PREDICTING HOUSE PRICE USING MACHINE LEARNING

TEAM MEMBER

au620121243003: Ashok S

Phase-4 submission document

Project Title: House Price Predictor

Phase 4: *Development Part 2*

Topic: *Continue building the house price prediction model by feature engineering, model training, and evaluation.*

House Price Prediction

Introduction:

❖ The process of building a house price prediction model is a critical endeavor in the realm of real estate, finance, and property valuation. Accurately estimating the price of a house is essential for *buyers, sellers, and investors* to make informed decisions. In this comprehensive guide, we will continue to delve deeper into the construction of a robust house price prediction model by focusing on three fundamental components: feature selection, model training, and evaluation.

❖ Feature selection is the process of identifying and selecting the most relevant features from a dataset to improve the performance of a machine learning model. This is an important step in building a house price prediction model, as it can help to reduce overfitting and improve the generalization ability of the model.

❖ Model training is the process of feeding the selected features to a machine learning algorithm and allowing it to learn the relationship between the features and the target variable *(i.e., house price)*. Once the model is trained, it can be used to predict the house prices of new houses, given their features.

Model evaluation is the process of assessing the performance of a trained machine learning model on a held-out test set. This is important to ensure that the model is generalizing well and that it is not overfitting the training data.

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\nDanieltown, 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

Overview of the process:

The following is an overview of the process of building a house price prediction model by feature selection, model training, and evaluation:

1. **Prepare the data:** This includes cleaning the data, removing outliers, and handling missing values.

2. **Perform feature selection:** This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3.   **Train the model:** There are many different machine learning algorithms that can be used for house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

4.   **Evaluate the model:** This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5.   **Deploy the model:** Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the house prices of new houses.

PROCEDURE:

Feature selection:

1.   **Identify the target variable.** This is the variable that you want to predict, such as house price.

2.   **Explore the data.** This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3.   **Remove redundant features.** If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

4.   **Remove irrelevant features.** If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

Feature Selection:

We are selecting numerical features which have more than 0.50 or less than -0.50 correlation rate based on Pearson Correlation Method—which is the default value of parameter "method" in corr() function. As for selecting categorical features, I selected the categorical values which I believe have significant effect on the target variable such as Heating and MSZoning.

In [1]:

```python
important_num_cols = list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.50) | (df.corr()["SalePrice"]<-0.50)].index)

cat_cols = ["MSZoning", "Utilities","BldgType","Heating","KitchenQual","SaleCondition","LandSlope"]

important_cols = important_num_cols + cat_cols

df = df[important_cols]
```

Checking for the missing values

In [2]:

```python
print("Missing Values by Column")

print("-"*30)

print(df.isna().sum())

print("-"*30)

print("TOTAL MISSING VALUES:",df.isna().sum().sum())
```

Missing Values by Column

-————--—————

OverallQual	0
YearBuilt	0
YearRemodAdd	0
TotalBsmtSF	0
1stFlrSF	0
GrLivArea	0
FullBath	0
TotRmsAbvGrd	0
GarageCars	0
GarageArea	0
SalePrice	0
MSZoning	0
Utilities	0
BldgType	0
Heating	0
KitchenQual	0
SaleCondition	0
LandSlope	0

dtype: int64

-_____-_-_____

TOTAL MISSING VALUES: 0

Model training:

1.   **Choose a machine learning algorithm.** There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests are Covered above.

