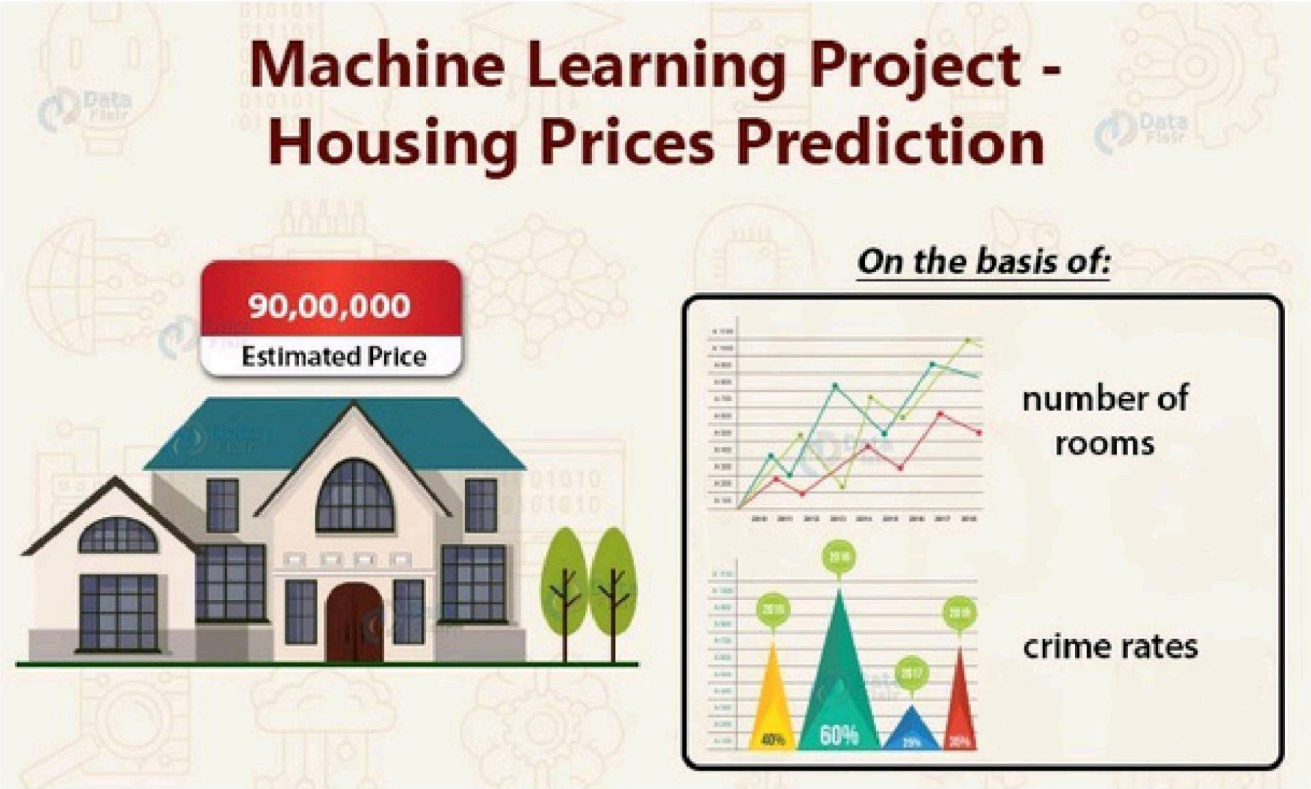
Project Title: House Price Predictor

Topic: In this section we will document the complete project and prepare it for submission.



House Price Prediction

Introduction:

- The real estate market is a dynamic and complex arena, where property values can fluctuate significantly due to a multitude of factors. For both homebuyers and sellers, accurately determining the fair market value of a property is of paramount importance.
- In this era of technological advancement, machine learning has emerged as a game-changing tool in the realm of real estate. One of its most compelling applications is predicting house prices with remarkable accuracy.
- Traditional methods of property valuation, relying on factors such as location, square footage, and recent sales data, are undoubtedly useful. However, they often fall short in capturing the intricacies and nuances that drive real estate market dynamics.
- Machine learning, on the other hand, has the capability to process vast volumes of data and identify patterns that human appraisers might overlook. This technology has the potential to revolutionize the way we value real estate, offering more precise and data-driven predictions.
- In this exploration, we delve into the exciting world of predicting house prices using machine learning. We will uncover how this cutting-edge technology harnesses the power of algorithms and data to create predictive models that consider an array of variables, such as neighborhood characteristics, property features, economic indicators, and even social trends.

- By doing so, machine learning enables us to make informed, databacked predictions about the future value of a property.
- This transformation of the real estate industry is not only beneficial for buyers and sellers but also for investors, developers, and policymakers. Accurate house price predictions can inform investment decisions, urban planning, and housing policy development, leading to a more efficient and equitable real estate market.
- As we embark on this journey into the realm of machine learning for house price prediction, we will explore the various techniques, data sources, and challenges involved.

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

Given data set:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanielltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 Rows x 7 Columns

Here's a list of tools and software commonly used in the process:

1. Programming Language:

- Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

1. Integrated Development Environment (IDE):

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

1. Machine Learning Libraries:

- You'll need various machine learning libraries, including:
- scikit-learn for building and evaluating machine learning models.
- TensorFlow or PyTorch for deep learning, if needed.

- XGBoost, LightGBM, or CatBoost for gradient boosting models.

2. Data Visualization Tools:
   - Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.
3. Data Preprocessing Tools:
   - Libraries like pandas help with data cleaning, manipulation, and preprocessing.
4. Data Collection and Storage:

- Depending on your data source, you might need web scraping tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

1. Version Control:

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

1. Notebooks and Documentation:

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

1. Hyperparameter Tuning:

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

1. Web Development Tools (for Deployment):

- If you plan to create a web application for model deployment, knowledge of web development tools like Flask or Django for backend development, and HTML, CSS, and JavaScript for the front-end can be useful.

1. Cloud Services (for Scalability):

- For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources.

1. External Data Sources (if applicable):

- Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

1. Data Annotation and Labeling Tools (if applicable):

- For specialized projects, tools for data annotation and labeling may be necessary, such as Labelbox or Supervisely.

1. Geospatial Tools (for location-based features):

- If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1.Empathize:

- Understand the needs and challenges of all stakeholders involved in the house price prediction process, including homebuyers, sellers, real estate professionals, appraisers, and investors.
- Conduct interviews and surveys to gather insights on what users value in property valuation and what information is most critical for their decision-making.

2.Define:

- Clearly articulate the problem statement, such as "How might we predict house prices more accurately and transparently using machine learning?"
- Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

3.Ideate:

- Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of house price predictions.
- Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative data, new algorithms, or improved visualization techniques.

4.Prototype:

- Create prototype machine learning models based on the ideas generated during the ideation phase.
- Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

5.Test:

- Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios.
- Assess how well the models meet the defined goals and success criteria, and make adjustments based on user feedback.

6.Implement:

- Develop a production-ready machine learning solution for predicting house prices, integrating the best-performing algorithms and data sources.
- Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

7.Evaluate:

- Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate and relevant in a changing real estate market.

- Gather feedback and insights from users to identify areas for improvement.

8.Iterate:

- Apply an iterative approach to refine the machine learning model based on ongoing feedback and changing user needs.
- Continuously seek ways to enhance prediction accuracy, transparency, and user satisfaction.

9.Scale and Deploy:

- Once the machine learning model has been optimized and validated, deploy it at scale to serve a broader audience, such as real estate professionals, investors, and homeowners.
- Ensure the model is accessible through user-friendly interfaces and integrates seamlessly into real estate workflows.

10.Educate and Train:

- Provide training and educational resources to help users understand how the machine learning model works, what factors it considers, and its limitations.
- Foster a culture of data literacy among stakeholders to enhance trust in the technology.

