**KINGSTON ENGINEERING COLLEGE-5113**

**ARTIFICIAL INTELLIGENCE - PHASE 4**

**TOPIC: PREDICTING HOUSE PRICES USING MACHINE LEARNING**

**TEAM MEMBERS:**

V.EBONICASALETH-au511321104025(ebonicasalethvincent@gmail.com)

G. KANIMOZHI- au511321104040

(kanimozhigopal2004@gmail.com)

D.AKSHAYA-au511321104004(Akshayadayalan03@gmail.com)

K.INDUJA-au511321104031(indujaindu137@gmail.com)

**CONTENT:**

**·** PROBLEM STATEMENT

· PYTHON CODE WITH OUTPUT CREATED WITH GOOGLE COLABORATORY

**· CODE EXPLANATION:**

**1. IMPORTING LIBRARIES**

**2. LOADING THE GIVEN DATASET**

**3. DATA EXPLORATION**

**4. DATA VISUALIZATION**

**5. SPLITTING THE DATASET INTO FEATURES (X) AND TARGET VARIABLE(y)**

**6. PREPROCESSING THE DATASET USING MINMAX SCALER**

**7. FEATURE SELECTION (EXCLUDING THE ADDRESS COLUMN)**

**8. SPLITTING THE DATASET INTO TRAINING AND TESTING SETS**

**9. PREPROCESSING THE DATASET USING STANDARD SCALER**

**10. CROSS-VALIDATION WITH DIFFERENT PREPROCESSING METHODS AND ALGORITHMS**

**11. BUILDING THE MODELS**

**· LINEAR REGRESSION MODEL**

**· RANDOM FOREST MODEL**

**· GRADIENT BOOSTING MODEL**

**· SUPPORT VECTOR REGRESSION MODEL**

**12. PERFORMANCE EVALUATION**

**13. CONCLUSION**

**LIBRARIES USED:**

**·**  PANDAS

· NUMPY

· SCI-KIT LEARN

· MATPLOTLIB

· SEABORN

**DATASET DETAILS:**

We will acquire our dataset from Kaggle, specifically the "USA Housing" dataset. This dataset will contain a wealth of information about houses in the USA, making it suitable for our predictive modeling task.

**· KAGGLE DATASET:**

**· LINK:** [**https://www.kaggle.com/datasets/vedavyasv/usa-housing**](https://www.kaggle.com/datasets/vedavyasv/usa-housing)

**PROBLEM STATEMENT:**

In this technology you will continue building your project by selecting a machine learning algorithm, training the model, and evaluating its performance. Perform different analysis as needed. After performing the relevant activities create a document around it and share the same for assessment.

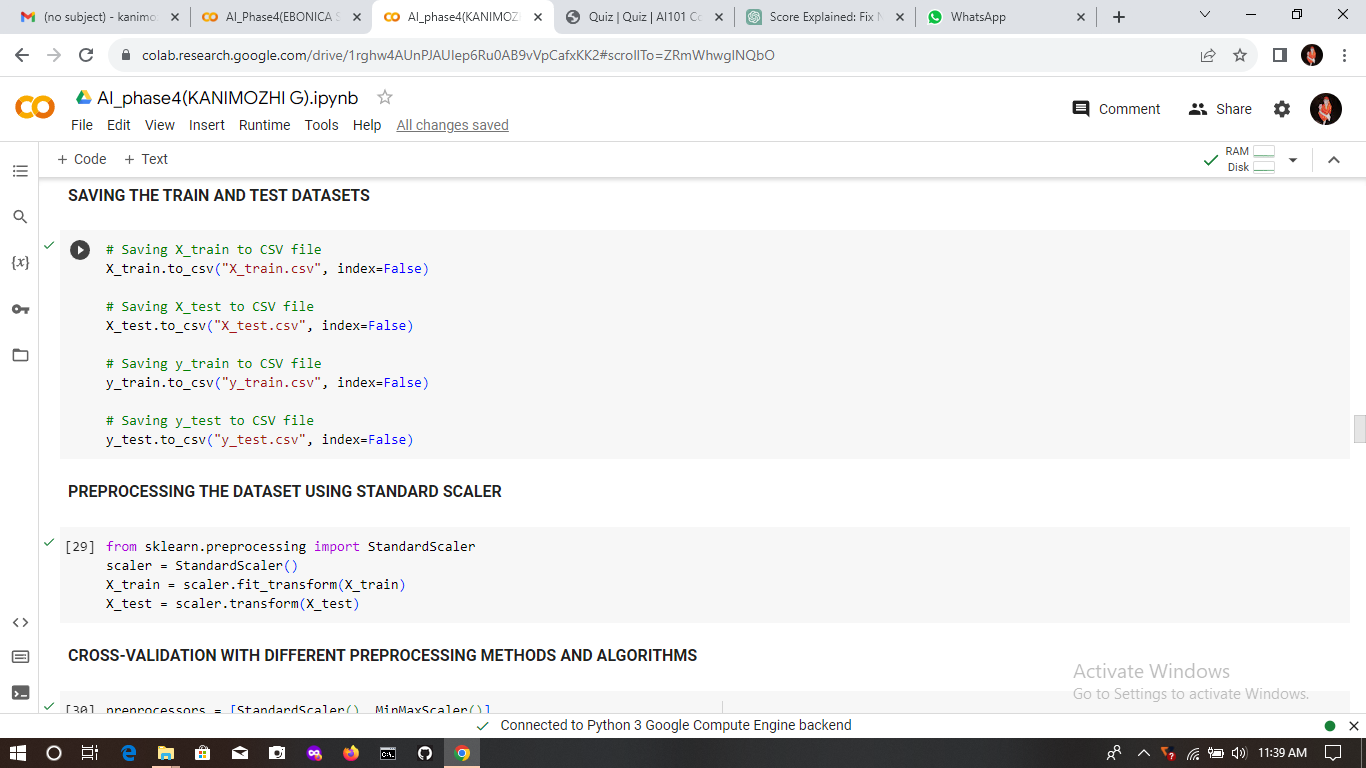
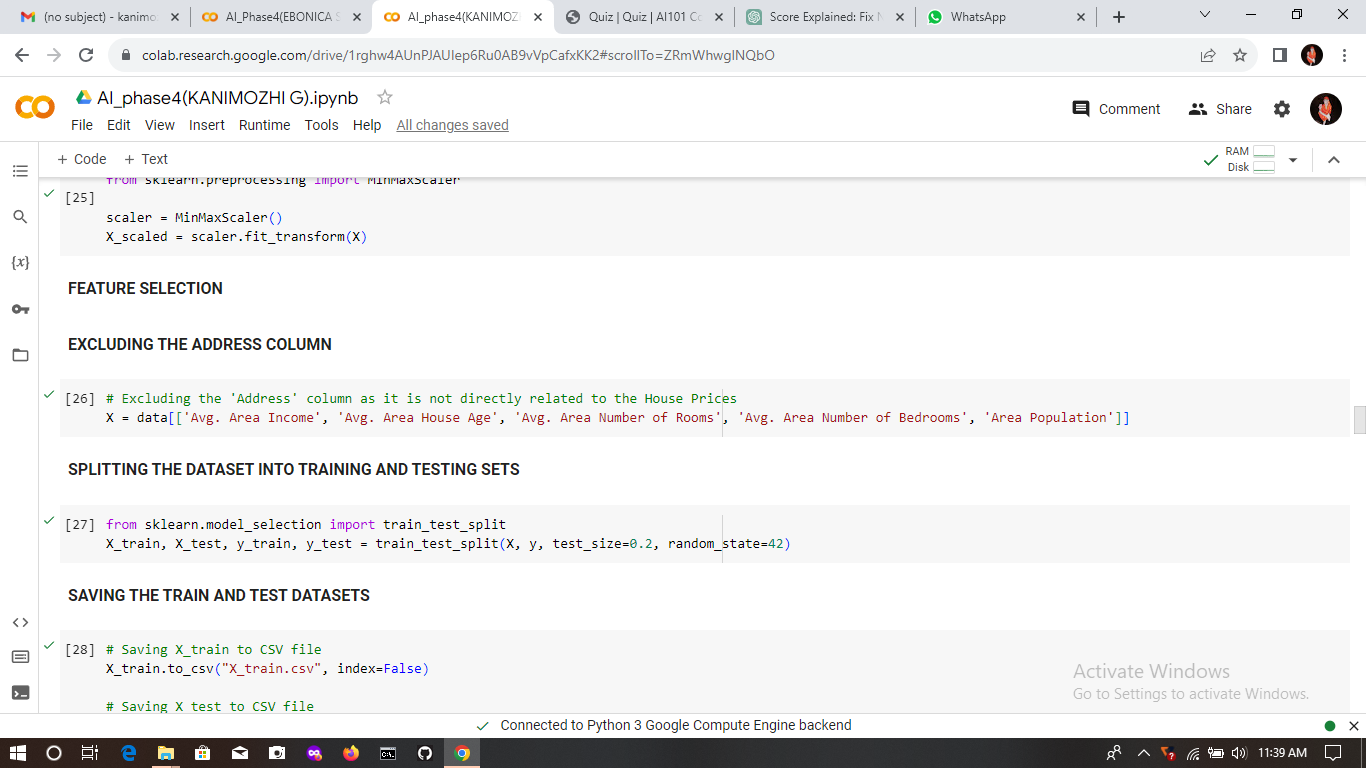
**PREDICTING HOUSE PRICES USING MACHINE LEARNING**

