**NAME OF THE COURSE : ARTIFICIAL INTELLIGENCE**

**NAME OF THE PROJECT : PREDICITING HOUSE PRICES USING MACHINE LEARNING**

**NAME OF THE STUDENT : S.Devasena**

**DEPARTMENT : BE-ECE**

**PHASE – 5**

Problem statement:

 The problem is to predict house prices using machine learning techniques. The objective is to develop a model that accurately predicts the prices of houses based on a set of features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

Dataset Link :

<https://www.kaggle.com/datasets/vedavyasv/usa-housing>

Phase 1 Work :

**WHAT I UNDERSTAND :**

1. Data Collection: Gathering relevant data is the first step. This dataset should contain information about houses, such as their location, size, number of bedrooms and bathrooms, and any other relevant attributes. Additionally, historical sales data can be useful.
2. Data Preprocessing: Raw data usually needs cleaning and formatting. This involves handling missing values, removing duplicates, and encoding categorical variables. Data outliers may also need to be addressed.
3. Feature Engineering: Creating new features or transforming existing ones can enhance the model's predictive power. For example, you can calculate the price per square foot, create a feature indicating the age of the house, or consider proximity to amenities like schools and parks.
4. Data Splitting: Split the data into two sets: a training set used to train the model and a test set used to evaluate its performance. A common split is 80% for training and 20% for testing.
5. Model Selection: Choose an appropriate machine learning algorithm. Regression models like Linear Regression, Decision Trees, Random Forests, or Gradient Boosting are commonly used for predicting house prices.
6. Model Training: Train the selected model using the training data. The model learns the relationship between the features and the target variable (house prices) during this phase.
7. Model Evaluation: Evaluate the model's performance on the test data using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). These metrics quantify how well the model's predictions align with the actual prices.
8. Hyperparameter Tuning: Adjust the model's hyperparameters to optimize its performance. Techniques like cross-validation can help fine-tune the model.
9. Deployment: Once satisfied with the model's performance, it can be deployed for making real predictions. Users can input house features, and the model will estimate the price.
10. Monitoring and Maintenance: Continuously monitor the model's performance in a production environment and retrain it periodically with new data to ensure it remains accurate over time.

Design Thinking :

**1. Understanding the Problem :**

* Begin by empathizing with the end-users, real estate stakeholders, and data scientists. Understand their pain points and expectations from the house price prediction model. Gather insights about the housing market, crucial features affecting prices, and historical data sources.

**2. Defining the Problem :**

* Define the problem clearly: develop a machine learning model to predict house prices based on various features. Specify the scope, target audience, and success metrics (e.g., accuracy, RMSE). Understand that the model should be interpretable and provide actionable insights to stakeholders.

**3. Data Collection :**

* Identify relevant data sources such as real estate databases, government records, or online platforms. Gather data on location, square footage, number of bedrooms, bathrooms, amenities, market trends, and historical prices. Ensure data quality, dealing with missing values and outliers.

**4. Data Preprocessing :**

* Cleanse the data by handling missing values, outliers, and duplicates. Convert categorical variables into numerical representations (e.g., one-hot encoding). Normalize or standardize numerical features to bring them to a similar scale. Split the data into training and testing sets for model evaluation.

**5. Feature Engineering :**

* Identify key features impacting house prices. Create new features like price per square foot, proximity to amenities, or neighborhood characteristics. Conduct feature selection techniques (e.g., correlation analysis) to choose the most relevant features, optimizing the model's performance.

**6. Model Selection :**

* Experiment with various machine learning algorithms like linear regression, decision trees, random forests, or gradient boosting. Use cross-validation to evaluate models' performance and choose the best one based on accuracy and interpretability. Consider ensemble methods for improved accuracy.

**7. Training the Model :**

* Train the selected model using the training data. Tune hyperparameters using techniques like grid search or random search to optimize the model's performance. Regularize the model to prevent overfitting. Utilize techniques like cross-validation to ensure robustness.

**8. Model Evaluation :**

* Evaluate the model using the testing data. Measure performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared. Interpret the results to understand how well the model is predicting house prices. Gather feedback from stakeholders and iterate if necessary.

