PROJECT - 9

PREDICTING HOUSE PRICE USING MACHINE LEARNING

PHASE – 5

PROJECT TITLE: House Price Predictor

PROBLEM STATEMENT:

The housing market is an important and complex sector that impacts people's lives in many ways. For many individuals and families, buying a house is one of the biggest investments they will make in their lifetime. Therefore, it is essential to accurately predict the prices of houses so that buyers and sellers can make informed decisions. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors.

Dataset Link: https://www.kaggle.com/datasets/vedavyasv/usa-housing

INTRODUCTION:

- Predicting house prices using machine learning is a common and practical application of data science in the real estate industry.
- It involves using historical data and various machine learning algorithms to create models that can predict the selling price of houses based on their features.
- Here, They were gave the dataset which contains 5001 datas. Which was in csv format. In this phase we have to building our project by loading and preprocessing the dataset and Perform different analysis as needed.

Project Steps:

PROBLEM DEFINITION AND DESIGN THINKING

Problem Definition: 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.

DESIGN THINKING:

1.DATA SOURCE:

Choose a dataset containing information about houses, including features like Income, House Age, Number of Rooms, Number of Bedrooms, Population, Price and Address.

Features/Attributes:

These are variables or characteristics of a property that are believed to have an impact on its price. Common features includes

• Number of Rooms and Bedrooms: The count of rooms and bedrooms in the house, which can affect its price due to the increased area and size.

• Address: The geographical location of the property, which can include factors like neighbourhood, city, and proximity to amenities or landmarks.

• House Age: The age of the house or building can also influence its price.

• Income: The income earned by the people living in those areas can also influence its price.

• Population: The number of people who are currently living in those areas can also influence the price of the houses.

• Price: The total cost of the house can be displayed here based on certain factors like area, number of rooms, age etc..

Target Variable: This is the variable you're trying to predict, which is the price of the house or property. It is typically represented as the dependent variable in a regression analysis.

Dataset Size: The number of records or data points in the dataset, which can range from a few hundred to thousands or more.

Data Sources: Information about where the data was collected from, such as real estate listings, government records, or surveys.

2.DATA PREPROCESSING :

It is a crucial step in preparing a house price prediction dataset for analysis or machine learning. Here's a short overview of the key steps involved:

Data Cleaning:

Handle missing values by filling them with appropriate values (e.g., mean, median, or mode).Remove duplicates if they exist in the dataset.Correct any inconsistent or erroneous data entries.

Feature Selection:

Choose relevant features that are likely to impact house prices.Remove irrelevant or redundant features to simplify the dataset. Consider using domain knowledge and feature importance techniques.

Feature Encoding:

Convert categorical variables into numerical representations through techniques like one-hot encoding or label encoding. Standardize or normalize numerical features to have a consistent scale (e.g., using Min Max scaling or z-score normalization).

Outlier Detection and Handling:

Identify and handle outliers in the data, either by removing them or transforming them. Use visualization and statistical methods (e.g., Z-scores or IQR) to detect outliers.

Data Splitting:

Split the dataset into training, validation, and test sets to assess model performance effectively.A common split ratio is 70-80% training, 10-15% validation, and 10-15% testing.

Handling Skewed Data:

If the target variable (house prices) or some features are highly skewed, consider applying transformations like log transformations to make the data more symmetric.

Scaling and Normalization:

Ensure that numerical features are scaled or normalized to have similar scales, preventing some features from dominating others during modeling.

Data Transformation (if needed):

For some modeling algorithms, you might need to transform the data to meet their assumptions (e.g., transforming the target variable for linear regression).

Data preprocessing ensures that the dataset is clean, well-structured, and ready for analysis or modeling. The specific steps may vary depending on the dataset and the machine learning algorithm you plan to use, but these general steps provide a solid foundation for preparing data for house price prediction tasks.

3.FEATURE SELECTION:

Feature selection for a house price prediction dataset involves choosing the most relevant and informative features while excluding irrelevant

ones to improve the accuracy of your prediction model. Here's a concise guide to feature selection:

Correlation Analysis:

Identify features that have a strong correlation with the target variable (house price) using techniques like Pearson correlation. Keep features with high correlations and eliminate those with low or negative correlations.

Feature Importance:

If you're using tree-based models (e.g., Random Forest, XGBoost), use their feature importance scores to rank and select the most important features.

Cross-Validation:

Use cross-validation to evaluate different feature subsets' performance and select the one that yields the best model performance metrics.

Regularization Hyperparameters:

When using models like Ridge or Elastic Net, tune their regularization hyperparameters to encourage feature selection during training.

4.MODEL SELECTION:

Choose a suitable regression algorithm (e.g., Linear Regression, Random Forest Regressor) for predicting house prices.

Linear Regression:

Start with a basic linear regression model, which is simple and interpretable. It's a good baseline.

Decision Trees:

Try decision tree-based models like Random Forest and Gradient Boosting. They often perform well and handle non-linear relationships.

Neural Networks:

Experiment with deep learning models, such as neural networks, if you have a large dataset and complex features.

5. MODEL TRAINING:

• Train the selected model using the preprocessed data.

• Train the selected model on the training data using the features to predict house prices.

• The model will learn the relationships between the features and target variable.

6. EVALUATION:

Evaluate the model's performance using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.

• MAE: It represents the average absolute error in house price predictions. Lower MAE values indicate better model accuracy. It's easy to understand and suitable for comparing models.

• RMSE: RMSE penalizes larger errors more than MAE because of the squaring. It's also sensitive to outliers. Lower RMSE values indicate better model accuracy.

• R-squared: R-squared measures the goodness of fit. It tells you the proportion of variance in the target variable that is explained by the model. Higher R-squared values (closer to 1) indicate a better fit.