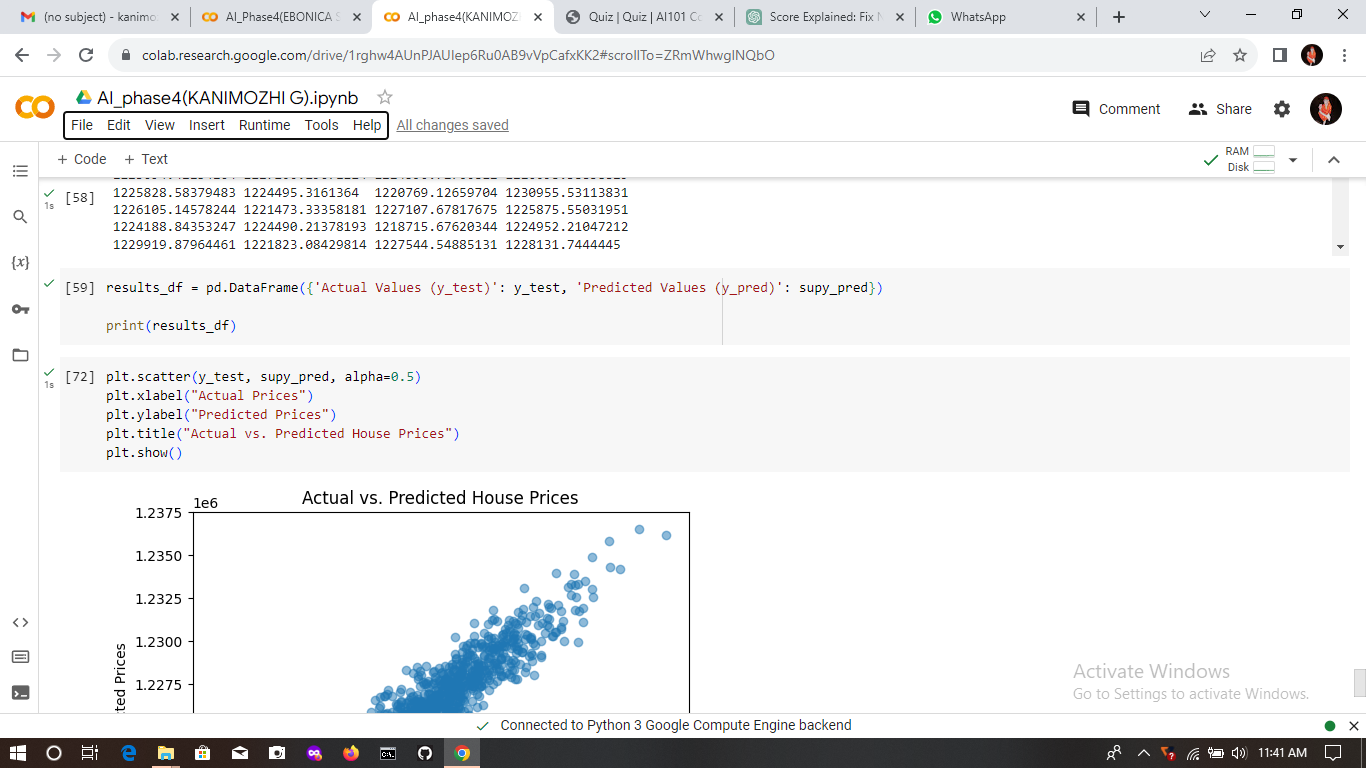
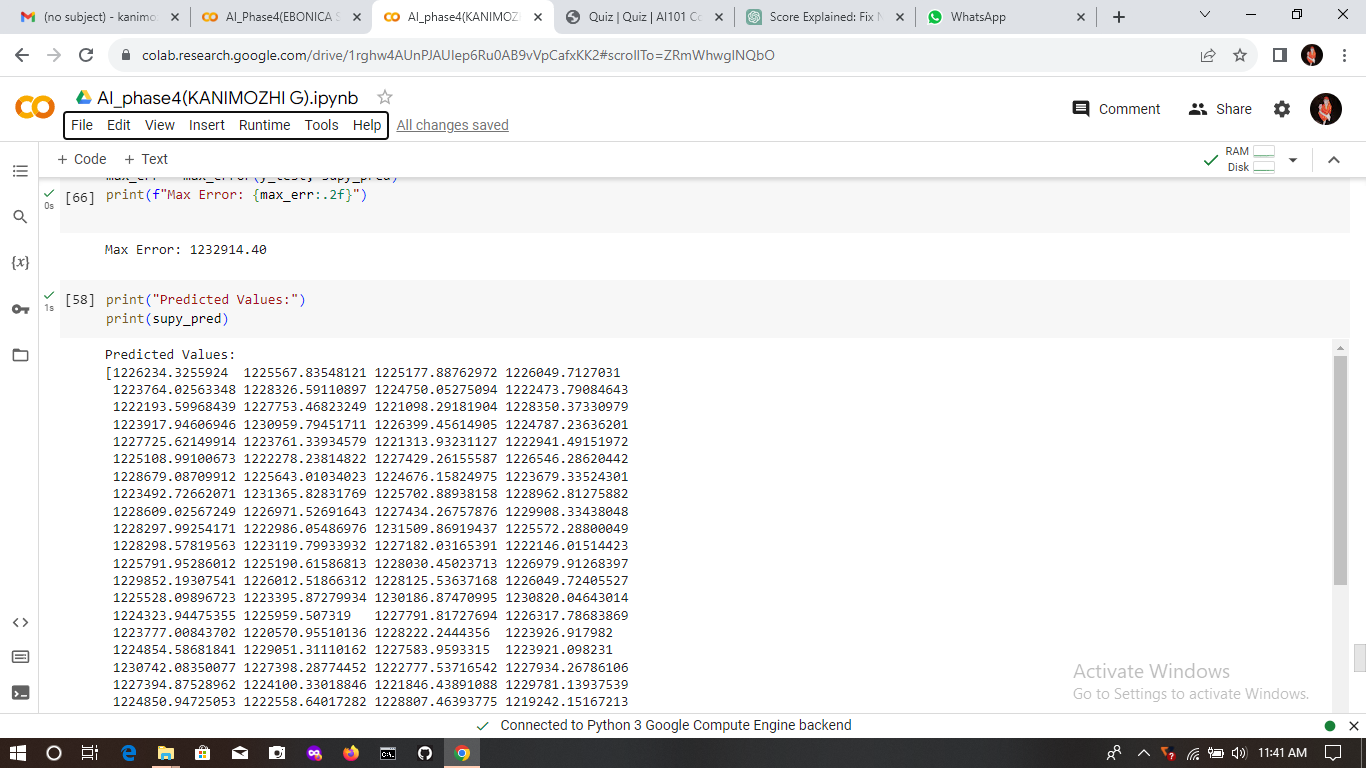
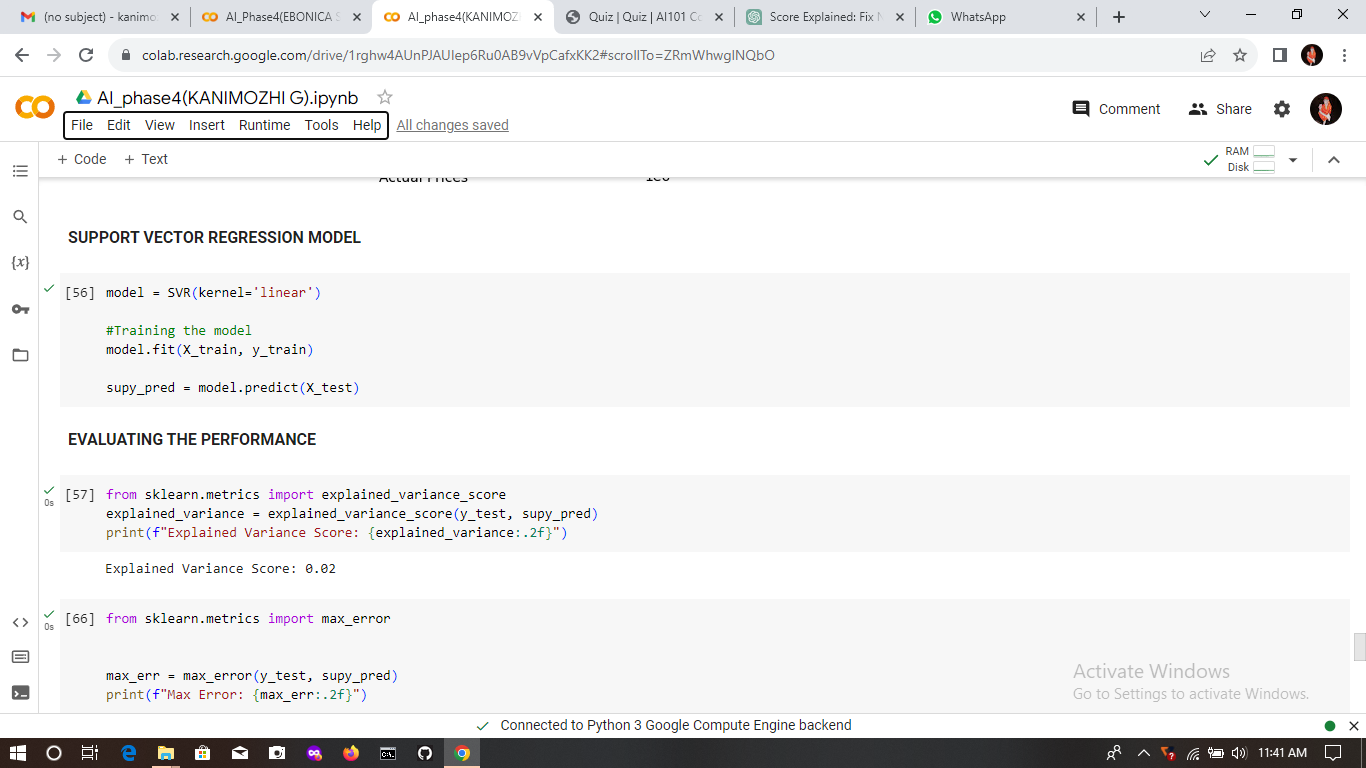
