Introduction to Linear Regression

Module 1, Lab 1: Understanding Linear Regression

Linear regression is one of the most fundamental algorithms in machine learning. In this lab, you will learn how to implement and understand linear regression from the ground up.

Learning Objectives

- Understand the mathematical foundation of linear regression
- Implement linear regression using scikit-learn
- Evaluate model performance using appropriate metrics
- Visualize the relationship between features and target variables

Business Problem

We will predict house prices based on various features like size, location, and number of bedrooms. This is a classic regression problem that helps understand how different factors influence property values.

Setup

Installing and Importing Libraries

First, we need to install the necessary libraries for our analysis.

```
# Install required packages
!pip install --upgrade pip
!pip install pandas numpy matplotlib seaborn scikit-learn
```

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

```
# Set plotting style
plt.style.use('seaborn-v0_8')
%matplotlib inline
```

Loading and Exploring the Data

We'll use the Boston Housing dataset, which is a classic dataset for regression problems.

```
# Create a synthetic housing dataset since Boston Housing has ethical concerns
np.random.seed(42)
n_samples = 500

# Generate synthetic features
size = np.random.normal(2500, 1000, n_samples)  # House size in sq ft
bedrooms = np.random.randint(1, 3, n_samples)  # Number of bedrooms
age = np.random.randint(1, 40, n_samples)      # Age of house
location_score = np.random.uniform(1, 15, n_samples)  # Location desirability score

# Generate target variable (price) with realistic relationships
price = (size * 150 + bedrooms * 10000 - age * 500 + location_score * 5000 +
         np.random.normal(0, 20000, n_samples))

# Create DataFrame
df = pd.DataFrame({
    'size': size,
    'bedrooms': bedrooms,
    'age': age,
    'location_score': location_score,
    'price': price
})

# Ensure positive prices
df['price'] = np.maximum(df['price'], 50000)

print(f"Dataset shape: {df.shape}")
df.head()
```

Dataset shape: (500, 5)

	size	bedrooms	age	location_score	price
0	2996.714153	2	2	7.732654	526140.555227
1	2361.735699	2	3	1.570557	381310.199779
2	3147.688538	2	23	3.295993	483461.368381
3	4023.029856	1	12	7.140501	657347.655162
4	2265.846625	2	20	11.147572	401388.131272