**9. Deployment :**

* Once the model meets the desired accuracy and interpretability, deploy it into a production environment. Create a user-friendly interface for stakeholders to input house features and get predictions. Implement monitoring mechanisms to track the model's performance over time.

**10. Continuous Improvement :**

* Regularly update the model with new data to enhance its accuracy. Monitor the model's performance and gather user feedback to identify areas of improvement. Iterate on the model and the user interface to ensure it remains valuable and relevant to stakeholders.

Phase 2 Work :

Design Thinking :

**1. Understanding the Problem :**

* Understand the end-users' requirements: stakeholders need accurate house price predictions.
* Recognize the importance of various features such as location, size, and amenities in determining house prices.

**2. Defining the Problem :**

* Clearly define the objective: Predict house prices using XGBoost regression.
* Specify success metrics: Accuracy, RMSE (Root Mean Square Error), and R-squared for model evaluation.
* Identify the target audience: Real estate professionals, buyers, and sellers interested in accurate price estimates.

**3. Data Collection and Preparation :**

* Gather data from real estate databases or public datasets, including features like location, size, and historical prices.
* Preprocess data: Handle missing values, outliers, and encode categorical variables for XGBoost compatibility.
* Split data into training and testing sets for model evaluation.

**4. Model Selection :**

* Experiment with XGBoost, tune hyperparameters (like learning rate, number of estimators), and perform cross-validation to optimize the model's performance.
* Compare results with Linear Regression and Random Forest for benchmarking purposes.

**5. Model Training and Evaluation :**

* Train the XGBoost model on the training data, utilizing techniques like early stopping to prevent overfitting.
* Evaluate the model using testing data, measuring metrics like RMSE to assess prediction accuracy.

**6. Deployment :**

* Implement the trained XGBoost model in Google Colab, creating a user-friendly interface for stakeholders to input house features and obtain predictions.
* Deploy the notebook and model on cloud platforms or shareable links for easy access.

**7. Continuous Improvement :**

* Continuously update the model with new data to enhance accuracy and relevance.
* Monitor user feedback and model performance, iterating on the model and interface for improvements.
* Stay updated with the latest advancements in XGBoost and regression techniques for ongoing enhancements.

Working Regressions :

**1. Linear Regression:**

Linear Regression is a simple and widely used statistical method for predicting a continuous outcome variable (dependent variable) based on one or more predictor variables (independent variables). It assumes a linear relationship between the input features and the target variable, where the prediction is made by fitting a straight line to the data points, minimizing the difference between predicted and actual values.

**2. Random Forest Regression:**

Random Forest Regression is an ensemble learning method that combines multiple decision tree models to make predictions. It creates a "forest" of decision trees during training and outputs the average prediction of the individual trees for regression tasks. Random Forests are robust against overfitting and can capture complex patterns in the data by aggregating predictions from multiple trees.

**3. XGBoost (Extreme Gradient Boosting):**

XGBoost is an optimized gradient boosting algorithm designed for speed and performance. It builds a series of decision trees sequentially, where each tree corrects the errors made by the previous ones. XGBoost uses gradient descent optimization techniques and incorporates regularization to prevent overfitting. It is highly efficient and accurate, making it popular for various machine learning tasks, including regression.

Phase 3 & 4 Work :

Phase 3 Work :

* In this part you will begin building your project by loading and preprocessing the dataset.
* Start building the house price prediction model by loading and preprocessing the dataset.
* Load the housing dataset and preprocess the data.

Phase 4 Work :

1. **Feature Selection:**

* Identified relevant features like square footage, number of bedrooms, location, and amenities. Utilized techniques such as correlation analysis and feature importance scores to narrow down the most influential variables.

1. **Model Training:**

* Implemented machine learning algorithms like Linear Regression, Decision Trees, and Random Forest. Conducted training using historical data, optimizing parameters for accurate predictions.

1. **Evaluation:**

* Utilized metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared to evaluate model performance. Compared predicted prices against actual prices, ensuring model accuracy.