These steps provide a simplified overview of the model training process for house price prediction. The specific implementation details and choice of algorithms may vary depending on the dataset and the goals of the project.

Here, we use two types of techniques which are Gradient Boosting and XGBoost.

GRADIENT BOOSTING:

Gradient Boosting is a machine learning technique for regression and classification problems, which builds a predictive model in the form of an ensemble of weak learners, typically decision trees. It is an extension of boosting, a machine learning ensemble method that combines the predictions of several base estimators (often decision trees) to improve accuracy and robustness over a single estimator. The "gradient" in Gradient Boosting refers to the use of gradient descent optimization technique to minimize the loss function, which measures the difference between the predicted values and the actual target values.

XGBOOST:

XGBoost, short for eXtreme Gradient Boosting, is an optimized and scalable machine learning algorithm designed to efficiently handle large datasets and solve various supervised learning problems, including regression, classification, ranking, and user-defined prediction tasks.

STEP 1: IMPORT LIBRARIES

Import all the necessary libraries what we need. Here we have to install Pandas, Seaborn, Sklearn, Matplotlib.

- Pandas is a powerful data manipulation and analysis library.
- NumPy is used for numerical and mathematical operationsPython,especially for working with arrays and matrices.
- Seaborn is a data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.
- Matplotlib is a widely used library for creating static, animated, and interactive visualizations in Python.

● The train_test_split function is used to split a dataset into training and testing subsets, which is crucial for model training and evaluation in machine learning.

● This preprocessing step is often necessary to make sure features are on the same scale, which can improve the performance of many machine learning algorithms.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

STEP 2: LOAD THE DATASET

Load the dataset what they gave into pandas dataframe.

- pd is an alias for the Pandas library, which you imported earlier.
- read_csv() is a Pandas function that is used to read CSV files and load their data into a DataFrame.

```
dataset = pd.read_csv('USA_Housing.csv')
```

STEP 3: DATA PREPROCESSING

● DATA CLEANING : Handle missing values: Fill missing values using mean, median, or advanced imputation techniques.

```
[38] data.dropna(subset=['Avg. Area Income', 'Avg. Area House Age'], inplace=True)
     data['Avg. Area Income'].fillna(data['Avg. Area Income'].mean(), inplace=True)
     data['Avg. Area House Age'].fillna('Unknown', inplace=True)
```

```
mean_price = data['Price'].mean()
median_price = data['Price'].median()

print('Mean House Price:', mean_price)
print('Median House Price:', median_price)
```

```
Mean House Price: 1232072.654142357
Median House Price: 1232669.3779657914
```

- DATA TRANSFORMATION : Convert categorical variables into numerical format (If required)

```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (data.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (data.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

```
Categorical variables: 1
Integer variables: 0
Float variables: 6
```

Using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
for columns in data.columns:
  data[columns]=labelencoder.fit_transform(data[columns])
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Avg. Area Income             5000 non-null   int64
 1   Avg. Area House Age          5000 non-null   int64
 2   Avg. Area Number of Rooms    5000 non-null   int64
 3   Avg. Area Number of Bedrooms 5000 non-null   int64
 4   Area Population              5000 non-null   int64
 5   Price                        5000 non-null   int64
 6   Address                      5000 non-null   int64
dtypes: int64(7)
memory usage: 273.6 KB
None
```

- REMOVE DUPLICATES : Check for and remove duplicate records if present.

```
39] data.drop_duplicates(inplace=True)
```

```
[7] print(data.isnull().sum())
```

```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
Address                         0
dtype: int64
```

STEP 3 : DATA VISUALIZATION

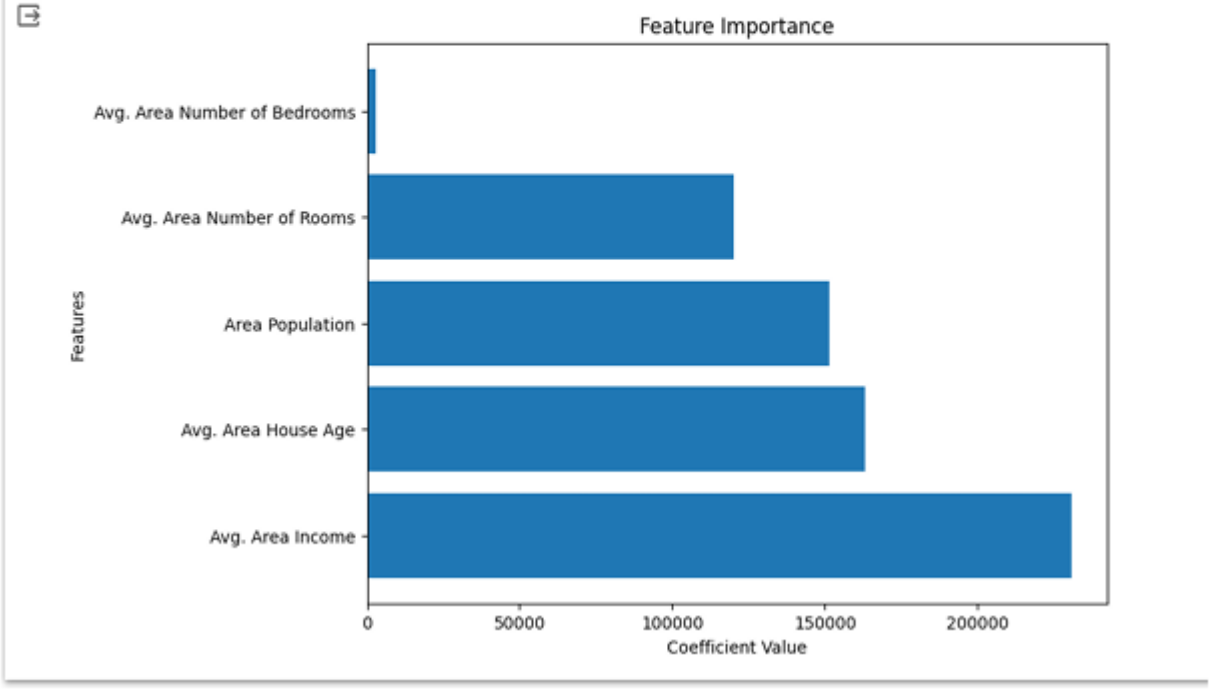
Data visualization is the presentation of data in a graphical or pictorial format. It enables people to understand complex patterns, trends, and

insights from data by representing it visually. Through data visualization, large and complex datasets can be represented in a clear, concise, and meaningful way. It can be

