CSST 102: Machine Problem No. 2

Data Exploration and Visualization

To begin the predictive modeling process for house prices using multiple regression, exploratory data analysis (EDA) is essential. This helps in understanding the relationships between the features and the target variable (house prices).

Data Overview

The dataset contains the following columns:

- **Size (sq. ft.)**: 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 from the house to the downtown area.
- **Price**: The actual price of the house (in thousands of dollars).

1. **Loading the Data**: Use `pandas` to load the dataset and inspect its structure.
2. **Visualizations**:
   - **Line Plot**: Size vs. Price
   - **Bar Plot**: Bedrooms vs. Price
   - **Histogram**: Age vs. Price
   - **Line Plot**: Proximity to Downtown vs. Price
   - **Density Plot**: Distribution of House Prices
   - **Correlation Matrix**: To identify relationships among features
3. **Correlation Analysis**

```
# Load the dataset
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('datasets_house_prices.csv')
print(df.info())
print(df.describe())
print(df.isnull().sum())
```

```
sns.set(style="whitegrid")

# Plot Size vs Price
plt.figure(figsize=(12, 8))
```

```
plt.subplot(2, 2, 1)
sns.scatterplot(data=df, x='Size (sq. ft.)', y='Price', marker='o', color='#5
plt.title('Size vs Price')

# Plot Bedrooms vs Price
plt.subplot(2, 2, 2)
sns.barplot(data=df, x='Bedrooms', y='Price', ci=None, color='#3E5058')
plt.title('Bedrooms vs Price')

# Plot Age vs Price
plt.subplot(2, 2, 3)
sns.histplot(data=df, x='Age', y='Price', bins=30, pthresh=.05, cmap="mako")
plt.title('Age vs Price')

# Plot Proximity to Downtown vs Price
plt.subplot(2, 2, 4)
sns.scatterplot(data=df, x='Proximity to Downtown (miles)', y='Price', marker
plt.title('Proximity to Downtown vs Price')

plt.tight_layout()
plt.show()

# Distribution of House Prices
plt.figure(figsize=(10, 5))
sns.kdeplot(df['Price'], shade=True, color='#55535B')
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Density')
plt.show()
```

```
# Correlation Matrix
plt.figure(figsize=(10, 6))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='viridis')
plt.title('Correlation Matrix')
plt.show()
```

Data Preprocessing

Handling Missing Data

Missing values can skew results, so it is crucial to handle them appropriately. In this case, filling missing values with the mean of each column is a common approach:

```
df.fillna(df.mean(), inplace=True)
```

Standardize the Data

Standardizing the features ensures that they are on a similar scale, which is particularly important for regression models:

```
#remove spaces
df.columns = df.columns.str.replace(' ', '')

# Define features and target
X = df[['Size(sqft)', 'Bedrooms', 'Age', 'ProximitytoDowntown(miles)']]
y = df['Price']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Model Development

Building the Regression Model

1. **Split Data and Train Model:** Split the data into training and test sets.
2. **Training the Model**: Implement a linear regression model.
3. **Coefficients Analysis**: Extract and display the model coefficients to understand the influence of each feature

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.
```

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

```
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
```

Model Evaluation

1. **Performance Metrics:** Calculate performance metrics and visualize the results.
2. **Visualization of Predictions:** To visualize the model's accuracy, plot the predicted prices against the actual prices:

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2:.2f}')
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, marker='o', s=50, alpha=0.7, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black
plt.xlabel('Actual Prices', fontsize=12)
plt.ylabel('Predicted Prices', fontsize=12)
plt.title('Comparison of Actual Prices vs. Predicted Prices', fontsize=14)
plt.grid(True)
plt.legend()
plt.show()
```

Conclusion

The analysis presented illustrates the utilization of multiple regression techniques to forecast house prices based on diverse features. The performance of the model can be assessed through MSE and R-squared metrics, providing insights into the model's data-fitting capabilities.

Challenges and Limitations

The challenges may involve handling outliers, addressing multicollinearity, and ensuring that the model's assumptions are met. Potential future improvements could include feature engineering, the exploration of non-linear models, or the utilization of more advanced machine-learning techniques. This predictive model has the potential to assist real estate agents in estimating house prices more accurately. However, it is crucial to consider external factors that may influence the housing market beyond the dataset used.