PREDICTING HOUSE PRICE USING MACHINE LEARNING

PHASE-4

Development Part-2

INTRODUCTION:

Already we developed our first part of the project which contains data loading, preprocessing and data visualization. In this phase we have to build our second part of the project by selecting a machine learning algorithm, which we want, training the model, and evaluating its performance. Perform different analysis as needed.

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:

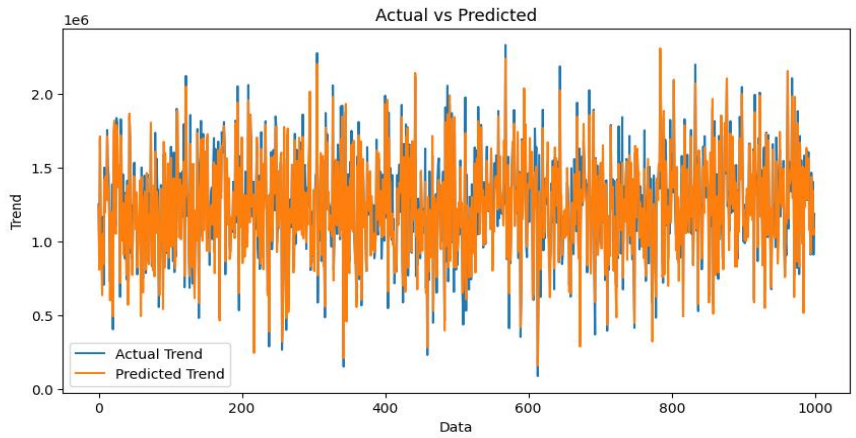
Use the trained model to make predictions on the test data.

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

EVALUATION OF PREDICTING DATA:

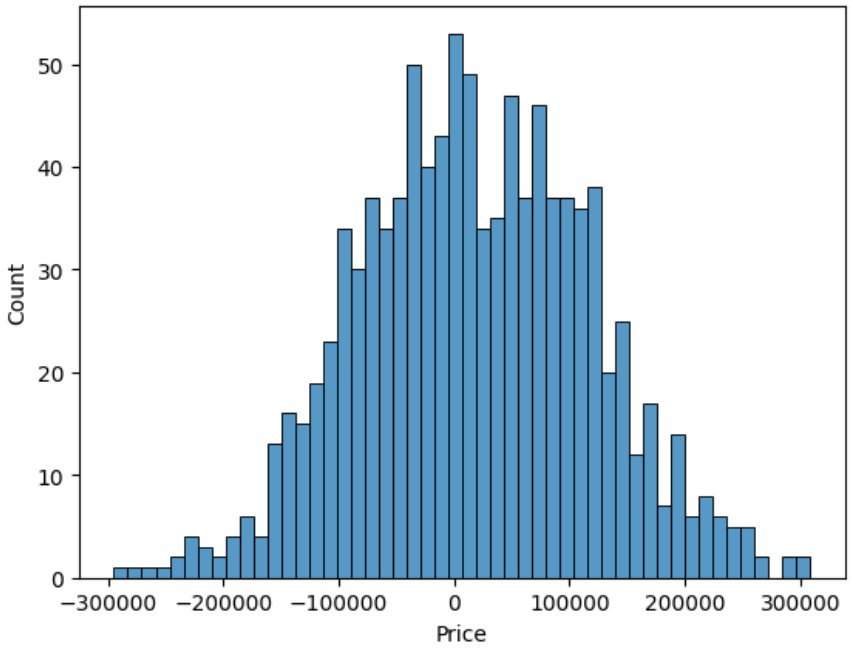
```
[24]  plt.figure(figsize=(12,6))
      plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
      plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
      plt.xlabel('Data')
      plt.ylabel('Trend')
      plt.legend()
      plt.title('Actual vs Predicted')
```

```
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:

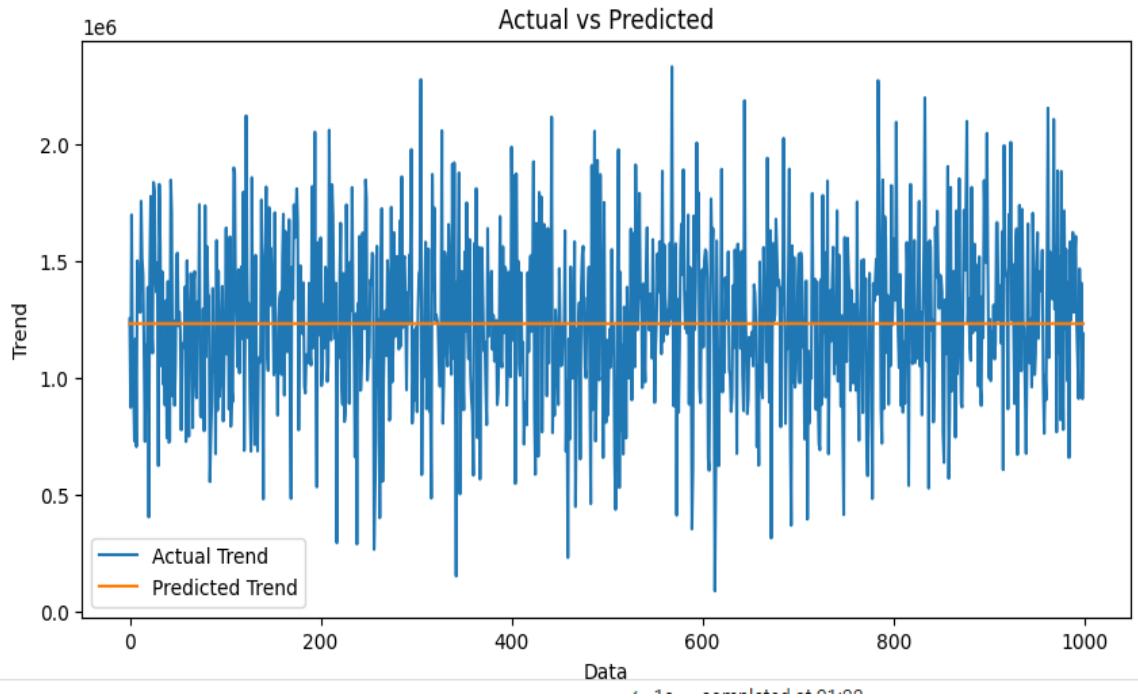
Use the trained model to make predictions on the test data.

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

EVALUATION OF PREDICTING DATA:

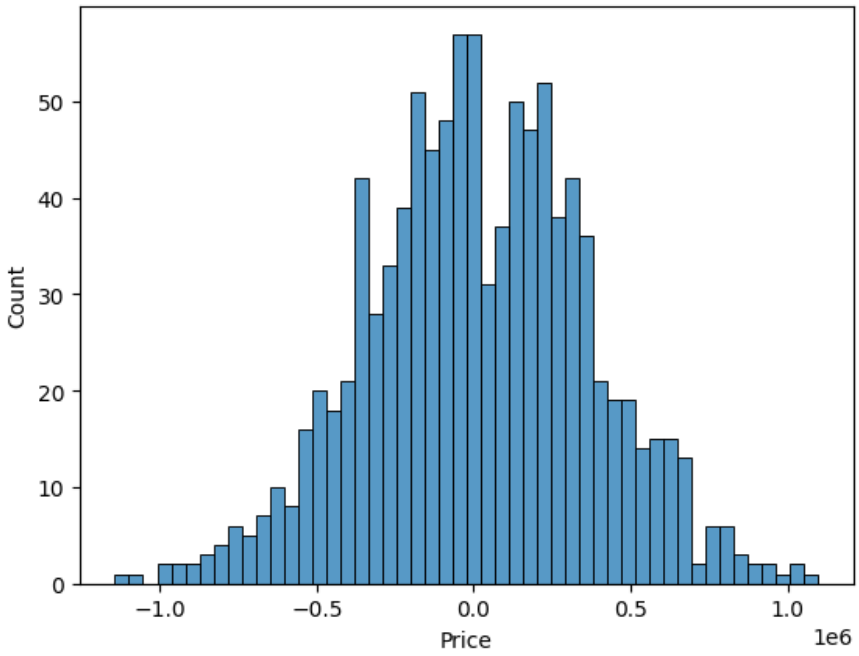
```
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
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:

