Synopsis Report

Here is the synopsis report for the laboratory activity, including the findings and the sections of model implementation while doing it. This lab activity aimed to build a multiple regression model to predict house costs. We used features like house size, number of bedrooms, house age, and distance from downtown. The dataset included these features and the goal was to create and evaluate a model to estimate house prices based on this information.

1. Data Exploration and Analysis

EDA

First, I loaded the dataset and checked the summary statistics to get an overview of the data. Scatter plots were used to see how each feature related to house prices. I also looked at the correlation matrix to understand how strongly features are related to the price.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
house_cost = pd.read_csv('datasets_house_prices.csv')

print(house_cost.describe())
```

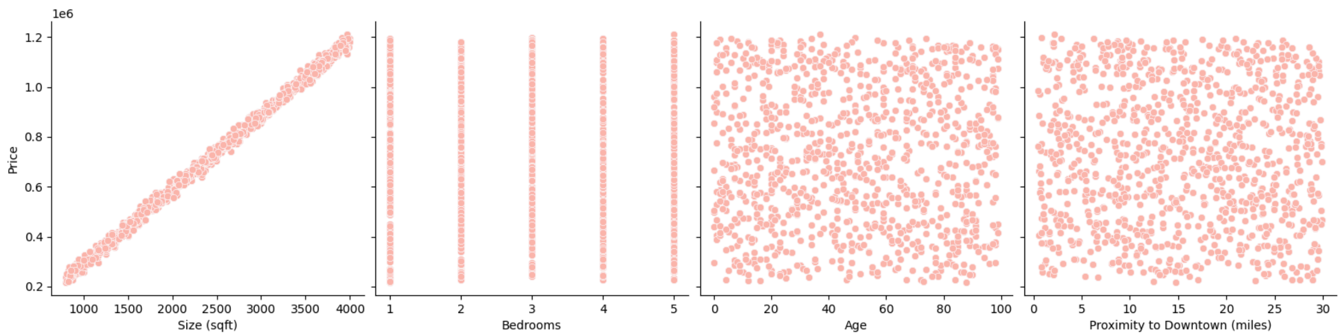
```
        Size (sqft)     Bedrooms          Age  Proximity to Downtown (miles)  \
count  1000.000000  1000.000000  1000.000000                    1000.000000
mean   2429.857000     2.993000    48.335000                      15.289063
std     929.914229     1.424423    29.203384                       8.546139
min     801.000000     1.000000     0.000000                       0.500343
25%    1629.500000     2.000000    22.000000                       8.475528
50%    2430.500000     3.000000    47.000000                      15.239628
75%    3254.250000     4.000000    74.000000                      22.765188
max    3997.000000     5.000000    99.000000                      29.935715