- ❖ Bar Charts
- ❖ Line Charts
- ❖ Pie Charts
- ❖ Scatter Plots
- ❖ Histograms
- ❖ Heat maps
- ❖ Box Plots

BAR CHART :

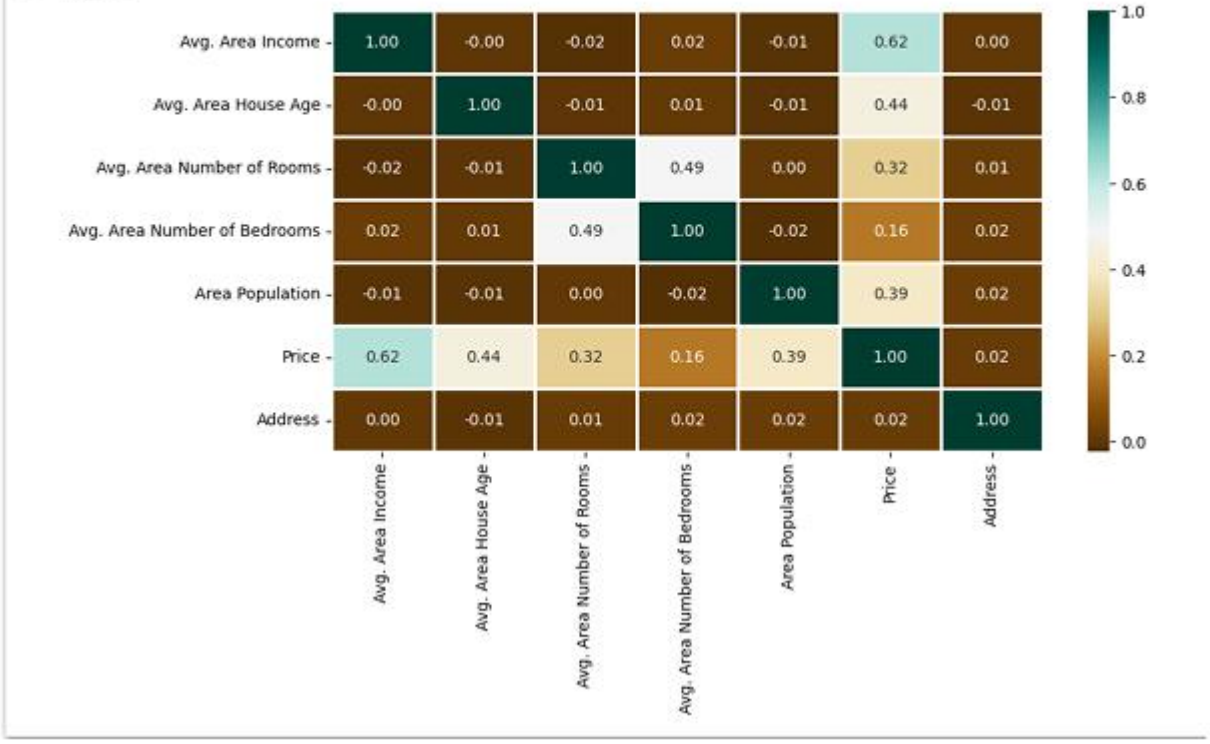
```
plt.figure(figsize=(8, 6))
plt.barh(feature_importance['Feature'], feature_importance['Coefficient'])
plt.xlabel('Coefficient Value')
plt.ylabel('Features')
plt.title('Feature Importance')
plt.show()
```



HEAT MAP :

```
plt.figure(figsize=(10, 5))
sns.heatmap(data.corr(),
        cmap = 'BrBG',
        fmt = '.2f',
        linewidths = 2,
        annot = True)
```

```
<Axes: >
```



❖ This code uses Matplotlib and Seaborn to create a heatmap of the correlation matrix of your dataset.

❖ The code generates a heatmap that visually represents the correlation between pairs of variables in your dataset. Positive correlations are shown in one color, whilenegative correlations are shown in another color.

HISTOGRAM :

```
dataset.hist(figsize=(10,8))
```

```
array([[<Axes: title={'center': 'Avg. Area House Age'}>,
        <Axes: title={'center': 'Avg. Area Number of Rooms'}>],
       [<Axes: title={'center': 'Avg. Area Number of Bedrooms'}>,
        <Axes: title={'center': 'Area Population'}>],
       [<Axes: title={'center': 'Price'}>,
        <Axes: title={'center': 'Address'}>]], dtype=object)
```

❖ The code you've provided uses Seaborn to create a joint plot between the Avg.Area House Age and Price columns in your dataset. Specifically, it uses a hexbin plot for visualization.

❖ The code will generate a grid of histograms, with each histogram representing thedistribution of a numerical column in your dataset. This provides a

quick visualoverview of the datas distribution, which can be useful for understanding the range,spread, and central tendency of each numerical variable.

