**Lab Exercise 3– Price Prediction in Amazon SageMaker**

This detailed guide walks you through:  
1️⃣ **House Price Prediction** (use case)  
2️⃣ **Where Price Prediction is Applicable** (industries & scenarios)  
3️⃣ **How It’s Done in Amazon SageMaker** (step-by-step with code)

It’s designed for **students, data engineers, and business analysts** learning predictive modeling using AWS tools.

**1. Lab Title**

**Price Prediction using Amazon SageMaker — House Price Prediction and Industry Applications**

**2. Objective**

The objective of this lab is to:

* Understand the **concept of price prediction** and its business relevance.
* Implement a **House Price Prediction model** using **Amazon SageMaker**.
* Explore how **SageMaker simplifies machine learning workflows** such as data preparation, training, and deployment.

By the end of this exercise, you will be able to:  
✅ Explain what price prediction is and where it applies.  
✅ Train and evaluate a regression model using SageMaker.  
✅ Deploy the model to make predictions.

**3. Prerequisites**

Before beginning this lab:

* You must have an **AWS Account** with **SageMaker**, **S3**, and **IAM** permissions.
* You should know **Python** and **basic machine learning concepts** (features, target variable, regression).
* You must have your dataset uploaded to **Amazon S3**, e.g.  
  s3://your-bucket/datasets/house\_prices\_dataset.csv.

**4. Introduction to Price Prediction**

**4.1 What is Price Prediction?**

**Price prediction** is a data-driven process of estimating the future or fair value of an item based on its characteristics and historical patterns.  
It uses **machine learning regression models** to learn relationships between **independent features** (inputs) and **target price** (output).

**4.2 Where Price Prediction is Applicable**

Price prediction models are used in almost every industry:

| **Industry** | **Application** |
| --- | --- |
| **Real Estate** | Predicting house or rent prices based on features (area, location, etc.) |
| **Finance** | Forecasting stock or crypto prices |
| **E-commerce** | Dynamic pricing based on demand, season, and competitors |
| **Agriculture** | Predicting crop prices or yield value |
| **Automotive** | Estimating resale or insurance value of vehicles |
| **Energy** | Forecasting electricity or oil prices |

**5. Use Case: House Price Prediction**

In this lab, we’ll build a **regression model** to predict **house prices** using features like:

* Location
* Area (square feet)
* Number of bedrooms and bathrooms

We’ll perform this end-to-end in **Amazon SageMaker Studio**.

**6. Lab Environment Setup**

**Step 1: Launch SageMaker Studio**

1. Log in to **AWS Management Console**.
2. Open **Amazon SageMaker → SageMaker Studio**.
3. Select your user profile and **Launch App → Studio**.
4. Create a new notebook with **Python 3 (Data Science)** kernel.
5. Rename it to:
6. House\_Price\_Prediction\_in\_SageMaker.ipynb

**Step 2: Import Required Libraries**

import pandas as pd

import numpy as np

import boto3

import sagemaker

from sagemaker import get\_execution\_role

**Step 3: Load Dataset from S3**

Make sure your dataset is stored in S3.

# Define S3 bucket and file

bucket = "your-bucket-name"

file\_key = "datasets/house\_prices\_dataset.csv"

s3\_path = f"s3://{bucket}/{file\_key}"

# Load the dataset

df = pd.read\_csv(s3\_path)

df.head()

**Output:** Preview of data with columns like location, area, bedrooms, bathrooms, and price.

**Step 4: Data Preprocessing**

Convert categorical columns (like location) to numerical format and split data.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

# Encode location

encoder = LabelEncoder()

df['location\_encoded'] = encoder.fit\_transform(df['location'])

# Select features and target

X = df[['area', 'bedrooms', 'bathrooms', 'location\_encoded']]

y = df['price']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step 5: Upload Training Data to S3**

Save your training data as CSV and upload it to your bucket.

train\_data = pd.concat([y\_train, X\_train], axis=1)

train\_data.to\_csv('train.csv', index=False)

s3 = boto3.client('s3')

s3.upload\_file('train.csv', bucket, 'house-price/train.csv')

print("Training data uploaded to S3.")

**7. Building the Model in SageMaker**

**Step 6: Initialize SageMaker Session**

sagemaker\_session = sagemaker.Session()

role = get\_execution\_role()

**Step 7: Use Built-in XGBoost Algorithm**

Amazon SageMaker provides **prebuilt algorithms** — here we’ll use **XGBoost**, one of the best for regression.

from sagemaker.inputs import TrainingInput

from sagemaker.amazon.amazon\_estimator import get\_image\_uri

# Get XGBoost container

container = sagemaker.image\_uris.retrieve("xgboost", boto3.Session().region\_name, "1.5-1")

# Define estimator

xgb\_model = sagemaker.estimator.Estimator(

container,

role,

instance\_count=1,

instance\_type="ml.m5.large",

output\_path=f"s3://{bucket}/house-price/output",

sagemaker\_session=sagemaker\_session

)

**Step 8: Set Hyperparameters**

xgb\_model.set\_hyperparameters(

objective="reg:squarederror",

num\_round=100,

max\_depth=5,

eta=0.2,

subsample=0.8

)

**Step 9: Train the Model**

train\_input = TrainingInput(

s3\_data=f"s3://{bucket}/house-price/train.csv",

content\_type="text/csv"

)

xgb\_model.fit({"train": train\_input})

**Output:** Training logs showing metrics such as RMSE decreasing over iterations.

**8. Deploying the Model**

**Step 10: Deploy the Trained Model**

predictor = xgb\_model.deploy(

initial\_instance\_count=1,

instance\_type="ml.m5.large"

)

**Step 11: Test Predictions**

Prepare sample data and send it to the deployed model:

import numpy as np

sample = np.array([[1500, 3, 2, 1]])

payload = ','.join(map(str, sample.flatten()))

# Explicitly set content\_type to 'text/csv'

result = predictor.predict(payload.encode('utf-8'), initial\_args={"ContentType": "text/csv"})

print("Predicted Price (INR):", result)

**Expected Output:**  
A numeric prediction representing the **estimated house price**.

**9. Evaluation**

**Step 12: Model Evaluation**

You can test accuracy using local predictions (not deployed endpoint):

!conda install -y -c conda-forge xgboost

from sklearn.metrics import mean\_squared\_error, r2\_score

from xgboost import XGBRegressor

# Local XGBoost model for evaluation

model = XGBRegressor()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluate

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R² Score:", r2)

**Interpretation:**

* **Lower MSE** = better accuracy.
* **R² near 1** = excellent prediction performance.

**10. Clean Up**

After testing, delete the endpoint to avoid extra charges:

sagemaker.Session().delete\_endpoint(predictor.endpoint\_name)

**11. Validation & Observations**

* Was the predicted price close to real-world expectations?
* Which features most influenced price (check feature importance)?
* What happens if you increase training data or model complexity?