KINGSTON ENGINEERING COLLEGE-5113

ARTIFICIAL INTELLIGENCE - PHASE 5

TOPIC: PREDICTING HOUSE PRICES USING

MACHINE LEARNING

TEAM MEMBERS:

V. EBONICA SALETH-

au511321104025(ebonicasalethvincent@gm

ail.com)

G. KANIMOZHI- au511321104040(

kanimozhigopal2004@gmail.com)

D. AKSHAYA-

au511321104004(Akshayadayalan03@gmail.

com)

K. INDUJA-

au511321104031(indujaindu137@gmail.com

)

PROJECT DOCUMENTATION<https://colab.research.google.com/drive/1Ni6DC-X7Sioi3yAU0ELe2ZoWex4618O-?usp=sharing>

PROBLEM STATEMENT:

The project aimed to address the critical task of predicting house

prices using machine learning techniques. The primary objective was

to develop a robust and accurate predictive model capable of

estimating the selling prices of houses based on a variety of relevant

features such as location, square footage, number of bedrooms and

bathrooms, and other factors. The project underwent a meticulously

structured process encompassing data exploration, preprocessing,

feature selection, model training, and comprehensive performance

evaluation.

DATASET DESCRIPTION:

This dataset is designed for the purpose of predicting house prices

using machine learning. Each row in the dataset represents a

different residential property or house. The dataset contains the

following features:

 Avg. Area Income: This numerical feature represents the

average income of residents in the area where each house is

located. It provides insight into the economic status of the area.

 Avg. Area House Age: This numerical feature indicates the

average age of houses in the area. It gives an idea of the age

and condition of properties in the neighbourhood.

 Avg. Area Number of Rooms: This numerical feature shows the

average number of rooms in houses within the area. It helps

understand the size of houses.

 Avg. Area Number of Bedrooms: This numerical feature

represents the average number of bedrooms in houses in the

area. It provides information about the accommodation

capacity.

 Area Population: This numerical feature reflects the population

of the area where each house is situated. It can be indicative of

the local community&#39;s size.

These features are used to predict the target variable:

 Price: The target variable is the price of each house. The goal of

the machine learning task is to build a predictive model that

can estimate house prices based on the provided features.

The dataset is a valuable resource for regression analysis and

predictive modelling in the real estate domain, allowing for the

development of machine learning models to make accurate house

price predictions based on the given property characteristics. The

&quot;Address&quot; column, while descriptive, may require further

preprocessing or encoding if included in the modelling process.

DESIGN THINKING PROCESS:

1. Problem Statement:

 The problem was to predict house prices based on various

features such as location, square footage, number of

bedrooms and bathrooms, and other factors.

 The problem was defined as predicting house prices as

accurately as possible.

2. Data Acquisition and Understanding:

 The necessary Python libraries for data analysis, visualization,

and machine learning were imported.

3. Data Collection:

 We obtained our dataset from Kaggle. This dataset includes

essential features such as location, square footage, number of

bedrooms and bathrooms, and, most importantly, the house

price. Having access to real-world data is essential for training a

predictive model.

 DATASET

LINK: https://www.kaggle.com/datasets/vedavyasv/usa-

housing

4. Data Exploration:

 The dataset was loaded into a Pandas Data Frame.

 The data was explored to understand its structure and

contents.

 The first and last few rows were displayed.

 Data types were checked.

 Missing values were identified.

 Summary statistics were calculated.

 The columns and shape of the dataset were examined.

 The data types of the columns were listed.

 The 21st row (index 20) was accessed.

 The number of unique values in each column was determined.

5. Data Visualization:

 A correlation heatmap was generated to visualize the

relationships between features.

 A histogram of house prices was created to understand the

distribution.

 Scatter plots were generated to explore the relationships

between average area income, house age, number of rooms,

population, and house prices.

6. Data Preprocessing:

 Feature selection was performed, excluding the &#39;Address&#39;

column.

 The dataset was split into training and testing sets for model

training and evaluation.

7. Data Preprocessing and Transformation:

 Data was normalized using Min Max Scaler and Standard Scaler

for feature scaling.

 Cross-validation was applied with different preprocessing

methods and algorithms to assess model performance.

8. Model Building:

 A linear regression model was built and trained on the pre

processed training data.

 A random forest regression model was created and trained on

the training data.

 A gradient boosting regression model was developed and

trained on the training data.

 A support vector regression model was implemented and

trained on the training data.

9. Model Evaluation:

·            Mean Absolute Error (MAE), also known as the Mean Absolute Deviation (MAD), is a common metric used in statistics and machine learning to measure the average absolute difference between the observed (actual) values and the predicted values in a dataset. Like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), MAE is used to evaluate the performance of regression models.

