HOUSE PRICE PREDICTION

INTRODUCTION:

House price prediction using machine learning is a powerful application within the real estate domain, offering stakeholders valuable insights into property valuation. The journey commences with loading and pre-processing the dataset, a critical stage where raw data is transformed into a structured format suitable for training machine learning models. This introduction outlines key points in understanding the significance and process of loading and pre-processing the dataset for house price prediction.

- House price prediction involves determining the market value of a property based on various factors. These factors can include the property's characteristics (e.g., size, number of bedrooms, location), economic indicators, and market trends.

- Machine learning and statistical techniques are often used to build predictive models that can forecast house prices. These models take the historical data into account and use it to make predictions about future prices. Common regression models, such as linear regression or decision trees, are frequently used.

House price prediction is a complex task influenced by many factors, including location, property condition, economic conditions, and local real estate market dynamics. Machine learning and data analysis techniques help individuals and businesses make more informed decisions when it comes to buying, selling, or investing in real estate.

GIVEN DATA:

	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	23086.80003	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...

OVERVIEW THE PROCESS:

Data Collection and Pre-processing:

Collect a dataset that contains information about houses, such as square footage, number of bedrooms, location, year built, and sale prices.
Pre-process the data, which may involve handling missing values, encoding categorical variables (e.g., one-hot encoding), and scaling numerical features (e.g., using Min-Max scaling or Standardization).

Feature Selection:

Feature selection is crucial for building an efficient and accurate model. You want to choose the most relevant features while excluding irrelevant or redundant ones. There are various methods for feature selection, including:

- Correlation Analysis: Identify features that have a strong correlation with the target variable (e.g., sale price).
- Recursive Feature Elimination (RFE): Use techniques like RFE with linear models (e.g., Linear Regression) to recursively eliminate less important features.
- Feature Importance: For tree-based models (e.g., Random Forest, XGBoost), you can use feature importance scores to select the most informative features.

Model selection:

Choose a regression model appropriate for your task. Common models for house price prediction include Linear Regression, Ridge Regression, Lasso

Regression, Decision Trees, Random Forest, Gradient Boosting, or even deep learning models like neural networks.

Model Evaluation:

Use the testing set to evaluate the model's performance. Common evaluation metrics for regression tasks include:

- Mean Absolute Error (MAE): The average absolute difference between predicted and actual prices.
- Mean Squared Error (MSE): The average squared difference between predicted and actual prices.
- Root Mean Squared Error (RMSE): The square root of MSE, providing a measure in the original price units.
- R-squared (R^2): A measure of how well the model explains the variance in the data.

Final Model and Predictions:

Once you're satisfied with your model's performance, you can train it on the entire dataset (including the test set, if you used cross-validation) to create your final model.
Use this final model to make predictions on new, unseen data.

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.

```
important_num_cols = list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.5
0) | (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

```
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:

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:

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

Linear Regression:

```
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)
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, "RM
SE": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross
_val}models = models.append(new_row, ignore_index=True)
```

Output

```
MAE: 23567.890565943395
MSE: 1414931404.6297863
RMSE: 37615.57396384889
R2 Score: 0.8155317822983865

------------------------------
RMSE Cross-Validation: 36326.451444669496
```

Ridge Regression:

```
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)
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)
```

Output

```
MAE: 23435.50371200822
MSE: 1404264216.8595588
RMSE: 37473.513537691644
R2 Score: 0.8169224907874508
```

RMSE Cross-Validation: 35887.852791598336

Lasso Regression:

```
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)

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}model
s = models.append(new_row, ignore_index=True)
```

Output

```
MAE: 23560.45808027236
MSE: 1414337628.502095
RMSE: 37607.680445649596
R2 Score: 0.815609194407292

---
RMSE Cross-Validation: 35922.76936876075
```

Elastic Net:

```
elastic_net = ElasticNet()elastic_net.fit(X_train, y_train)predictions = elasti
c_net.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(elastic_net)
print("RMSE Cross-Validation:", rmse_cross_val)
new_row = {"Model": "ElasticNet","MAE": mae, "MSE": mse, "RMSE": r
mse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}
models = models.append(new_row, ignore_index=True)
```