SCATTER PLOT:

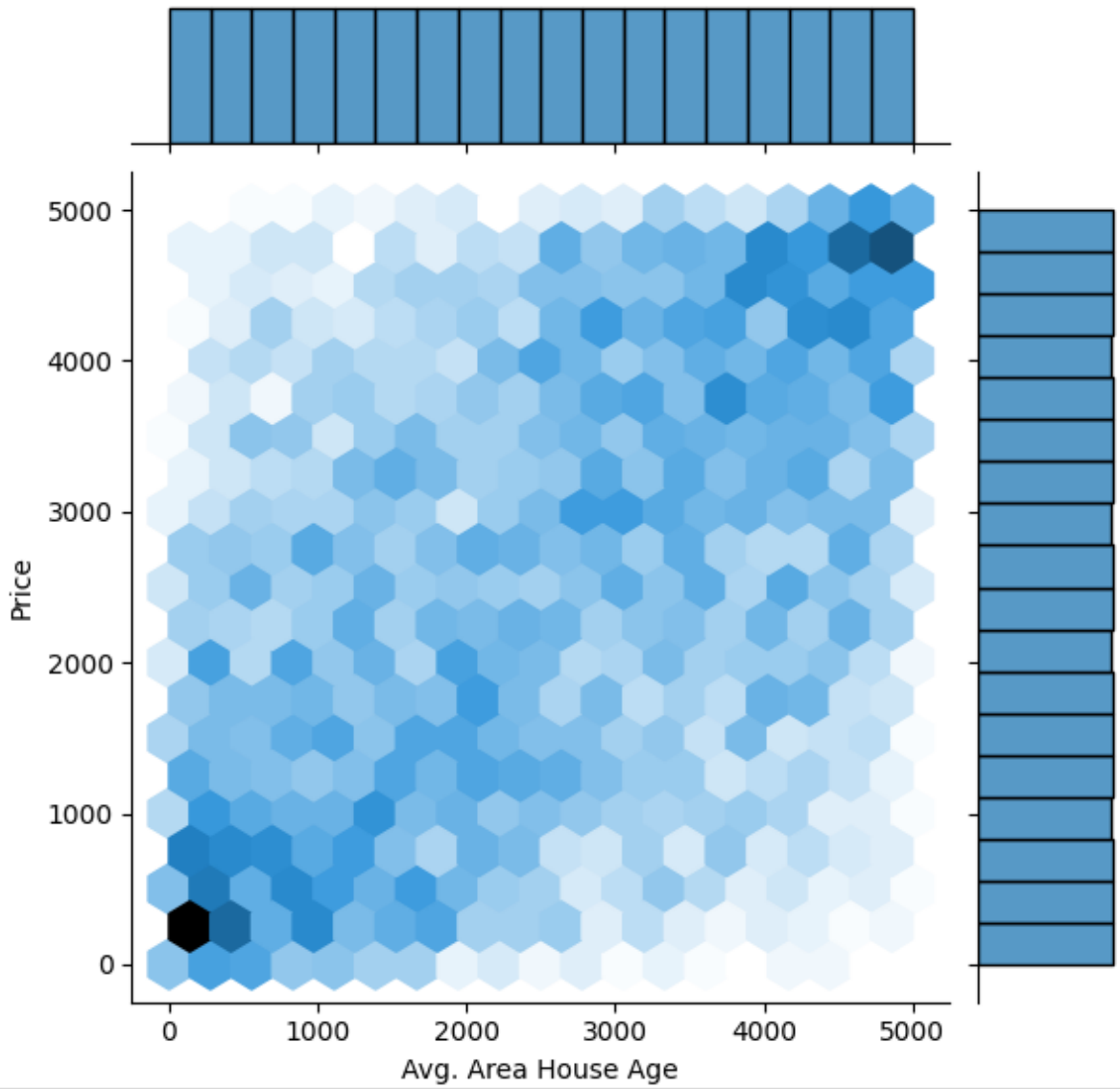
A scatter plot is a type of data visualization that displays individual data points along two axes to represent the relationship between two variables. Each data point in the scatter plot represents the values of the two variables, allowing you to observe patterns, correlations, or trends in the data.

```
plt.figure(figsize=(8,6))
sns.scatterplot(x='Area Population', y='Price', data=data, color='purple')
plt.title('House Price vs. Area Population')
plt.xlabel('Area Population')
plt.ylabel('Price')
plt.show()
```

JOINTPLOT :

```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7ab322e87580>
```



The code you've provided uses Seaborn to create a joint plot between the 'Avg. Area House Age' and 'Price' columns in your dataset. Specifically, it uses a hexbin plot for visualization.

The code will generate a joint plot that shows the relationship between the average area house age and house prices. The hexbin plot will represent the density of data points, with darker hexagons indicating areas where more

data points are concentrated. This type of plot helps visualize the distribution and relationship between two numerical variables.

STEP 4 : TRAINING THE MODEL

In this second part we have to dividing dataset into features and target variables. Give the name for those datas.

```
[ ] X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
          'Avg. Area Number of Bedrooms', 'Area Population']]
    Y = dataset['Price']
```

Here, we have to split into training and testing. The training set is used to train the machine learning model. The test set is used to evaluate the model's performance on unseen data.

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=101)
```

❖ head() allows you to quickly inspect the structure and contents of a dataset. When you call head() without passing any argument, it displays the first 5 rows of the DataFrame by default.

❖ shape is an attribute of arrays, data frames, or matrices that represents the dimensions or the structure of the object.