Machine Learning Models:

In [3]:

```python
models = pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 Score","RMSE (Cross-Validation)"])
```

Linear_Regression:

In [4]:

```python
lin_reg = LinearRegression()

lin_reg.fit(X_train, y_train)

predictions = lin_reg.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)
```

```python
print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(lin_reg)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "LinearRegression","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[4]:

MAE: 23567.890565943395

MSE: 1414931404.6297863

RMSE: 37615.57396384889

R2 Score: 0.8155317822983865

-_____-_-_____

RMSE Cross-Validation: 36326.451444669496

Ridge Regression:

In [5]:

```python
ridge = Ridge()ridge.fit(X_train, y_train)predictions = ridge.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)
```

```python
print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(ridge)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "Ridge","MAE": mae, "MSE": mse, "RMSE": rmse,
"R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}model
s = models.append(new_row, ignore_index=True)
```

Out[5]:

MAE: 23435.50371200822

MSE: 1404264216.8595588

RMSE: 37473.513537691644

R2 Score: 0.8169224907874508

-_____- -_____

RMSE Cross-Validation: 35887.852791598336

Lasso Regression:

In [6]:

```python
lasso = Lasso()lasso.fit(X_train, y_train)predictions = lasso.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)
```

```python
print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(lasso)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "Lasso","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[6]:

MAE: 23560.45808027236

MSE: 1414337628.502095

RMSE: 37607.680445649596

R2 Score: 0.815609194407292

RMSE Cross-Validation: 35922.76936876075

Elastic_Net:

In [7]:

```python
elastic_net = ElasticNet()elastic_net.fit(X_train, y_train)predictions = elastic_net.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
```

```python
print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(elastic_net)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "ElasticNet","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}
models = models.append(new_row, ignore_index=True)
```

Out[7]:

MAE: 23792.743784996732

MSE: 1718445790.1371393

RMSE: 41454.14080809225

R2 Score: 0.775961837382229

-_____-_-_____

RMSE Cross-Validation: 38449.00864609558

Support Vector Machines:

In [8]:

```python
svr = SVR(C=100000)svr.fit(X_train, y_train)predictions = svr.predict(X_test)
```

```python
mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(svr)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "SVR","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[9]:

MAE: 17843.16228084976

MSE: 1132136370.3413317

RMSE: 33647.234215330864

R2 Score: 0.852400492526574

RMSE Cross-Validation: 30745.475239075837

Random Forest Regressor:

In [9]:

```python
random_forest = RandomForestRegressor(n_estimators=100)random_forest.fit(X_train, y_train)predictions = random_forest.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(random_forest)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[9]:

MAE: 18115.11067351598

MSE: 1004422414.0219476

RMSE: 31692.623968708358

R2 Score: 0.869050886899595

-‾‾‾‾‾‾‾-_-‾‾‾‾‾‾‾

RMSE Cross-Validation: 31138.863315259332

XGBoost Regressor:

In [10]:

```python
xgb = XGBRegressor(n_estimators=1000, learning_rate=0.01)xgb.fit(X_train, y_train)predictions = xgb.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(xgb)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "XGBRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[10]:

MAE: 17439.918396832192

MSE: 716579004.5214689

RMSE: 26768.993341578403

R2 Score: 0.9065777666861116

-_____-_-_____

RMSE Cross-Validation: 29698.84961808251

Polynomial Regression (Degree=2)

In [11]:

```python
poly_reg = PolynomialFeatures(degree=2)X_train_2d = poly_reg.fit_transform(X_train)X_test_2d = poly_reg.transform(X_test)

lin_reg = LinearRegression()lin_reg.fit(X_train_2d, y_train)predictions = lin_reg.predict(X_test_2d)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r_squared)

print("-"*30)rmse_cross_val = rmse_cv(lin_reg)

print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "Polynomial Regression (degree=2)","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}models = models.append(new_row, ignore_index=True)
```

Out[11]:

MAE: 2382228327828308.5

MSE: 1.5139911544182342e+32

RMSE: 1.230443478758059e+16

R2 Score: -1.9738289005226644e+22

RMSE Cross-Validation: 36326.451444669496

Model training:

➢ Model training is the process of teaching a machine learning model to predict house prices. It involves feeding the model historical data on house prices and features, such as square footage, number of bedrooms, and location. The model then learns the relationships between these features and house prices.

➢ Once the model is trained, it can be used to predict house prices for new data. For example, you could use the model to predict the price of a house that you are interested in buying.

1. **Prepare the data.** This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.

2. **Split the data into training and test sets.** The training set will be used to train the model, and the test set will be used to evaluate the performance of the model on unseen data.

3. **Choose a machine learning algorithm.** There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests.

4. **Tune the hyperparameters of the algorithm.** The hyperparameters of a machine learning algorithm are parameters that control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.

5.   **Train the model on the training set.** This involves feeding the training data to the model and allowing it to learn the relationships between the features and house prices.

6.   **Evaluate the model on the test set.** This involves feeding the test data to the model and measuring how well it predicts the house prices.

        If the model performs well on the test set, then you can be confident that it will generalize well to new data.

Dividing Dataset in to features and target variable:

In [12]:

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

Y = dataset['Price']
```

