KINGSTON ENGINEERING COLLEGE-5113

ARTIFICIAL INTELLIGENCE - PHASE 3

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

)

**CONTENT:**

PYTHON CODE WITH OUTPUT CREATED WITH GOOGLE

COLABORATORY

**CODE EXPLANATION:**

1. IMPORTING LIBRARIES

2. LOADING THE GIVEN DATASET

3. READING THE GIVEN DATASET

4. DATA EXPLORATION

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

VARIABLE (y)

6. PREPROCESSING THE DATASET USING MINMAX SCALER

7. EXCLUDING THE ADDRESS COLUMN

8. SPLITTING THE DATASET INTO TRAINING AND TESTING SETS

9. SAVING THE TRAIN AND TEST DATASETS

10. PREPROCESSING THE DATASET USING STANDARD SCALER

11. CROSS-VALIDATION WITH DIFFERENT PREPROCESSING METHODS

AND ALGORITHMS

12. BUILDING THE MODELS

**LIBRARIES USED:**

Pandas

Sklearn

Numpy

**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

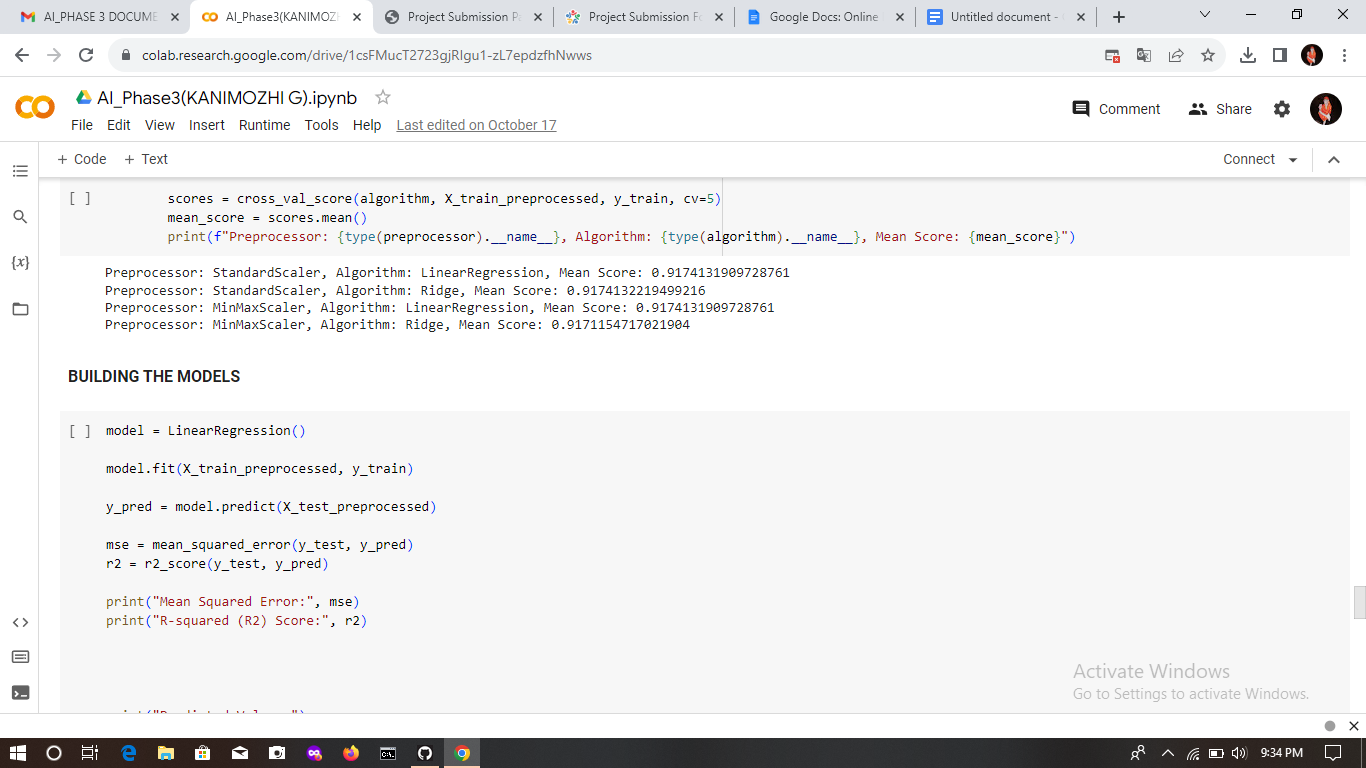
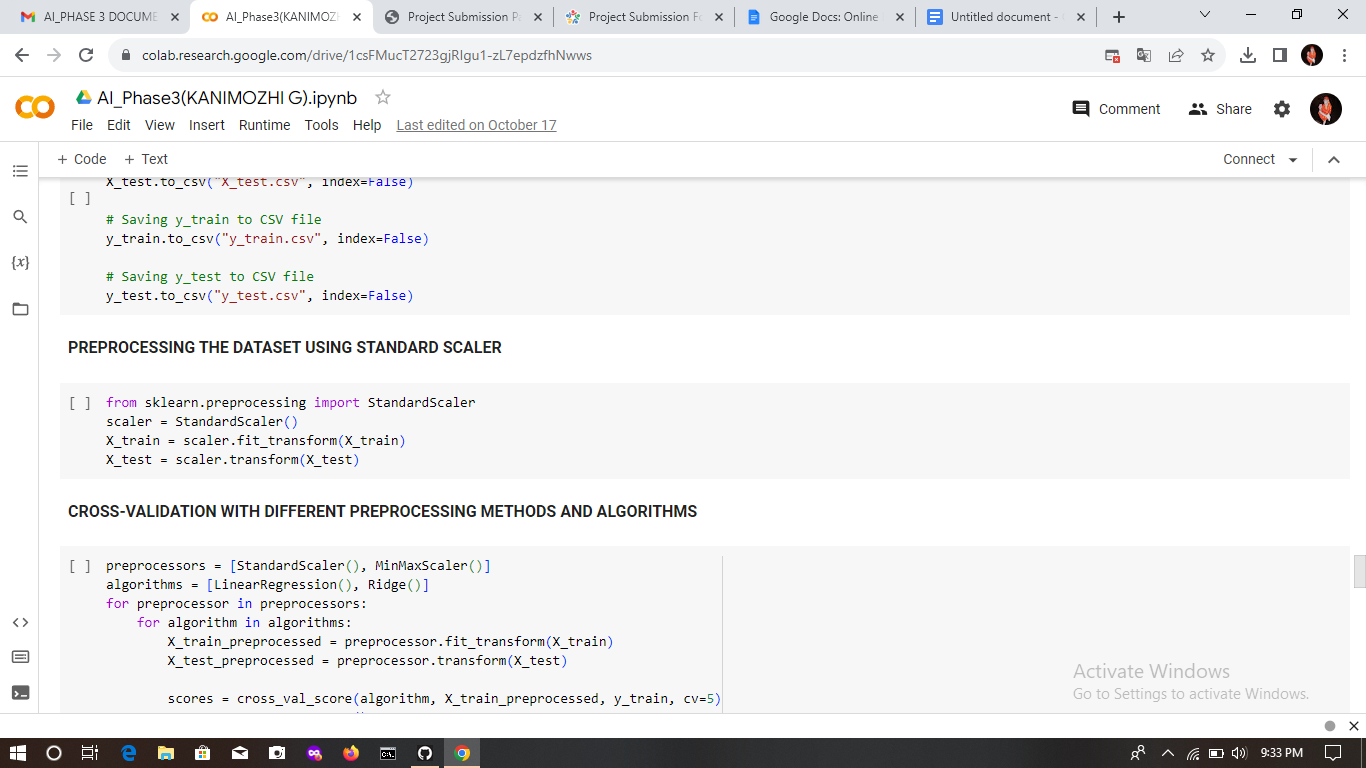
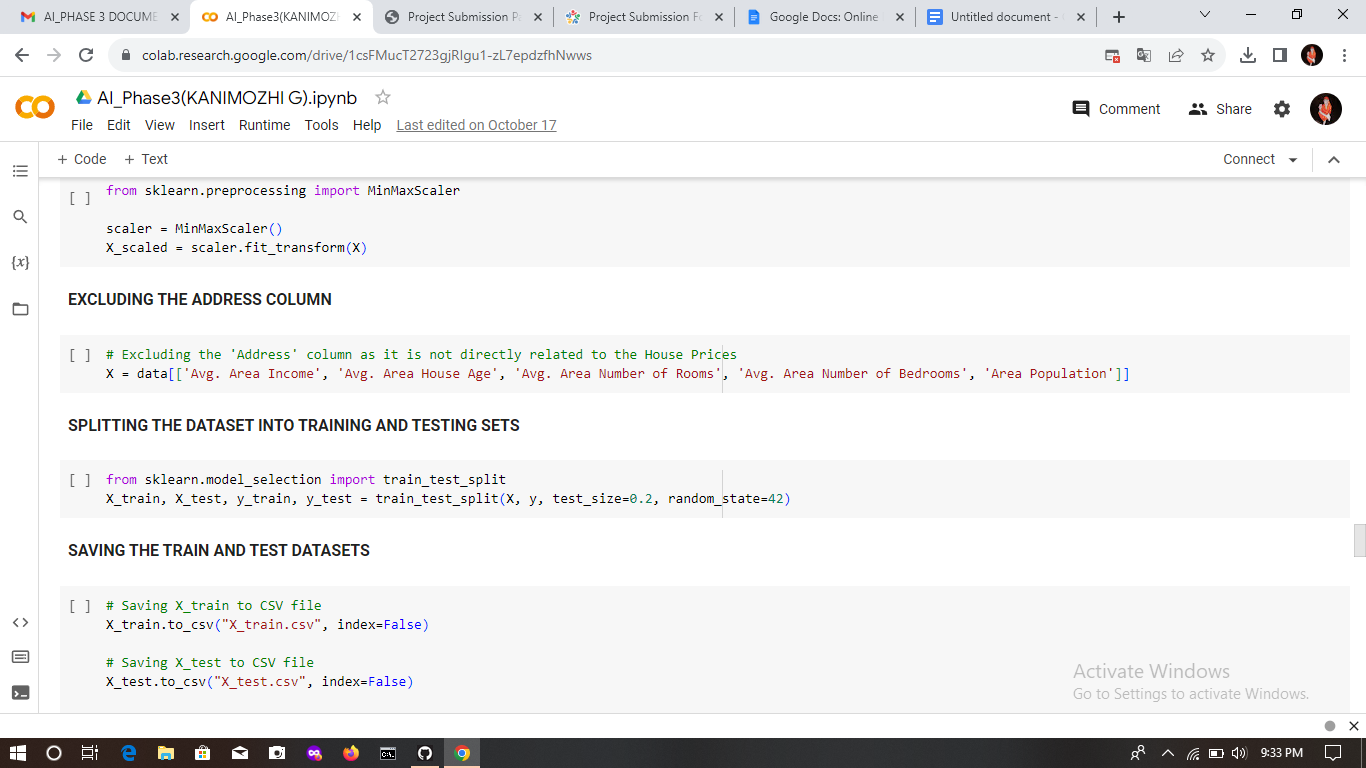
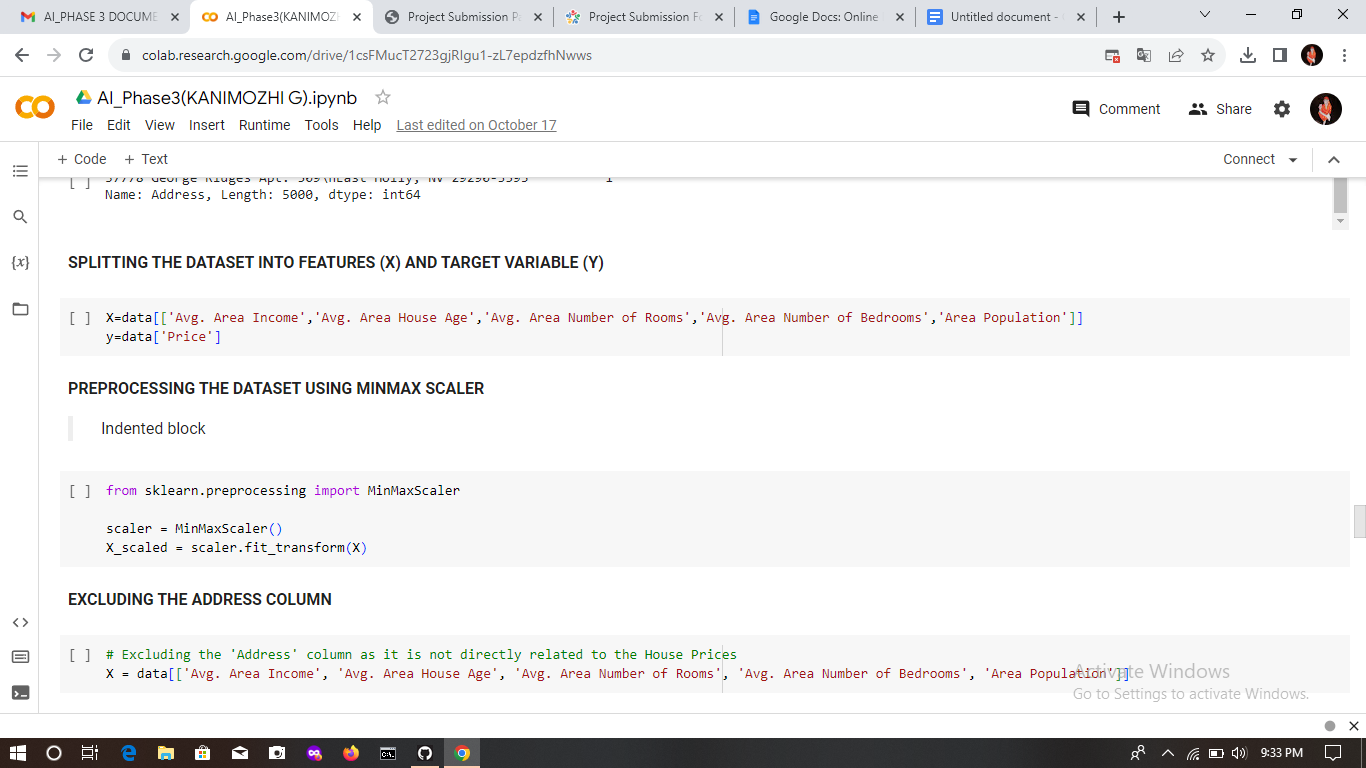