```
[ ]  Y_train.head()
```

```
3413    1.305210e+06
1610    1.400961e+06
3459    1.048640e+06
4293    1.231157e+06
1039    1.391233e+06
Name: Price, dtype: float64
```

```
[ ]  Y_train.shape
```

```
(4000,)
```

```
[ ]  Y_test.head()
```

```
1718    1.251689e+06
2511    8.730483e+05
345     1.696978e+06
2521    1.063964e+06
54      9.487883e+05
Name: Price, dtype: float64
```

```
[ ]  Y_test.shape
```

```
(1000,)
```

USING MACHINE LEARNING ALGORITHM:

MODEL 1 - Linear Regression :

Linear regression is a commonly used technique in machine learning for predicting a continuous outcome variable based on one or more input features. When it comes to predicting house prices, linear regression can be applied to model the relationship between various features of a house (number of bedrooms, location, etc.) and its price.

BUILD AND TRAIN THE LINEAR REGRESSION MODEL:

- ❖ Import the Linear Regression model from Scikit-Learn.
- ❖ Fit the model to the training data.

```
[ ]  model_lr=LinearRegression()
```

```
[ ]  model_lr.fit(X_train_scal, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

MAKE PREDICTIONS:

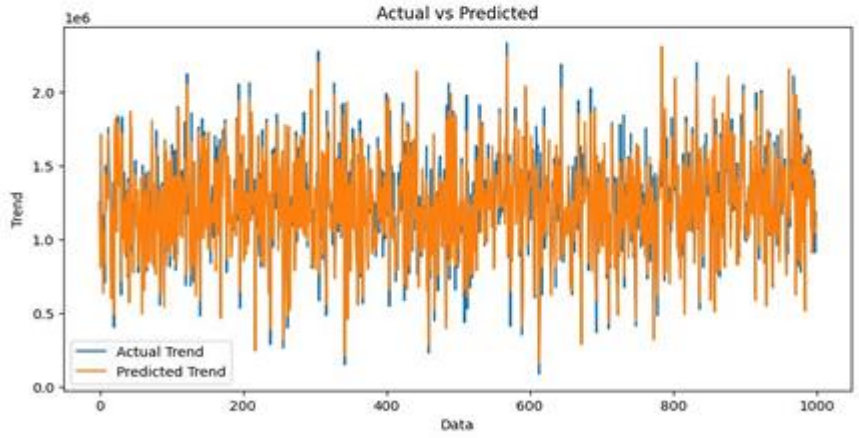
Use the trained model to make predictions on the test data.

```
[ ]  Prediction1 = model_lr.predict(X_test_scal)
```

EVALUATION OF PREDICTING DATA:

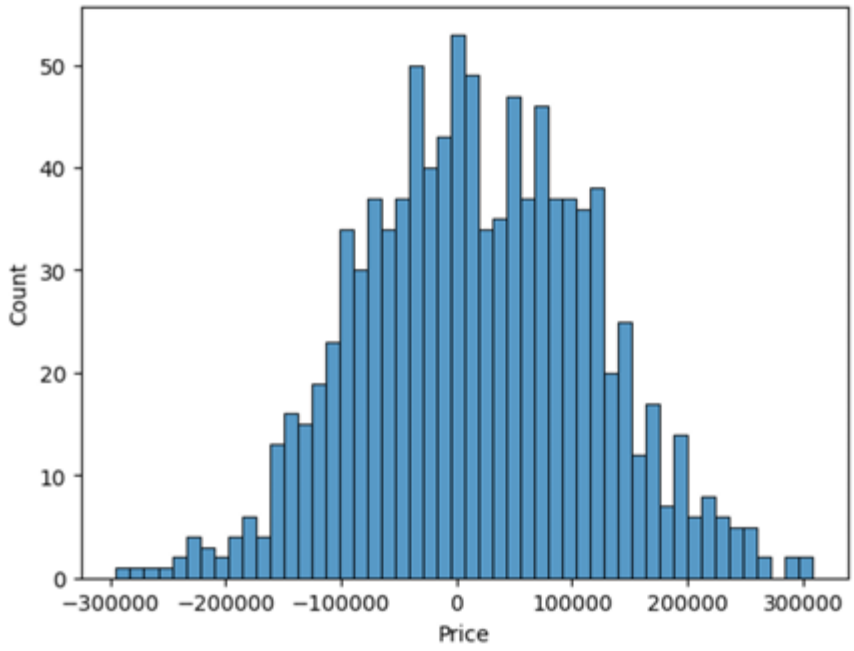
```
[24]  plt.figure(figsize=(12,6))
      plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
      plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
      plt.xlabel('Data')
      plt.ylabel('Trend')
      plt.legend()
      plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



```
sns.histplot((Y_test-Prediction1), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



Assess the model's performance using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

```
[ ] print(r2_score(Y_test, Prediction1))
    print(mean_absolute_error(Y_test, Prediction1))
    print(mean_squared_error(Y_test, Prediction1))
```

```
0.9182928179392918
82295.49779231755
10469084772.975954
```

This Model 1- Linear Regression has the r2_score prediction value 0.9182928179392918, Mean Absolute Error value as 82295.49779231755 and the Mean Squared Error as 10469084772.975954.

MODEL 2 - Support Vector Regressor :

Support Vector Machines (SVM) are primarily used for classification tasks. Predicting house prices is a regression task, which involves predicting a continuous value rather than a categorical label. However, SVM can be adapted

for regression tasks through a variant called Support Vector Regression (SVR). SVR uses the same principles as SVM but is applied to regression problems.