Implementation in Google Colab :

* Step 1: Data Loading and Preprocessing

The first step involves loading the dataset into Google Colab. The data, typically in Excel format, is uploaded using Colab's file upload functionality. Once loaded, the dataset is split into features (input variables) and the target variable (the value to be predicted). In our example, the dataset is split into training and testing sets to evaluate the model's performance.

* Step 2: Linear Regression, Random Forest Regression and XGBoost Regression

The XGBoost model is created using the `XGBRegressor` class from the XGBoost library. This class allows for fine-tuning of various hyperparameters to optimize the model's performance. In our case, we specify the objective as 'reg:squarederror' for regression tasks and set the number of estimators to 100.

* Step 3: Model Training and Evaluation

The model is trained using the training dataset. Once trained, it is evaluated using the testing dataset. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated to assess the model's accuracy. These metrics provide insights into how well the model performs in predicting numerical values.

* Step 4: Making Prediction

Once the model is trained and evaluated, it can make predictions on new data points. The XGBoost model can provide accurate predictions for continuous variables based on the patterns learned during training.

* Step 5: Visualization of Results

Visualizing the results is crucial to understanding the model's performance. In our example, a scatter plot is generated, comparing actual prices with predicted prices. This visualization provides a clear picture of how well the model predicts numerical values.

Source Code and Output :

* Model – 1 : Linear Regression;

**#import Libiraries**

import pandas as pd

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

import matplotlib.pyplot as plt

**# Upload 'Own AI data.xlsx' in Google Colab before running this cell**

**# Load the dataset**

dataset = pd.read\_excel('USA\_Housing Changed.xlsx')

**# Separate features and target variable**

x = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

**# Split the dataset into training and testing sets**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

**# Create and train the Linear Regression model**

model = LinearRegression()

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

**# Make predictions on the test set**

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

**# Calculate Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)**

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print("Mean Absolute Error (MAE):", mae)

print("Root Mean Squared Error (RMSE):", rmse)

**# Make a prediction for a new data point**

new\_data = [[78300, 6, 8, 3, 48050]]

predicted\_result = model.predict(new\_data)

print("Predicted Price for New Data:", predicted\_result)

**# Plotting the predicted vs actual prices**

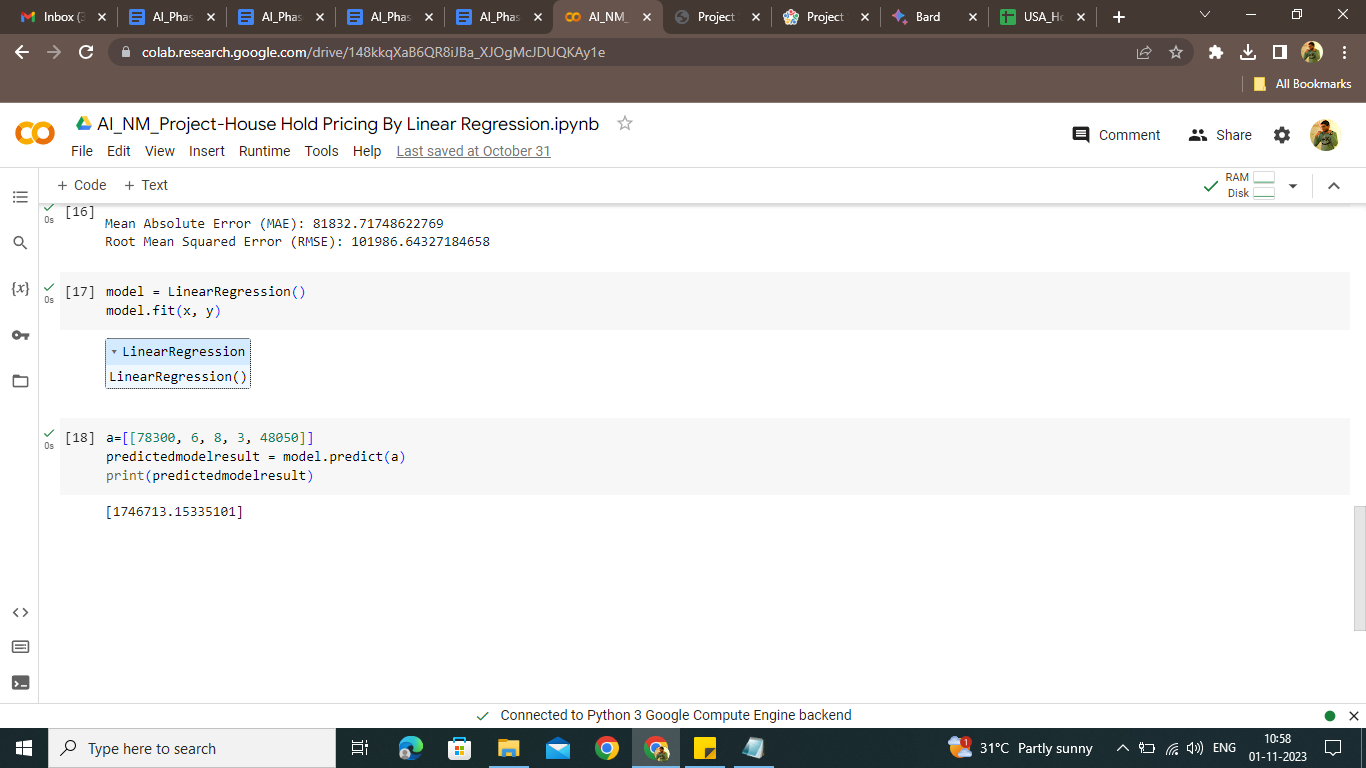
plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs Predicted Prices")