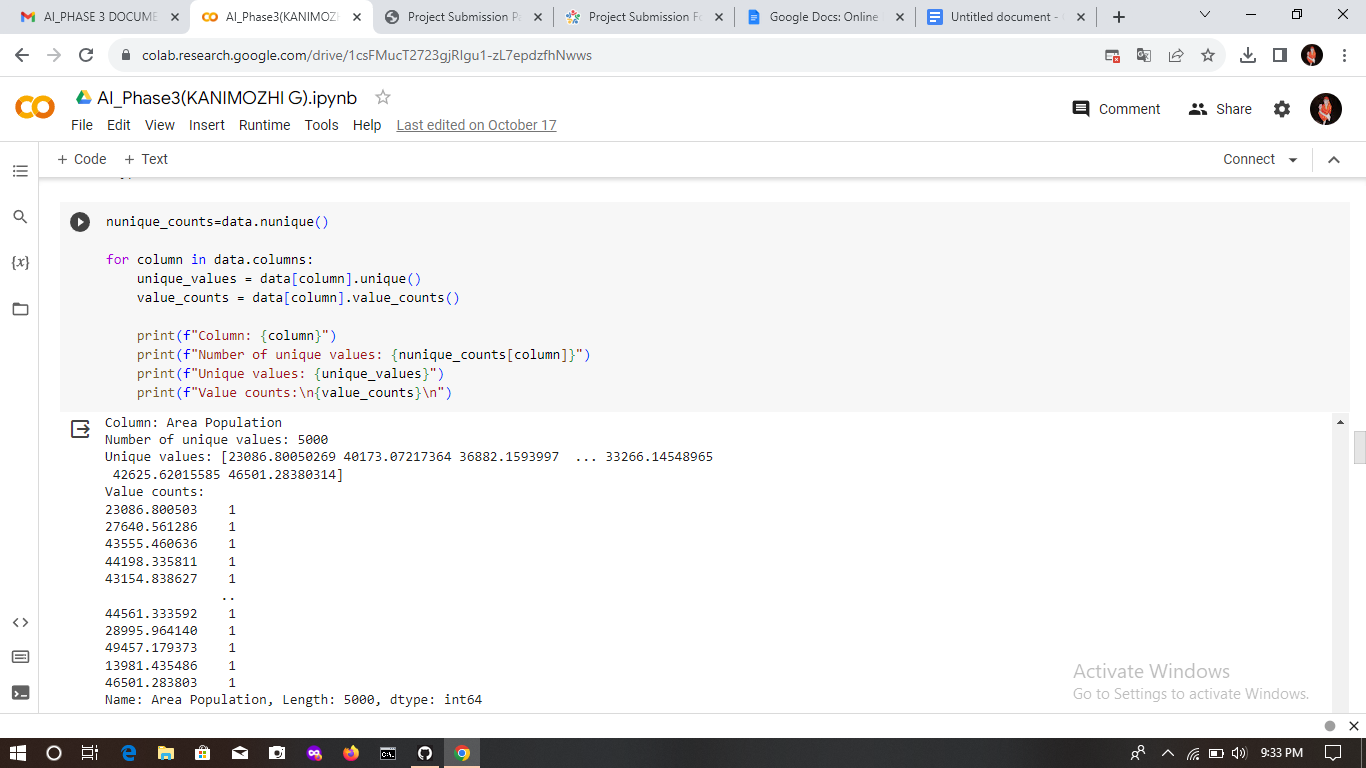
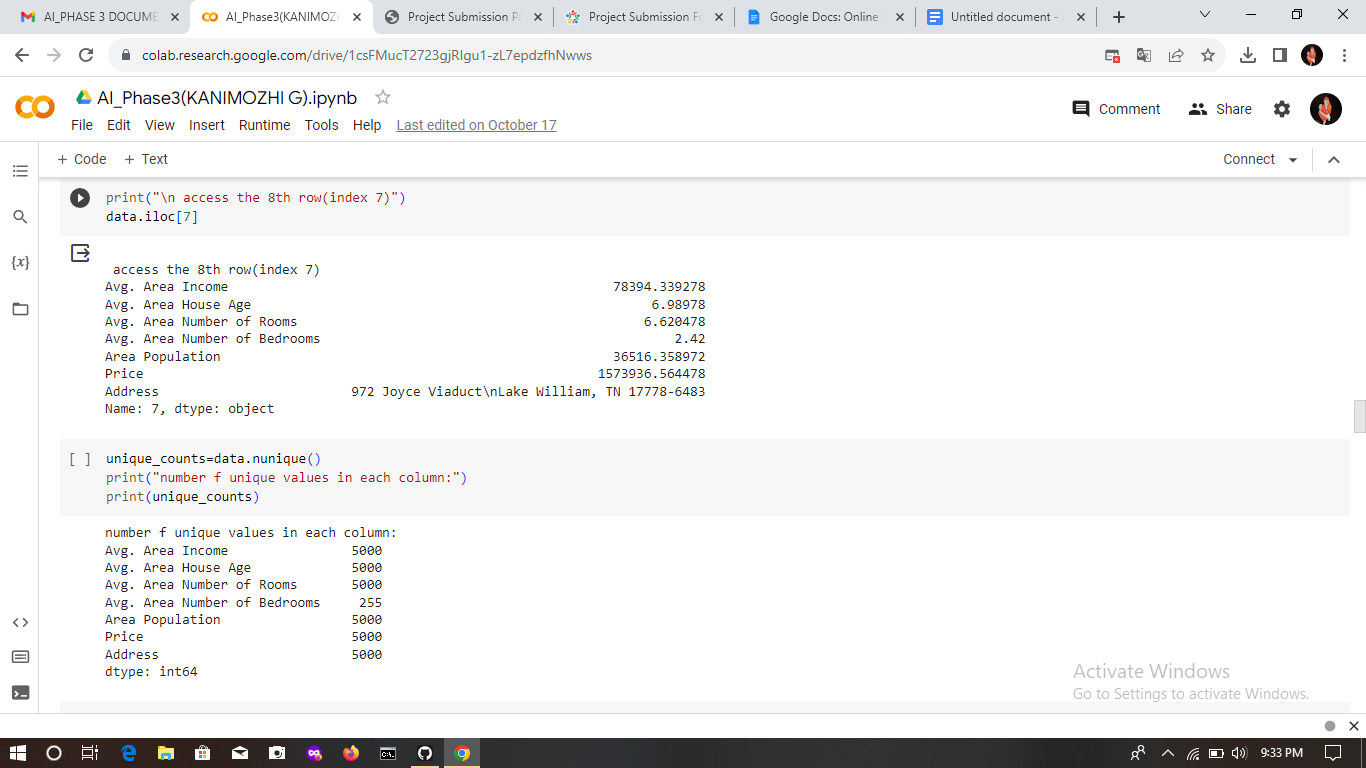
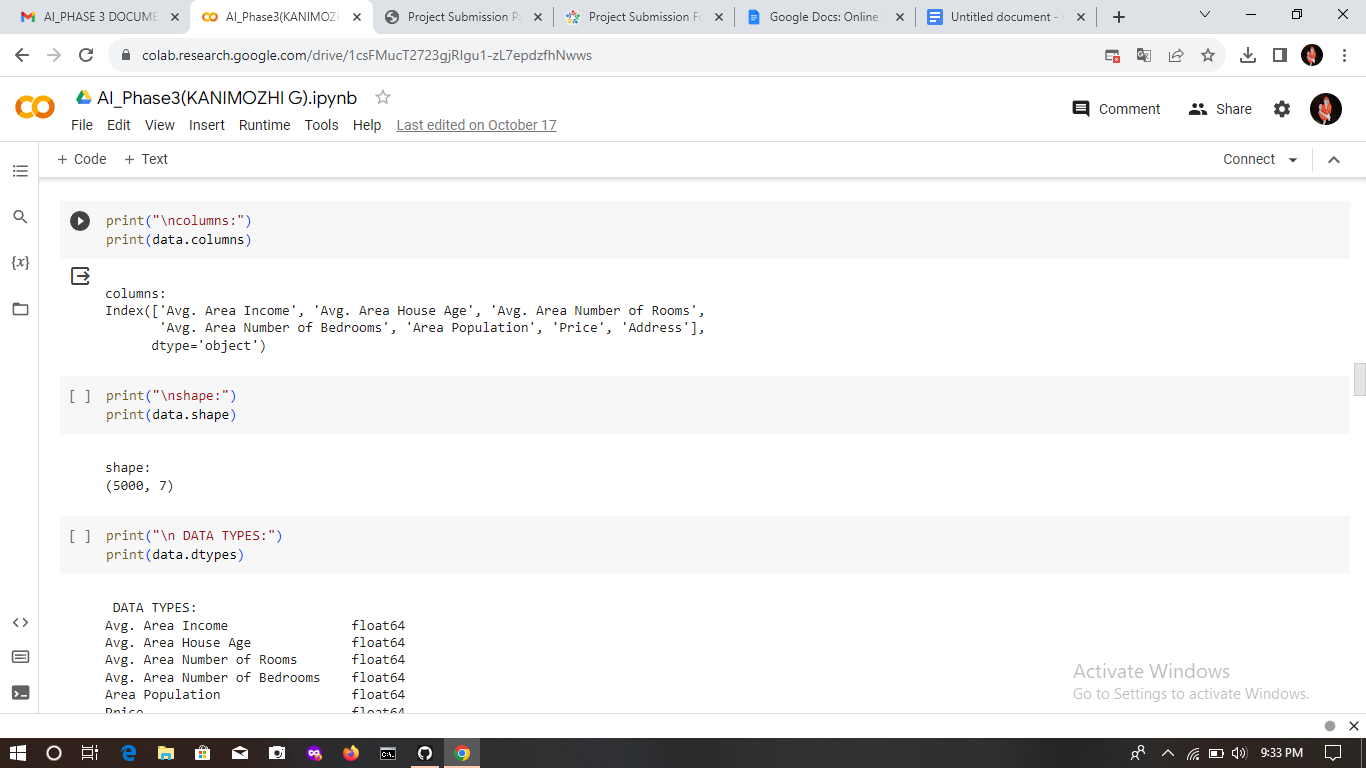
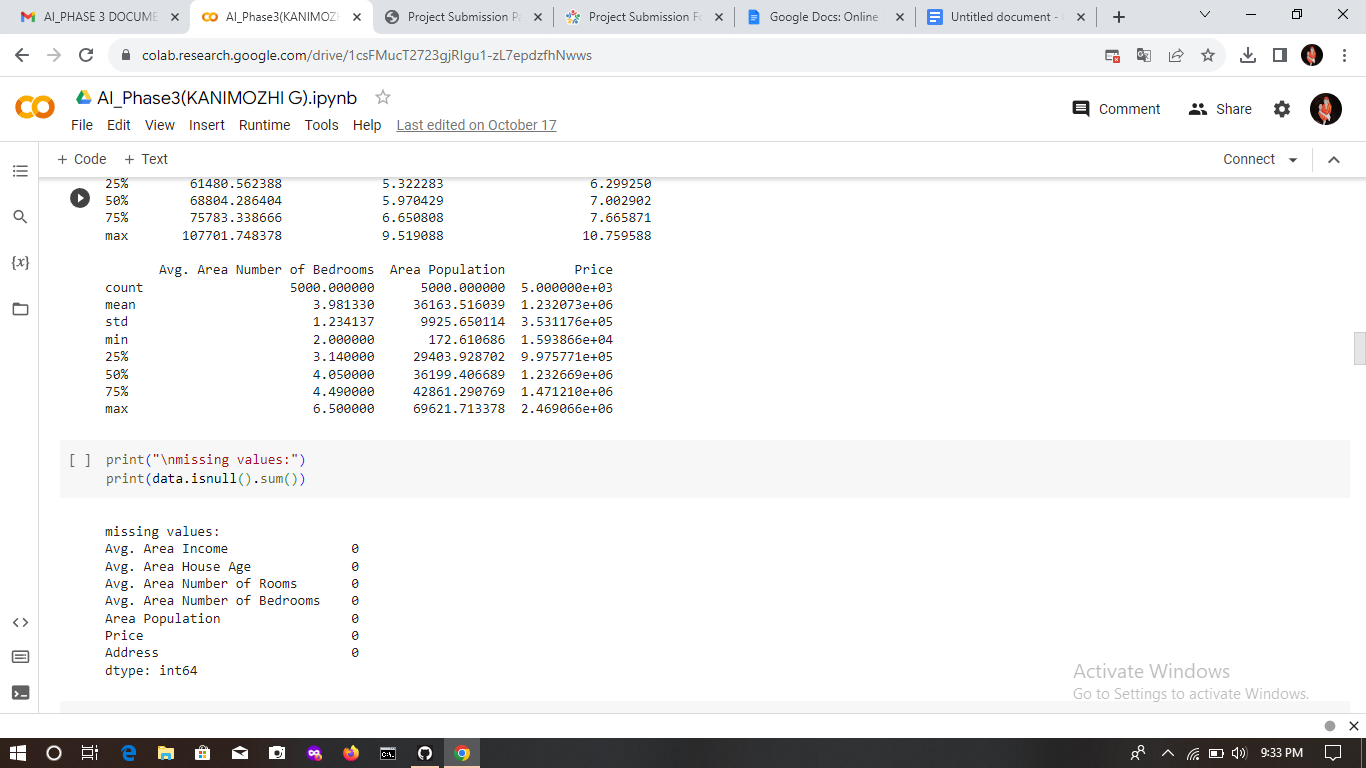
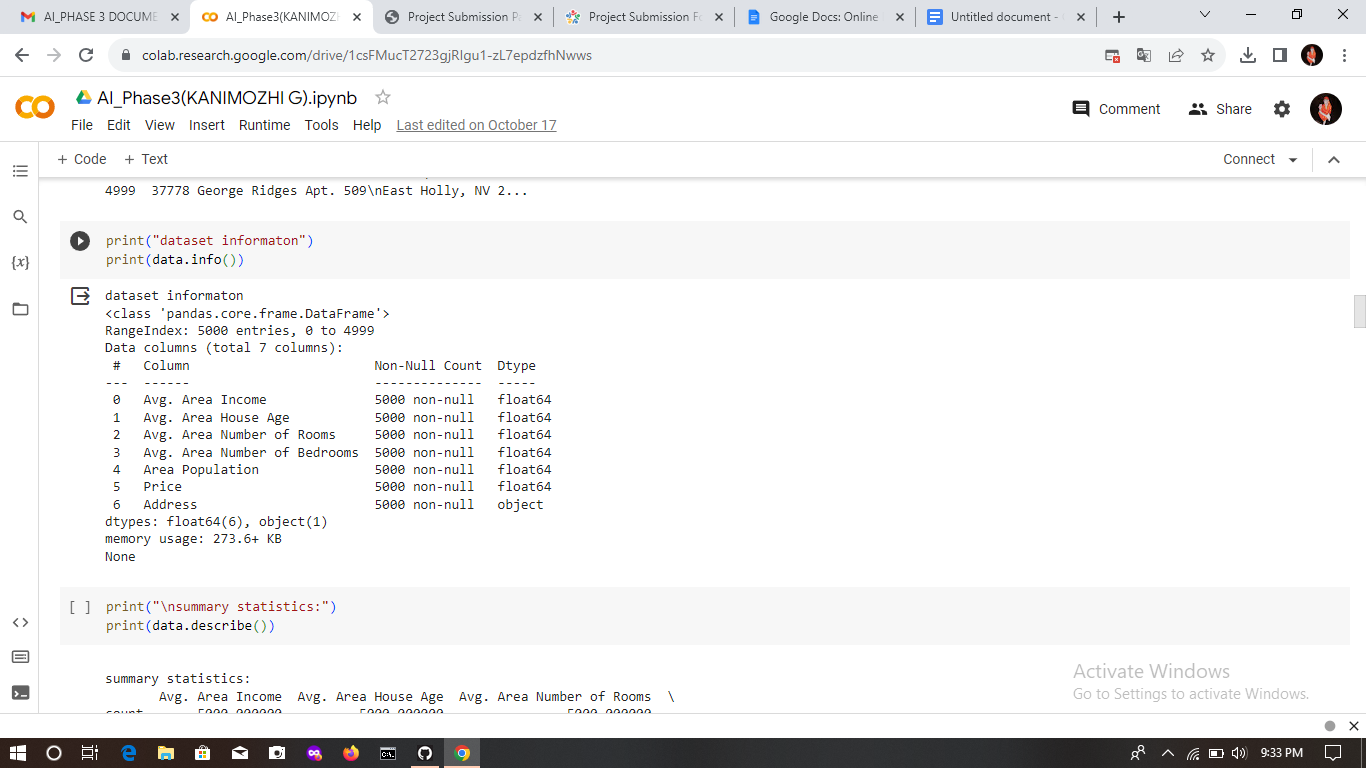
