Predicting House Prices with Multiple Regression

Introduction

In this project, I analyzed the dataset of house prices with features like size, number of bedrooms, age, proximity to downtown, and their respective prices. The goal was to build a linear regression model that predicts house prices based on these features. Below, I will explain the steps I took for data preprocessing, model development, evaluation, and the conclusions drawn from the results.

1. Data Preprocessing

Dataset Overview

The dataset contains the following columns:

- **Size (sqft)**: The size of the house in square feet.
- **Bedrooms**: The number of bedrooms in the house.
- **Age**: The age of the house in years.
- **Proximity to Downtown (miles)**: The distance of the house from the city center.
- **Price**: The selling price of the house.

```
df.info()
```

The dataset contains no missing values, which made preprocessing easier. There are 1,000 rows and 5 columns, with no null values.

Data Description

To understand the range and distribution of each feature, I looked at basic statistical summaries:

```
df.describe()
```

- **Size (sqft)** ranges from 801 to 3,997, with an average of 2,429 sqft.
- **Bedrooms** ranges from 1 to 5, with an average of 3 bedrooms.
- **Age** of the houses ranges from 0 to 99 years.
- **Proximity to Downtown (miles)** ranges from 0.5 to 29.93 miles.
- **Price** ranges from $215,945 to $1,212,350, with an average price of $719,053.

2. Data Visualization

To explore the relationships between features and house prices, I created visualizations using Seaborn and Matplotlib:

1. **Size vs. Price**: A line plot showed that larger houses generally have higher prices.
2. **Bedrooms vs. Price**: A bar plot indicated that houses with more bedrooms tend to have higher prices, though the relationship is weaker.
3. **Age vs. Price**: A histogram showed that older houses are generally less expensive.
4. **Proximity to Downtown vs. Price**: A line plot demonstrated that houses farther from downtown are usually cheaper.
5. **Price Distribution**: A KDE plot revealed that the prices are mostly clustered around $500,000 to $1,000,000.

3. Model Development

Feature Scaling

Since linear regression can be sensitive to the scales of the features, I standardized the features using StandardScaler:

```
scaler = StandardScaler()

X = scaler.fit_transform(df[['Size (sqft)', 'Bedrooms', 'Age', 'Proximity to Downtown (miles)']])

y = df['Price']
```

This ensured that all features were on a similar scale, with a mean of 0 and a standard deviation of 1.

Train-Test Split

I split the data into a training set (70%) and a test set (30%) to evaluate the model's performance on unseen data:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Model Training

I used Scikit-Learn's LinearRegression model to predict house prices:

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

After training, I examined the coefficients of each feature to understand their impact on house prices:

```
coefficients = pd.DataFrame(model.coef_, ['Size (sq. ft.)', 'Bedrooms', 'Age', 'Proximity to Downtown (miles)'], columns=['Coefficient'])

print(coefficients)
```

The coefficients revealed the following:

- **Size (sq. ft.)**: Strongly positive, indicating that larger houses lead to higher prices.
- **Bedrooms**: Weakly positive, indicating that the number of bedrooms has a small impact on price.
- **Age**: Negative, meaning older houses tend to be less expensive.
- **Proximity to Downtown (miles)**: Negative, indicating that houses farther from downtown are cheaper.

4. Model Evaluation

Performance Metrics

To assess the model's performance, I used two key metrics:

- **Mean Squared Error (MSE)**: Measures the average squared difference between actual and predicted prices. A lower value indicates better performance.
- **R-squared (R^2)**: Indicates how much of the variance in the target variable (price) is explained by the model. An R^2 value closer to 1 is ideal.

```
mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)
```

The results were:

- **Mean Squared Error**: 100,214,724.63, indicating the average prediction error in price.
- **R-squared**: 1.00, indicating the model explained nearly all the variance in house prices.

Visualization of Results

I plotted the actual vs predicted house prices to visually assess the model's performance:

```
plt.scatter(y_test, y_pred, marker='o', s=50, alpha=0.7, color='purple')

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='yellow', linewidth=2, label='Perfect Prediction')

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')
```

The scatter plot shows that the predicted prices align well with the actual prices, indicating the model performed well.

5. Conclusion

Challenges

The biggest challenge was ensuring that the features were correctly scaled for the linear regression model. I used StandardScaler to standardize the features, which helped the model perform better.

Model Applicability

The linear regression model is simple and interpretable, making it suitable for real-world scenarios where a straightforward model is needed. However, it assumes a linear relationship between the features and the target, which might not always be the case in more complex housing markets. The high R^2 score suggests that the model fits this dataset well, but the performance could vary with different data.

Potential Limitations

- **Limited Features**: The model only uses four features, but other factors (like neighborhood, number of bathrooms, etc.) could improve the model's predictions.
- **Overfitting**: The very high R^2 score might indicate overfitting, especially if the model is tested on different data.
- **Linearity Assumption**: The model assumes a linear relationship between features and price, which may not always hold in reality.

In conclusion, while the model performed well on this dataset, expanding the feature set and testing on more varied data could lead to better, more generalizable predictions.