·      The formula for calculating Mean Absolute Error is:

                          MAE = (1/n) \* Σ|yi - ŷi|

**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.

 Model performance was evaluated by calculating Mean

Squared Error (MSE) and max error score for each model.

 Actual vs. predicted house prices were visualized using

scatterplots.

 Feature importance was visualized for the Random Forest

model.

10. Conclusion:

 This project underscored the significance of data

preprocessing, feature selection, and model evaluation in

constructing accurate predictive models. These models can

offer invaluable support in making informed decisions in

the real estate domain.

PHASES OF DEVELOPMENT:

Phase 1: Problem Definition and Design

Thinking

In this phase, we aimed to understand the problem statement and

create a clear plan for solving it. We started by comprehensively

defining the problem and our approach.

Problem Definition:

We thoroughly analyzed the problem statement, which was to

predict house prices based on several key features. The dataset,

&quot;USA\_Housing&quot;, was provided for this task.

Design Thinking:

To solve this problem, we followed a structured design thinking

process. We broke it down into the following steps:

 Empathize: We empathized with the problem by understanding

the dataset and the features it contained.

 Define: We clearly defined the problem statement and our

goal, which was to build a predictive model for house prices.

 Ideate: We brainstormed potential approaches, including

innovative techniques and algorithms.

 Prototype: We created a plan for the project, outlining the

phases of development.

Phase 2: Innovation

The innovation phase was a critical step in which we explored

cutting-edge techniques to maximize the prediction system&#39;s

accuracy and resilience. We delved into advanced regression

methods, including Gradient Boosting and XG Boost, to enhance the

predictive accuracy of our models. The integration of these advanced

algorithms was a pivotal aspect of our project, aimed at pushing the

boundaries of traditional regression analysis.

Phase 3: Development Part 1

This phase marked the inception of our house price prediction

model. We initiated the journey by loading the dataset and diligently

preparing it for subsequent stages. The pivotal steps involved in this

phase were as follows:

 Dataset Loading:

We sourced the housing dataset from an external repository,

ensuring that it provided comprehensive and relevant

information.

 Data Exploration:

To gain an insightful understanding of the dataset, we

conducted an exploratory data analysis. We inspected the

initial and concluding rows of the dataset, assessed its overall

structure, and scrutinized summary statistics. Moreover, we

meticulously inspected the data for any missing values. An in-

depth examination of the dataset&#39;s columns and their data

types was carried out to understand the scope and nature of

the data.

 Specific Row Access:

We went a step further to access a specific row (in this case,

the 21st row) to analyze the dataset&#39;s granularity and integrity.

 Unique Value Counts:

We meticulously assessed the number of unique values in each

column, providing detailed insight into the diversity and

distribution of data within the dataset.

 Preprocessing the Dataset:

training. We applied data scaling techniques, such as Min-Max

We also pre processed the dataset to prepare it for model

scaling and Standard scaling, to standardize the feature values.

Min-Max scaling transformed the data to a common scale,

typically between 0 and 1, while Standard scaling standardized

the data by centring it around the mean and scaling it to have a

standard deviation of 1. These preprocessing steps were

necessary to ensure that all features had equal influence on the

models and to improve their convergence during training.

 Cross-Validation:

To further assess the performance of our models, we

conducted cross-validation with different preprocessing

methods and algorithms. Specifically, we used a 5-fold cross-

validation strategy to train and validate our models multiple

times with different subsets of the data. This helped us

estimate how well our models would generalize to unseen data

and provided a more robust evaluation of their performance.

Phase 4: Development Part 2

Building upon the groundwork laid in the previous phase, we

continued the development process. This phase encompassed four

fundamental activities:

Data Visualization:

 Correlation Heatmap:

We used a correlation heatmap to visualize the relationships

between various features, helping us identify which features

were most influential.

 Histogram of House Prices:

A histogram with a kernel density estimate revealed the

distribution and patterns in house prices.

 Scatter Plots:

We created scatter plots to explore the relationships between

house prices and key factors like average area income, house

age, number of rooms, and population density. These plots

provided a quick visual assessment of potential trends and

outliers.

Feature Selection:

To enhance the quality of our models, we performed feature

selection. In this process, we decided to exclude the &#39;Address&#39;

column from our dataset. We made this decision because &#39;Address&#39;

was not directly related to house prices and was, therefore, not

considered a valuable feature for our predictive models. This careful

feature selection process helped streamline our data and improve

the overall performance of our models by focusing on the most

relevant attributes.

Model Training:

We initiated the training of our prediction model. This was a pivotal

stage where we harnessed the power of various machine learning

algorithms, including Linear Regression, Random Forest, Gradient

Boosting, and Support Vector Regression (SVR). Each of these

algorithms played a crucial role in providing diverse perspectives on

the data and delivering predictive models. Specifically, we have

trained the models by fitting them to the training data using the

historical features of houses, such as location, square footage,

number of bedrooms and bathrooms. The algorithms were employed

to learn the underlying patterns and relationships within the dataset.