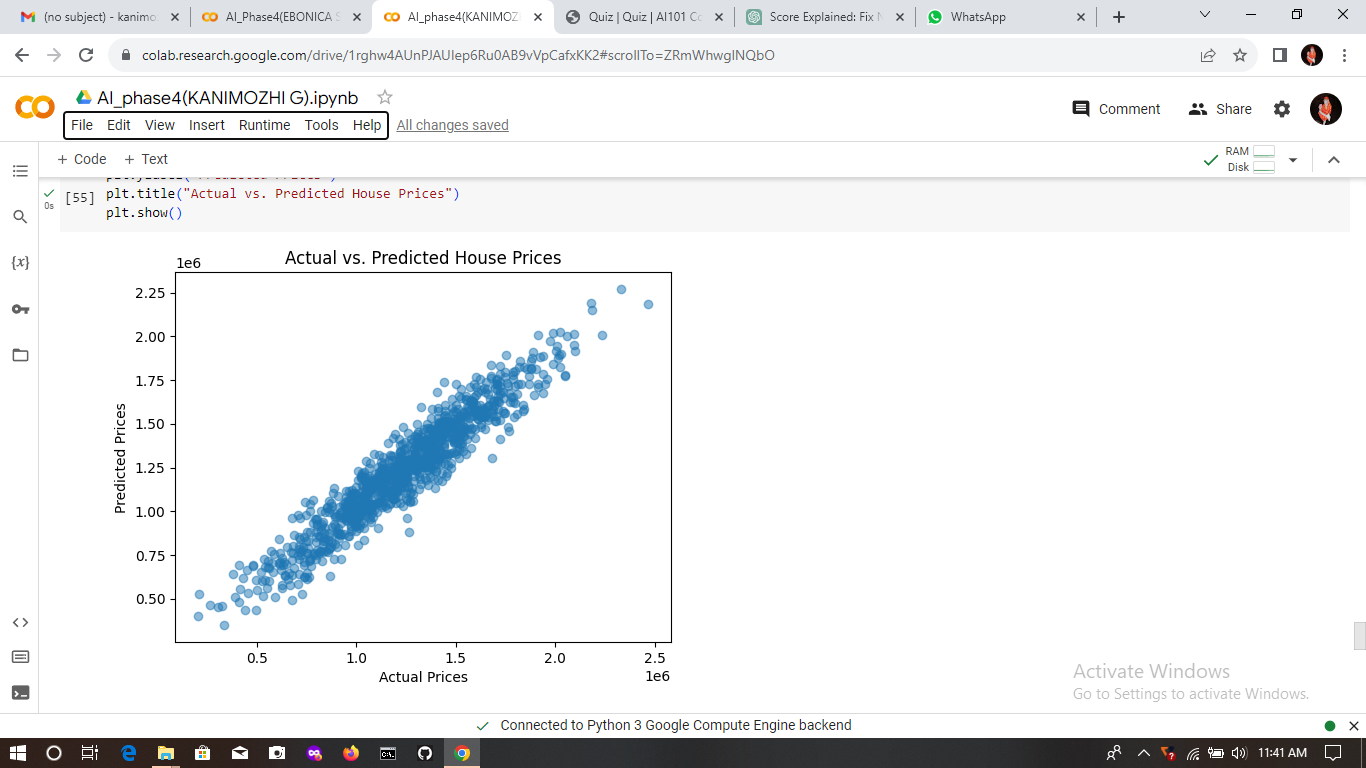
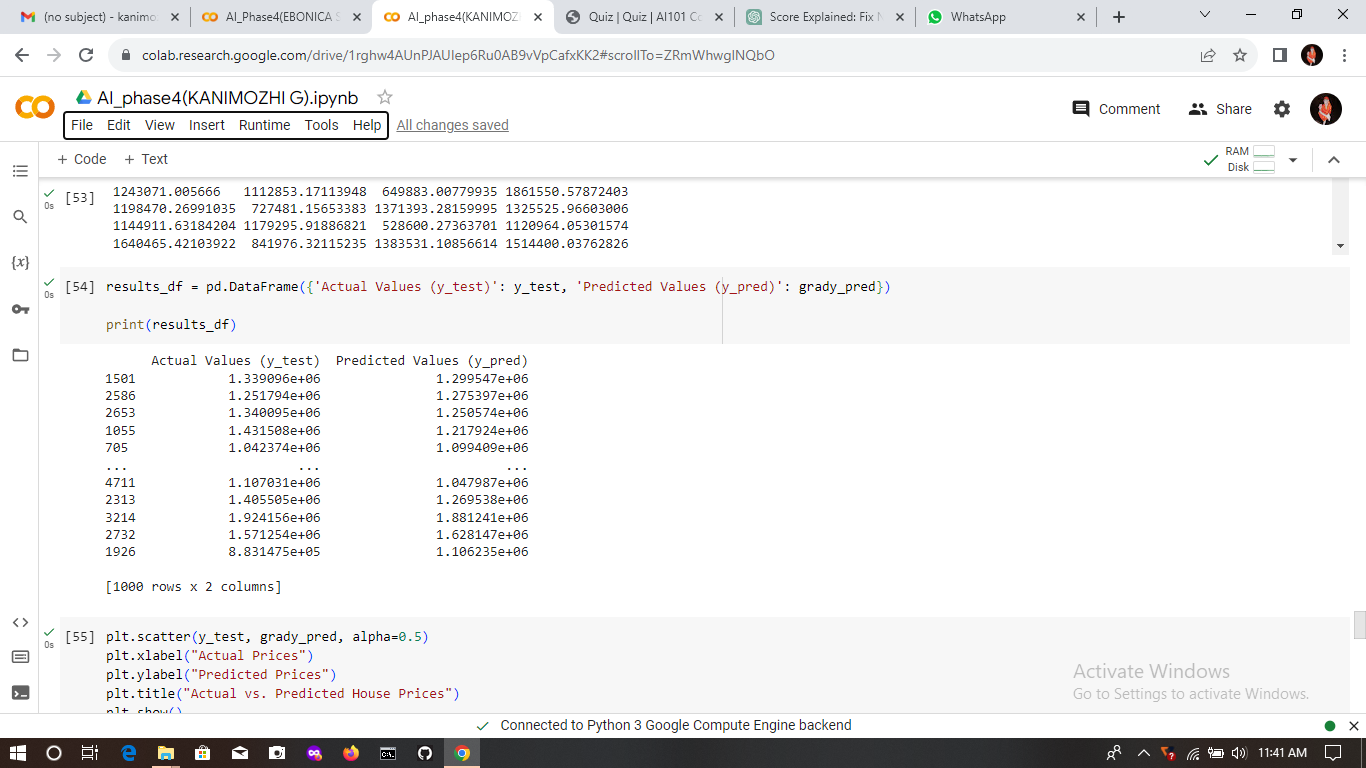
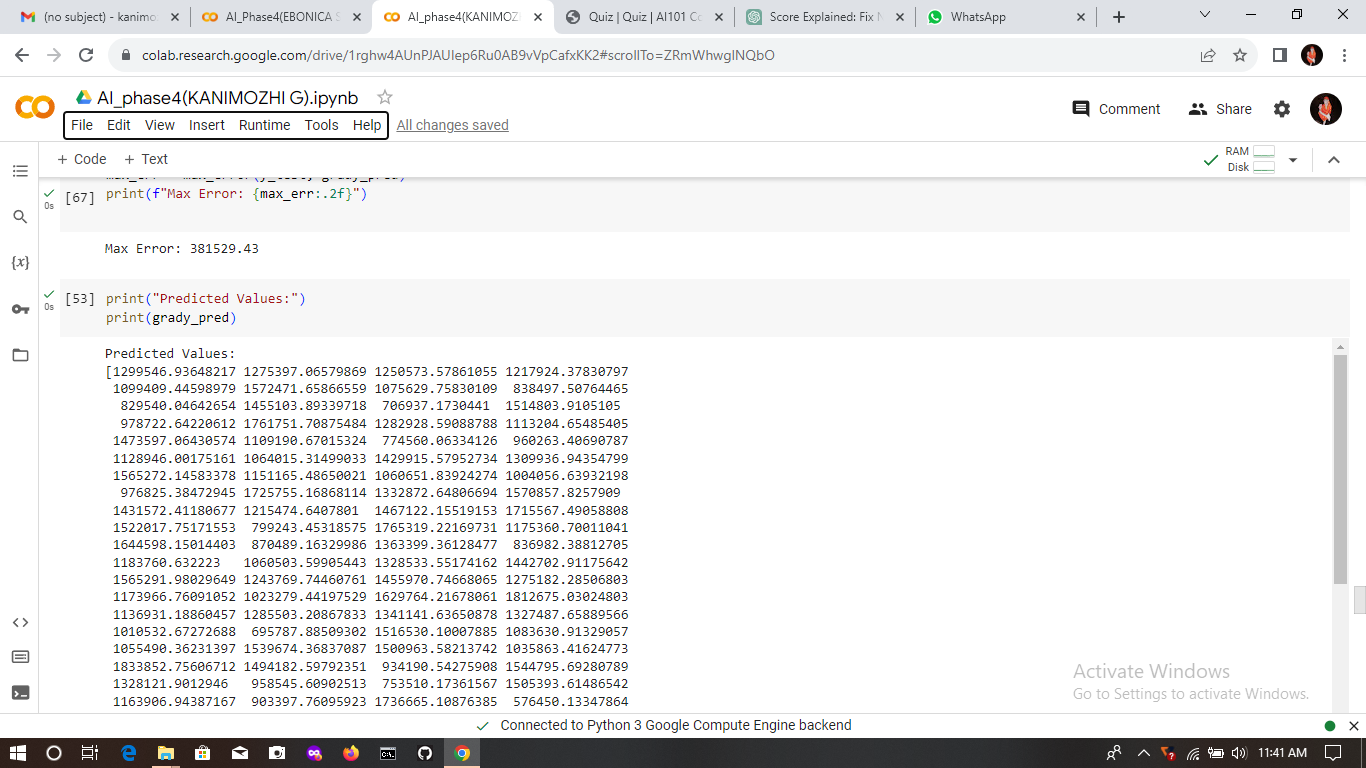