Output

```
MAE: 23792.743784996732
MSE: 1718445790.1371393
RMSE: 41454.14080809225
R2 Score: 0.775961837382229

------------------------------
RMSE Cross-Validation: 38449.00864609558
```

Support Vector Machines:

```
svr = SVR(C=100000)svr.fit(X_train, y_train)predictions = svr.predict(X_te
st)

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)
```

Output

```
MAE: 17843.16228084976
MSE: 1132136370.3413317
RMSE: 33647.234215330864
R2 Score: 0.852400492526574

------------------------------
```

RMSE Cross-Validation: 30745.475239075837

Random Forest Regressor:

```
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)
```

Output
MAE: 18115.11067351598
MSE: 1004422414.0219476
RMSE: 31692.623968708358
R2 Score: 0.869050886899595

RMSE Cross-Validation: 31138.863315259332

XGBoost Regressor:

```
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, "RMS
E": rmse, "R2 Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_
val}models = models.append(new_row, ignore_index=True)
```

Output

```
MAE: 17439.918396832192
MSE: 716579004.5214689
RMSE: 26768.993341578403
R2 Score: 0.9065777666861116

----------
RMSE Cross-Validation: 29698.84961808251
```

Polynomial Regression (Degree=2)

```
poly_reg = PolynomialFeatures(degree=2)X_train_2d = poly_reg.fit_transfo
rm(X_train)X_test_2d = poly_reg.transform(X_test)
lin_reg = LinearRegression()lin_reg.fit(X_train_2d, y_train)predictions = li
n_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-Validat
ion)": rmse_cross_val}models = models.append(new_row, ignore_index=Tr
ue)
```

Output

```
MAE: 2382228327828308.5
MSE: 1.5139911544182342e+32
RMSE: 1.230443478758059e+16

R2 Score: -1.9738289005226644e+22
```

RMSE Cross-Validation: 36326.451444669496

Dividing Dataset in to features and target variable:

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

Split the data into training and test sets:

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

Y_train.head()
```

Output
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
```