Performance Evaluation:

After training each of these models, we rigorously evaluated their

performance. For each model, we calculated Mean Squared Error

(MSE) and R-squared (R2) scores to quantitatively assess their

predictive accuracy and capability. These evaluation metrics allowed

us to compare the performance of the different models and select

the one that performed the best on our dataset. Additionally, we

visualized the results by creating scatter plots, which allowed us to

visualize the alignment between actual house prices and the

predicted values. These plots helped us to identify any trends or

discrepancies in the models predictions.

THE CHOICE OF REGRESSION ALGORITHM

AND EVALUATION METRICS:

Choice of Regression Algorithm:

1. Linear Regression:

When the relationship between the independent variables and the

target variable appears to be approximately linear, linear regression

is a straightforward choice. It’s a simple and interpretable model.

2. Support Vector Regression:

Support Vector Regression (SVR) is useful when having a clear margin

of separation between different target values and want to find a

hyperplane that fits as many data points as possible.

3. Gradient Boosting Regressors:

Algorithms like Gradient Boosting (e.g., XGBoost, LightGBM, or

CatBoost) are powerful for complex regression tasks. They build

ensembles of decision trees to capture nonlinear relationships.

4. Random Forests:

Random Forest, are useful when the relationship between features

and the target variable is nonlinear. They can handle both numerical

and categorical features.

Choice of Evaluation Metrics:

**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.

MEAN ABSOLUTE ERROR

·            Mean Absolute Error (MAE), also known as the Mean Absolute Deviation (MAD), is a common metric used in statistics and machine learning to measure the average absolute difference between the observed (actual) values and the predicted values in a dataset. Like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), MAE is used to evaluate the performance of regression models.

·      The formula for calculating Mean Absolute Error is:

                          MAE = (1/n) \* Σ|yi - ŷi|

PYTHON CODE WITH OUTPUT:

AI\_Phase5(AKSHAYA D).ipynb

Original file is located at:

https://colab.research.google.com/drive/11An0zXsS5h0b\_qorMMZh

-4WOuaCOP74a?usp=sharing

IMPLEMENTATION DETAILS:

DATASET DETAILS:

We will acquire our dataset from Kaggle, specifically the &quot;USA

Housing&quot; 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://colab.research.google.com/drive/1Ni6DC-X7Sioi3yAU0ELe2ZoWex4618O-?usp=sharing>

 Before uploading file (downloaded from Kaggle) in the Google

Colab convert the zip file to csv file using online converter:

 https://www.ezyzip.com/convert-zip-to-csv.html#

 Now download the csv file from the website and upload it in

the Google Colab.

Step 1: Importing Libraries

The code begins by importing the necessary Python libraries for data

manipulation, machine learning, and 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

Step 2: Loading the Given Dataset

This code snippet uses Google Colab&#39;s file upload feature to allow the

user to upload the dataset file, which is named &quot;USA\_Housing.csv.&quot;

CODE:

from google.colab import files

uploaded = files.upload()

Step 3: Reading the Given Dataset

This line reads the dataset from the uploaded CSV file into a pandas

DataFrame named &#39;data&#39;.

CODE:

data = pd.read\_csv(&quot;USA\_Housing.csv&quot;)

Step 4: Data Exploration

In this section, the code explores the dataset, providing various

insights about it. These code snippets print the first few and last few

rows, general information, summary statistics, missing values,

columns, shape, data types, and access the 21st row of the dataset.

It also calculates and prints the number of unique values in each

column.

CODE:

print(&quot;First few rows of the dataset:&quot;)

print(data.head())

print(&quot;Last few rows of the dataset:&quot;)

print(data.tail())

print(&quot;Dataset Information:&quot;)

print(data.info())

print(&quot;\nSummary statistics:&quot;)

print(data.describe())

print(&quot;\nMissing Values:&quot;)

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

print(&quot;\nColumns:&quot;)

print(data.columns)

print(&quot;\nShape:&quot;)

print(data.shape)

print(&quot;\nDATA TYPES:&quot;)

print(data.dtypes)print(&quot;\nAccess the 21st row (index 20)&quot;)

data.iloc[20]

unique\_counts = data.nunique()

print(&quot;Number of unique values in each column:&quot;)

print(unique\_counts)

Step 5: Data Visualization

In this section, the code generates visualizations to explore the data.

These code snippets create a correlation heatmap, a histogram of

house prices, and multiple scatter plots to visualize the relationships

between various features and house prices.