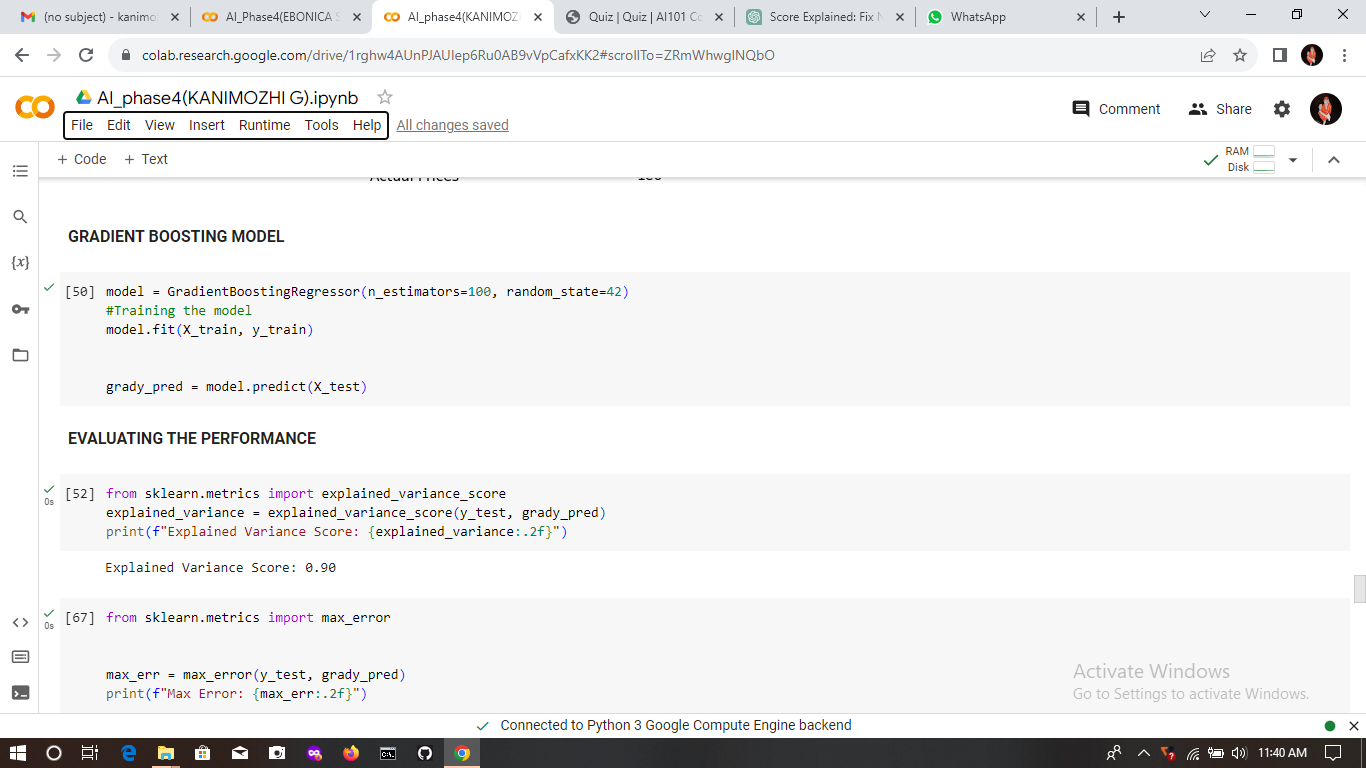
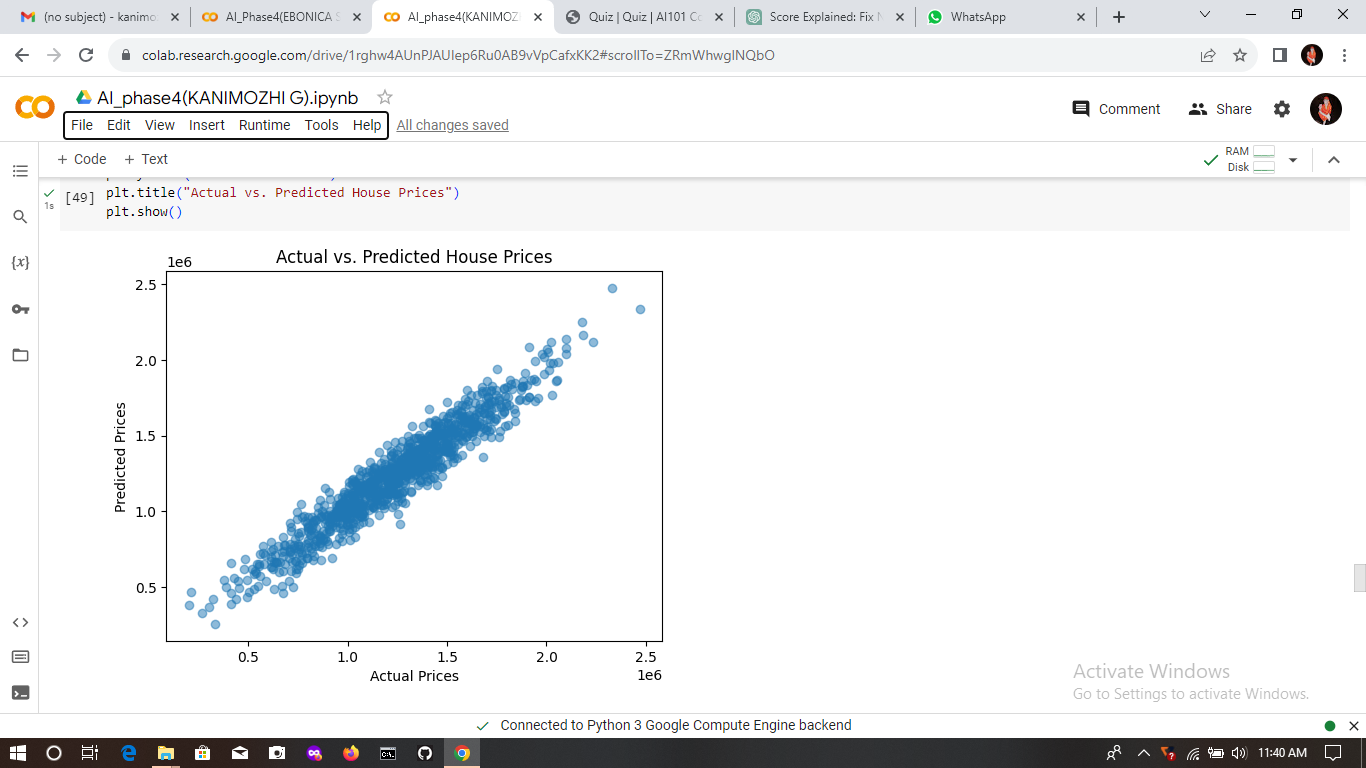
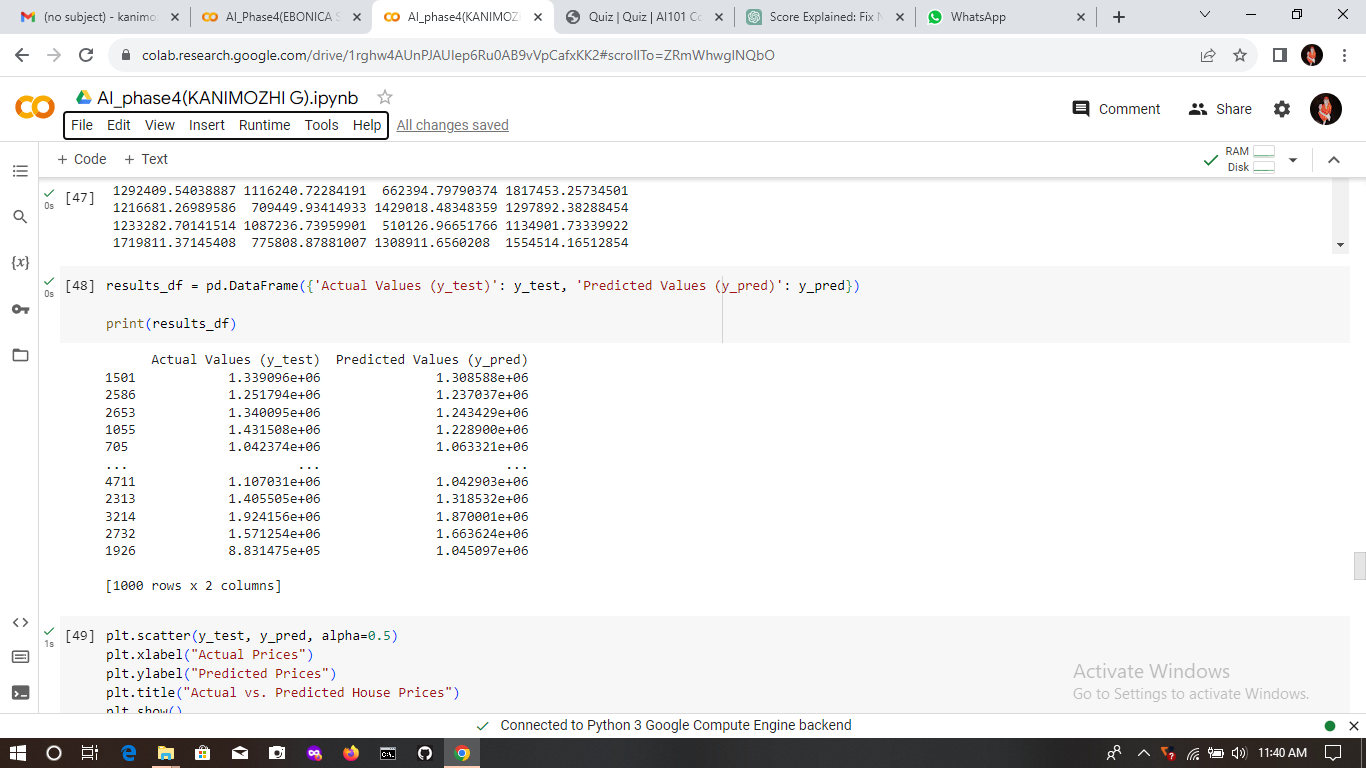