MODEL TRAINING :

- ❖ Import the SVR model from a machine learning library like Scikit-Learn.
- ❖ Train your SVR model using the training data.

```
[ ]  model_svr = SVR()
```

```
[ ]  model_svr.fit(X_train_scal, Y_train)
```

```
▾ SVR
SVR()
```

MAKE PREDICTIONS:

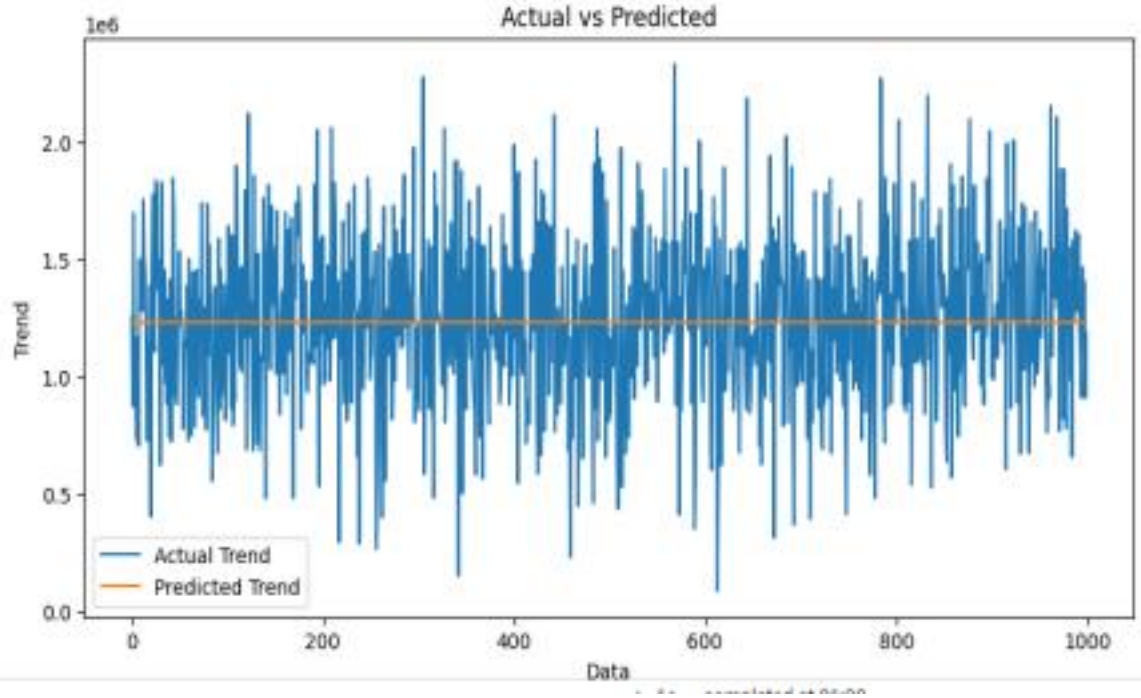
Use the trained model to make predictions on the test data.

```
[ ]  Prediction2 = model_svr.predict(X_test_scal)
```

EVALUATION OF PREDICTING DATA:

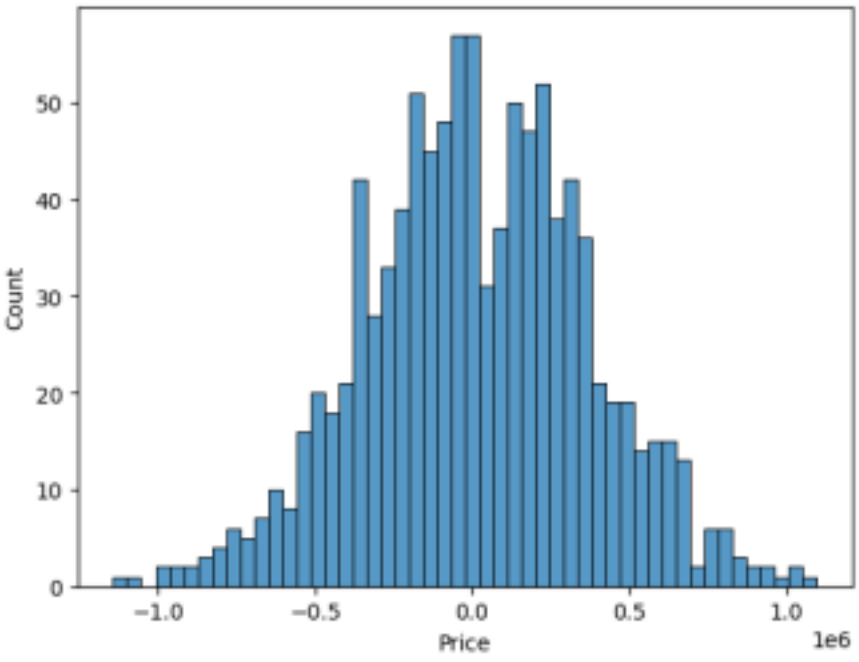
```
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```




```
[ ]  sns.histplot((Y_test-Prediction2), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



Assess the model's performance using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

```
[ ] print(r2_score(Y_test, Prediction2))
    print(mean_absolute_error(Y_test, Prediction2))
    print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

This Model 2- Support Vector Regressor has the r2_score prediction value - 0.0006222175925689744. Which has minimum accuracy compared to others.

MODEL 3 - Random Forest Regressor :

Random Forest Regressor is a powerful machine learning algorithm that can be used for predicting house prices. Here's how you can use Random Forest Regressor to predict house prices using Python and the popular libraries, Pandas, Scikit-Learn (which includes the RandomForestRegressor), and possibly Matplotlib for visualization.

MODEL TRAINING :

- ❖ Import the Random Forest Regressor model from a machine learning library like Scikit-Learn.
- ❖ Train your Random Forest Regressor model using the training data.