Next steps: Generate code with df New interactive sheet

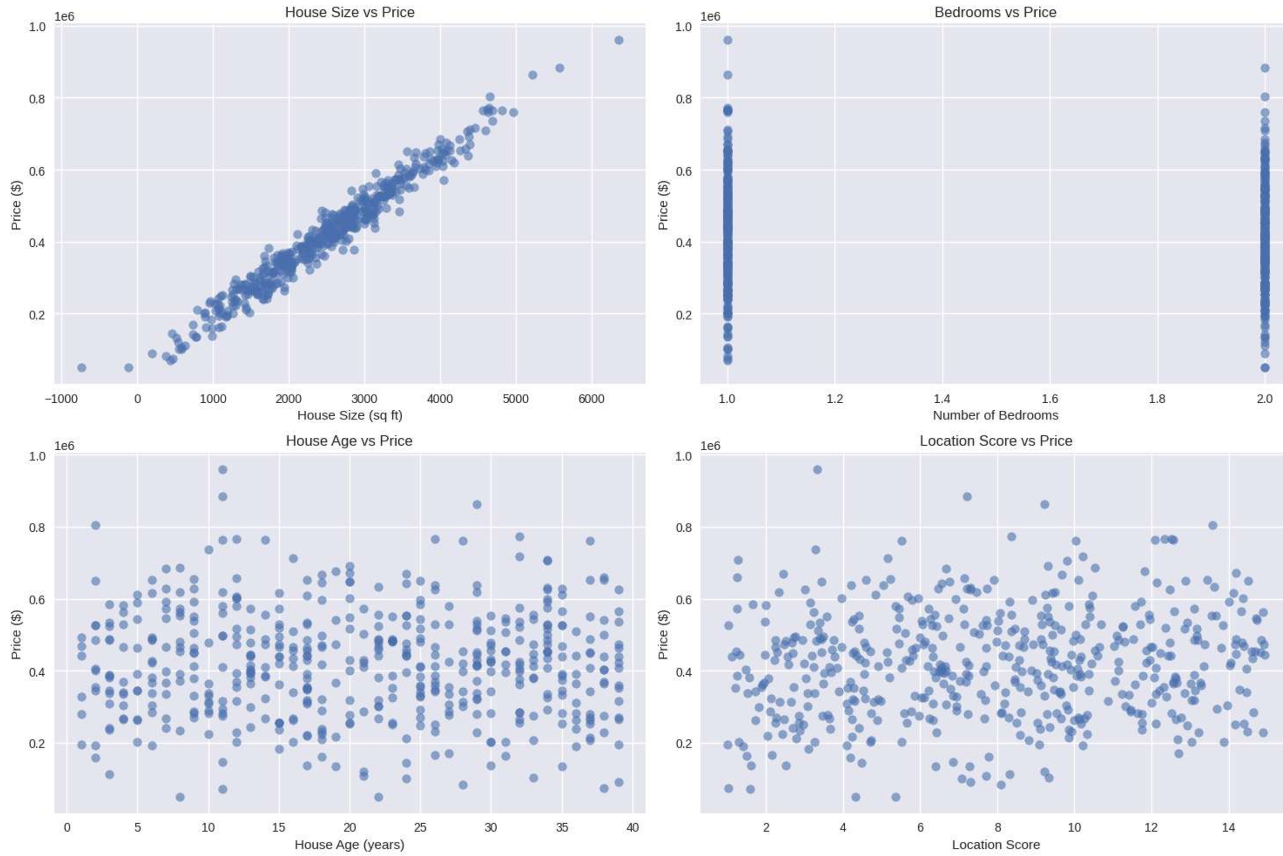
Exploratory Data Analysis

Let's explore our dataset to understand the relationships between variables.

```
# Basic statistics
print("Dataset Info:")
print(df.info())
print("\nBasic Statistics:")
print(df.describe())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   size            500 non-null    float64
 1   bedrooms        500 non-null    int64
 2   age             500 non-null    int64
 3   location_score  500 non-null    float64
 4   price           500 non-null    float64
dtypes: float64(3), int64(2)
memory usage: 19.7 KB
None

Basic Statistics:
              size  bedrooms         age  location_score          price
count   500.000000  500.0000  500.000000      500.000000     500.000000
mean   2506.837995    1.4800   20.706000        7.842993  419596.027891
std     981.253247    0.5001   11.057309        3.833236  148733.192245
min    -741.267340    1.0000    1.000000        1.004655   50000.000000
25%    1799.692596    1.0000   11.000000        4.521573  314973.790697
50%    2512.797146    1.0000   21.000000        7.763655  417765.925236
75%    3136.783254    2.0000   30.000000       10.730839  518168.632731
max    6352.731491    2.0000   39.000000       14.936727  960282.865982
```

```
# Visualize the relationships
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Size vs Price
axes[0, 0].scatter(df['size'], df['price'], alpha=0.6)
axes[0, 0].set_xlabel('House Size (sq ft)')
axes[0, 0].set_ylabel('Price ($)')
axes[0, 0].set_title('House Size vs Price')

# Bedrooms vs Price
axes[0, 1].scatter(df['bedrooms'], df['price'], alpha=0.6)
axes[0, 1].set_xlabel('Number of Bedrooms')
axes[0, 1].set_ylabel('Price ($)')
axes[0, 1].set_title('Bedrooms vs Price')

# Age vs Price
axes[1, 0].scatter(df['age'], df['price'], alpha=0.6)
axes[1, 0].set_xlabel('House Age (years)')
axes[1, 0].set_ylabel('Price ($)')
axes[1, 0].set_title('House Age vs Price')

# Location Score vs Price
axes[1, 1].scatter(df['location_score'], df['price'], alpha=0.6)
axes[1, 1].set_xlabel('Location Score')
axes[1, 1].set_ylabel('Price ($)')
axes[1, 1].set_title('Location Score vs Price')

plt.tight_layout()
plt.show()
```