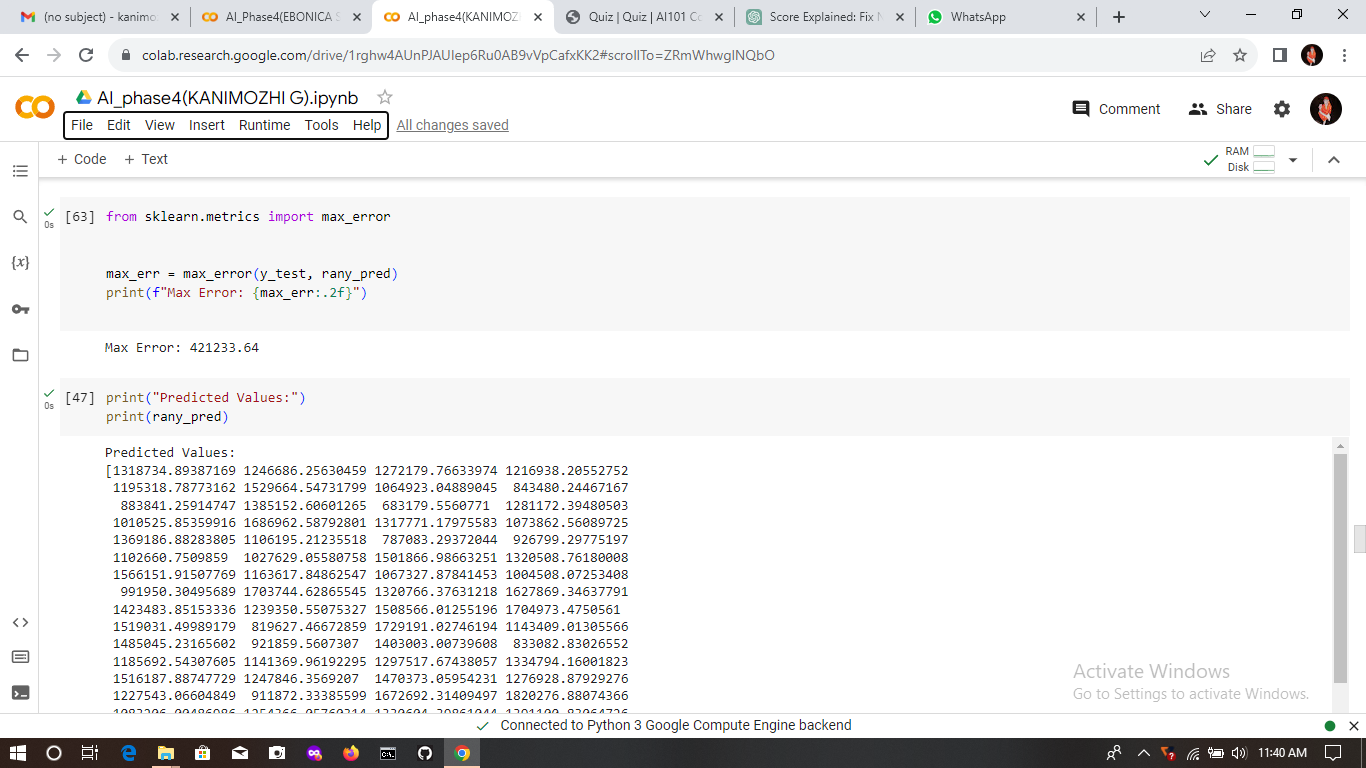
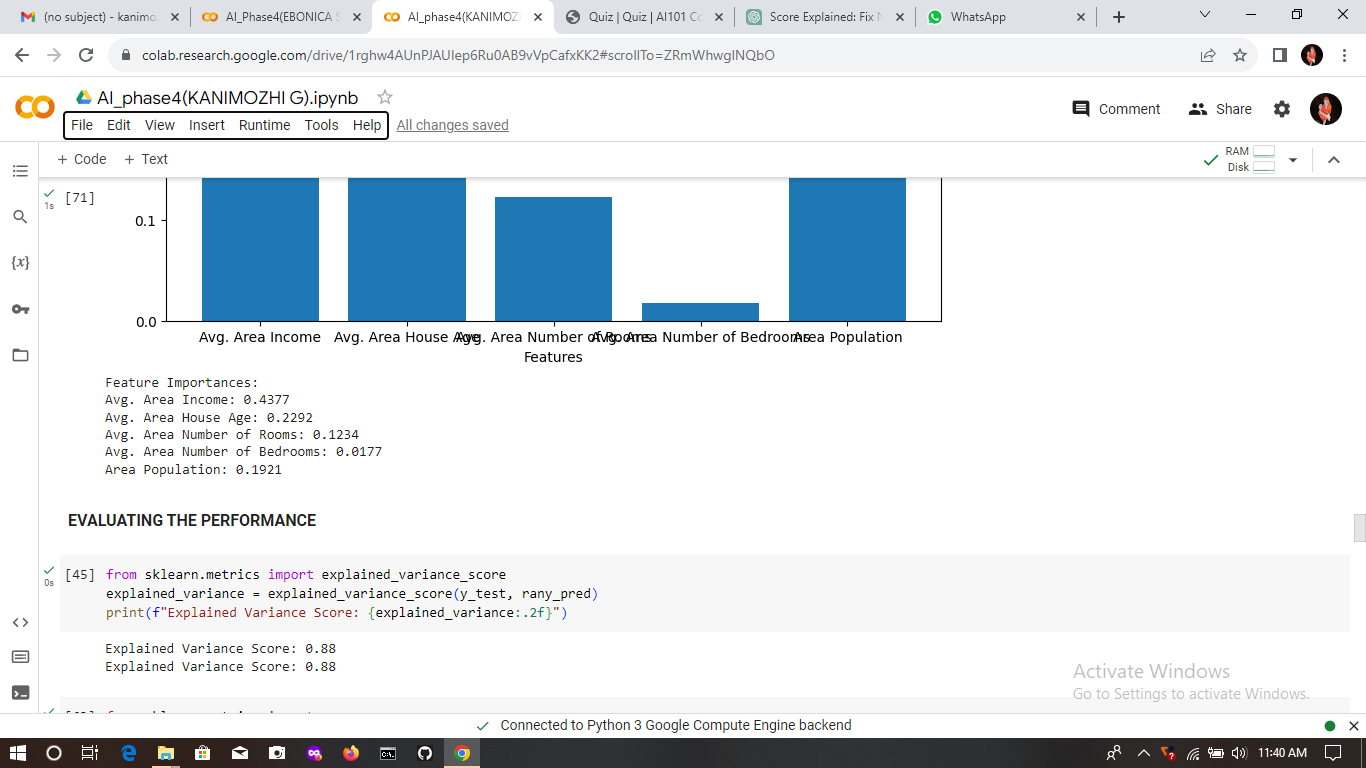
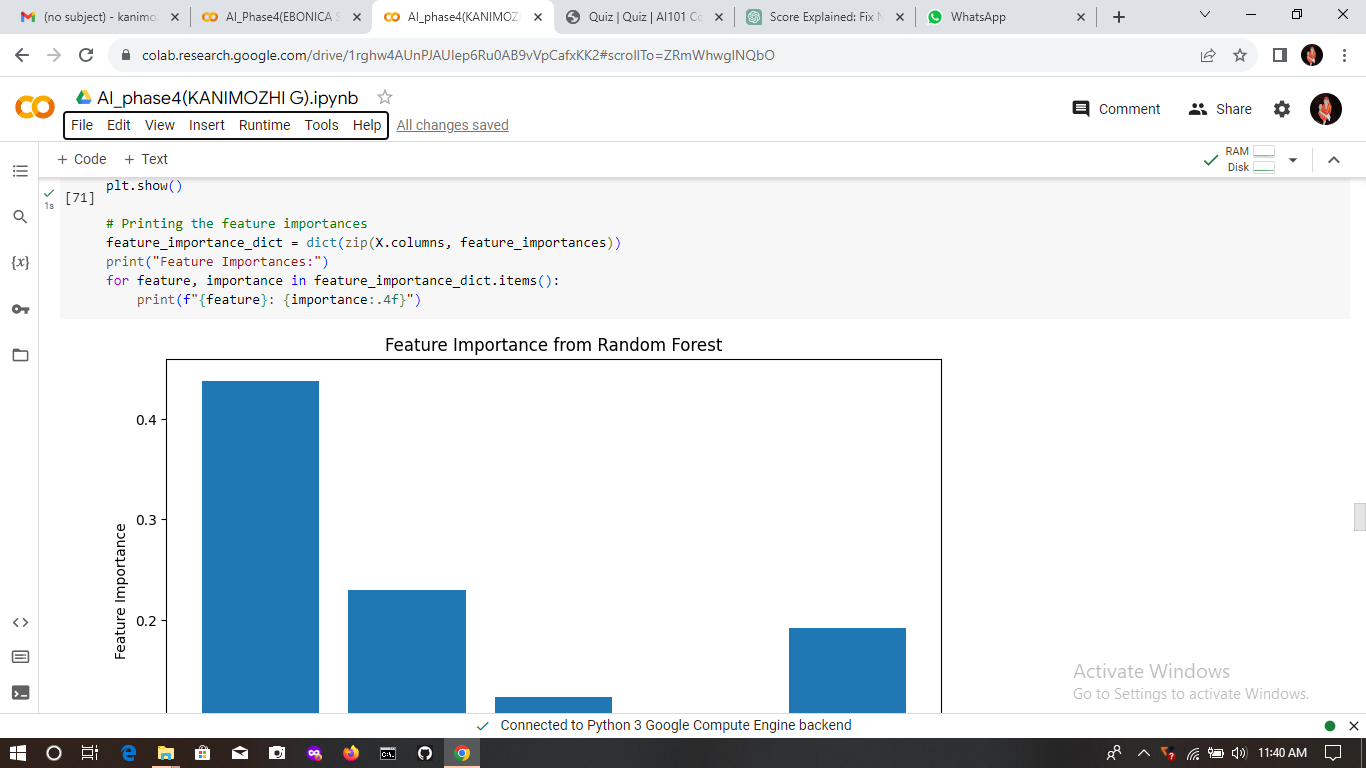
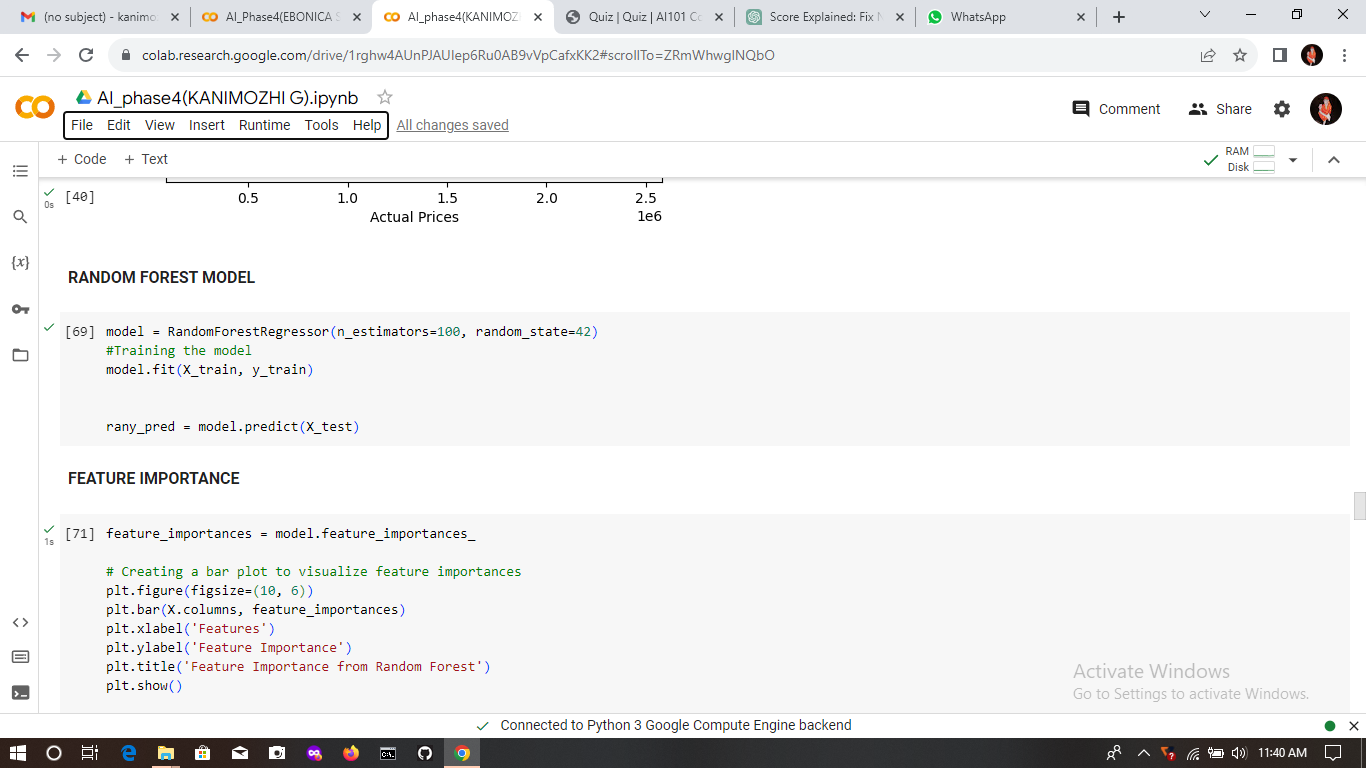