CODE:

#Correlation Heatmap

correlation\_matrix = data.corr()

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

sns.heatmap(correlation\_matrix, annot=True, cmap=&quot;coolwarm&quot;)

plt.title(&quot;Correlation Heatmap&quot;)

plt.show()

#Histogram of House Prices

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

sns.histplot(data[&#39;Price&#39;], kde=True)

plt.xlabel(&quot;House Price&quot;)

plt.ylabel(&quot;Frequency&quot;)

plt.title(&quot;Histogram of House Prices&quot;)

plt.show()

# Scatter plot of Avg. Area Income vs. Price

sns.scatterplot(x=&#39;Avg. Area Income&#39;, y=&#39;Price&#39;, data=data)

plt.title(&quot;Price vs. Avg. Area Income&quot;)

plt.xlabel(&quot;Avg. Area Income&quot;)

plt.ylabel(&quot;Price&quot;)

plt.show()

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

sns.scatterplot(x=&#39;Avg. Area House Age&#39;, y=&#39;Price&#39;, data=data)

plt.title(&quot;Price vs. Avg. Area House Age&quot;)

plt.xlabel(&quot;Avg. Area House Age&quot;)

plt.ylabel(&quot;Price&quot;)

plt.show()

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

sns.scatterplot(x=&#39;Avg. Area Number of Rooms&#39;, y=&#39;Price&#39;, data=data)

plt.title(&quot;Price vs. Avg. Area Number of Rooms&quot;)

plt.xlabel(&quot;Avg. Area Number of Rooms&quot;)

plt.ylabel(&quot;Price&quot;)

plt.show()

# Scatter plot of Area Population vs. Price

sns.scatterplot(x=&#39;Area Population&#39;, y=&#39;Price&#39;, data=data)

plt.title(&quot;Price vs. Area Population&quot;)

plt.xlabel(&quot;Area Population&quot;)

plt.ylabel(&quot;Price&quot;)

plt.show()

Step 6: Splitting the Dataset into Features (X)

and Target Variable (y)

This code snippet splits the dataset into feature columns (&#39;X&#39;) and the

target variable (&#39;y&#39;).

CODE:

X = data[[&#39;Avg. Area Income&#39;, &#39;Avg. Area House Age&#39;, &#39;Avg. Area

Number of Rooms&#39;, &#39;Avg. Area Number of Bedrooms&#39;, &#39;Area

Population&#39;]]

y = data[&#39;Price&#39;]

Step 7: Preprocessing the Dataset Using Min

Max Scaler

It uses Min Max Scaler to scale the feature values between 0 and 1.

CODE:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

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

Step 8: Feature Selection

The &#39;Address&#39; column is excluded from the features since it is not

directly related to house prices.

CODE:

X = data[[&#39;Avg. Area Income&#39;, &#39;Avg. Area House Age&#39;, &#39;Avg. Area

Number of Rooms&#39;, &#39;Avg. Area Number of Bedrooms&#39;, &#39;Area

Population&#39;]]

Step 9: Splitting the Dataset into Training

and Testing Sets

The dataset is split into training and testing sets, with 80% of the

data used for training and 20% for testing. The random\_state

parameter ensures reproducibility.

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)

Step 10: Saving the Train and Test Datasets

This code saves the training and testing datasets to CSV files.

CODE:

X\_train.to\_csv(&quot;X\_train.csv&quot;, index=False)

X\_test.to\_csv(&quot;X\_test.csv&quot;, index=False)

y\_train.to\_csv(&quot;y\_train.csv&quot;, index=False)

y\_test.to\_csv(&quot;y\_test.csv&quot;, index=False)

Step 11: Preprocessing the Dataset Using

Standard Scaler

In this step, the code uses the Standard Scaler to standardize the

feature values. This means it scales the features to have a mean of 0

and a standard deviation of 1.

CODE:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

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

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

Step 12: Cross-Validation with Different

Preprocessing Methods and Algorithms

In this step, the code performs cross-validation with different

preprocessing methods and algorithms. It iterates through the

preprocessors and algorithms lists, applies the selected

preprocessing to the training and testing data, and then evaluates

the models using 5-fold cross-validation. It prints the mean score for

each combination of preprocessing and algorithm.

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&quot;Preprocessor: {type(preprocessor).\_\_name\_\_},

Algorithm: {type(algorithm).\_\_name\_\_}, Mean Score: {mean\_score}&quot;)

Step 13: Building the Models - Linear

Regression Model

Here, a Linear Regression model is created using LinearRegression()

from scikit-learn. The model is trained on the pre processed training

data, and predictions are made on the pre processed test data.

CODE:

model = LinearRegression()

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

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

Step 14: Evaluating the Performance

The code calculates and prints the Mean Squared Error (MSE) and

the R-squared (R2) score, which are metrics used to evaluate the

performance of the Linear Regression model. It also prints the

predicted values and displays a scatter plot of actual vs. predicted

house prices.