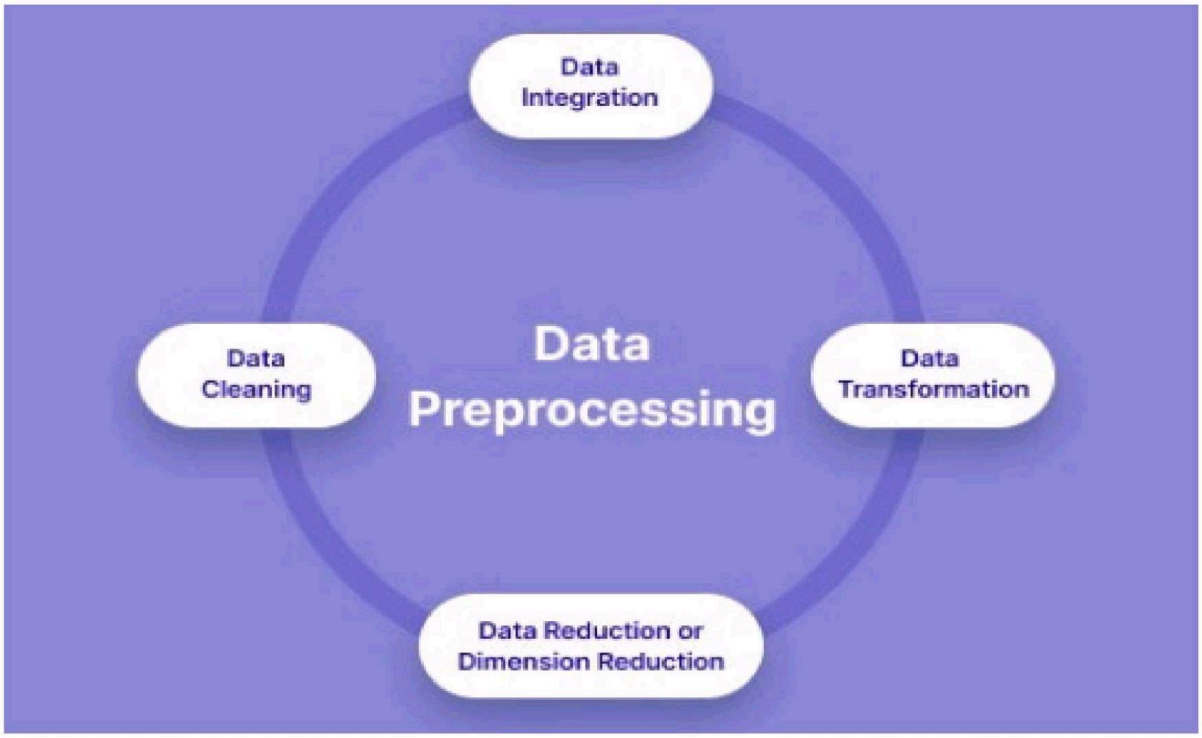
2.DESIGN INTO INNOVATION

1. Data Collection:

Gather a comprehensive dataset that includes features such as location, size, age, amenities, nearby schools, crime rates, and other relevant variables.

2.Data Preprocessing:

Clean the data by handling missing values, outliers, and encoding categorical variables. Standardize or normalize numerical features as necessary.



3.BUILD LOADING AND PREPROCESSING THE DATASET

1. Data Collection:

Obtain a dataset that contains information about houses and their corresponding prices. This dataset can be obtained from sources like real estate websites, government records, or other reliable data providers.

1. Load the Dataset:

- Import relevant libraries, such as pandas for data manipulation and numpy for numerical operations.
- Load the dataset into a pandas DataFrame for easy data handling. You can use pd.read_csv() for CSV files or other appropriate functions for different file formats.

Program:

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")

/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:

UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np_minversion} and

<{np_maxversion}"

Loading Dataset:

dataset = pd.read_csv('E:/USA_Housing.csv')

Output:

KitchenQual0

SaleCondition0

LandSlope0

dtype: int64

TOTAL MISSING VALUES: 0

1. Feature Engineering:

Create new features or transform existing ones to capture additional information that may impact house prices. For example, you can calculate the price per square foot.

1. Data Encoding:

Convert categorical variables (e.g., location) into numerical format using techniques like one-hot encoding.

1. Train-Test Split:

Split the dataset into training and testing sets to evaluate the machine learning model's performance.

Program:

X = df.drop('price', axis=1) # Features y = df['price'] # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION etc.,

1. Feature Engineering:

- As mentioned earlier, feature engineering is crucial. It involves creating new features or transforming existing ones to provide meaningful information for your model.
- Extracting information from textual descriptions (e.g., presence of keywords like "pool" or "granite countertops").
- Calculating distances to key locations (e.g., schools, parks) if you have location data.

2. Data Preprocessing & Visualisation:

Continue data preprocessing by handling any remaining

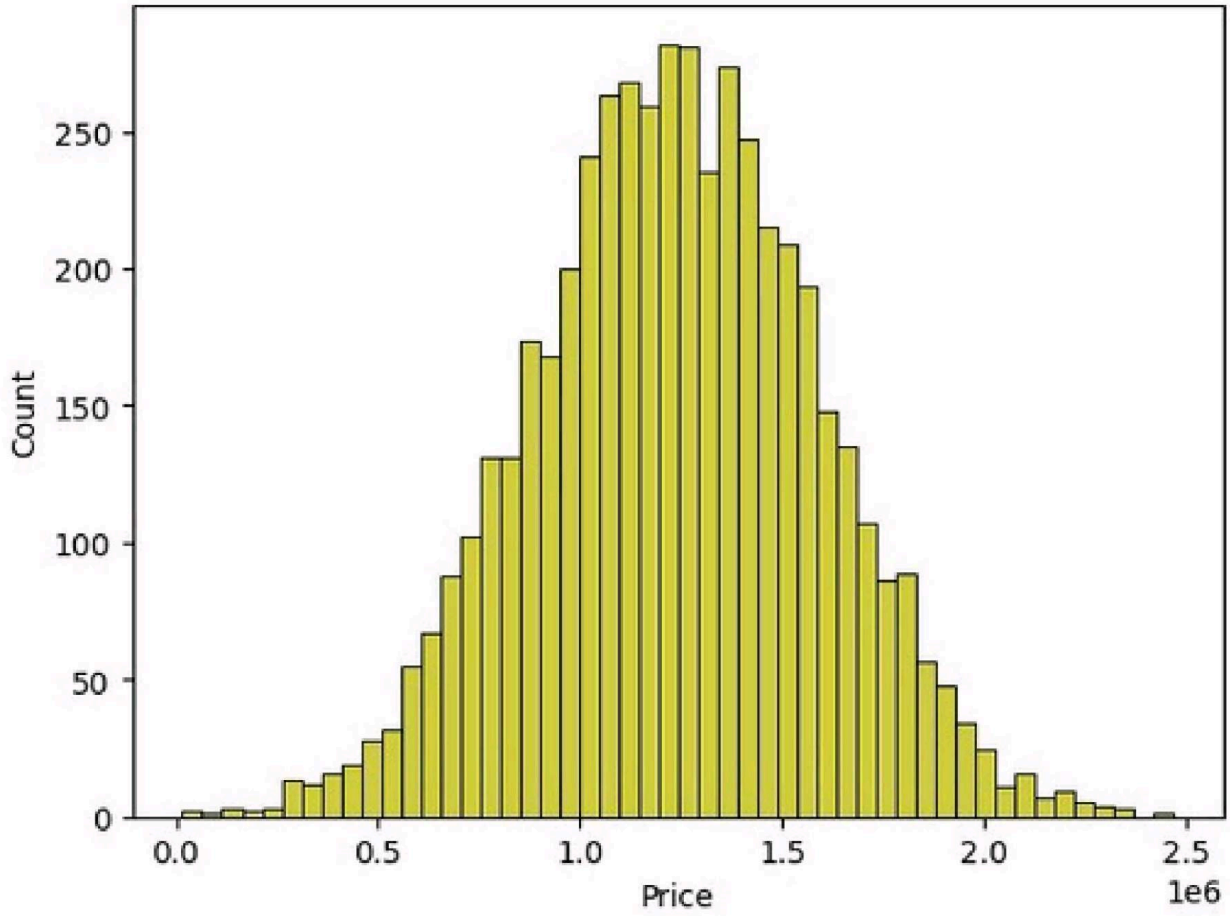
missing values or outliers based on insights from your data exploration.

Visualisation and Pre-Processing of Data:

In [1]: sns.histplot(dataset, x='Price', bins=50, color='y')

Out[1]:

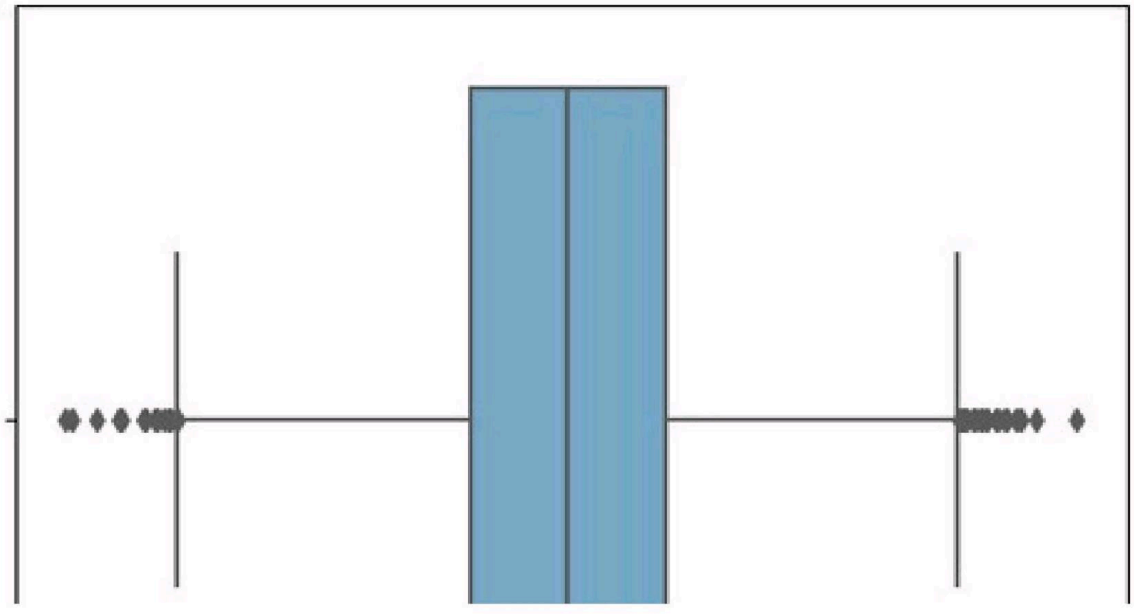
<Axes: xlabel='Price', ylabel='Count'>

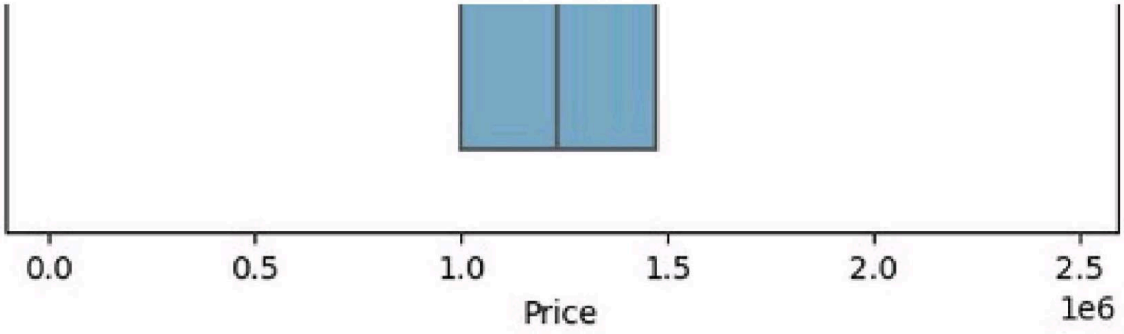


In [2]: sns.boxplot(dataset, x='Price', palette='Blues')

Out[2]:

<Axes: xlabel='Price'>

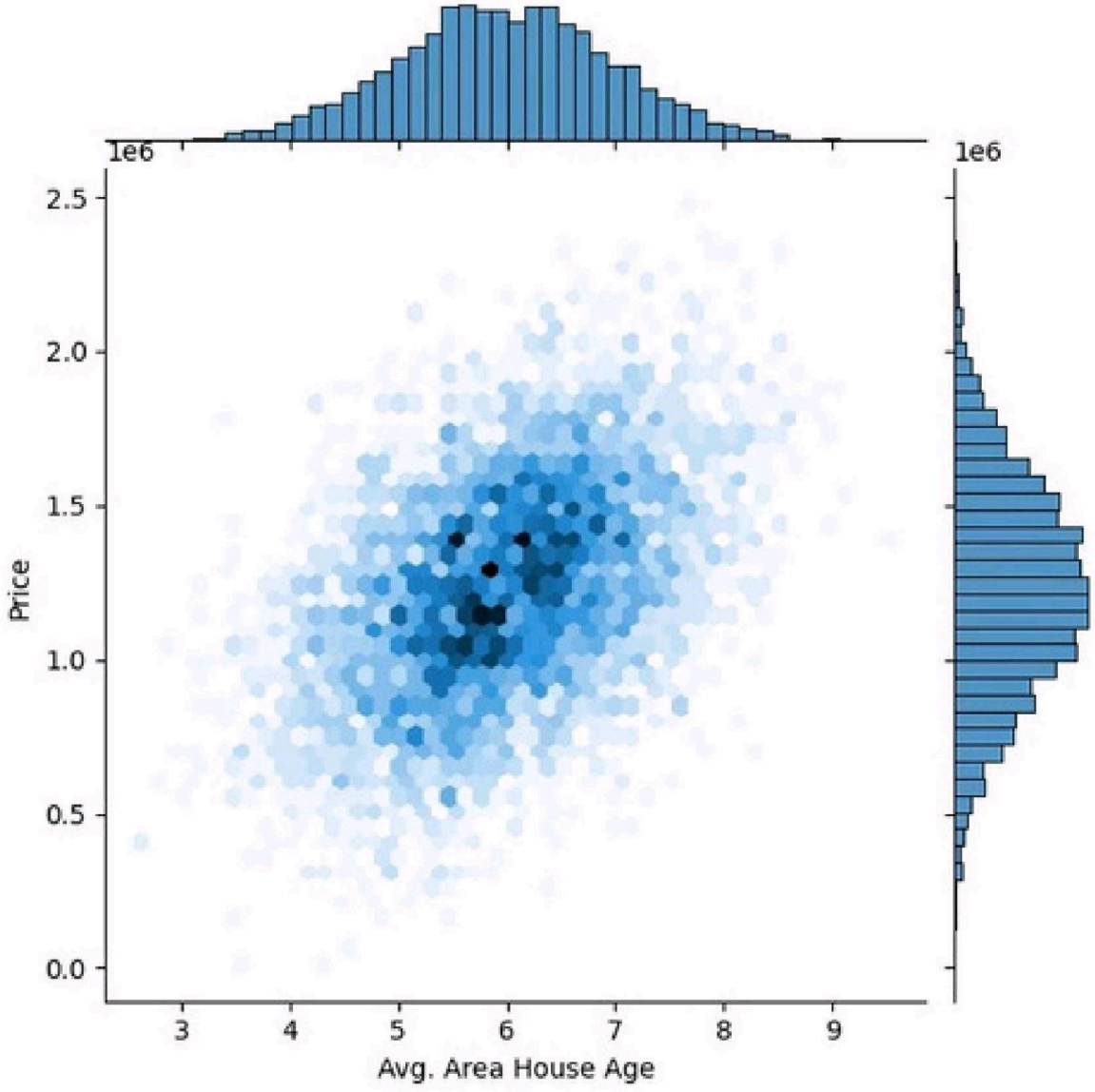




In [3]: sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')

Out[3]:

<seaborn.axisgrid.JointGrid at 0x7caf1d571810>

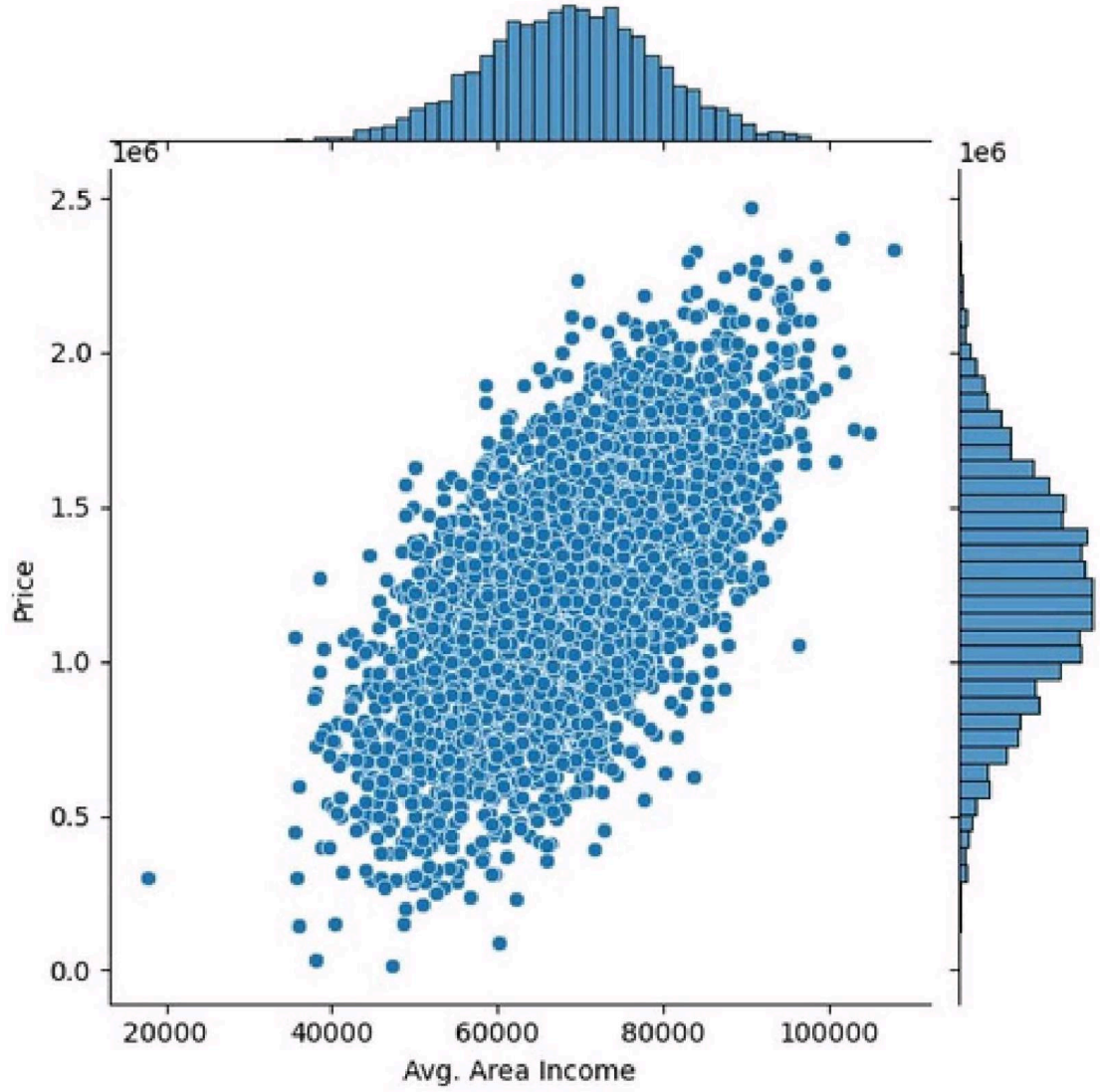


In [4]:

sns.jointplot(dataset, x='Avg. Area Income', y='Price')