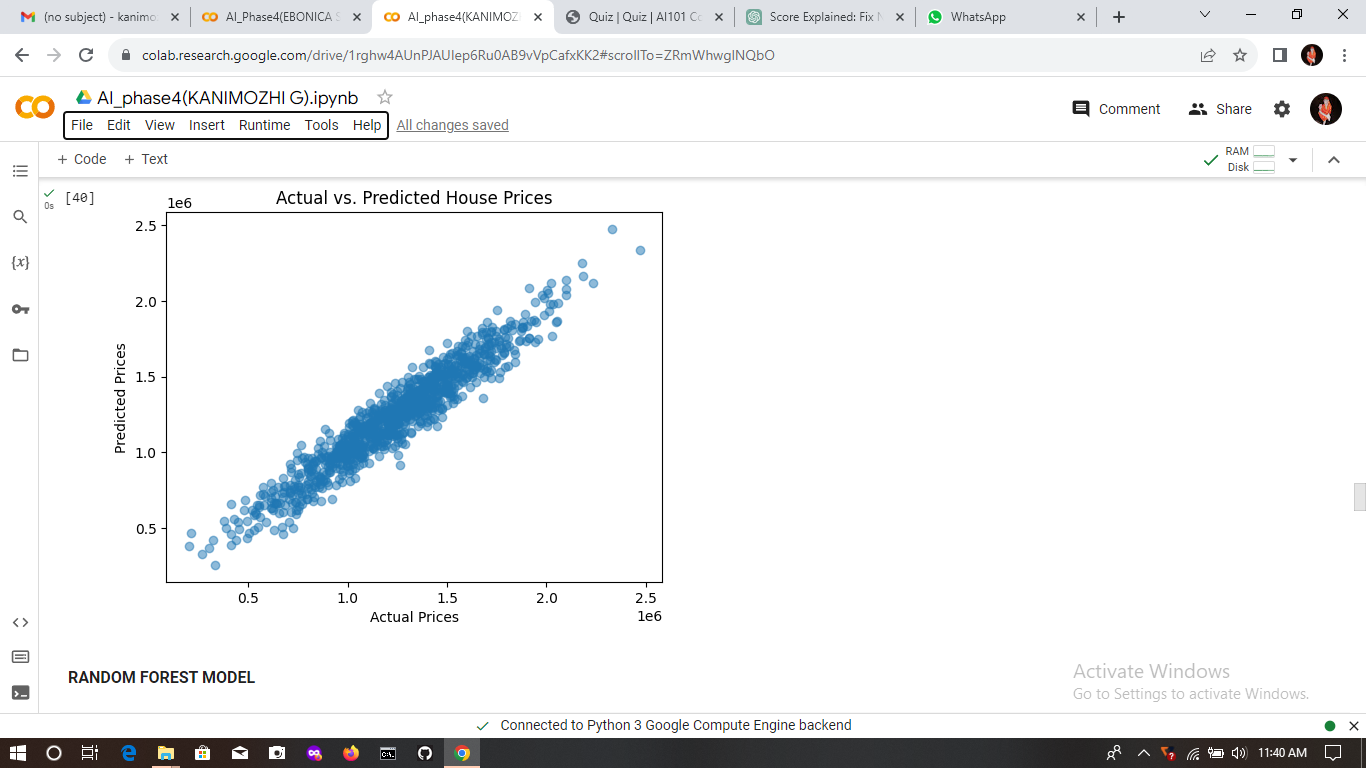
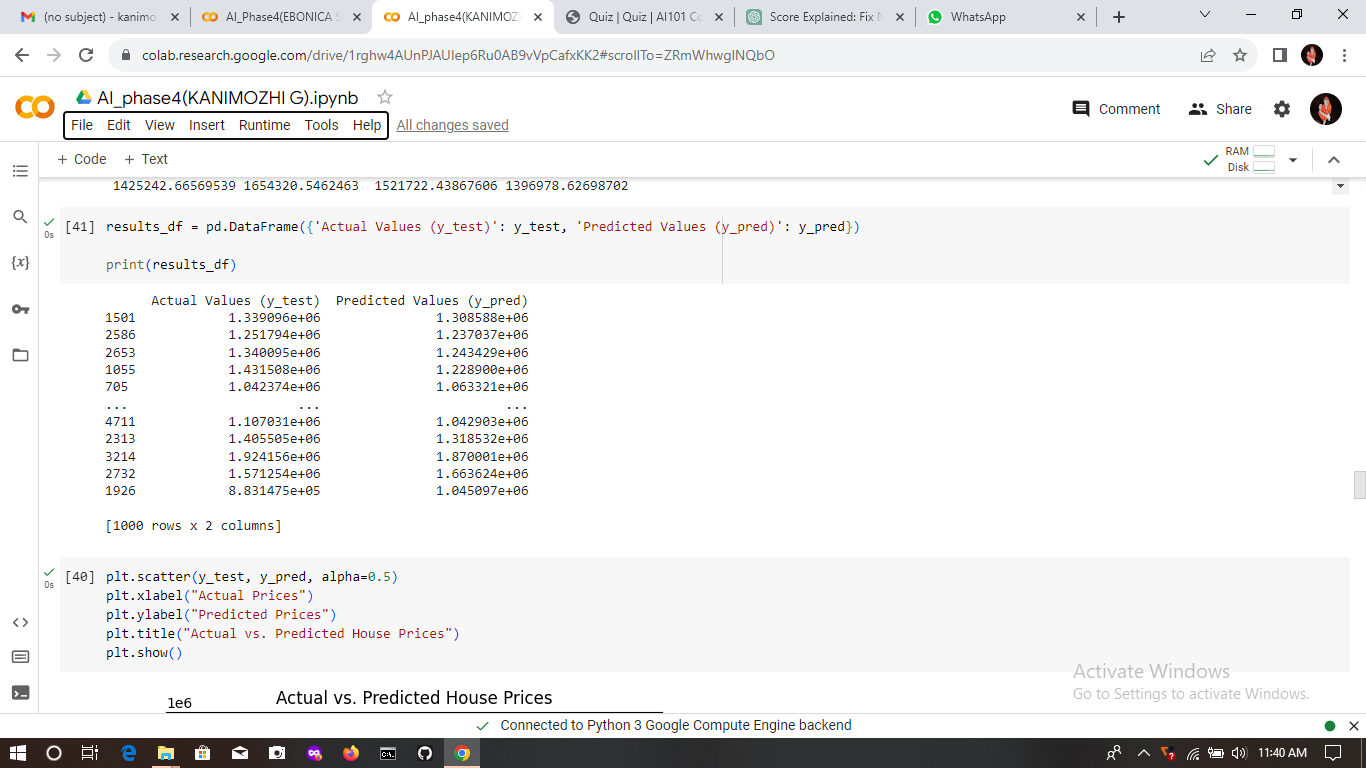
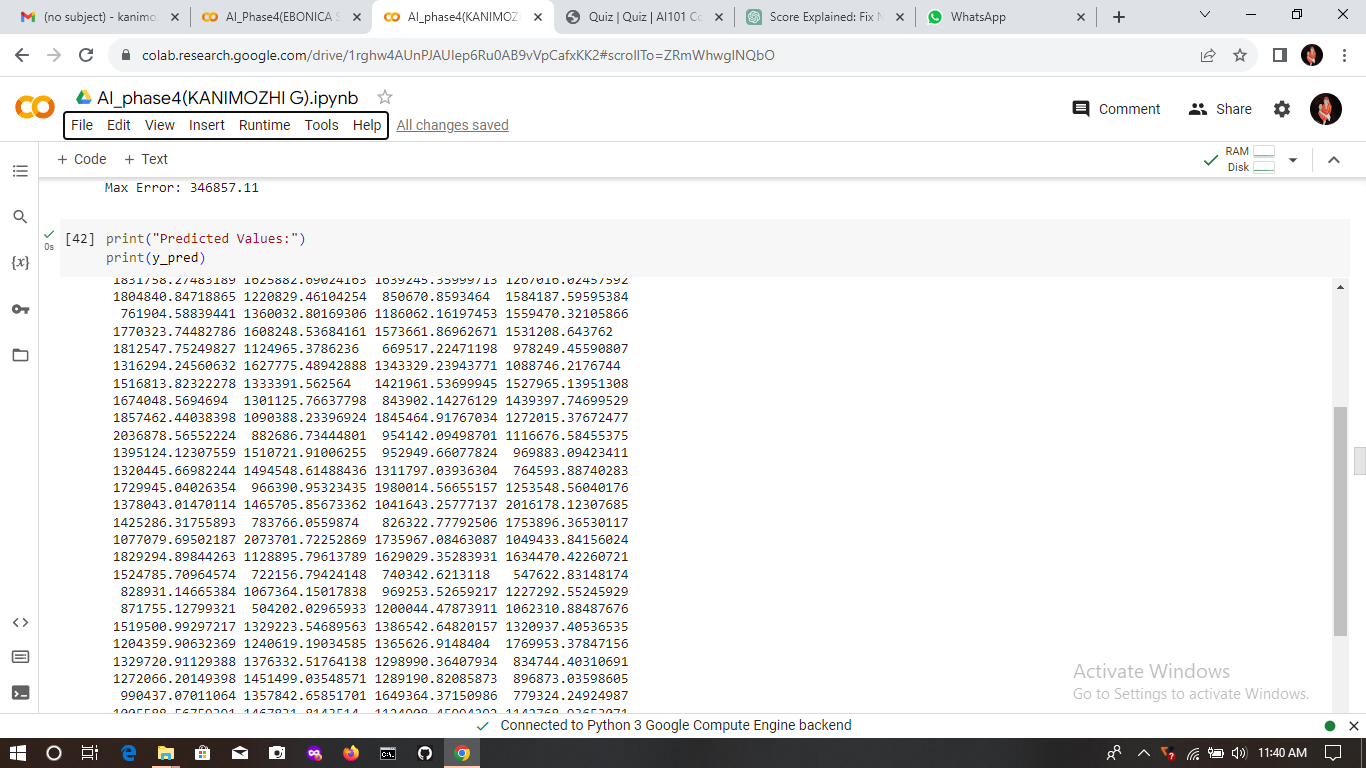