CODE:

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

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

print(&quot;Mean Squared Error:&quot;, mse)

print(&quot;R-squared (R2) Score:&quot;, r2)

results\_df = pd.DataFrame({&#39;Actual Values (y\_test)&#39;: y\_test,

&#39;Predicted Values (y\_pred)&#39;: y\_pred})

print(results\_df)

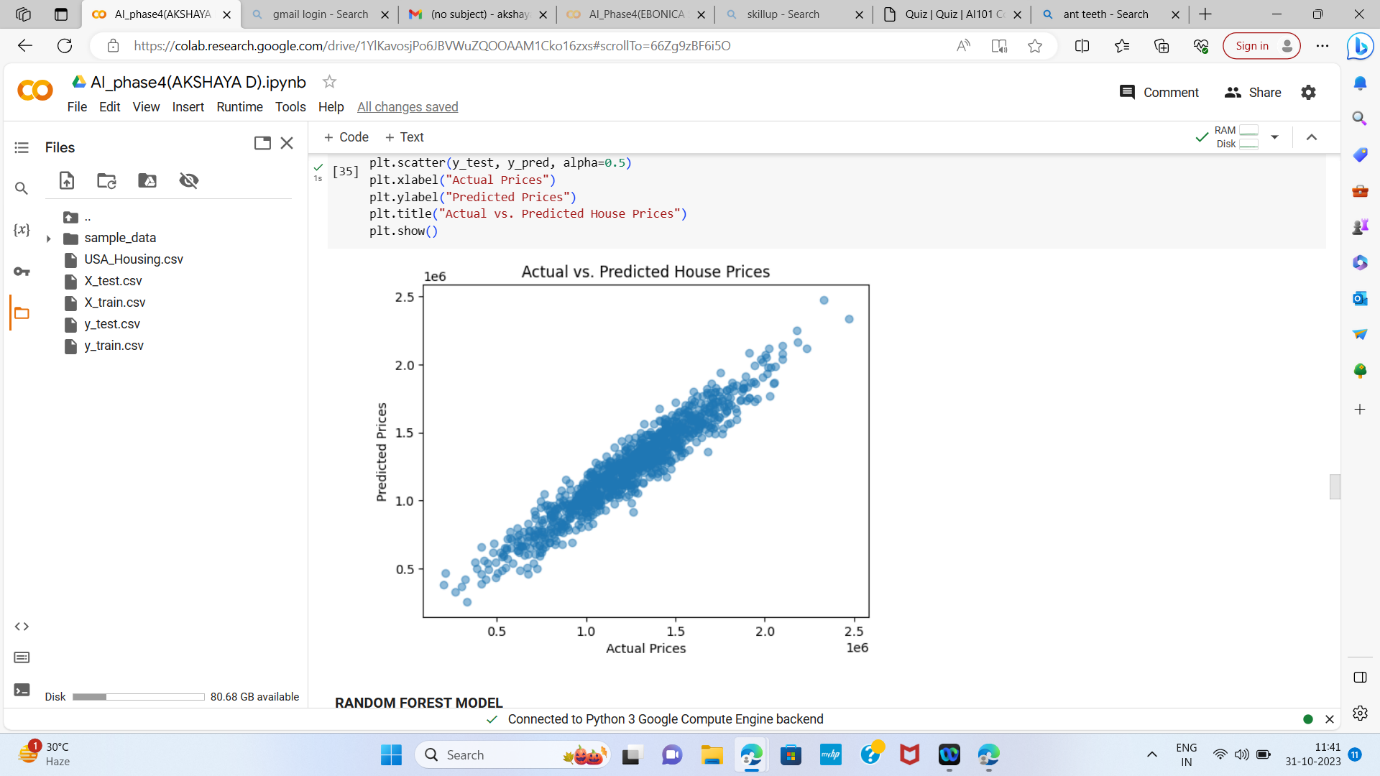
plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.xlabel(&quot;Actual Prices&quot;)

plt.ylabel(&quot;Predicted Prices&quot;)

plt.title(&quot;Actual vs. Predicted House Prices&quot;)

plt.show()



Step 15: Random Forest Model

Here, a Random Forest Regressor model is created with 100 trees.

The model is trained on the original (unprocessed) training data, and

predictions are made on the testing data.

CODE:

model = RandomForestRegressor(n\_estimators=100,

random\_state=42)

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

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

Step 16: Feature Importance

This section calculates and visualizes feature importances for the

Random Forest model. Feature importances indicate the

contribution of each feature to the model&#39;s predictions.