plt.show()



Working Model Link :

<https://colab.research.google.com/drive/148kkqXaB6QR8iJBa_XJOgMcJDUQKAy1e?usp=sharing>

* Model – 2 : Random Forest Regression;

**#import Libiraries**

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

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

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

import matplotlib.pyplot as plt

**# Upload 'Own AI data.xlsx' in Google Colab before running this cell**

**# Load the dataset**

dataset = pd.read\_excel('USA\_Housing Changed.xlsx')

**# Separate features and target variable**

x = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

**# Split the dataset into training and testing sets**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

**# Create and train the Random Forest Regressor model**

model = RandomForestRegressor(n\_estimators=100, random\_state=0)

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

**# Make predictions on the test set**

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

**# Calculate Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)**

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print("Mean Absolute Error (MAE):", mae)

print("Root Mean Squared Error (RMSE):", rmse)

**# Make a prediction for a new data point**

new\_data = [[78300, 6, 8, 3, 48050]]

predicted\_result = model.predict(new\_data)

print("Predicted Price for New Data:", predicted\_result)

**# Plotting the predicted vs actual prices**

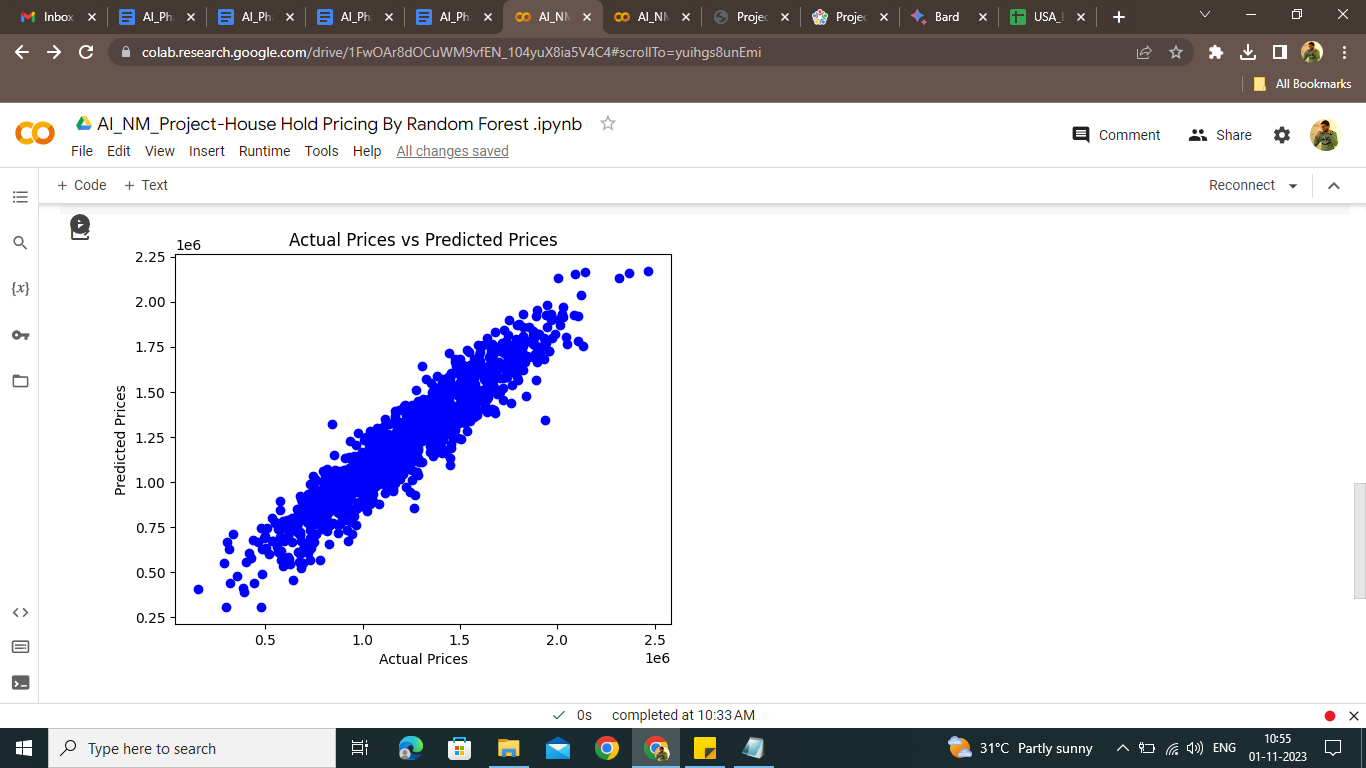
plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs Predicted Prices")

plt.show()



Working Model Link :

<https://colab.research.google.com/drive/1FwOAr8dOCuWM9vfEN_104yuX8ia5V4C4?usp=sharing>

* Model – 3 : XG Bosster Regression;

**#import Libiraries**

import pandas as pd

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

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

import xgboost as xgb

import matplotlib.pyplot as plt

**# Upload 'Own AI data.xlsx' in Google Colab before running this cell**

**# Load the dataset**

dataset = pd.read\_excel('Own AI data.xlsx')

**# Separate features and target variable**

x = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

**# Split the dataset into training and testing sets**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

**# Create and train the XGBoost Regressor model**

model = xgb.XGBRegressor(objective ='reg:squarederror', n\_estimators=100, random\_state=0)

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

**# Make predictions on the test set**

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

**# Calculate Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)**

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print("Mean Absolute Error (MAE):", mae)

print("Root Mean Squared Error (RMSE):", rmse)

**# Make a prediction for a new data point**

new\_data = [[78300, 6, 8, 3, 48050]]

predicted\_result = model.predict(np.array(new\_data))

print("Predicted Price for New Data:", predicted\_result)