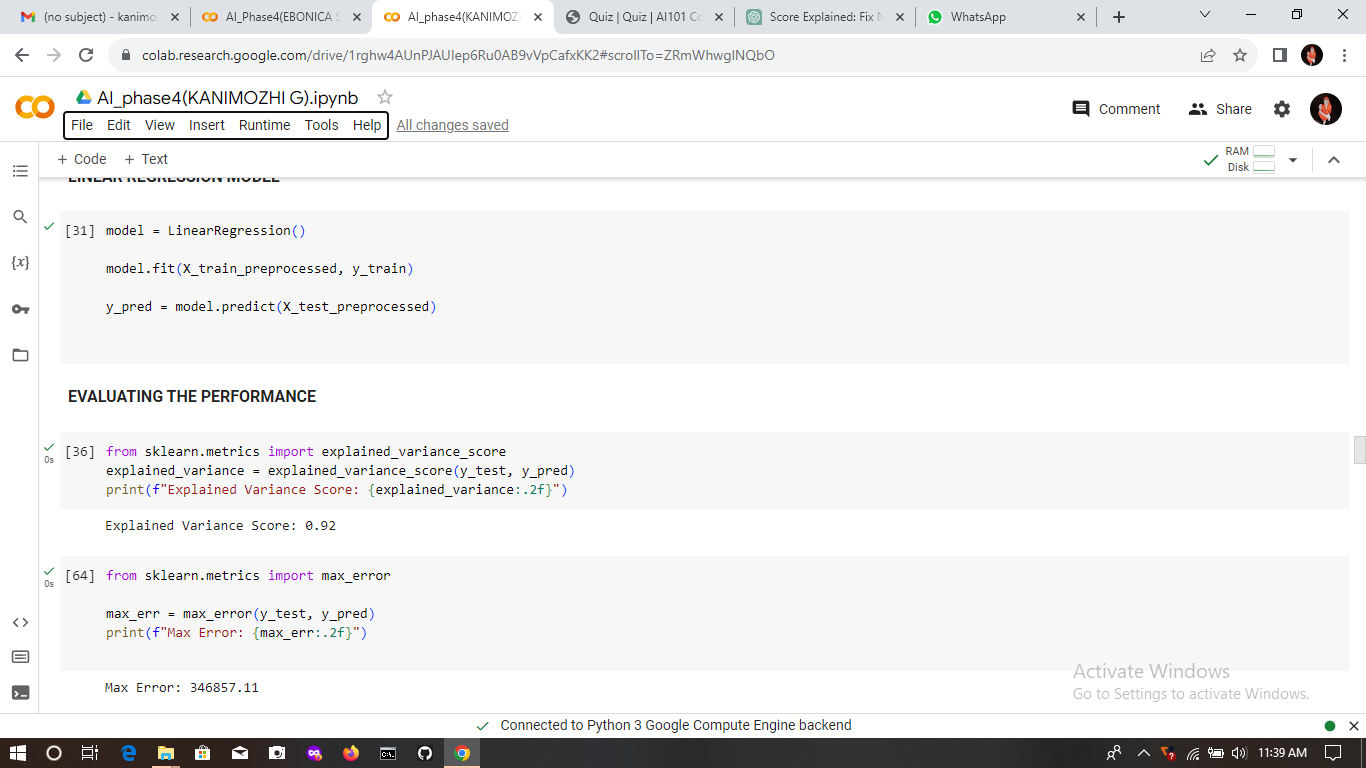
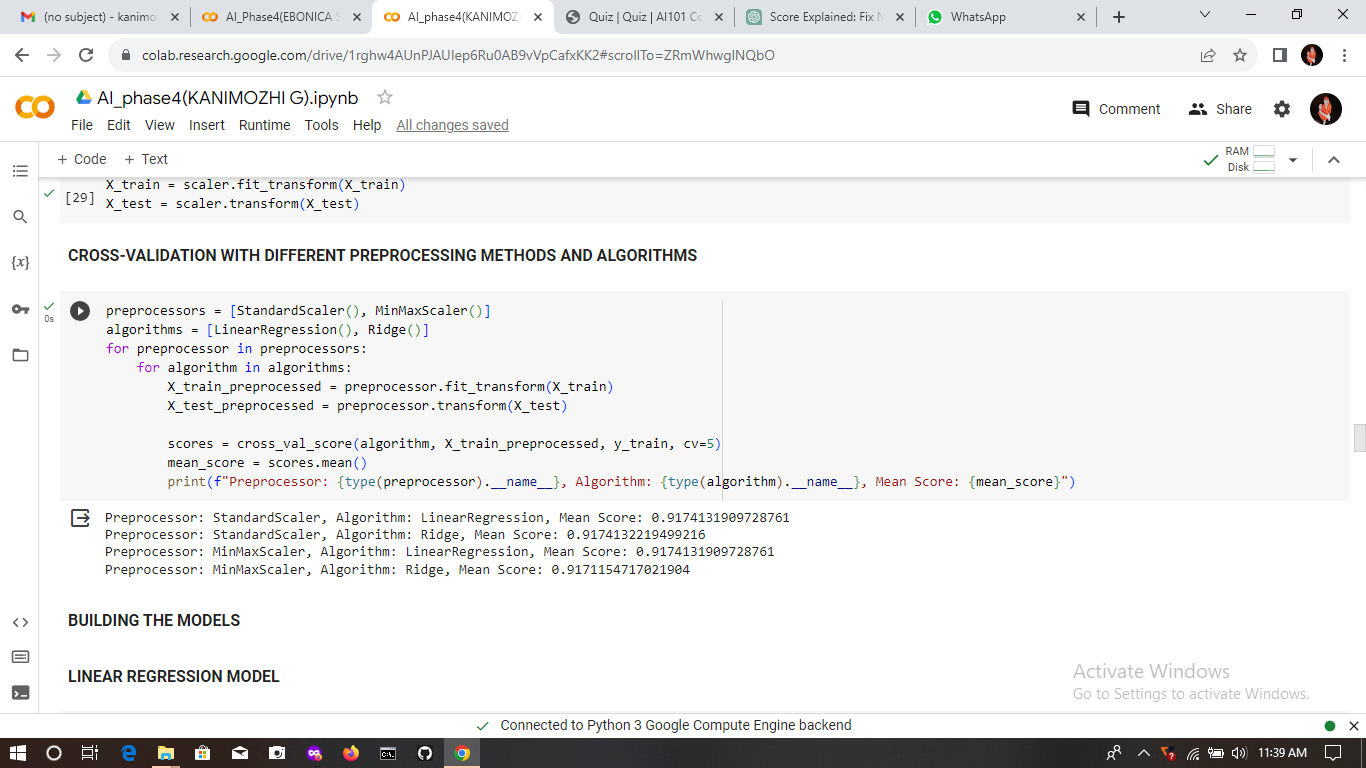
AI\_Phase4(KANIMOZHI G).ipynb