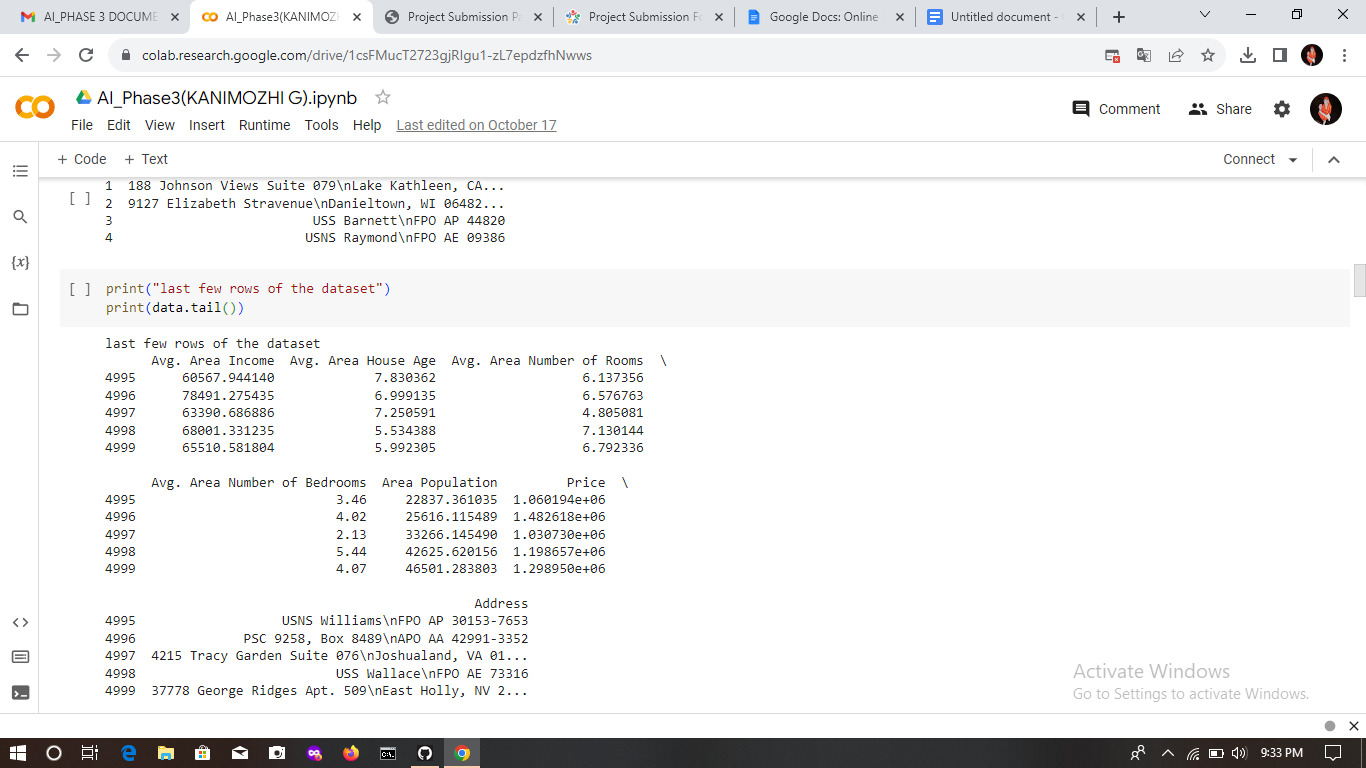
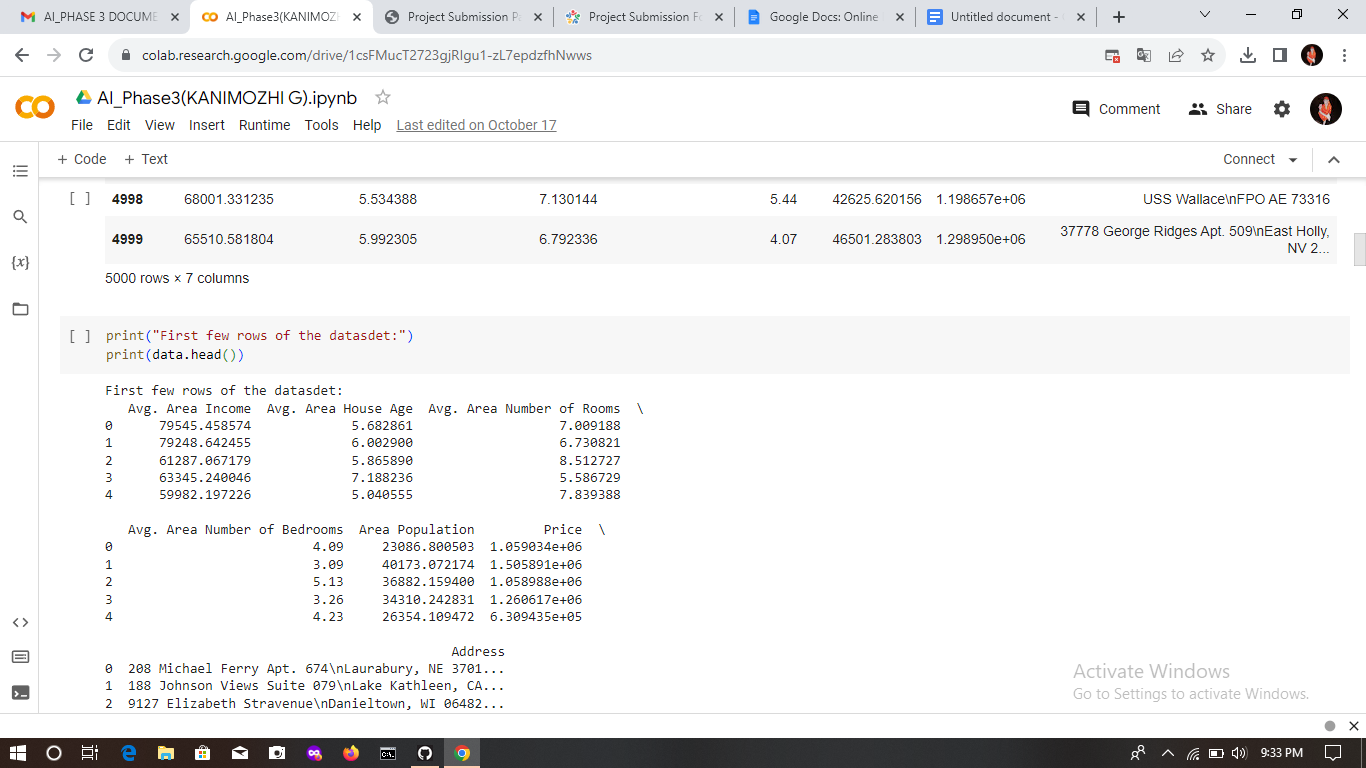
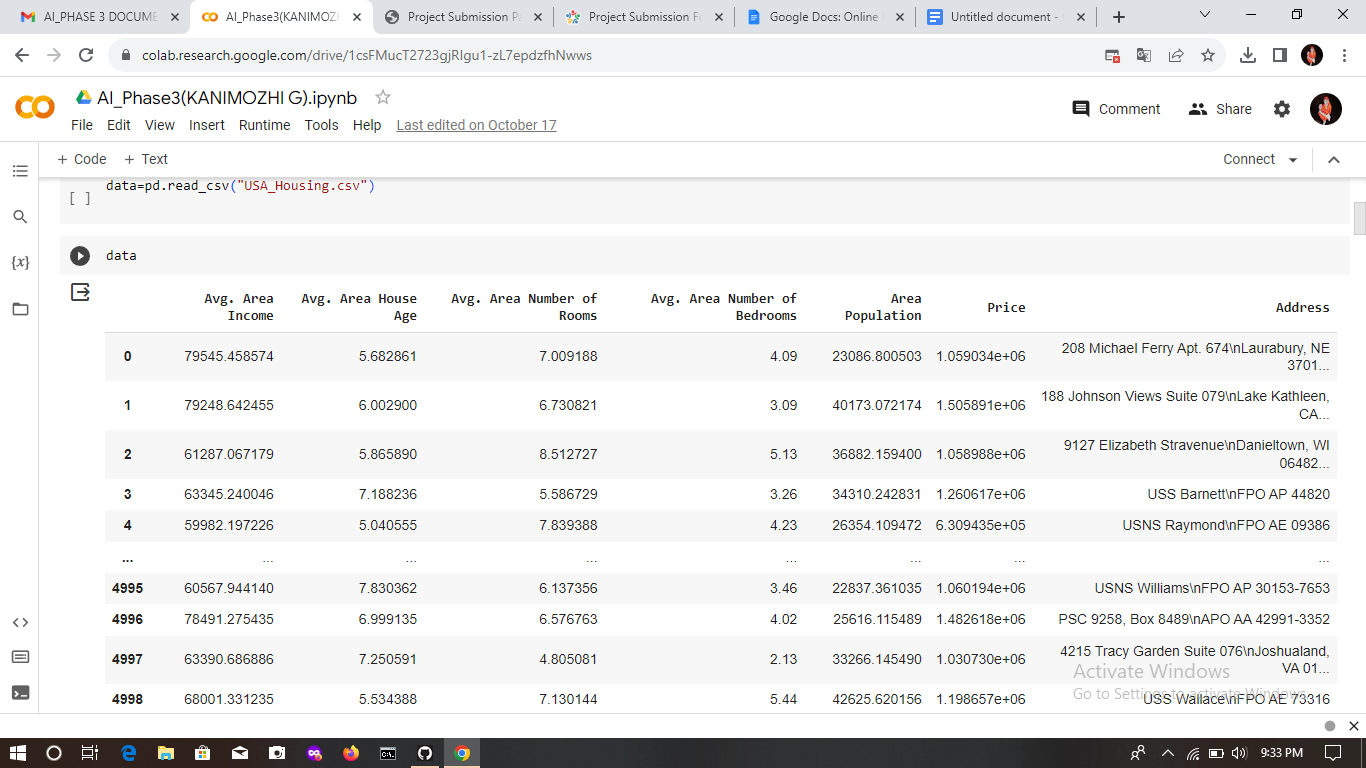
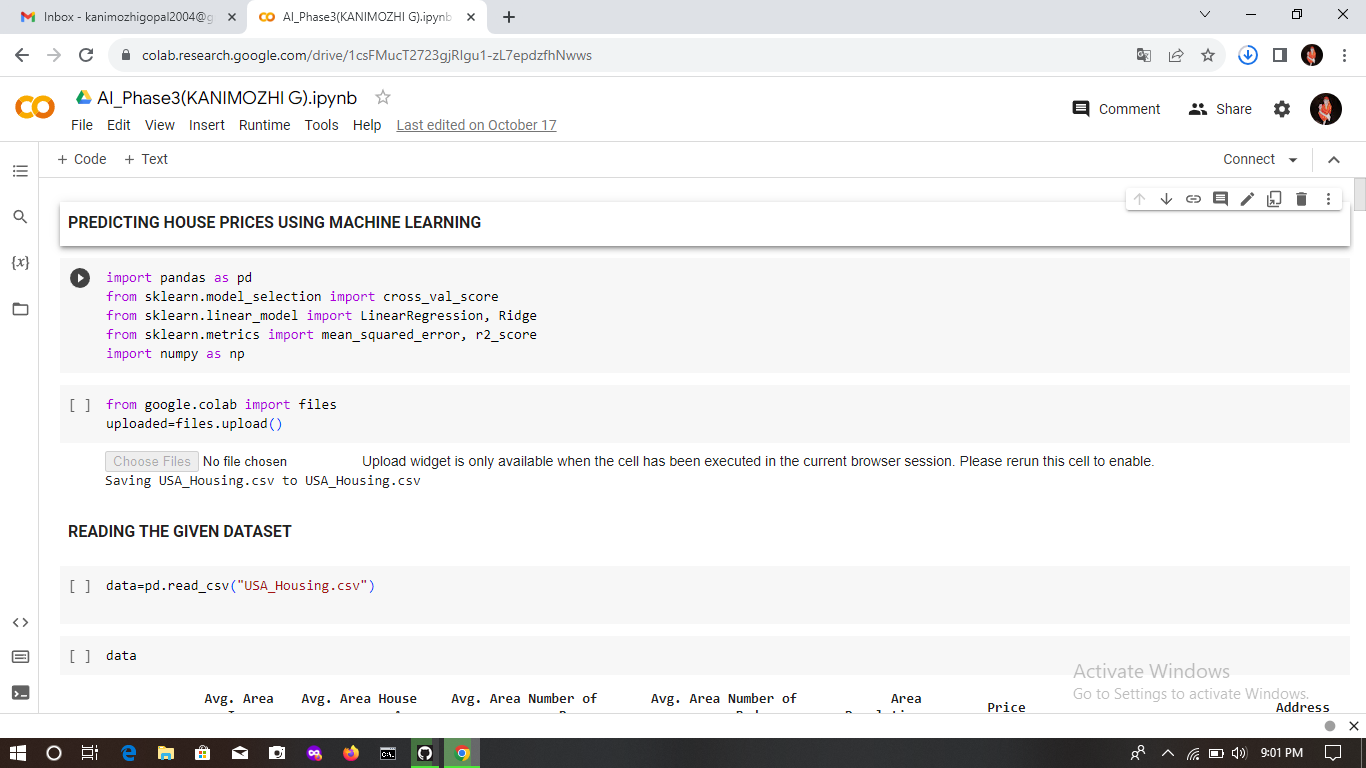
PREDICTING HOUSE PRICES USING MACHINE LEARNING