Out[4]:

<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>

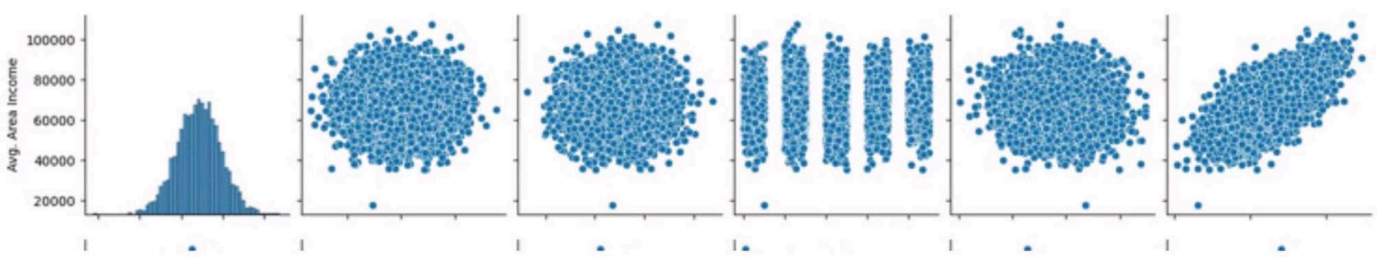


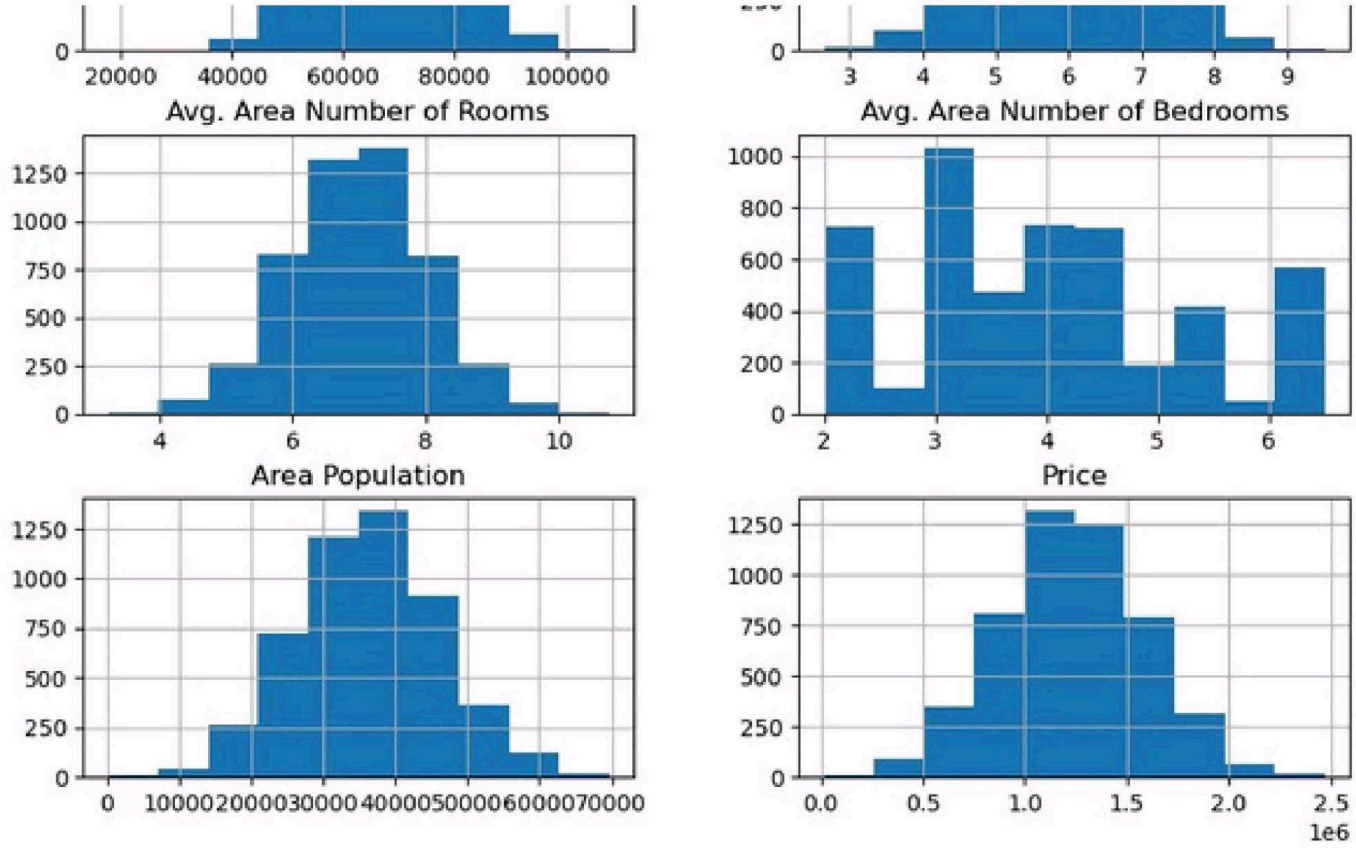
In [5]: plt.figure(figsize=(12,8))sns.pairplot(dataset)

Out[5]:

<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>

<Figure size 1200x800 with 0 Axes>





Visualising Correlation:

In [7]: dataset.corr(numeric_only=True)

Out[7]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of	-	-	1.000000	0.462695	0.002040	0.335664

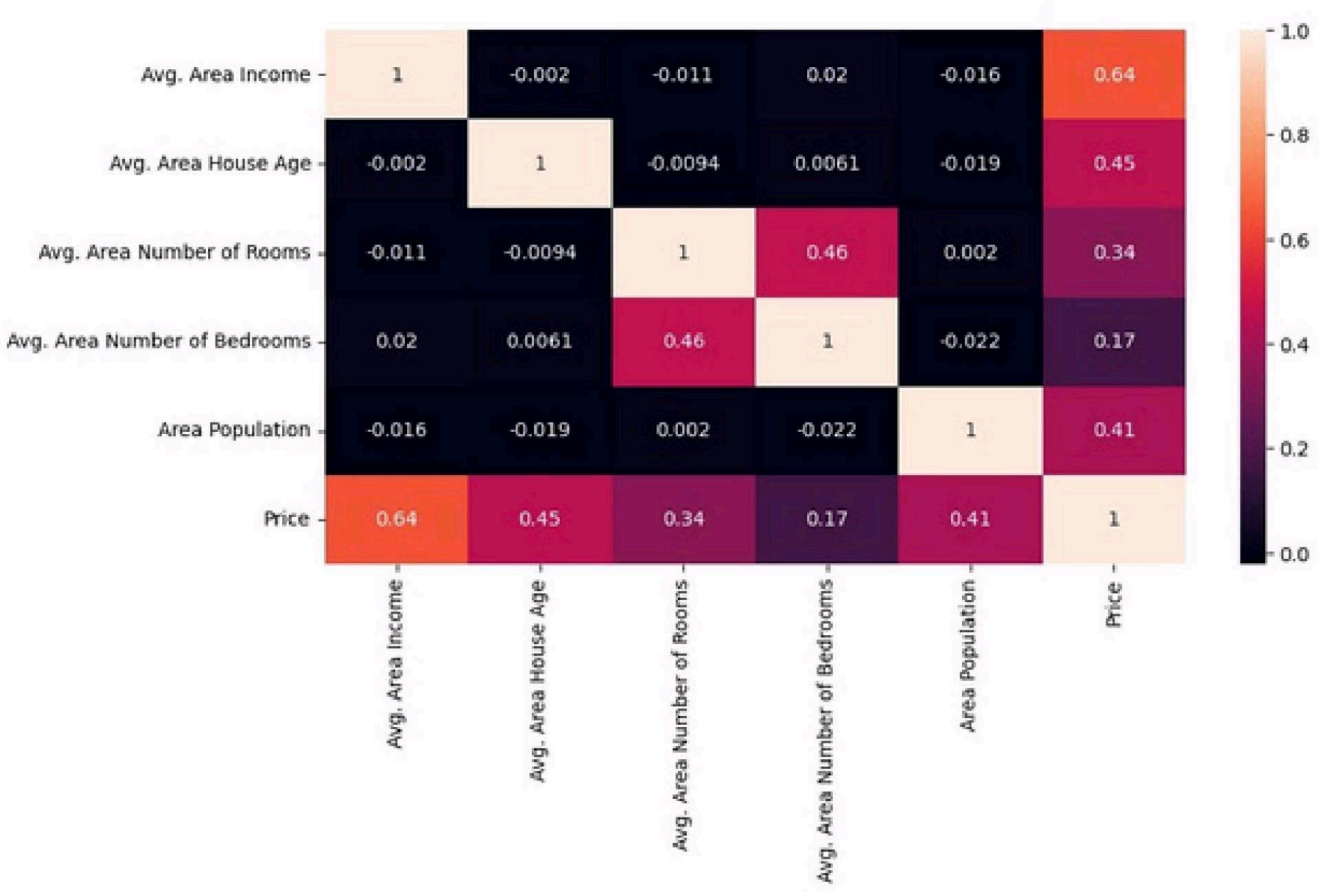
Rooms	0.011032	0.009428				
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