House Size vs Price
Bedrooms vs Price
House Age vs Price
Location Score vs Price
Price ($)
1e6
House Size (sq ft)
Number of Bedrooms
House Age (years)
Location Score

Building the Linear Regression Model

Now let's build our linear regression model step by step.

```
# Prepare features and target
X = df[['size', 'bedrooms', 'age', 'location_score']]
y = df['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

print(f"Training set size: {X_train.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")
```

```
Training set size: 400
Test set size: 100
```

```
# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

print("Model trained successfully!")
```

```
Model trained successfully!
```

Model Evaluation

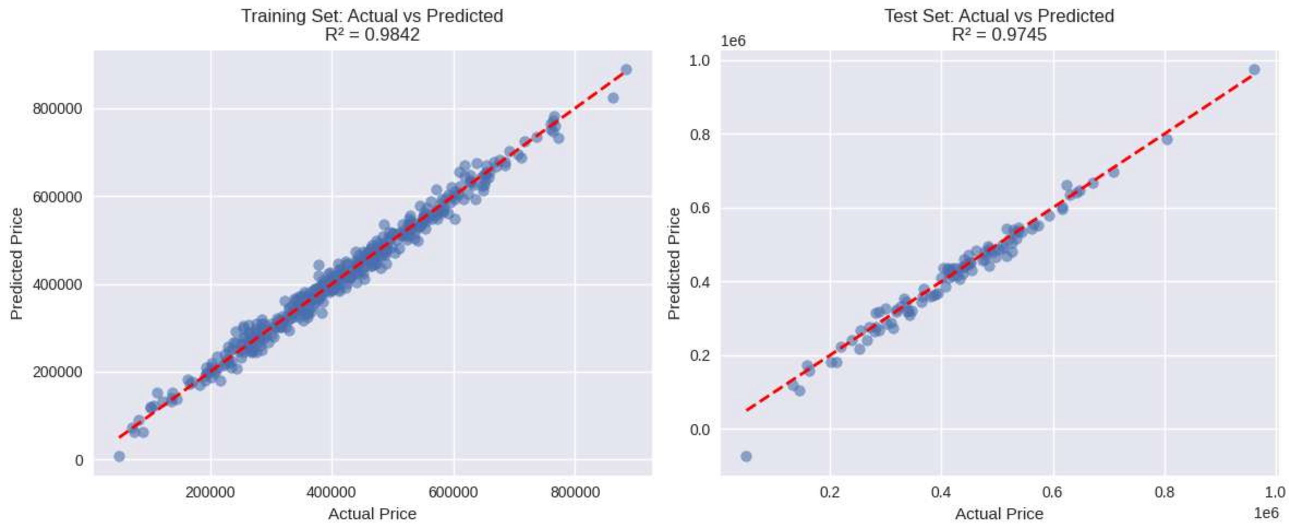
Let's evaluate how well our model performs.

```
# Calculate evaluation metrics
train_mse = mean_squared_error(y_train, y_pred_train)
test_mse = mean_squared_error(y_test, y_pred_test)
train_r2 = r2_score(y_train, y_pred_train)
test_r2 = r2_score(y_test, y_pred_test)
train_mae = mean_absolute_error(y_train, y_pred_train)
test_mae = mean_absolute_error(y_test, y_pred_test)

print("Model Performance:")
print(f"Training R² Score: {train_r2:.4f}")
print(f"Test R² Score: {test_r2:.4f}")
print(f"Training RMSE: ${np.sqrt(train_mse):,.2f}")
print(f"Test RMSE: ${np.sqrt(test_mse):,.2f}")
```

```
print(f"Training MAE: ${train_mae:,.2f}")
print(f"Test MAE: ${test_mae:,.2f}")
```

```
Model Performance:
Training R² Score: 0.9842
Test R² Score: 0.9745
Training RMSE: $18,826.43
Test RMSE: $22,962.57
Training MAE: $14,772.90
Test MAE: $17,011.32
```

```
# Visualize predictions vs actual values
plt.figure(figsize=(12, 5))

# Training set
plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, alpha=0.6)
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'r--', lw=
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title(f'Training Set: Actual vs Predicted\nR² = {train_r2:.4f}')

# Test set
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title(f'Test Set: Actual vs Predicted\nR² = {test_r2:.4f}')

plt.tight_layout()
plt.show()
```