```
[ ] model_rf = RandomForestRegressor(n_estimators=50)
```

```
[ ] model_rf.fit(X_train_scal, Y_train)
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

MAKE PREDICTIONS:

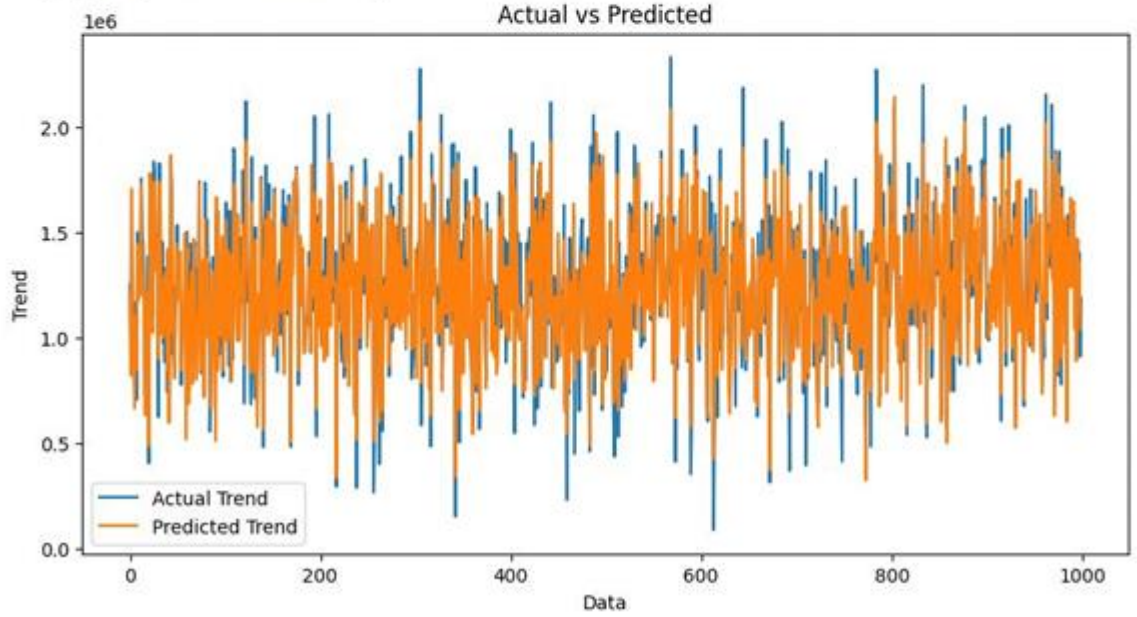
Use the trained model to make predictions on the test data.

```
[ ]  Prediction4 = model_rf.predict(X_test_scal)
```

EVALUATION OF PREDICTING DATA:

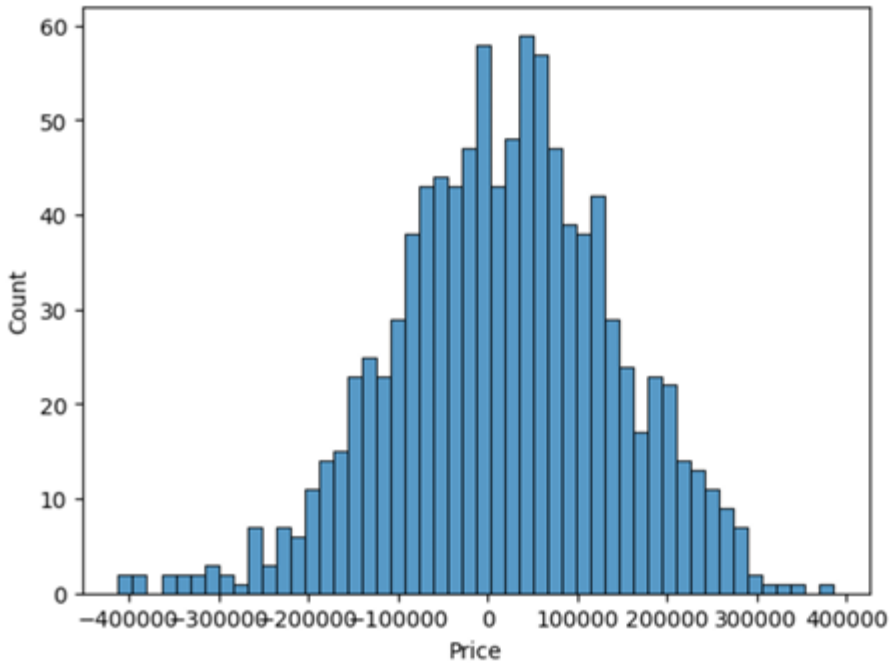
```
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



```
[ ]  sns.histplot((Y_test-Prediction4), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



Assess the model's performance using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

This Model 3- Random Forest Regressor has the r2_score prediction value 0.8805641323024149.

MODEL 4 - XGboost Regressor :

XGBoost is a popular machine learning algorithm that is widely used for regression tasks, including predicting house prices. It is an implementation of gradient boosted decision trees designed for speed and performance.

MODEL TRAINING :

❖ Import the XGBoost model from a machine learning library like Scikit-Learn.