In [8]:

```
plt.figure(figsize=(10,5))sns.heatmap(dataset.corr(numeric_only = Tru

e), annot=True)
```

Out[8]:

<Axes: >



In [3]:

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

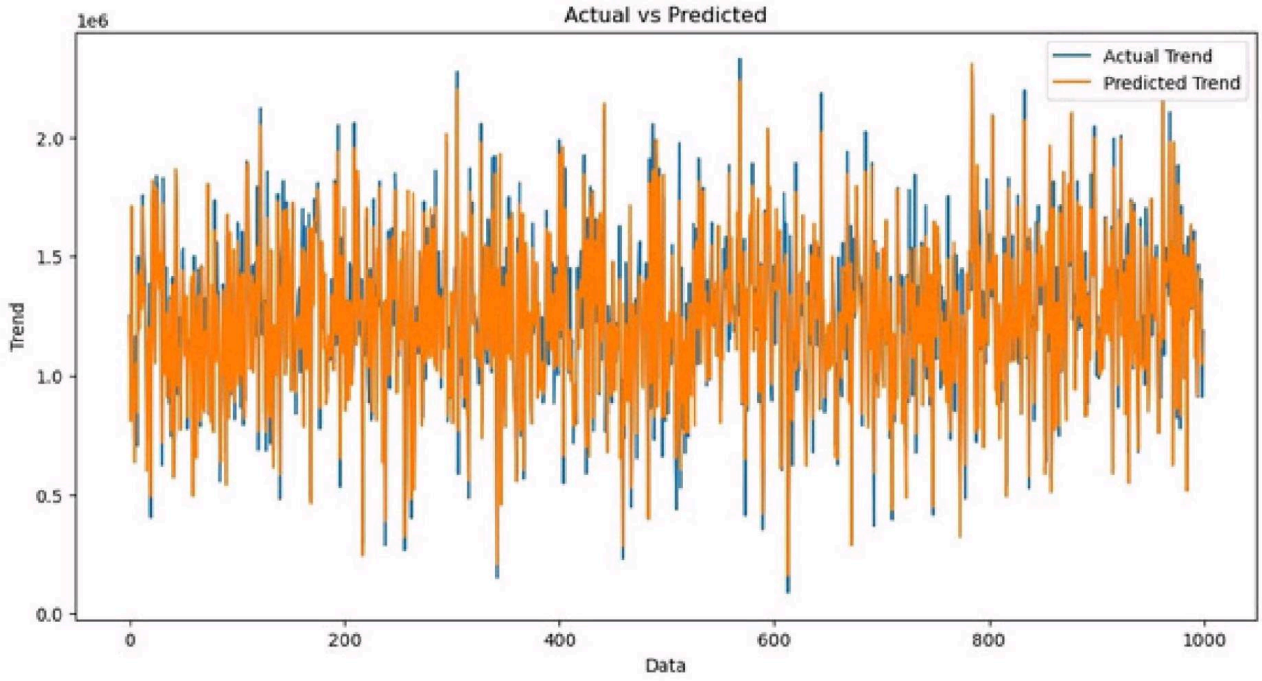
Evaluation of Predicted Data

In [4]:

```
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 Tr

end')

plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')
```

Out[4]:

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

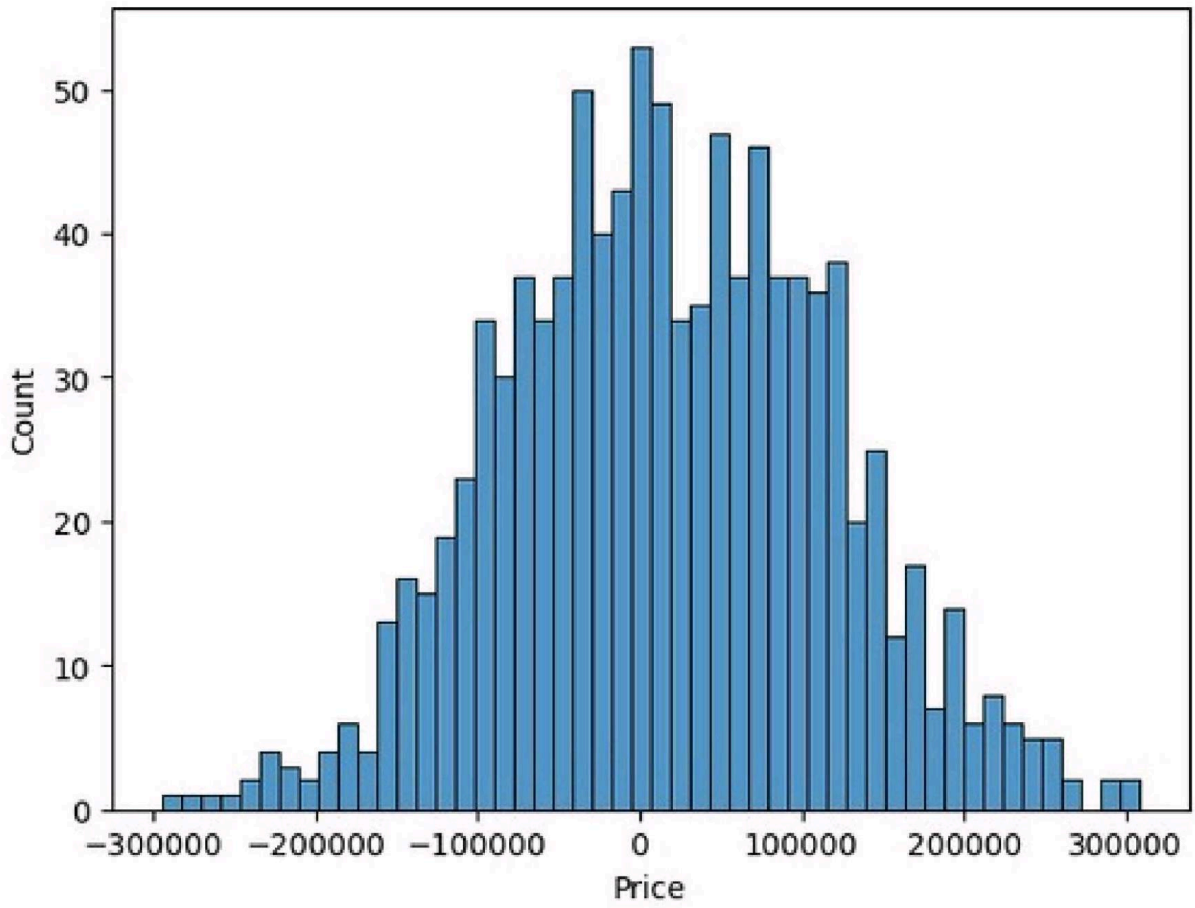


In [5]:

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

Out[5]:

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



In [6]:

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

Out[6]:

0.9182928179392918 82295.49779231755

10469084772.975954

Model 2 – Support Vector Regressor

In [7]:

```
model_svr = SVR()
```

In [8]:

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

Out[8]:

SVR

SVR()

Predicting Prices

In [9]:

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

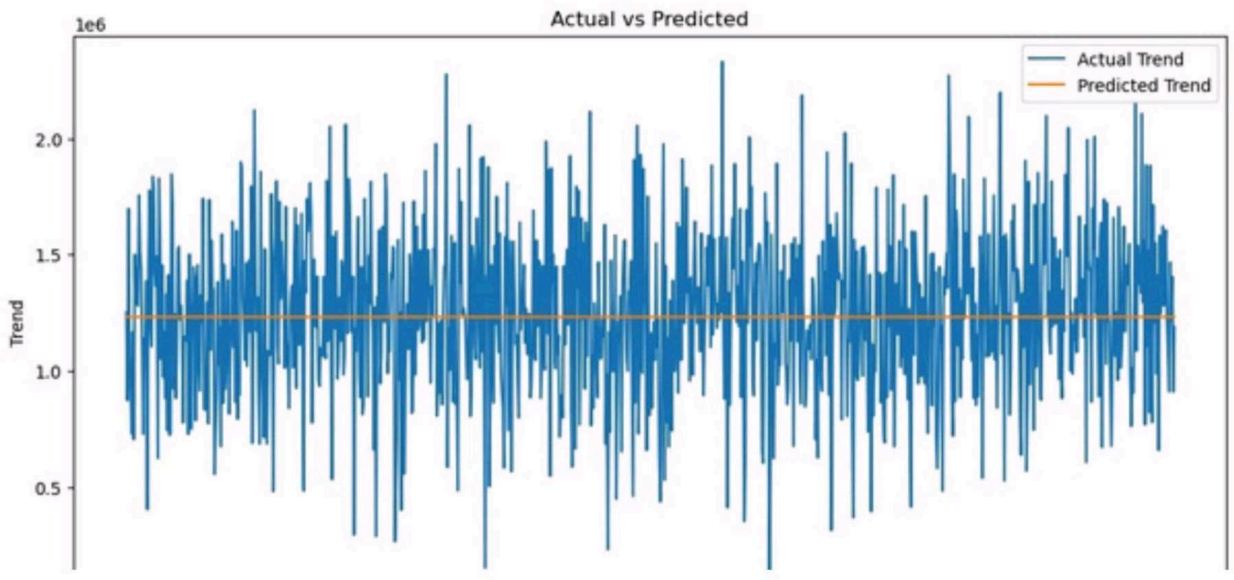
Evaluation of Predicted Data

In [10]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Tr
end')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[10]:

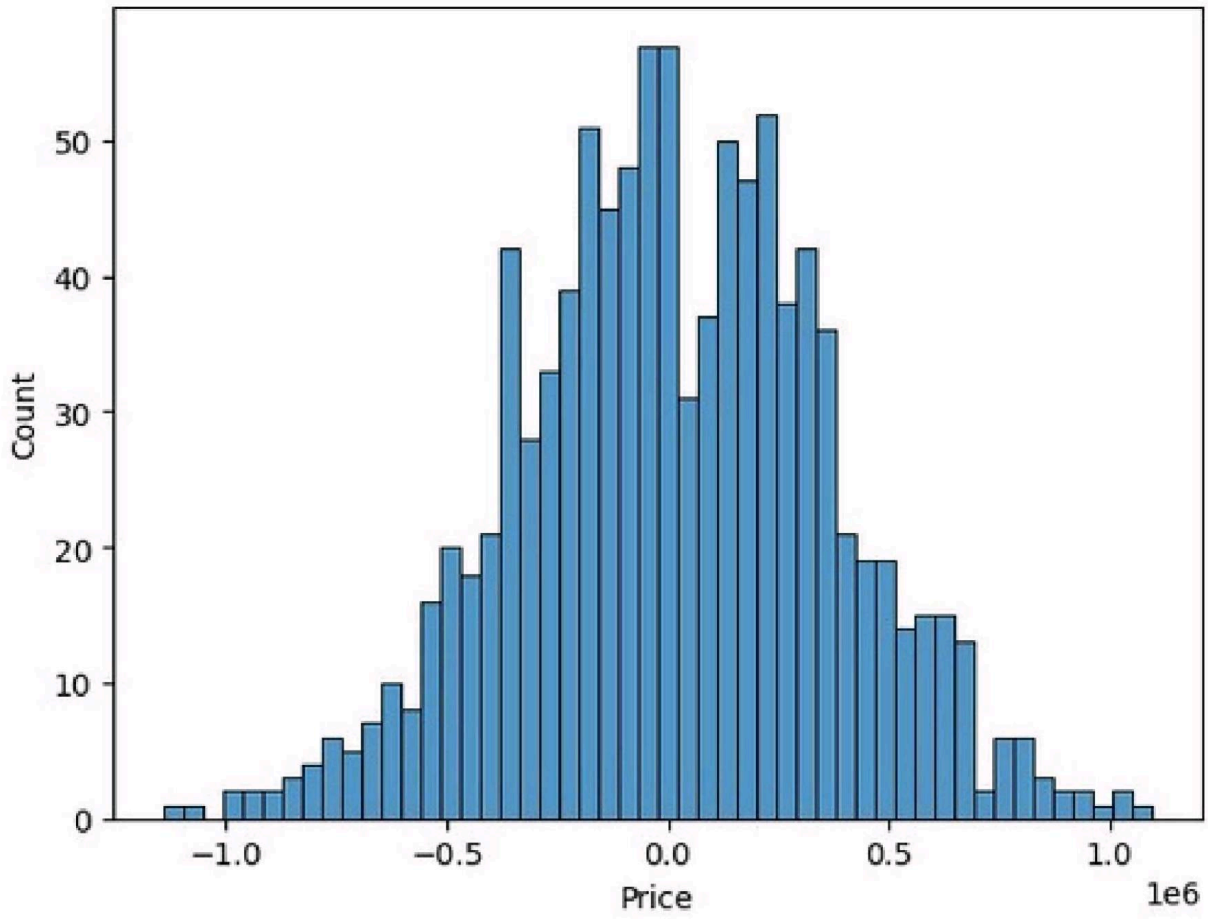
Text(0.5, 1.0, 'Actual vs Predicted')



In [11]: sns.histplot((Y_test-Prediction2), bins=50)

Out[12]:

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



In [12]:

```
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

Model 3 – Lasso Regression

In [13]:

```
model_lar = Lasso(alpha=1)
```

In [14]:

```
model_lar.fit(X_train_scal,Y_train)
```

Out[14]:

```
Lasso(alpha=1)
```

Predicting Prices

In [15]:

```
Prediction3 = model_lar.predict(X_test_scal)
```

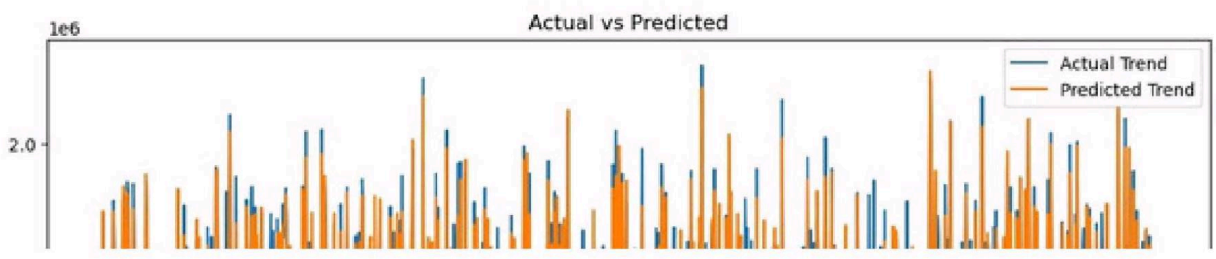
Evaluation of Predicted Data

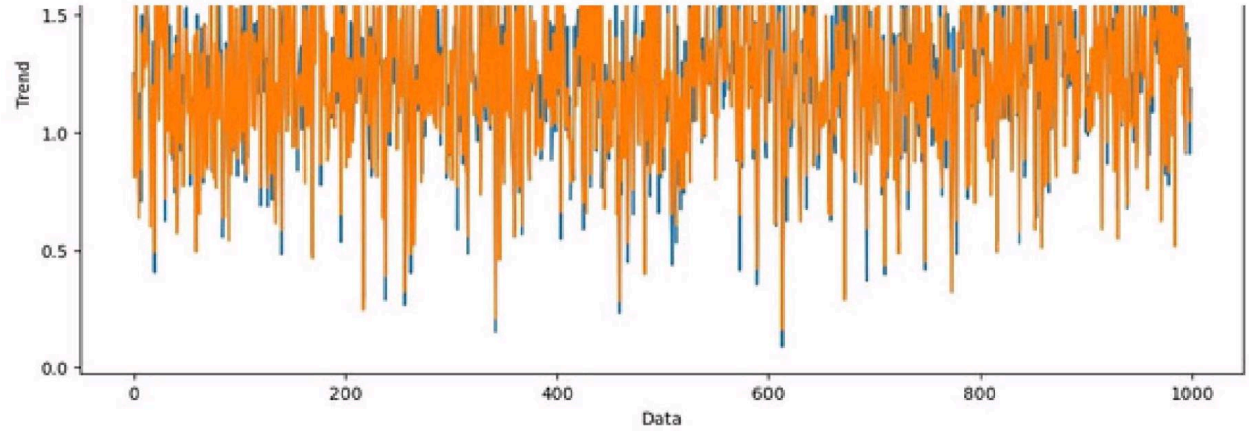
In [16]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction3, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[16]:

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

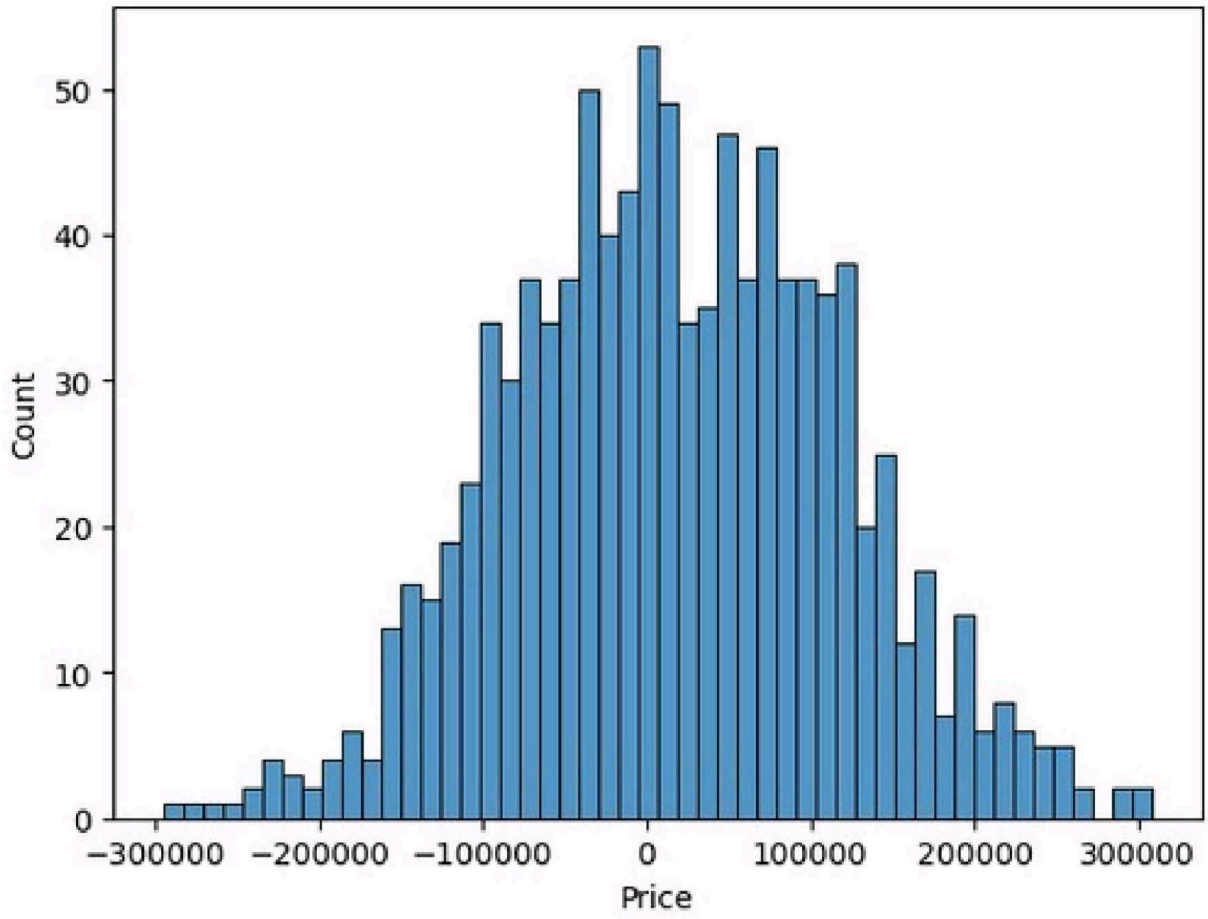




In [17]: sns.histplot((Y_test-Prediction3), bins=50)

Out[17]:

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



In [18]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
```