              Price
count  1.000000e+03
```

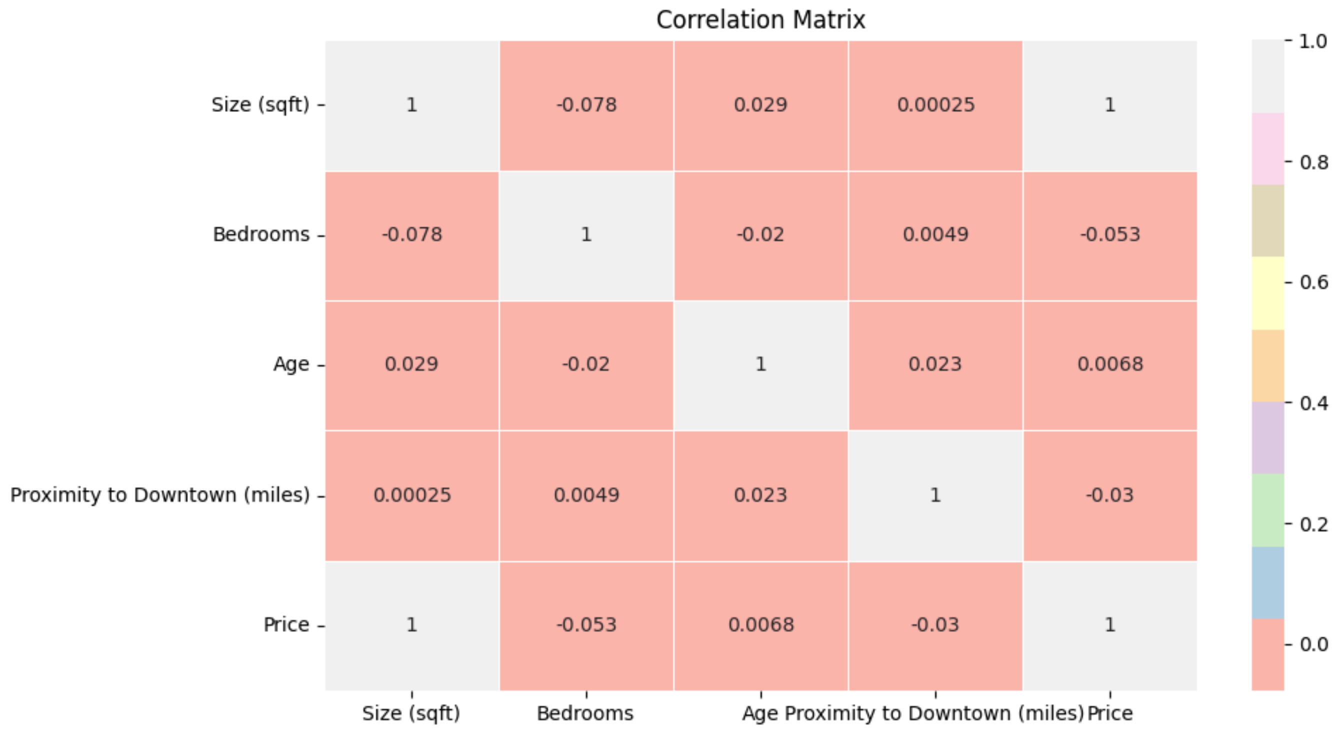
```
mean    7.190532e+05
std     2.789818e+05
min     2.159455e+05
25%     4.789045e+05
50%     7.128781e+05
75%     9.680664e+05
max     1.212350e+06
```

Visualization

```
plot_kws = {'color': sns.color_palette('Pastel1', 1)[0]}
sns.pairplot(house_cost, x_vars=['Size (sqft)', 'Bedrooms', 'Age', 'Proximity to Downtown (m
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.heatmap(house_cost.corr(), annot=True, cmap='Pastel1', linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```



2. Data Preprocessing

I checked for missing values and filled any gaps with the median value. To handle the differences in feature scales (like size in sqft and proximity in miles), I standardized the data so all features were on the same scale.

Handling missing data

```
house_cost.fillna(house_cost.median(), inplace = True)
print(house_cost.isnull().sum())
```

```
Size (sqft)                      0
Bedrooms                         0
Age                              0
Proximity to Downtown (miles)    0
Price                            0
dtype: int64
```

Normalization

```
X = house_cost[['Size (sqft)', 'Bedrooms', 'Age', 'Proximity to Downtown (miles)']]
y = house_cost['Price']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled[:5])
```

```
[[ 1.66135285 -1.39986345  1.66725032 -1.5519263 ]
 [-0.82829383  1.40969691  1.35891265  0.98411104]
 [-0.36135059  0.70730682  0.02278273 -1.03593679]
 [-0.53779919 -0.69747336 -0.69667185 -0.83819507]
 [-0.5754559  -1.39986345  0.26260092 -1.1625361 ]]
```

3. Model Development

*I built a multiple regression model using Scikit-learn's **LinearRegression.** The data was split into training and testing sets (70% training, 30% testing). The model was trained and I examined the coeficients and intercept to understand the influence of each feature on house prices. I also looked at feature importance based on these coefficients.*

Split dataset

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state
```

Multiple regression model

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

print("Model Coefficients", model.coef_)
print("Model Intercept", model.intercept_)
```

```
Model Coefficients [278975.28593843   6804.51900082  -6082.93925798  -8459.85395639]
Model Intercept 718607.7680535176
```

Feature selection

```
feature_importance = pd.Series(model.coef_, index=['Size(sqft)', 'Bedrooms', 'Age', 'Proximi
print(feature_importance.sort_values(ascending = False))
```

```
Size(sqft)                      278975.285938
Bedrooms                          6804.519001
Age                              -6082.939258
Proximity to Downtown (miles)    -8459.853956
dtype: float64
```

4. Model Evaluation

The model was evaluated with Mean Squared Error (MSE), R-squared and Adjusted R-squared. MSE showed how well the model predicted house prices on the test set. R-squared and Adjusted R-squared indicated how well the model explained the variance in house prices. A scatter plot of actual vs. predicted prices showed that the model's predictions were quite close to their actual values.

```
y_test_pred = model.predict(X_test)
```