AI\_Phase3(KANIMOZHI G).ipynb

**Original file is located at:**

<https://colab.research.google.com/drive/1csFMucT2723gjRIgu1-zL7epdzfhNwws?usp=sharing>

**PYTHON CODE WITH OUTPUT CREATED WITH GOOGLE COLAB:**



**CODE EXPLANATION:**

**IMPORTING LIBRARIES:**

In this section, the necessary Python libraries are imported to

perform various tasks in building a house price prediction model.

These libraries include pandas for data handling, scikit-learn for

machine learning, and NumPy for numerical operations.

**CODE:**

import pandas as pd

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

from sklearn.linear\_model import LinearRegression, Ridge

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

import numpy as np

**LOADING THE GIVEN DATASET:**

The provided dataset is loaded into the environment using Google

Colab&#39;s file upload feature.

**CODE:**

from google.colab import files

uploaded = files.upload()

**READING THE GIVEN DATASET:**

The dataset is read into a pandas dataframe (&#39;data&#39;) for further

analysis and modeling.

**CODE:**

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

**DATA EXPLORATION:**

This section is dedicated to exploring the dataset to gain insights into

its structure and content. Various operations are performed,

including displaying the first and last few rows, dataset information,

summary statistics, missing values, columns, shape, data types,

accessing a specific row, and calculating the number of unique values

in each column.

**CODE:**

# Display the first and last few rows of the dataset

print(data.head())

print(data.tail())

# Display dataset information

print(data.info())

# Display summary statistics

print(data.describe())

# Check for missing values

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

# List the columns

print(data.columns)

# Get the shape of the dataset

print(data.shape)

# Display data types of columns

print(data.dtypes)

# Access the 21st row (index 20)

data.iloc[20]

# Calculate the number of unique values in each column

unique\_counts = data.nunique()

SPLITTING THE DATASET INTO FEATURES (X) AND TARGET VARIABLE

(y)

The dataset is divided into two parts: the feature matrix (&#39;X&#39;) and the target variable (&#39;y&#39;). In this case, the selected features include &#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; and &#39;Area Population,&#39; while the target variable is &#39;Price.&#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;]

PREPROCESSING THE DATASET USING MINMAX SCALER

The MinMaxScaler is applied to the feature matrix &#39;X&#39; to scale the values within the range [0, 1].

**CODE:**

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

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

EXCLUDING THE ADDRESS COLUMN

The &#39;Address&#39; column is excluded from the feature matrix &#39;X&#39; 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;]]

SPLITTING THE DATASET INTO TRAINING AND TESTING SETS

The dataset is divided into training and testing sets using an 80-20

split ratio. The &#39;X\_train&#39; and &#39;X\_test&#39; datasets are created for

features, and &#39;y\_train&#39; and &#39;y\_test&#39; for the target variable.

**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)

SAVING THE TRAIN AND TEST DATASETS

The training and testing datasets are saved as CSV files for future

use.

**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)

PREPROCESSING THE DATASET USING STANDARD SCALER

The StandardScaler is applied to standardize the training and testing

datasets.

**CODE:**

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

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

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

CROSS-VALIDATION WITH DIFFERENT PREPROCESSING METHODS

AND ALGORITHMS

Cross-validation is performed using different preprocessing methods

(StandardScaler and MinMaxScaler) and algorithms

(LinearRegression and Ridge). The code calculates the mean score for

each combination.

**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;)

BUILDING THE MODELS

The code trains a Linear Regression model on the preprocessed

training data and evaluates its performance on the testing data. It

calculates and displays the Mean Squared Error (MSE) and R-squared

(R2) Score as evaluation metrics for the Linear Regression model. The

predicted values are also printed.

**CODE:**

model = LinearRegression()

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

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

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)

print(&quot;Predicted Values:&quot;)

print(y\_pred)

**CONCLUSION:**

This code illustrates a house price prediction model using Linear

Regression, and it includes data exploration, preprocessing, model

building, and evaluation. Further enhancement of the model and

analysis will be done