CODE:

feature\_importances = model.feature\_importances\_

# Creating a bar plot to visualize feature importances

plt.figure(figsize=(10, 6))

plt.bar(X.columns, feature\_importances)

plt.xlabel(&#39;Features&#39;)

plt.ylabel(&#39;Feature Importance&#39;)

plt.title(&#39;Feature Importance from Random Forest&#39;)

plt.show()

# Printing the feature importances

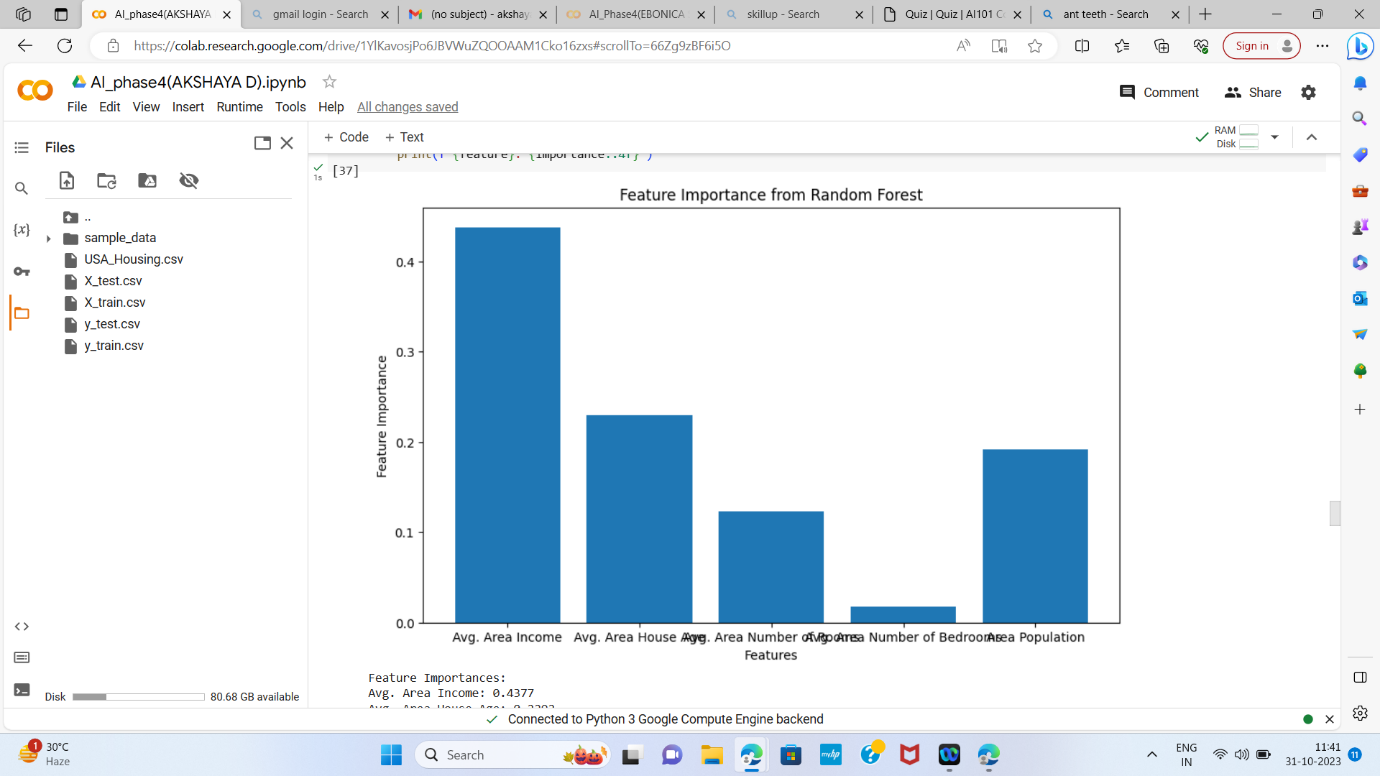
feature\_importance\_dict = dict(zip(X.columns,

feature\_importances))

print(&quot;Feature Importances:&quot;)

for feature, importance in feature\_importance\_dict.items():

print(f&quot;{feature}: {importance:.4f}&quot;)



Step 17: Evaluating the Performance -

Random Forest

The code calculates and prints the performance metrics (MSE and

R2) for the Random Forest model. It also prints the predicted values

and displays a scatter plot of actual vs. predicted house prices.