2.   **Split the data into training and test sets.** The training set will be used to train the model, and the test set will be used to evaluate the performance of the model.

In [13]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_st
ate=101)
```

In [14]:

```
Y_train.head()
```

Out[14]:

3413   1.305210e+06

1610   1.400961e+06

3459   1.048640e+06

4293   1.231157e+06

1039   1.391233e+06

Name: Price, dtype: float64

In [15]:

```
Y_train.shape
```

Out[15]:

(4000,)

In [16]:

```
Y_test.head()
```

Out[16]:

1718   1.251689e+06

2511   8.730483e+05

345   1.696978e+06

2521   1.063964e+06

54   9.487883e+05

Name: Price, dtype: float64

In [17]:

```
Y_test.shape
```

Out[17]: (1000)

3.   **Train the model on the training set.** This involves feeding the training data to the model and allowing it to learn the relationships between the features and the target variable.

4.   **Evaluate the model on the test set.** This involves feeding the test data to the model and measuring how well it predicts the target variable.

Model evaluation:

1.   **Calculate the evaluation metrics.** There are a number of different evaluation metrics that can be used to assess the performance of a machine learning model, such as ***R-squared, mean squared error (MSE), and root mean squared error (RMSE).***

2.   **Interpret the evaluation metrics.** The evaluation metrics will give you an idea of how well the model is performing on unseen data. If the model is performing well, then you can be confident that it will generalize well to new data. However, if the model is performing poorly, then you may need to try a different model or retune the hyperparameters of the current model.

Model evaluation:

❖ Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.

❖ There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:

- **Mean squared error (MSE):** This metric measures the average squared difference between the predicted and actual house prices.
- **Root mean squared error (RMSE):** This metric is the square root of the MSE.
- **Mean absolute error (MAE):** This metric measures the average absolute difference between the predicted and actual house prices.
- **R-squared:** This metric measures how well the model explains the variation in the actual house prices.

In addition to these metrics, it is also important to consider the following factors when evaluating a house price prediction model:

- **Bias:** Bias is the tendency of a model to consistently over- or underestimate house prices.
- **Variance:** Variance is the measure of how much the predictions of a model vary around the true house prices.
- **Interpretability:** Interpretability is the ability to understand how the model makes its predictions. This is important for house price prediction models, as it allows users to understand the factors that influence the predicted house prices.

Evaluation of Predicted Data:

In [18]:

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

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')
```
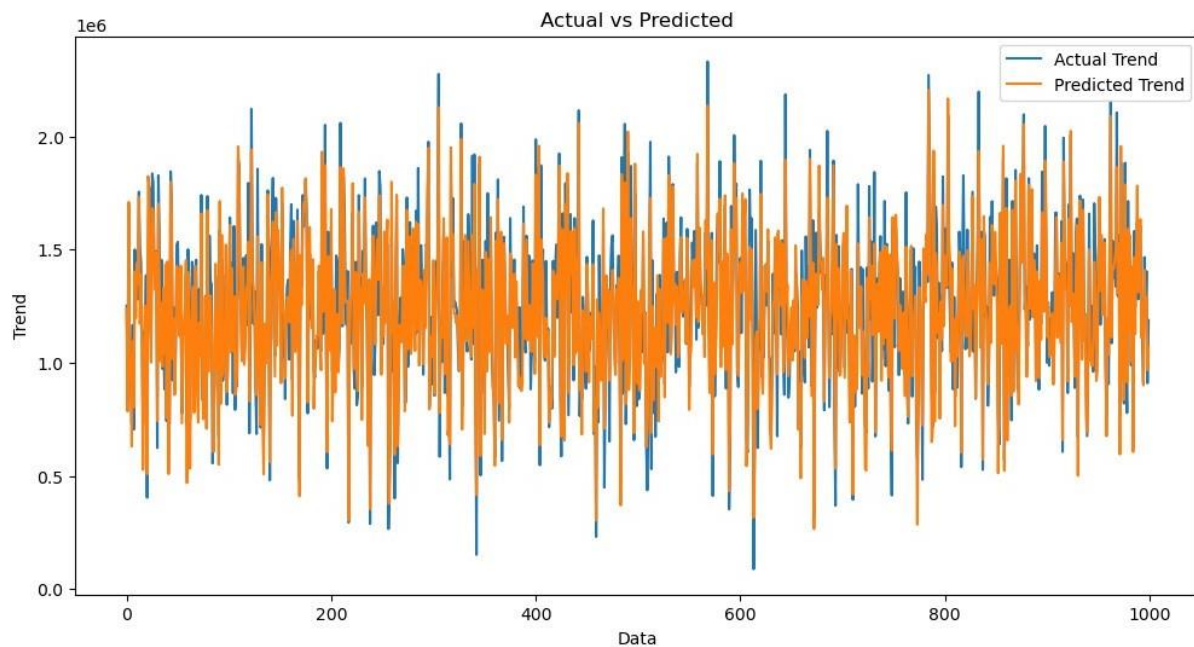
Out[18]:

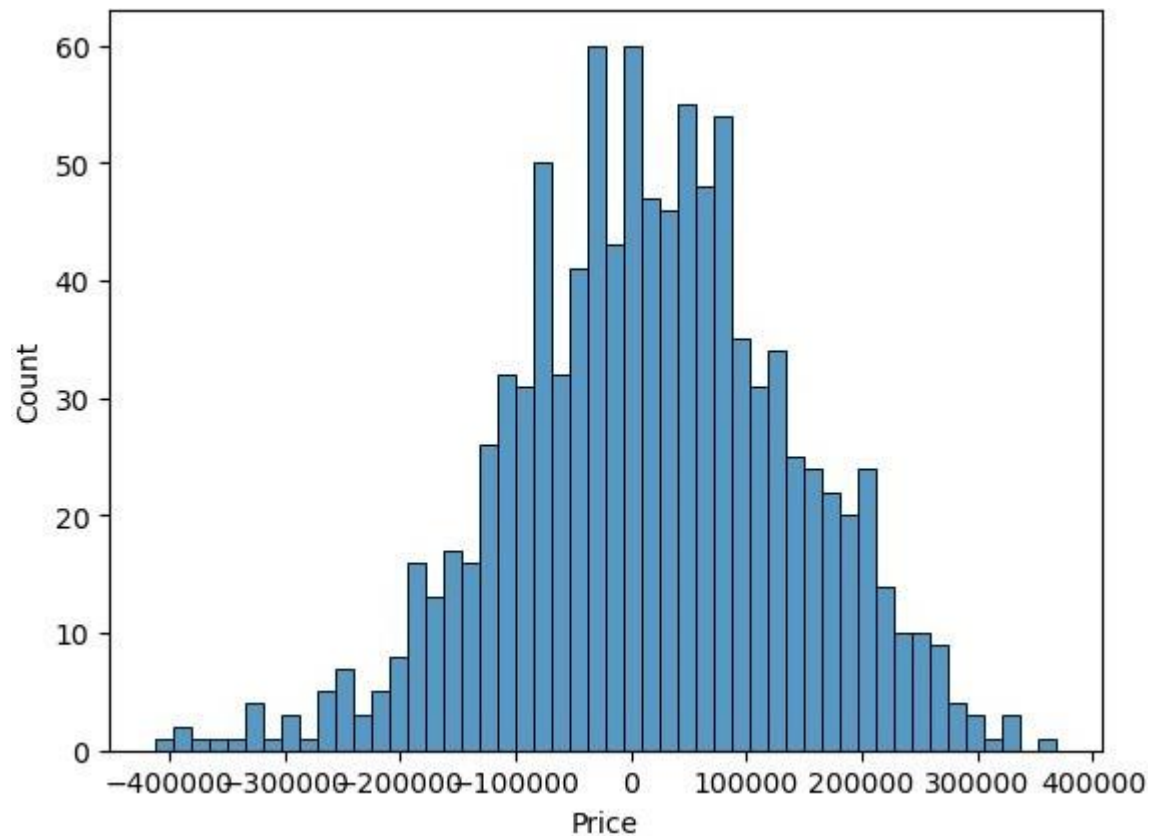Text(0.5, 1.0, 'Actual vs Predicted')

