dataset:** Historical e-commerce sales data (**structured CSV files**).
- **Key Features Used for Review Prediction:**
  - **Product ID** (to understand product-specific trends).
  - **Price** (higher/lower prices influencing ratings).
  - **Freight Value** (shipping cost impact on customer satisfaction).
- **Data Source:**
  - Raw data stored in **AWS S3 (Raw Data Bucket).**
  - Used **AWS Glue Crawler** to catalog and structure data.
- **Challenge:**
  - Large dataset processing with Pandas was slow.
- **Solution:**
  - Used **AWS Glue & PySpark** for distributed processing.

2️⃣ Data Preprocessing & Transformation (ETL & Feature Engineering)

- **Challenges Faced:**
  - Missing values in **freight_value & price**.
  - Some products had **skewed review distribution** (imbalanced data).
- **Solution:**
  - **AWS Glue ETL pipeline:**
    - **Filled missing values** using mean imputation for price & freight value.
    - **Removed outliers** using IQR (Interquartile Range) method.
    - **Created additional features:**
      - **Price-to-Freight Ratio:** To capture impact on review scores.
      - **Product Popularity Index:** Based on sales frequency.
  - Processed data stored in **AWS S3 (Processed Data Bucket).**

SageMaker Workflow

1. **Data Preparation**
   - **Source:** Processed data is fetched from the **AWS S3 bucket** (Processed Data Bucket).
   - **Tool:** Use **Boto3** to interact with S3 for accessing the processed data.
   - **Action:** Load data into SageMaker and prepare it for training.
2. **Model Training**
   - **Model Selection:** Trained models using **XGBoost** for review prediction.
   - **Tool:** Use **Boto3** to interact with **SageMaker** for initiating the training job.
   - **Action:**
     - **Boto3** starts the training job with the processed dataset.
     - **Training:** Use **SageMaker's built-in algorithm for XGBoost** and specify hyperparameters.
3. **Hyperparameter Tuning**
   - **Tuning:** Used **SageMaker's automatic hyperparameter tuning** to optimize the model.
   - **Tool: Boto3** was used to start and monitor the hyperparameter tuning job.
4. **Model Evaluation**
   - **Evaluation Metrics:** Accuracy, F1-score, precision, and recall.
   - **Tool:** Use **Boto3** to track model performance and retrieve the best model based on evaluation metrics.
5. **Model Deployment**
   - **Model Saving:** After training, the **XGBoost model** was saved as a `.pkl` file in an **S3 bucket**.

3️⃣ Machine Learning Pipeline (Model Training & Optimization)

- **Task 1: Review Prediction (Classifying customer review ratings)**
- **Models Tested:**
  - **Logistic Regression:** Poor accuracy (~60%) due to high bias.

    - **Decision Tree:** Overfit on training data (~70% accuracy but poor generalization).
    - **Random Forest:** Improved accuracy (~78%) but computationally expensive.
    - **XGBoost (Final Model):** Best performance (~85% accuracy after hyperparameter tuning).
- **Training Process:**
    - Used **AWS SageMaker** for training & hyperparameter tuning.
    - Stored **XGBoost trained model as a .pkl file** in AWS S3.

Why XGBoost?

- Handles missing data efficiently.
- Performs well on tabular structured data.
- Optimized training speed with AWS SageMaker tuning.

4️⃣ Product Recommendation (KNN-Based Approach)

- **Challenge:**
    - Collaborative filtering didn't work well due to sparse data.
- **Solution:**
    - Used **K-Nearest Neighbors (KNN)** for recommendations.
    - **Personalized Recommendation:** Suggested products based on purchase patterns.
    - **Hybrid Approach:** Combined **price, product_id, and review prediction score** for better recommendations.
- **Performance Optimization:**
    - Stored recommendation results in **AWS S3 (Processed Data Bucket).**

5️⃣ Deployment & Cloud-Based Automation

- **Deployment Workflow:**
    - **No Docker Used.**
    - Uploaded **trained XGBoost & KNN models to AWS S3.**
    - **Streamlit-based Web App for User Interaction.**
    - **EC2 Deployment:**
        - Launched **AWS EC2 instance.**
        - Installed dependencies using **requirements.txt**.
        - Hosted the **Streamlit app** for live predictions.
- **Challenges Faced & Solutions:**
    - **Issue:** Model inference was slow.
    - **Solution:** Optimized EC2 instance selection and used efficient model loading.

6️⃣ Data Visualization & Insights (Power BI)

- **Challenge:**
  - Direct Power BI connection to AWS S3 was complex.
- **Solution:**
  - Used **AWS Athena to query S3 data** and connected Power BI via DirectQuery.
  - Built **interactive dashboards** showcasing:
    - **Review Prediction Analysis** (Predicted vs. Actual Ratings).
    - **Product Recommendation Insights** (Most recommended products by category).
    - **E-commerce Sales Trends** (Revenue, most sold products).

📊 Workflow Diagram

1️⃣ **Data Ingestion** → (Fetched from AWS S3 Raw Data Bucket, AWS Glue Crawler)
2️⃣ **Data Preprocessing & Feature Engineering** → (AWS Glue processes structured data, stores in new S3 bucket)
3️⃣ **ML Model Training** → (XGBoost for Review Prediction, KNN for Recommendations on AWS SageMaker)
4️⃣ **Cloud Deployment** → (Models stored in S3, Streamlit app deployed on EC2)
5️⃣ **Data Visualization** → (Power BI dashboards via AWS Athena)

🚀 Problems Faced & Solutions

🎤 Common Interview Questions & Answers

Q1: How does your project handle Big Data?

Problem	Solution
Pandas was slow for processing large CSVs	Used **AWS Glue & PySpark** for efficient ETL

Low accuracy with Logistic Regression & Decision Tree	Used **XGBoost with hyperparameter tuning**
Product Recommendation was inaccurate using Matrix Factorization	Switched to **KNN-based approach**
Model inference was slow	Optimized **EC2 instance selection & model loading**
Power BI couldn't directly access AWS S3	Used **AWS Athena for querying S3 data**

🗨️ "I leveraged **AWS Glue & PySpark** for distributed processing and stored large datasets in AWS S3."

Q2: Why AWS for deployment?

🗨️ "AWS provides a full ML lifecycle with **SageMaker for training, EC2 for deployment, and Glue for ETL.**"

Q3: How did you ensure model accuracy?

🗨️ "We initially tested multiple models, but **XGBoost performed best** for review prediction. Used **Grid Search & feature engineering** for optimization."

Q4: Why did you use KNN for recommendation?

🗨️ "We first tried Matrix Factorization, but due to sparse data issues, **KNN gave better recommendations by leveraging user-product similarity.**"

Q5: How does Power BI integrate with AWS?

🗨️ "We connected **AWS S3 via Athena queries**, allowing Power BI to generate real-time dashboards."

🔍 Enhancements for Future Implementation

✅ **CI/CD automation:** Using **AWS CodePipeline** for model retraining and deployment.
✅ **Kubernetes Scaling:** Deploy ML models with **AWS Fargate & Kubernetes** for better

scalability.
✅ **Optimized Model Inference:** Using **TensorRT** to reduce latency in predictions.

This structured workflow aligns with your **updated project approach (structured features for review prediction, no NLP)** while maintaining **a strong AWS & ML pipeline**. 🚀 Let me know if you need further refinements!