**Original file is located at:**

**PYTHON CODE WITH OUTPUT:**

**(FEATURE SELECTION, MODEL TRAINING AND MODEL EVALUATION STEPS)**

****

****

**CODE EXPLANATION:**

**Introduction:**

In this document, we will walk through the process of predicting house prices using various machine learning techniques. This comprehensive workflow involves data exploration, data preprocessing, model training, and performance evaluation. The primary objective is to create a model that can accurately predict house prices based on a set of features.

**1. Importing Libraries:**

We begin by importing the necessary libraries and modules to facilitate our analysis. These libraries include pandas for data manipulation, scikit-learn for machine learning, numpy for numerical operations, and matplotlib and seaborn for data visualization.

**CODE:**

import pandas as pd

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression, Ridge

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.svm import SVR

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

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**2. Loading the Dataset:**

We loaded the dataset named "USA\_Housing.csv" into our working environment(Google colab) using Google Colab. This dataset serves as the foundation for our house price prediction analysis.

**CODE:**

from google.colab import files

uploaded = files.upload()

data = pd.read\_csv("USA\_Housing.csv")

**3. Data Exploration:**

In this phase, we conducted a thorough exploration of the dataset to gain a better understanding of its structure and content.

**CODE:**

**# Displays the first few rows of the dataset**

print("First few rows of the dataset:")

print(data.head())

**# Displays the last few rows of the dataset**

print("Last few rows of the dataset:")

print(data.tail())

**# Provides dataset information, including data types, non-null values, and memory usage**

print("Dataset Information:")

print(data.info())

**# Calculates summary statistics to obtain a statistical overview of the numerical variables**

print("\nSummary statistics:")

print(data.describe())

**# Identifies and addresses missing values within the dataset**

print("\nMissing Values:")

print(data.isnull().sum())

**# Lists the column names for reference**

print("\nColumns:")

print(data.columns)

**# Determines the shape of the dataset in terms of rows and columns**

print("\nShape:")

print(data.shape)

**# Examines data types**

print("\nDATA TYPES:")

print(data.dtypes)

**# Accesses a specific row (index 20)**

data.iloc[20]

**# Calculates the number of unique values in each column**

unique\_counts = data.nunique()

print("Number of unique values in each column:")

print(unique\_counts)

**4. Data Visualization:**

Data visualization is crucial for understanding the relationships between variables and identifying patterns within the data. We performed several data visualization tasks.

**CODE:**

**# Creates a correlation heatmap to visualize relationships between features**

correlation\_matrix = data.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.show()

**# Constructs a histogram of house prices to understand their distribution**

plt.figure(figsize=(8, 6))

sns.histplot(data['Price'], kde=True)

plt.xlabel("House Price")

plt ylabel("Frequency")

plt.title("Histogram of House Prices")

plt.show()

**# Generates scatter plots to visualize the relationships between house prices and specific features**

**# Scatter plot of Price vs. Avg. Area Income**

sns.scatterplot(x='Avg. Area Income', y='Price', data=data)

plt.title("Price vs. Avg. Area Income")

plt.xlabel("Avg. Area Income")

plt.ylabel("Price")

plt.show()

**# Scatter plot of Avg. Area House Age vs. Price**

sns.scatterplot(x='Avg. Area House Age', y='Price', data=data)

plt.title("Price vs. Avg. Area House Age")

plt.xlabel("Avg. Area House Age")

plt.ylabel("Price")

plt.show()

**# Scatter plot of Avg. Area Number of Rooms vs. Price**

sns.scatterplot(x='Avg. Area Number of Rooms', y='Price', data=data)

plt.title("Price vs. Avg. Area Number of Rooms")

plt.xlabel("Avg. Area Number of Rooms")

plt.ylabel("Price")

plt.show()

**# Scatter plot of Area Population vs. Price**

sns.scatterplot(x='Area Population', y='Price', data=data)

plt.title("Price vs. Area Population")

plt.xlabel("Area Population")

plt.ylabel("Price")

plt.show()

**5. Splitting the Dataset into Features and Target Variable:**

The dataset is divided into two main components:

**Features (X):** These are the predictor variables, such as Avg. Area Income, Avg. Area House Age, Avg. Area Number of Rooms, Avg. Area Number of Bedrooms and Area Population.

**Target Variable (y):** This represents the variable we aim to predict, which is the "Price" of the houses.

**CODE:**

X = data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]

y = data['Price']

**6. Preprocessing the Dataset Using MinMax Scaler:**

We preprocessed the features using MinMaxScaler to standardize the feature values.

**CODE:**

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

**7. Feature Selection:**

We excluded the "Address" column from the features, as it is not directly related to the prediction of house prices.

**CODE:**

**# Excluding the 'Address' column as it is not directly related to the House Prices**

X = data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]

**8. Splitting the Dataset into Training and Testing Sets:**

To assess the model's performance, we divided the dataset into training and testing sets with an 80/20 split. This allows us to train the model on one portion of the data and evaluate its performance on another, unseen portion.

**CODE:**

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

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

**9. Preprocessing the Dataset Using Standard Scaler:**

In addition to MinMaxScaler, we also applied StandardScaler to preprocess the features. This further standardizes the data for modeling.

**CODE:**

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**10. Cross-Validation with Different Preprocessing Methods and Algorithms:**

To determine the effectiveness of different preprocessing methods and machine learning algorithms, we performed cross-validation. The methods tested include StandardScaler and MinMaxScaler, and the algorithms tested are Linear Regression and Ridge Regression. Mean scores are calculated to evaluate each combination's performance.

**CODE:**

preprocessors = [StandardScaler(), MinMaxScaler()]

algorithms = [LinearRegression(), Ridge()]

for preprocessor in preprocessors:

for algorithm in algorithms:

X\_train\_preprocessed = preprocessor.fit\_transform(X\_train)

X\_test\_preprocessed = preprocessor.transform(X\_test)

scores = cross\_val\_score(algorithm, X\_train\_preprocessed, y\_train, cv=5)

mean\_score = scores.mean()

print(f"Preprocessor: {type(preprocessor).\_\_name\_\_}, Algorithm: {type(algorithm).\_\_name\_\_}, Mean Score: {mean\_score}")

**11. Building the Models:**

**Linear Regression Model:**

We trained a Linear Regression model using the preprocessed training data. This model is commonly used for regression tasks.

**CODE:**

model = LinearRegression()

**# Training the model**

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

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

**Random Forest Model:**

A Random Forest Regressor, a powerful ensemble model, is trained and evaluated. Feature importance is calculated to understand which features have the most significant impact on the predictions.

**CODE:**

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

**# Training the model**

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

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

**Gradient Boosting Model:**

We also trained and evaluated a Gradient Boosting Regressor, another ensemble method known for its predictive power.

**CODE:**

model=GradientBoostingRegressor(n\_estimators=100,random\_state=42)

**# Training the model**

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

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

**Support Vector Regression Model:**

We applied Support Vector Regression (SVR) with a linear kernel to the dataset and evaluated its performance.

**CODE:**

model = SVR(kernel='linear')

**# Training the model**

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

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

**12. Performance Evaluation:**

For each model, we calculated two key metrics:

**Explained variance score:**The explained\_variance\_score is a function in scikit-learn, a popular machine learning library in Python. It is used to calculate the explained variance of a prediction. This metric is often used to evaluate the performance of regression models.

**Max error:**The max\_error function in scikit-learn is a metric used for evaluating the performance of regression models. It calculates the maximum residual error between the true target values and the predicted values. The residual error is the absolute difference between the actual and predicted values, and the max\_error function identifies the largest such difference.

We also created scatter plots to visualize the relationship between actual and predicted house prices, providing an intuitive view of model performance.We assessed the performance of each machine learning model using various metrics, including explained variance score and max error. Additionally, we created scatter plots to visualize how well the models predicted house prices compared to actual values.

**CODE:**

**# Linear Regression Performance Evaluation**

from sklearn.metrics import explained\_variance\_score explained\_variance = explained\_variance\_score(y\_test, y\_pred) print(f"Explained Variance Score: {explained\_variance:.2f}")

from sklearn.metrics import max\_error max\_err = max\_error(y\_test, y\_pred) print(f"Max Error: {max\_err:.2f}") print("Predicted Values:") print(y\_pred) results\_df = pd.DataFrame({'Actual Values (y\_test)': y\_test, 'Predicted Values (y\_pred)': y\_pred}) print(results\_df) plt.scatter(y\_test, y\_pred, alpha=0.5) plt.xlabel("Actual Prices") plt.ylabel("Predicted Prices") plt.title("Actual vs. Predicted House Prices") plt.show()

**# Random Forest Performance Evaluation**

from sklearn.metrics import explained\_variance\_score explained\_variance = explained\_variance\_score(y\_test, rany\_pred) print(f"Explained VarianceScore:{explained\_variance:.2f}") from sklearn.metrics import max\_error max\_err = max\_error(y\_test, rany\_pred) print(f"Max Error: {max\_err:.2f}") print("Predicted Values:") print(rany\_pred) results\_df = pd.DataFrame({'Actual Values (y\_test)': y\_test, 'Predicted Values (y\_pred)': y\_pred}) print(results\_df) plt.scatter(y\_test, y\_pred, alpha=0.5) plt.xlabel("Actual Prices") plt.ylabel("Predicted Prices") plt.title("Actual vs. Predicted House Prices") plt.show()

**# Gradient Boosting Performance Evaluation**

from sklearn.metrics import explained\_variance\_score explained\_variance = explained\_variance\_score(y\_test, grady\_pred) print(f"Explained Variance Score: {explained\_variance:.2f}") from sklearn.metrics import max\_error max\_err = max\_error(y\_test, grady\_pred) print(f"Max Error: {max\_err:.2f}") print("Predicted Values:") print(grady\_pred) results\_df = pd.DataFrame({'Actual Values (y\_test)': y\_test, 'Predicted Values (y\_pred)': grady\_pred}) print(results\_df) plt.scatter(y\_test, grady\_pred, alpha=0.5) plt.xlabel("Actual Prices") plt.ylabel("Predicted Prices") plt.title("Actual vs. Predicted House Prices") plt.show()

**#Support vector regression performance evaluation**

from sklearn.metrics import explained\_variance\_score explained\_variance = explained\_variance\_score(y\_test, supy\_pred) print(f"Explained Variance Score: {explained\_variance:.2f}") from sklearn.metrics import max\_error max\_err = max\_error(y\_test, supy\_pred) print(f"Max Error: {max\_err:.2f}") print("Predicted Values:") print(supy\_pred) results\_df = pd.DataFrame({'Actual Values (y\_test)': y\_test, 'Predicted Values (y\_pred)': supy\_pred}) print(results\_df) plt.scatter(y\_test, supy\_pred, alpha=0.5) plt.xlabel("Actual Prices") plt.ylabel("Predicted Prices") plt.title("Actual vs. Predicted House Prices") plt.show()

**Conclusion:**

In conclusion, this document outlines the entire process of predicting house prices using machine learning, from data exploration to model building and evaluation. The results obtained from different models and preprocessing techniques can guide decisions on selecting the most suitable approach for accurately predicting house prices.