In [19]:

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

Out[19]:

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

In [20]:

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

print(mean_absolute_error(Y_test, Prediction2))

print(mean_squared_error(Y_test, Prediction2))
```

Out[20]:

-0.0006222175925689744

286137.81086908665

128209033251.4034

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 (Cross-Validation)
6	XGBRegressor	1.743992 e+04	7.165790 e+08	2.676899 e+04	9.065778 e-01	29698.84 9618
4	SVR	1.784316 e+04	1.132136 e+09	3.364723 e+04	8.524005 e-01	30745.47 5239
5	RandomForestRe gressor	1.811511 e+04	1.004422 e+09	3.169262 e+04	8.690509 e-01	31138.86 3315
1	Ridge	2.343550 e+04	1.404264 e+09	3.747351 e+04	8.169225 e-01	35887.85 2792

|	Model	MAE	MSE	RMSE	R2 Score	RMSE (Cross-Validation)
2	Lasso	2.356046e+04	1.414338e+09	3.760768e+04	8.156092e-01	35922.769369
0	LinearRegression	2.356789e+04	1.414931e+09	3.761557e+04	8.155318e-01	36326.451445
7	Polynomial Regression (degree=2)	2.382228e+15	1.513991e+32	1.230443e+16	-1.973829e+22	36326.451445
3	ElasticNet	2.379274e+04	1.718446e+09	4.145414e+04	7.759618e-01	38449.008646

In [31]:

```python
plt.figure(figsize=(12,8))

sns.barplot(x=models["Model"], y=models["RMSE (Cross-Validation)"])

plt.title("Models' RMSE Scores (Cross-Validated)", size=15)

plt.xticks(rotation=30, size=12)