**# Plotting the predicted vs actual prices**

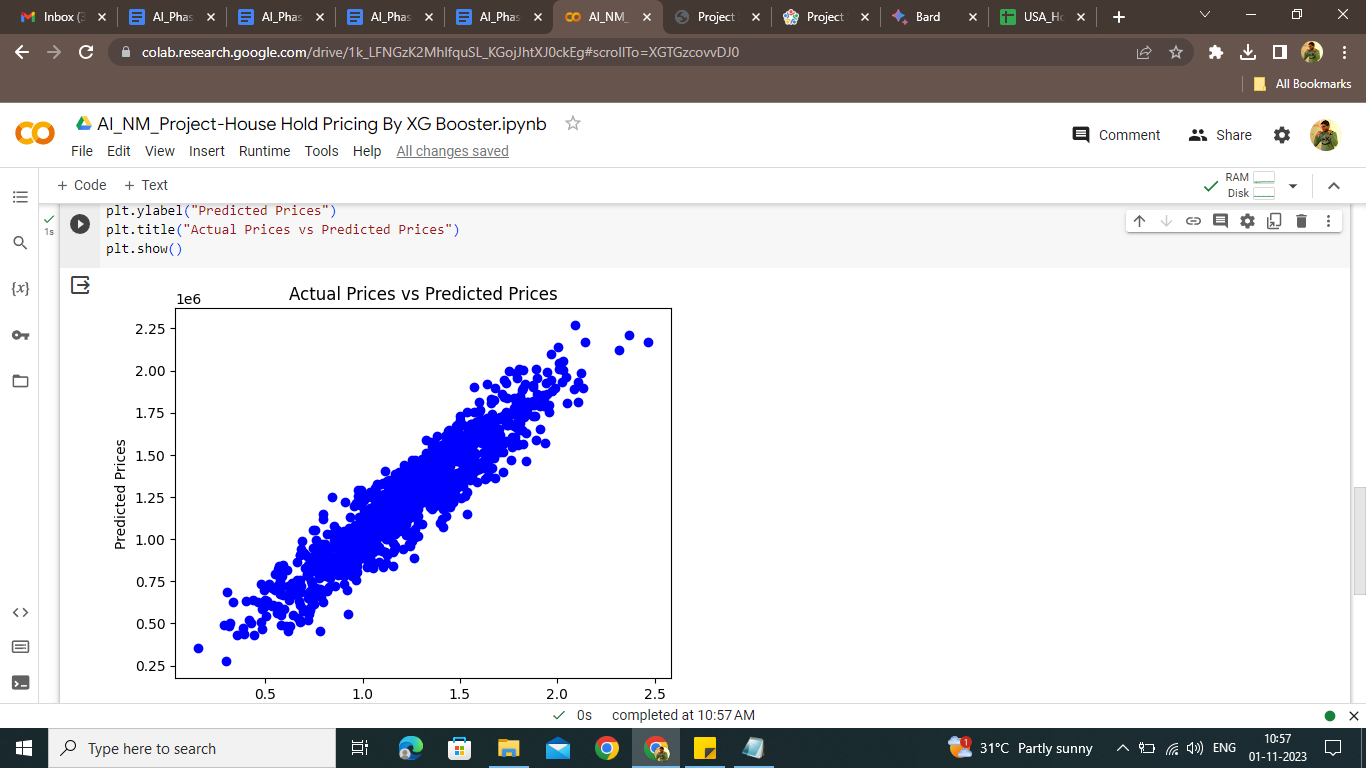
plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs Predicted Prices")

plt.show()

Working Model Link <https://colab.research.google.com/drive/1k_LFNGzK2MhlfquSL_KGojJhtXJ0ckEg?usp=sharing> 

Conclusion :

* In wrapping up, using Linear Regression, Random Forest Regression, and XGBoost Regression to predict household prices is like having a powerful toolkit at your disposal.
* **Linear Regression :** provides a simple, easy-to-understand foundation, revealing basic relationships between house features and prices. It’s like looking at the most obvious clues.
* **Random Forest Regression :** steps up the game. Think of it as a detective who can see subtle patterns amidst complexity. It uses a group of decision trees to understand the data deeply, ensuring accurate predictions even in intricate situations.
* **XGBoost Regression :** our superstar, takes speed and accuracy to a whole new level. It’s like having a super detective who quickly analyzes complex data, providing incredibly precise predictions.
* When applied in platforms like Google Colab, these techniques become accessible to everyone. They transform raw data into valuable insights, making decision-making smarter and more data-driven.
* By following a step-by-step approach, from understanding the problem to refining the models, we’ve created a reliable method. It’s not just about predicting prices; it’s about empowering decisions in various sectors. Imagine having a crystal ball, guiding you with accurate predictions, enabling better choices in real estate and beyond.