CODE:

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

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

print(f&quot;Random Forest - Mean Squared Error: {mse}&quot;)

print(f&quot;Random Forest - R-squared (R2) Score: {r2}&quot;)

results\_df = pd.DataFrame({&#39;Actual Values (y\_test)&#39;: y\_test,

&#39;Predicted Values (y\_pred)&#39;: rany\_pred})

print(results\_df)

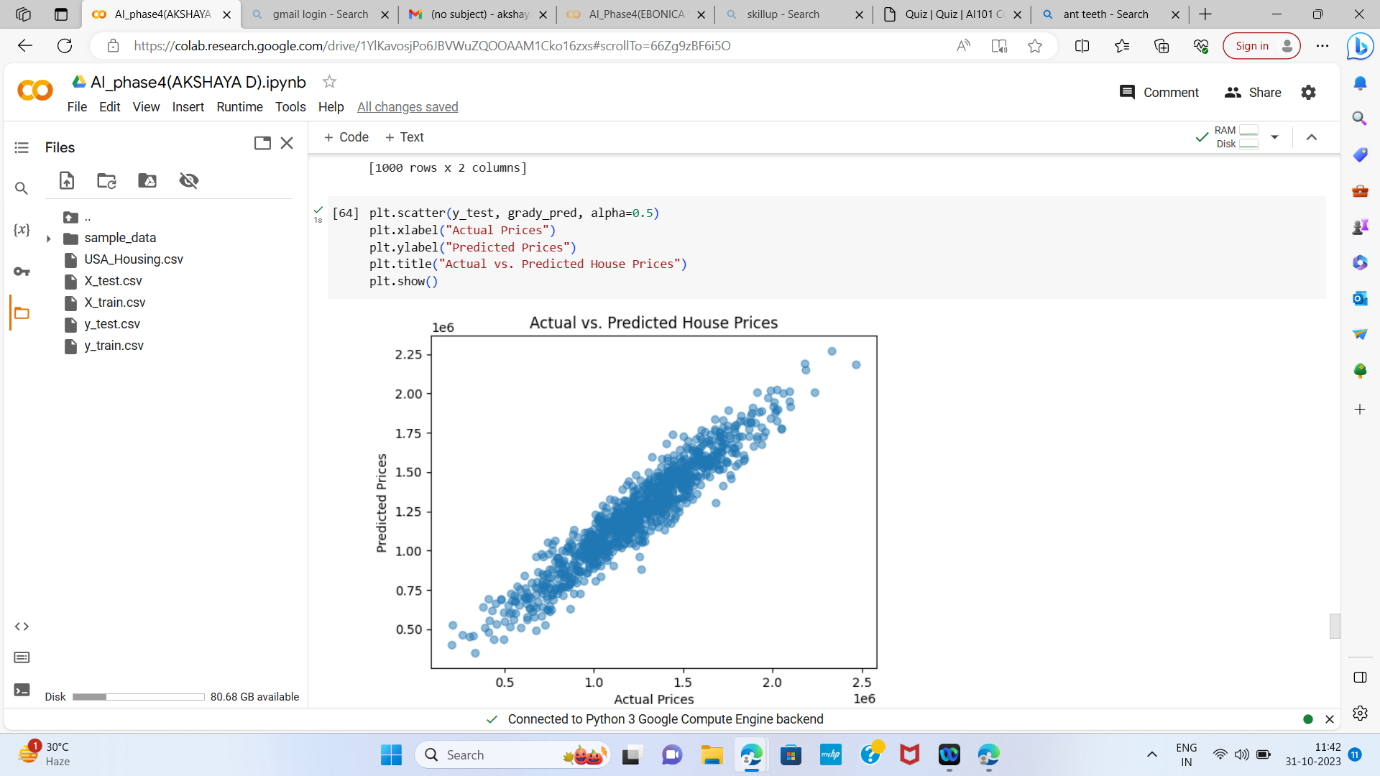
plt.scatter(y\_test, rany\_pred, alpha=0.5)

plt.xlabel(&quot;Actual Prices&quot;)

plt.ylabel(&quot;Predicted Prices&quot;)

plt.title(&quot;Actual vs. Predicted House Prices&quot;)

plt.show()



Step 18: Gradient Boosting Model

In this step, a Gradient Boosting Regressor model is created with 100

boosting iterations. The model is trained on the original training

data, and predictions are made on the testing data.

CODE:

model = GradientBoostingRegressor(n\_estimators=100,

random\_state=42)

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

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

Step 19: Evaluating the Performance -

Gradient Boosting

The code calculates and prints the performance metrics (MSE and

R2) for the Gradient Boosting model. It also prints the predicted

values and displays a scatter plot of actual vs. predicted house prices.

CODE:

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

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

print(f&quot;Gradient Boosting - Mean Squared Error: {mse}&quot;)

print(f&quot;Gradient Boosting - R-squared (R2) Score: {r2}&quot;)

results\_df = pd.DataFrame({&#39;Actual Values (y\_test)&#39;: y\_test,

&#39;Predicted Values (y\_pred)&#39;: grady\_pred})

print(results\_df)

plt.scatter(y\_test, grady\_pred, alpha=0.5)

plt.xlabel(&quot;Actual Prices&quot;)

plt.ylabel(&quot;Predicted Prices&quot;)

plt.title(&quot;Actual vs. Predicted House Prices&quot;)

plt.show()

Step 20: Support Vector Regression (SVR)

Model

In this step, a Support Vector Regression model with a linear kernel is

created. The model is trained on the original training data, and

predictions are made on the testing data.