plt.show()
```

Models' RMSE Scores (Cross-Validated)
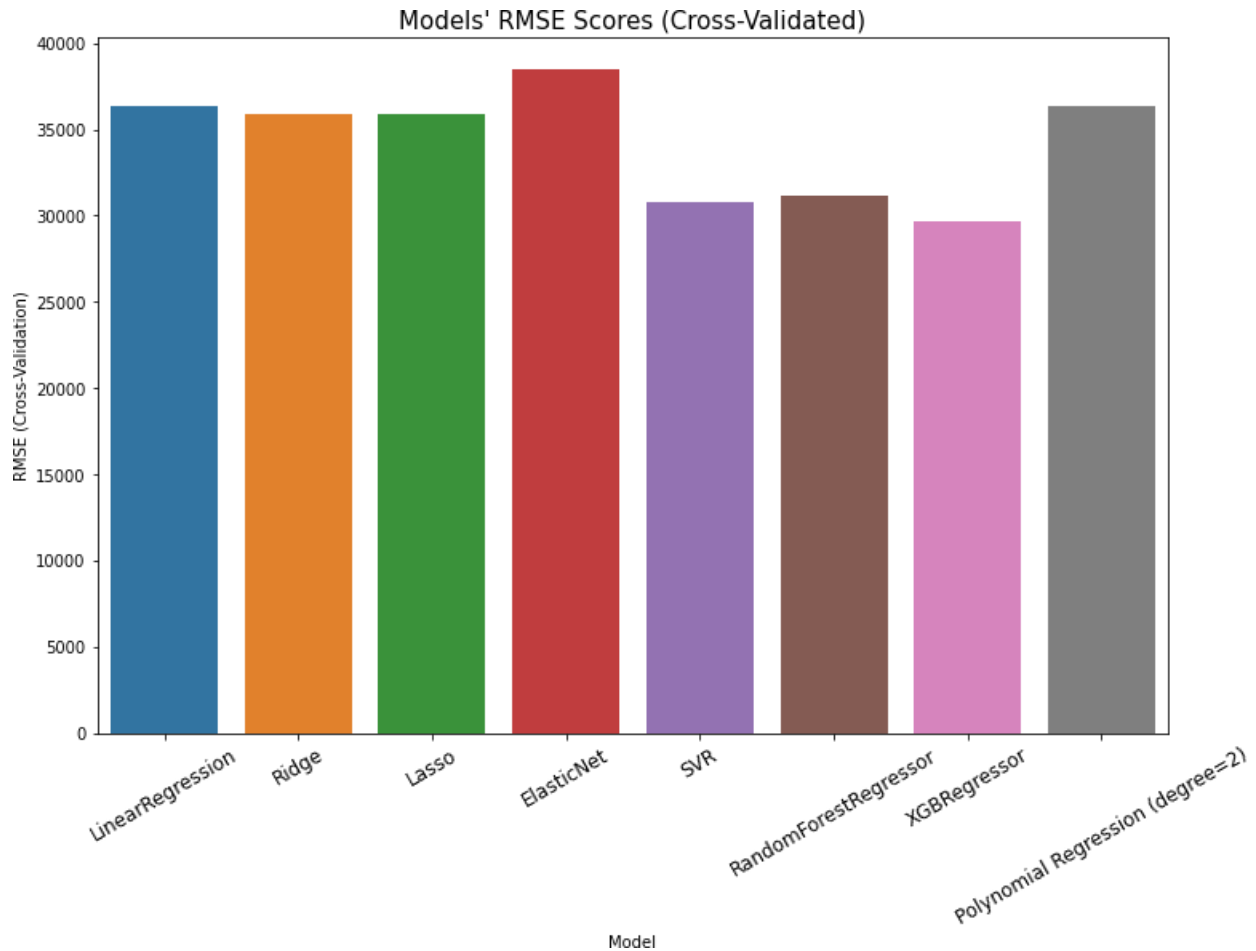
Feature Engineering:

Feature engineering is a crucial aspect of building a house price prediction model using machine learning. It involves creating new features, transforming existing ones, and selecting the most relevant variables to improve the model's predictive power. Here are some feature engineering ideas for house price prediction:

1.Total Area Features:

Combine individual room areas to create features like "Total Living Area," "Total Bedroom Area," or "Total Bathroom Area." These can be significant predictors of house price.

2.Ratio Features:

Create features that represent ratios, such as the "Bedroom to Bathroom Ratio" or "Living Area to Lot Area Ratio." These ratios may capture the property's layout and functionality.

3.Age of the Property:

Calculate the age of the property by subtracting the construction year from the current year. Newer properties might have higher values.

4.Neighborhood Statistics:

Aggregate neighborhood-level statistics, such as the average income, crime rate, school ratings, or proximity to amenities, and use these as features.

5.Distance to Key Locations:

Calculate distances from the property to essential places like schools, parks, shopping centers, or public transportation hubs. Closer proximity to such amenities can affect the price.

6.Categorical Encodings:

Use techniques like one-hot encoding, label encoding, or target encoding for categorical variables, such as property type, heating system, or garage type.

7.Seasonal Features:

Create features indicating the season during which the house was sold. Seasonality can influence property demand and prices.

8.Historical Data:

Incorporate historical data on house prices and local real estate market trends. This can help the model account for cyclical patterns.

9. Exterior Features:

Develop features related to the property's exterior, such as the presence of a swimming pool, patio, or garden. These features can be valuable for determining a property's appeal.

10. Quality Scores:

Create a combined quality score by aggregating the quality ratings of various components of the property, such as kitchen quality, bathroom quality, and overall house quality.

11. Logarithmic Transformations:

Apply logarithmic transformations to features like "Lot Area" or "Number of Bedrooms" to make their distributions more normal.

12. Interaction Features:

Create interaction terms by multiplying or dividing relevant features. For example, "Number of Bathrooms" multiplied by "Total Living Area" can represent the total bathroom area.

13. Missing Value Indicators:

Create binary indicators for missing values in the dataset. The presence of missing data can be an informative feature.

14. Density Features:

Compute population density in the neighborhood or the density of certain property types. High density might impact property prices.

15. Sentiment Analysis:

Analyze online reviews or social media sentiment related to the property or neighborhood to capture public perception.

16. Time-Related Features:

Incorporate time-related features like day of the week, month, or year when the property was listed or sold.

17. Zoning Information:

Include zoning information that can affect property use, such as residential, commercial, or mixed-use zoning.

18. Accessibility Features:

Create features to represent accessibility, like the number of nearby public transport stations or major highways.

19. Energy Efficiency:

Include features related to energy-efficient components, such as insulation, energy-efficient appliances, or solar panels.

20. Demographic Data:

Use demographic data for the area to understand the potential buyer's income levels, family sizes, and preferences.

Various feature to perform model training:

- **Use a variety of feature engineering techniques.**

Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models. By using a variety of feature engineering techniques, you can create a set of features that will help your model to predict house prices more accurately.

- **Use cross-validation.**

Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use cross-validation to evaluate the performance of your model during the training process. This will help you to avoid overfitting and to ensure that your model will generalize well to new data.

- **Use ensemble methods.**

Ensemble methods are machine learning methods that combine the predictions of multiple models to produce a more accurate prediction. Ensemble methods can often achieve better performance than individual machine learning models.

- **Use cross-validation.**

  Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use cross-validation to evaluate the performance of your model during the evaluation process. This will help you to avoid overfitting and to ensure that the model will generalize well to new data.

- **Use a holdout test set.**

  A holdout test set is a set of data that is not used to train or evaluate the model during the training process. This data is used to evaluate the performance of the model on unseen data after the training process is complete.

- **Compare the model to a baseline.**

  A baseline is a simple model that is used to compare the performance of your model to. For example, you could use the mean house price as a baseline.

- **Analyze the model's predictions.**

  Once you have evaluated the performance of the model, you can analyze the model's predictions to identify any patterns or biases. This will help you to understand the strengths and weaknesses of the model and to improve it.

<u>Conclusion:</u>

   In the quest to build an accurate and reliable house price prediction model, we have embarked on a journey that encompasses critical phases, from feature selection to model training and evaluation. Each of these stages plays an indispensable role in crafting a model that can provide meaningful insights and estimates for one of the most significant financial decisions individuals and businesses make—real estate transactions.

❖ Model training is where the model's predictive power is forged. We have explored a variety of regression techniques, fine-tuning their parameters to learn from historical data patterns. This step allows the model to capture the intricate relationships between features and house prices, giving it the ability to generalize beyond the training dataset.

❖ Finally, model evaluation is the litmus test for our predictive prowess. Using metrics like *Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, and R-squared*, we've quantified the model's performance. This phase provides us with the confidence to trust the model's predictions and assess its ability to adapt to unseen data.

❖ In the ever-evolving world of real estate and finance, a robust house price prediction model is an invaluable tool. It aids *buyers, sellers, and investors* in making informed decisions, mitigating risks, and seizing opportunities. As more data becomes available and market dynamics change, the model can be retrained and refined to maintain its accuracy.