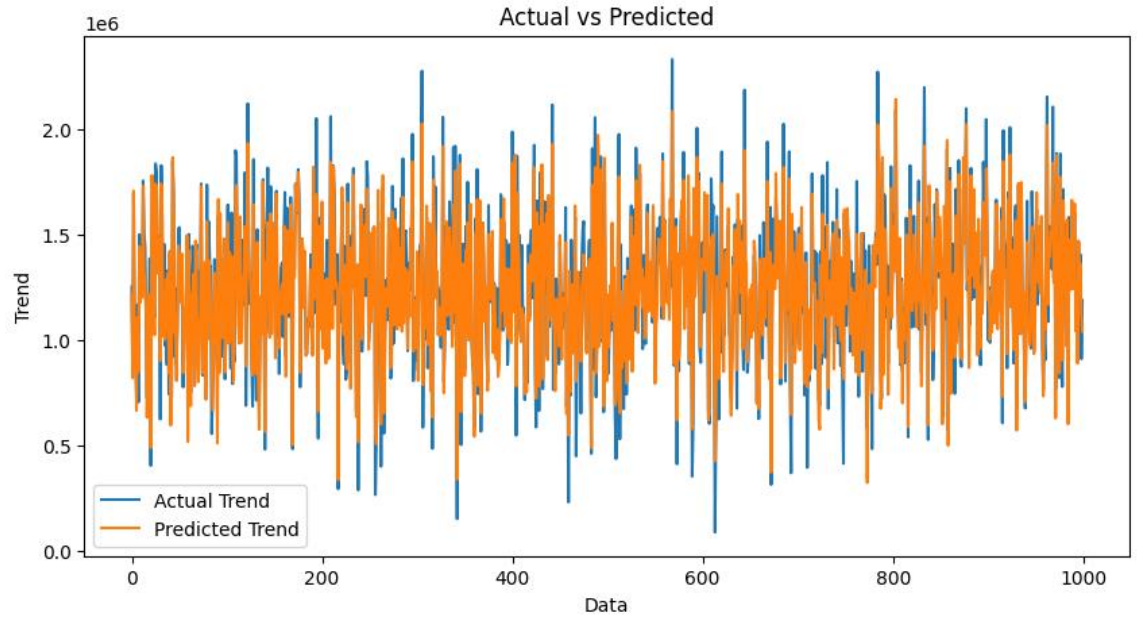
Use the trained model to make predictions on the test data.

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

EVALUATION OF PREDICTING DATA:

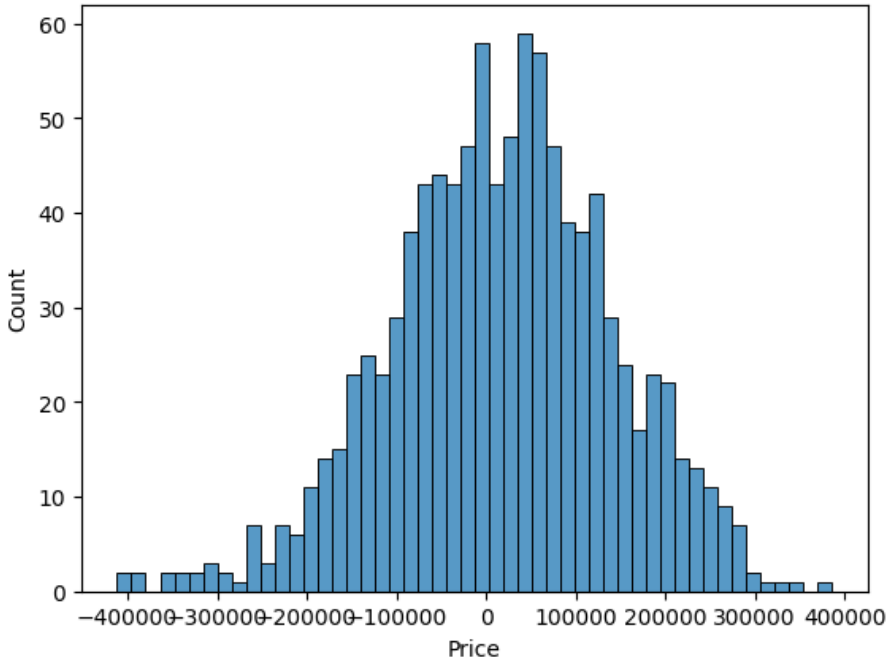
```
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
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).