CODE:

model = SVR(kernel=&#39;linear&#39;)

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

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

Step 21: Evaluating the Performance – SVR

The code calculates and prints the performance metrics (MSE and

R2) for the SVR model. It also prints the predicted values and displays

a scatter plot of actual vs. predicted house prices.

CODE:

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

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

print(f&quot;SVR - Mean Squared Error: {mse}&quot;)

print(f&quot;SVR - R-squared (R2) Score: {r2}&quot;)

results\_df = pd.DataFrame({&#39;Actual Values (y\_test)&#39;: y\_test,

&#39;Predicted Values (y\_pred)&#39;: supy\_pred})

print(results\_df)

plt.scatter(y\_test, supy\_pred, alpha=0.5)

plt.xlabel(&quot;Actual Prices&quot;)

plt.ylabel(&quot;Predicted Prices&quot;)

plt.title(&quot;Actual vs. Predicted House Prices&quot;)

plt.show()

RESULTS:

 The project yielded notable results, each indicative of the

model&#39;s predictive accuracy and performance.

 Linear Regression Model:

The Linear Regression model achieved a Mean Squared Error (MSE)

of approximately 10089009300.89399 indicating that, on average,

the squared difference between the model&#39;s predictions and the

actual house prices is around this value. A lower MSE is desirable,

and this value suggests that the model&#39;s predictions are relatively

close to the actual prices.

The R-squared (R2) score for the Linear Regression model is

approximately 0.918. The R2 score measures how well the model

explains the variance in the target variable. An R2 score of 1.0

represents a perfect fit, and an R2 score of 0.918 suggests that the

model explains about 91.8% of the variance in house prices. This

indicates a strong linear relationship between the features and the

target variable.

 Random Forest Model:

The Random Forest model achieved a Mean Squared Error (MSE) of

approximately 14,462,012,668.45. While the MSE is higher than that

of the Linear Regression model, it is still a relatively low value,

suggesting that the model&#39;s predictions are reasonably close to

actual prices.

The R-squared (R2) score for the Random Forest model is

approximately 0.882. This R2 score indicates that the model explains

about 88.2% of the variance in house prices, which is a strong

performance.

 Gradient Boosting Model:

The Gradient Boosting model achieved a Mean Squared Error (MSE)

of approximately 11,983,338,273.94. This MSE is lower than the

Random Forest model&#39;s MSE, indicating that the Gradient Boosting

model&#39;s predictions are even closer to the actual prices.

The R-squared (R2) score for the Gradient Boosting model is

approximately 0.903. This R2 score suggests that the model explains

about 90.3% of the variance in house prices, which is a very strong

performance.

 Support Vector Regression (SVR) Model:

The Support Vector Regression (SVR) model achieved a significantly

higher Mean Squared Error (MSE) of approximately 121,260,874,519.

99678 This indicates that the SVR model&#39;s predictions have a larger

squared difference from the actual prices. A high MSE suggests that

the model&#39;s predictions are less accurate.

The R-squared (R2) score for the SVR model is approximately 0.0144.

This R2 score is quite low, suggesting that the model explains only

about 1.44% of the variance in house prices. The SVR model appears

to perform poorly compared to the other models, as the R2 score is

close to 0, indicating weak predictive capabilities.

 In summary, the Linear Regression, Random Forest, and Gradient

Boosting models all demonstrate strong predictive accuracy, with

relatively low MSE values and high R2 scores. The Support Vector

Regression (SVR) model, on the other hand, exhibits significantly

higher MSE and a low R2 score, indicating a weaker predictive

performance in comparison to the other models.

CONCLUSION:

The project, through rigorous data exploration, preprocessing,

feature selection, model training, and evaluation, successfully

constructed predictive models for house price estimation. These

models not only provided valuable insights into the determinants of

house prices but also demonstrated a high level of predictive

accuracy. Such models hold great potential for real-world

applications, such as real estate market analysis, property valuation,

and investment decision-making.

FUTURE WORK:

To further enhance the predictive accuracy and applicability of the

models, future iterations of this project can explore more advanced

regression techniques and delve into deep learning architectures.

Moreover, expanding the dataset and incorporating additional

relevant features could provide a more comprehensive

understanding of the factors influencing house prices. Continuous

model refinement and optimization will be pivotal in maximizing the

utility of the predictive models.