Model's performance using metrics

```
mse = mean_squared_error (y_test, y_test_pred)
r2 = r2_score(y_test, y_test_pred)
adjusted_r2 = 1 -(1-r2)* (len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)

print(f"Test MSE: {mse}")
print(f"Test R-squared: {r2}")
print(f"Adjusted R-squared: {adjusted_r2}")
```

```
Test MSE: 100214724.63128743
Test R-squared: 0.9986314443568995
Adjusted R-squared: 0.9986128876702134
```

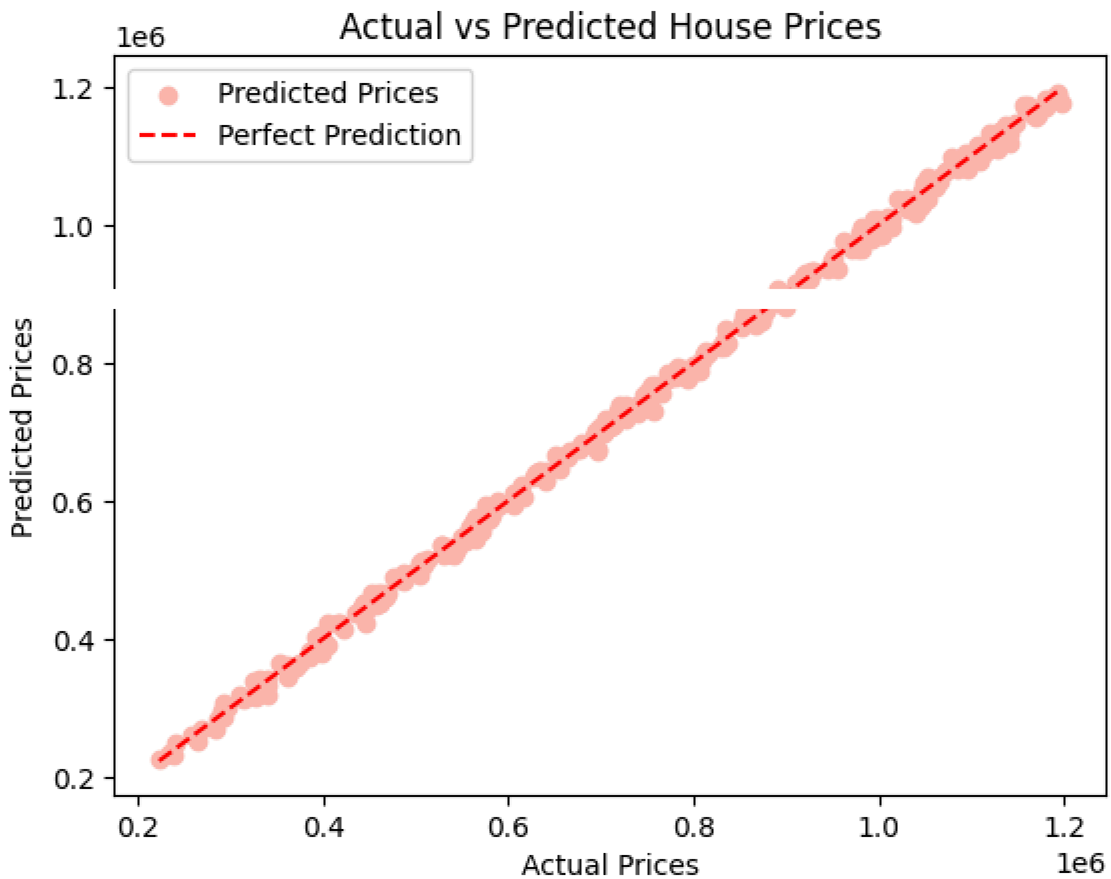
Plotting the predicted prices

```
colors = sns.color_palette('Pastel1')
plt.scatter(y_test, y_test_pred, color=colors[0], label="Predicted Prices")

plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--'

plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
```

```
plt.legend()
plt.show()
```



Conclusion:

As you can see, the multiple regression model performed well, predicting house prices accurately. The standardization of features was key to this success. In the future works, I could explore improving the model by adding polynomial features or experimenting with different types of regression tenchniques to better capture non-linear relationships in data.

References (libraries used):

1. Pandas
2. Scikit-learn
3. Matplotlib
4. Seaborn