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,

n_estimators=100, n_jobs=None,

num_parallel_tree=None, predictor=None, random_state=None, ...)

1. Model Training:

Split your dataset into training and testing sets (as shown earlier) and train the selected model on the training data. Here's an example using Linear Regression:

1. Model Evaluation:

Evaluate your model's performance using appropriate regression metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). For example:

PYTHON PROGRAM:

Import necessary libraries from sklearn.feature_selection import SelectKBest, f_regression from sklearn.linear_model

import LinearRegression from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import

mean_squared_error, r2_score import numpy as np selector = SelectKBest(score_func=f_regression, k=k)

X_train_selected = selector.fit_transform(X_train, y_train)

Model Selection

Let's choose both Linear Regression and Random Forest Regressor for comparison.

linear_reg_model = LinearRegression() random_forest_model = RandomForestRegressor(n_estimators=100,

random_state=42)

Train the models on the selected features linear_reg_model.fit(X_train_selected, y_train)

random_forest_model.fit(X_train_selected, y_train)

Evaluate the models on the test set

X_test_selected = selector.transform(X_test)

Make predictions linear_reg_predictions = linear_reg_model.predict(X_test_selected) random_forest_predictions =

random_forest_model.predict(X_test_selected)

```
# Evaluate model performance def evaluate_model(predictions, model_name): mse = mean_squared_error(y_test, predictions) r2 = r2_score(y_test, predictions) print(f"{model_name} Model Evaluation:") print(f"Mean Squared Error (MSE): {mse}") print(f"R-squared (R2) Score: {r2}\n") # Performance Measure elr_mse = mean_squared_error(y_test, pred) elr_rmse = np.sqrt(lr_mse) elr_r2 = r2_score(y_test, pred)

# Show Measures

result = '''

MSE : {}

RMSE : {}

R^2 : {}

'''.format(lr_mse, lr_rmse, lr_r2) print(result)

# Model Comparision

names.append("elr") mses.append(elr_mse) rmses.append(elr_rmse) r2s.append(elr_r2)

evaluate_model(linear_reg_predictions, "Linear Regression") evaluate_model(random_forest_predictions, "Random Forest Regressor")
```

OUTPUT:

Linear Regression Model Evaluation:

Mean Squared Error (MSE): 10089009300.893988

R-squared (R2) Score: 0.9179971706834331

Random Forest Regressor Model Evaluation:

Mean Squared Error (MSE): 14463028828.265167

R-squared (R2) Score: 0.8824454166872736

MSE : 10141766848.330585

RMSE : 100706.33966305491

R^2 : 0.913302484308253

Model Comparison:

The less the Root Mean Squared Error (RMSE), The better the model is.

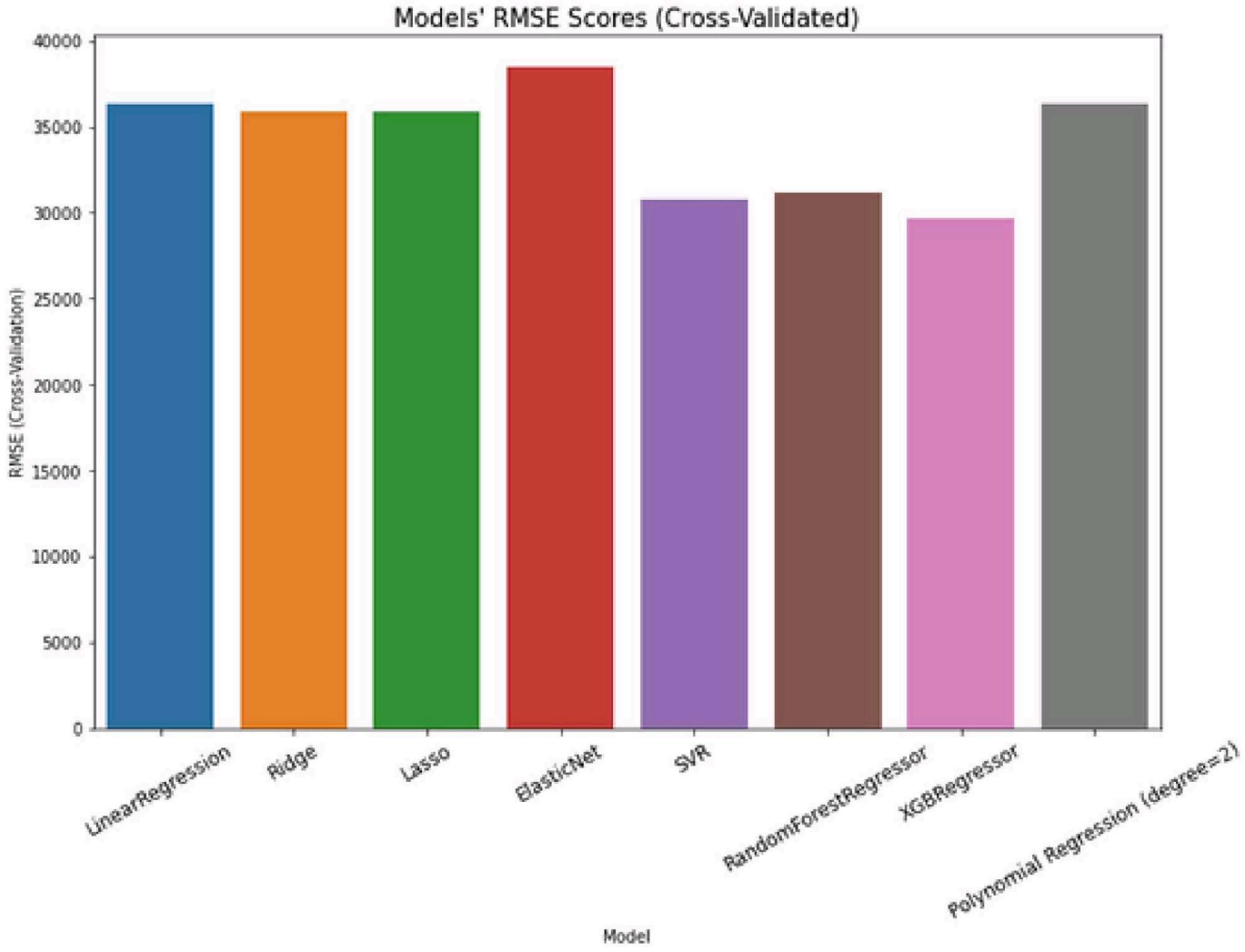
In [30]: models.sort_values(by="RMSE (Cross-Validation)")

