```
Zillow Predictive Power <br>
<h3 style="text-align: center; color: white; font-size: 15px; font-
family: 'Arial', sans-serif;">  House Price Prediction Model</h3>
```

</div

The goal of this project is to build and evaluate a regression model to predict house prices using the "Ames Housing" dataset from Kaggle.

Students will apply supervised learning techniques to develop and refine their models, analyzing the results to improve accuracy.

1. Model Code
2. Evaluation Metric
3. Jupyter Notebook - Well documented ipynb that includes:
   - Data exploration and preprocessing steps
   - Model training and evaluation results
   - Conclusions and future improvement suggestions

1. Familiarize yourself with the dataset.
2. Load and explore the data.
3. Clean the data.
4. Handle missing values.
5. Convert categorical variables to numerical.
6. Select relevant features.
7. Split the data into training and testing sets.
8. Choose a model.
9. Train your model.
10. Evaluate your model's performance.
11. Optimize model parameters.
12. Document your process in a Jupyter Notebook.
13. Create visualizations to support your findings.
14. Summarize key insights and conclusions.
15. Review your notebook for clarity and completeness.
16. Prepare your final submission materials.

```
# Import necessary libraries
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical operations
import sklearn
from sklearn.model_selection import train_test_split  # To split the
dataset
from sklearn.linear_model import LinearRegression  # A simple
regression model
from sklearn.metrics import root_mean_squared_error,r2_score  # For
model evaluation
```

```
import matplotlib.pyplot as plt  # For data visualization
import seaborn as sns  # For enhanced visualization

# Load the dataset
# Replace 'path_to_your_dataset.csv' with the actual path to your
dataset
# For example, you might have a file named 'ames_housing.csv' in your
directory
data = pd.read_csv('your_dataset.csv')

# Explore the dataset
print("Dataset Shape:", data.shape)  # Print the dimensions of the
dataset
print(data.head())  # Print the first few rows of the dataset for a
quick overview

# Data Cleaning
# Check for missing values
print("Missing Values:", data.isnull().sum())  # Print the count of
missing values for each column

# Drop rows with missing target values (house prices)
data.dropna(subset=['SalePrice'], inplace=True)

# Fill missing values for other columns with the mean (you can choose
different strategies)
data.fillna(data.mean(), inplace=True)

# Feature Selection
# Selecting features that are likely to affect house prices
# Here we will assume 'GrLivArea' (Ground Living Area) and
'GarageCars' (Number of Cars Garage Holds) are our features
X = data[['GrLivArea', 'GarageCars']]  # Features
y = data['SalePrice']  # Target variable (house prices)

# Split the dataset into training and testing sets
# test_size=0.2 means 20% of the data will be used for testing
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Model Training
# Initialize the Linear Regression model
model = LinearRegression()
# Fit the model on the training data
model.fit(X_train, y_train)

# Model Evaluation
# Make predictions on the test data
y_pred = model.predict(X_test)
```

```
# Calculate evaluation metrics
mse = root_mean_squared_error(y_test, y_pred)  # Mean Squared Error
r2 = r2_score(y_test, y_pred)  # R-squared score

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")

# Visualize the results
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred)  # Scatter plot of actual vs
predicted prices
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted House Prices')
plt.plot([y.min(), y.max()], [y.min(), y.max()], '--', color='red')  #
Diagonal line for perfect predictions
plt.show()

# End of program
```