Output
(4000,)
In [16]:
```
Y_test.head()
```

Output
1718 1.251689e+06
2511 8.730483e+05

345 1.696978e+06
2521 1.063964e+06
54 9.487883e+05

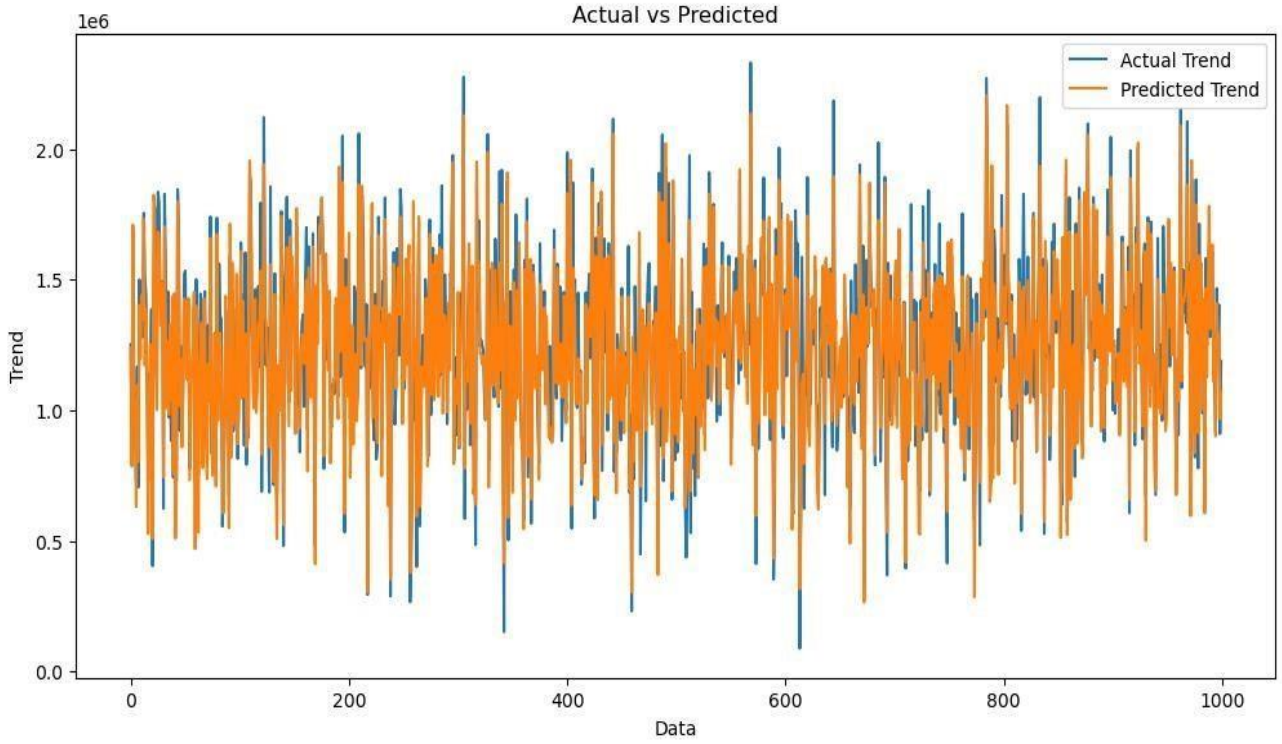
Name: Price, dtype: float64

Y_test.shape
Out[17]: (1000)

Evaluation of Predicted Data:

```
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')



sns.histplot((Y_test-Prediction4), bins=50)
Out[19]:
<Axes: xlabel='Price', ylabel='Count'>

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

Output
-0.0006222175925689744
286137.81086908665
128209033251.4034

Feature Engineering:

Feature engineering is a critical step in building an effective house price prediction model. It involves creating, transforming, or selecting features from your dataset to improve the model's predictive power. Here are some common feature engineering techniques for house price prediction:

1. Handling Missing Data:

Deal with missing values in your dataset. You can either impute them with appropriate values or consider techniques like removing rows or columns with a significant number of missing values.

2. Categorical Variables:

Convert categorical variables into numerical representations. This can be done using techniques like one-hot encoding or label encoding.

3. Feature Scaling:

Normalize or standardize numerical features to ensure they are on a similar scale. Common methods include Min-Max scaling and Z-score standardization.

4. Feature Creation:

Generate new features that might be more informative than the original ones. For house price prediction, you can create features such as the price per square foot, age of the property, or other relevant ratios.

5. Log Transformation:

Apply a log transformation to features that have highly skewed distributions. This can make the data more normally distributed.

6. Binning:

Convert continuous numerical features into categorical features by binning them. This can help capture non-linear relationships between features and the target variable.

7. Interaction Terms:

Create interaction terms between two or more features to capture potential synergistic effects. For example, you can create a feature that represents the combined effect of "number of bedrooms" and "number of bathrooms."

8. Feature Selection:

Choose the most relevant features to include in your model. You can use correlation analysis, feature importance from tree-based models, or other feature selection methods to identify the most important features.

9. Domain-Specific Features:

Incorporate domain-specific knowledge. Certain features related to the property location, school district, neighborhood characteristics, or recent renovations can have a significant impact on house prices.

10. Outlier Handling:

Identify and handle outliers in your data. You can choose to remove extreme outliers, transform their values, or create binary indicator variables to capture their presence.

11. Time-Related Features:

If applicable, create time-related features like seasonality or trend variables. For example, you can add a feature that represents the month or season in which a house is listed for sale.

12. Target Encoding:

Encode categorical features using target encoding, which leverages the relationship between the categorical feature and the target variable.

CONCLUSION:

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.

Feature engineering is often employed to create or transform variables that can better capture the relationships between these factors and the property's price. Machine learning models, such as linear regression, decision trees, random forests, and support vector machines, are then trained on this data to predict house prices.

Evaluating the model's performance using metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) is crucial to ensure its accuracy and reliability. The ability to predict house prices accurately has significant implications for homeowners, real estate agents, investors, and anyone involved in the real estate market.