Out[30]:

	Model	MAE	MSE	RMSE	R2 Score	RMSE

"])

```
plt.title("Models' RMSE Scores (Cross-Validated)", size=15)
plt.xticks(rotation=30, size=12)
plt.show()
```



Evaluation of Predicted Data

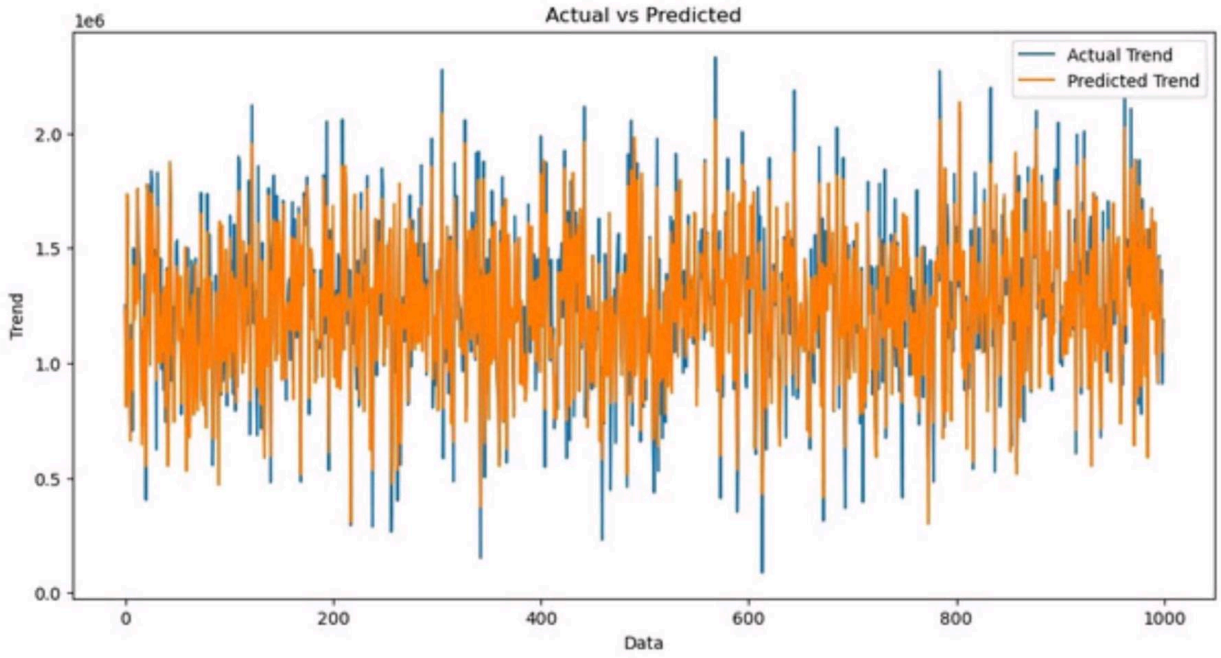
In [22]:

```
plt.figure(figsize=(12,6))
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')
```

Out[22]:

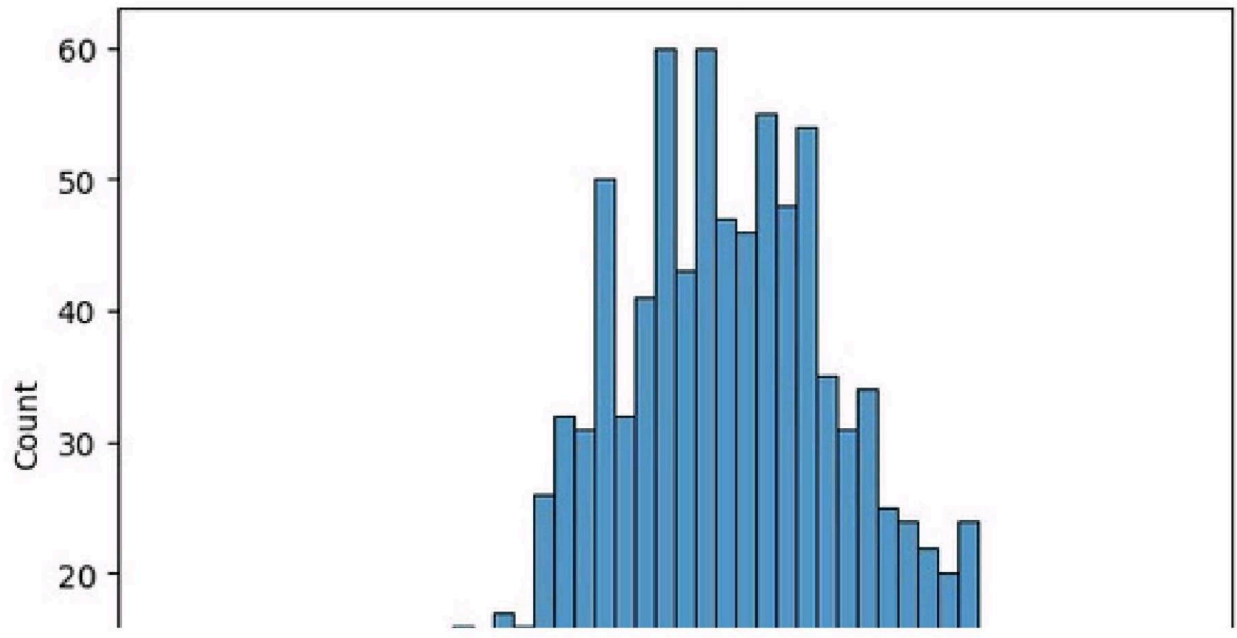
```
Text(0.5, 1.0, 'Actual vs Predicted')
```



In [23]:
```
sns.histplot((Y_test-Prediction4), bins=50)
```

Out[23]:

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



costs associated with manual appraisals, benefiting both businesses and individuals in terms of appraisal expenses.

1. Scalability:

Machine learning models can be applied at scale, making it

possible to assess property values in large real estate portfolios, entire neighborhoods, or even entire cities.

1. Fairness and Consistency:

Machine learning models evaluate properties objectively based on data, reducing potential human bias in property valuation and promoting fairness and consistency in pricing.

1. Real-Time Monitoring:

Machine learning models can provide real-time monitoring of property values, allowing stakeholders to react promptly to market changes or anomalies.

1. Market Forecasting:

By analyzing historical data and current market conditions, machine learning models can make forecasts about future property values, enabling more informed investment decisions.

1. Urban Planning:

Accurate property valuations can inform urban planning and development decisions, ensuring that communities are built in a way that aligns with market dynamics and housing needs.

1. Market Competitiveness:

Real estate professionals can gain a competitive edge by using machine learning to provide more accurate property valuations and better serve clients.

CONCLUSION:

Predicting house prices using machine learning is a transformative and promising approach that has the potential to revolutionize the real estate industry. Throughout this exploration, we have uncovered the remarkable capabilities of machine learning in providing more accurate, data-driven, and nuanced predictions for property values. As we conclude, several key takeaways and implications emerge:

Improved Accuracy: Machine learning models consider a myriad of variables, many of which may be overlooked by traditional methods. This results in more accurate predictions, benefiting both buyers and sellers who can make informed decisions based on a property's true value.

Data-Driven Insights: These models provide valuable insights into the real estate market by identifying trends, neighborhood characteristics, and other factors that influence property prices. This information can be invaluable for investors, developers, and policymakers seeking to make strategic decisions.

Market Efficiency: The increased accuracy in pricing predictions can lead to a more efficient real estate market, reducing overvaluation and undervaluation of properties. This contributes to a fairer and more transparent marketplace.

Challenges and Considerations: Machine learning for house price prediction is not without its challenges. Data quality, model interpretability, and ethical concerns are important considerations. Addressing these issues is crucial for the responsible and ethical deployment of this technology.

Continual Advancement: The field of machine learning is continually evolving, and as it does, so will the accuracy and capabilities of predictive models. As more data becomes available and algorithms improve, we can expect even more sophisticated predictions in the future.

In conclusion, the application of machine learning in predicting house prices is a groundbreaking development with farreaching implications. It empowers individuals, businesses, and governments to navigate the real estate market with more confidence and precision. However, it is essential to approach this technology with a clear understanding of its potential and limitations, ensuring that its benefits are harnessed responsibly for the betterment of the real estate industry and society as a whole. As machine learning continues to advance, we can look forward to a future where property valuation becomes increasingly precise and data-informed.

PREPARED BY,

JAYASURYA.M