```
print(r2_score(Y_test, Prediction4))
print(mean_absolute_error(Y_test, Prediction4))
print(mean_squared_error(Y_test, Prediction4))
```

```
0.8805641323024149
98883.8118245981
15303235190.890911
```

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:

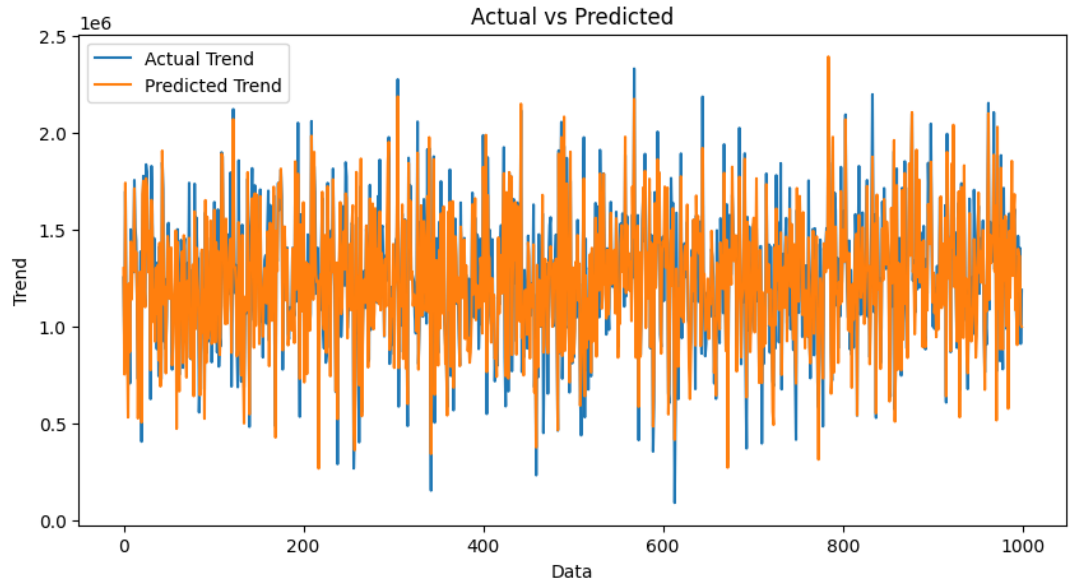
Use the trained model to make predictions on the test data.

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

EVALUATION OF PREDICTING DATA:

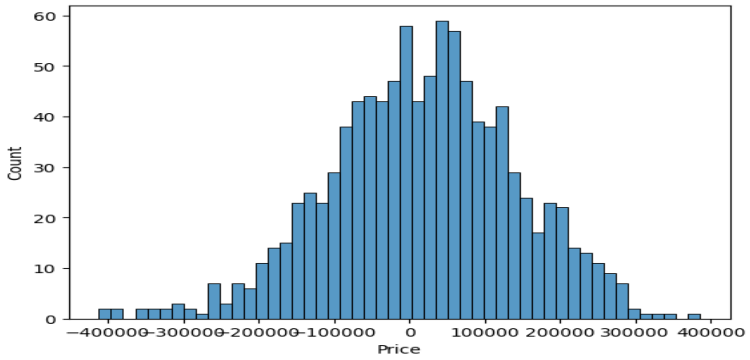
```
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction5, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

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




```
[ ]  sns.histplot((Y_test-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).

```
[50]  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 :

In this phase we had built our second part of the project by selecting a machine learning algorithm, which we want, 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.