Understanding the Model Coefficients

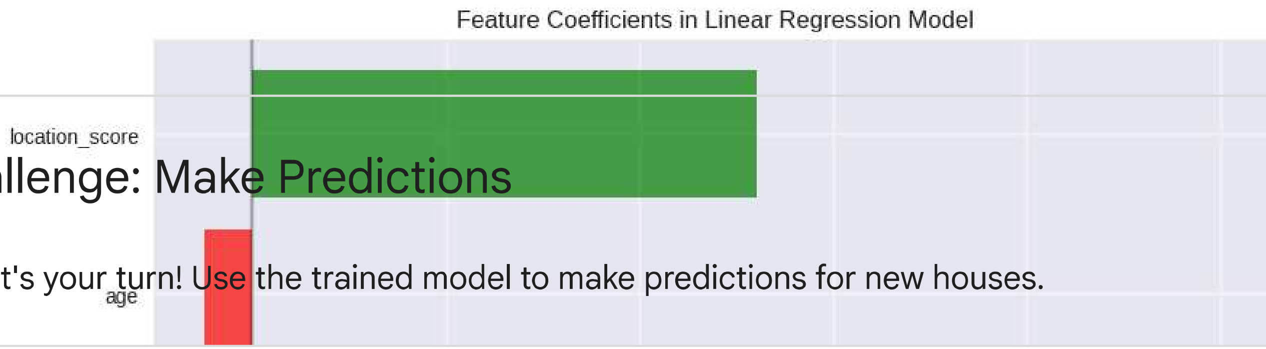
Let's examine what our model learned about the relationship between features and house prices.

```
# Display model coefficients
feature_names = X.columns
coefficients = model.coef_
intercept = model.intercept_

print("Linear Regression Equation:")
print(f"Price = {intercept:.2f}", end="")
for name, coef in zip(feature_names, coefficients):
    print(f" + ({coef:.2f} × {name})", end="")
print("\n")

# Create a DataFrame for better visualization
coef_df = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefficients,
    'Abs_Coefficient': np.abs(coefficients)
```

```
}).sort_values('Abs_Coefficient', ascending=False)

print("Feature Importance (by coefficient magnitude):")
print(coef_df)
```

```
Linear Regression Equation:
Price = -5364.60 + (150.86 × size) + (10051.77 × bedrooms) + (-497.18 × age) + (5209

Feature Importance (by coefficient magnitude):
          Feature   Coefficient  Abs_Coefficient
1        bedrooms  10051.769439     10051.769439
3  location_score   5209.693806      5209.693806
2             age   -497.178564       497.178564
0            size    150.855713       150.855713
```

```
# Visualize feature importance
plt.figure(figsize=(10, 6))
colors = ['green' if x > 0 else 'red' for x in coefficients]
plt.barh(feature_names, coefficients, color=colors, alpha=0.7)
plt.xlabel('Coefficient Value')
plt.title('Feature Coefficients in Linear Regression Model')
plt.axvline(x=0, color='black', linestyle='-', alpha=0.3)
plt.grid(True, alpha=0.3)
plt.show()
```



Challenge: Make Predictions

Now it's your turn! Use the trained model to make predictions for new houses.

```
# Challenge: Predict prices for these houses
new_houses = pd.DataFrame({
    'size': [1800, 2500, 1200],
    'bedrooms': [3, 4, 2],
    'age': [10, 5, 25],
    'location_score': [8.5, 9.2, 6.0]
})

# TODO: Use the model to predict prices for these houses
# predictions = model.predict(new_houses)

# Uncomment the lines below after making predictions
# for i, price in enumerate(predictions):
#     print(f"House {i+1}: ${price:,.2f}")

print("Houses to predict:")
print(new_houses)
```