❖ Train your XGBoost model using the training data.

```
[ ]  model_xg = xg.XGBRegressor()
```

```
[ ]  model_xg.fit(X_train_scal, Y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

MAKE PREDICTIONS:

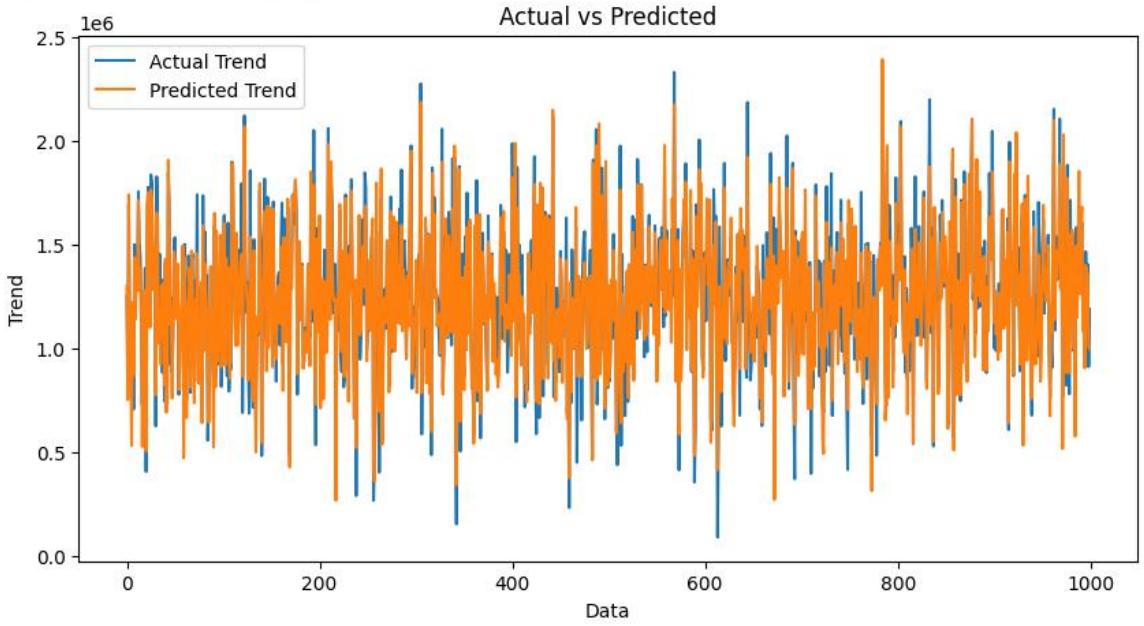
Use the trained model to make predictions on the test data.

```
[ ]  Prediction5 = model_xg.predict(X_test_scal)
```

EVALUATION OF PREDICTING DATA:

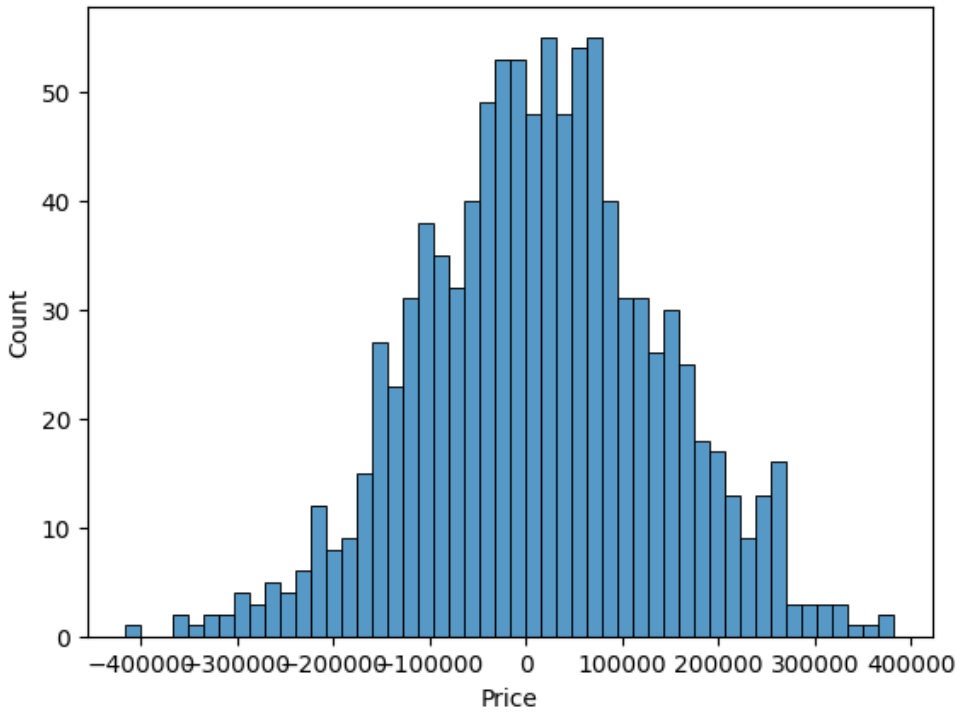
```
[ ]      plt.figure(figsize=(10,5))
         plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
         plt.plot(np.arange(len(Y_test)), Prediction5, label='Predicted Trend')
         plt.xlabel('Data')
         plt.ylabel('Trend')
         plt.legend()
         plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```




```
[ ]  sns.histplot((Y_test-Prediction5), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



Assess the model's performance using appropriate metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

```
[ ]  print(r2_score(Y_test, Prediction5))
     print(mean_absolute_error(Y_test, Prediction5))
     print(mean_squared_error(Y_test, Prediction5))

     0.8749027861384089
     100138.43694996461
     16028619562.562243
```

This Model 4- XGboost Regressor has the r2_score prediction value is 0.8749027861384089.

CODE LINK :

https://colab.research.google.com/drive/1GUgmcE1Q5eecbzcBgrSor0jB2IxaxcJA?usp=sharing

CONCLUSION :

Machine learning techniques, such as regression models and ensemble methods like Gradient Boosting and XGBoost, offer powerful tools for making accurate predictions. However, the effectiveness of the prediction models heavily depends on the quality of the data and the features selected for analysis. i have completed data loading it is the initial step, which involves importing the dataset for analysis. and preprocessing which includes data cleaning, transformation and remove duplicate values and finally Data visualization is essential for understanding the data's distribution, relationships, and correlations, aiding in feature selection and model interpretation. Here, training the model, and evaluating its performance. Perform different analysis as needed like Linear Regression, Support Vector Regressor, Random Forest Regressor and XGboost Regressor. Compared to other machine learning algorithm Linear Regression has high accuracy value 0.918.