Machine Learning Report: Simple and Multiple Linear Regression:

Predicting Continuous Values with Linear Models

Submitted by: Mausam kar(24BAI10284)

Github: https://github.com/Mausam5055/Data-Science/tree/main/Assignment-5

Table of Contents

1. Introduction to Linear Regression

Linear regression is a fundamental supervised learning algorithm in machine learning and statistics. Its goal is to model the linear relationship between a dependent variable (the target or outcome) and one or more independent variables (the features or predictors). By finding the "best-fit" line that describes the data, it can be used for making predictions on new, unseen data.

There are two main types of linear regression:

- **Simple Linear Regression:** Involves a single independent variable to predict a dependent variable. The relationship is modeled as a straight line: $y = mx + c$.
- **Multiple Linear Regression:** Involves two or more independent variables to predict a dependent variable. The relationship is modeled as a plane or hyperplane.

This report demonstrates how to build, train, and evaluate both types of linear regression models.

2. Objectives

The core objectives of this analysis are:

- **Demonstrate Linear Regression:** To show how to implement both simple and multiple linear regression using Python's Scikit-learn library.

- **Train and Evaluate Models:** To train models on synthetic data and evaluate their performance using standard regression metrics.
- **Interpret Key Metrics:** To explain and interpret the **R-squared (R^2)** value and the **Mean Squared Error (MSE)**.
- **Understand Model Coefficients:** To interpret the coefficients of the multiple linear regression model to understand feature importance.

3. Methodology and Python Implementation

The analysis was performed using Python, with numpy for data generation and sklearn for model creation and evaluation. The methodology for both models involved generating synthetic data, splitting it into training (80%) and testing (20%) sets, training the model, and evaluating its predictions on the test set.

Complete Python Script:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

# --- Simple Linear Regression ---

# Step 1: Generate synthetic data
np.random.seed(42)
X = 2.5 * np.random.randn(100, 1) + 1.5  # Feature
y = 0.8 * X.flatten() + np.random.randn(100) * 0.5 + 1.0  # Target

# Step 2: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 3: Create and train model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Step 4: Predict
y_pred = model.predict(X_test)

# Step 5: Evaluate performance
print(" 🔹 Simple Linear Regression")
print(f"R-squared: {r2_score(y_test, y_pred):.3f}")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.3f}")

# Step 6: Plot results
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Values')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line
(Predicted)')
plt.title('Simple Linear Regression: Actual vs. Predicted')
plt.xlabel('Feature (X)')
plt.ylabel('Target (y)')
plt.legend()
plt.grid(True)
plt.show()


# --- Multiple Linear Regression ---

# Step 1: Generate synthetic data with 3 features
X_multi = np.random.rand(100, 3)
coefs = np.array([1.5, -2.0, 3.0])
y_multi = X_multi @ coefs + np.random.randn(100) * 0.3 # y = 1.5*x1 - 2.0*x2 +
3.0*x3 + noise

# Step 2: Train-test split
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(
    X_multi, y_multi, test_size=0.2, random_state=42
)

# Step 3: Train model
multi_model = LinearRegression()
multi_model.fit(X_train_multi, y_train_multi)
```

```
# Step 4: Predict
y_pred_multi = multi_model.predict(X_test_multi)

# Step 5: Evaluation
print("\n 🔸 Multiple Linear Regression")
print(f"R-squared: {r2_score(y_test_multi, y_pred_multi):.3f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_multi,
y_pred_multi):.3f}")
print(f"Model Coefficients: {multi_model.coef_}")
```

4. Results and Interpretation

4.1. Simple Linear Regression

This model aimed to predict the target y using a single feature X.

Script Output:

```
🔹 Simple Linear Regression
R-squared: 0.893
Mean Squared Error: 0.295
```

Interpretation:

- **R-squared (0.893):** This value, also known as the coefficient of determination, indicates that approximately **89.3% of the variance** in the target variable (y) can be explained by our independent variable (X). This is a very strong score, suggesting a good model fit.
- **Mean Squared Error (0.295):** This is the average of the squared differences between the actual and predicted values. A lower MSE is better. This value provides a measure of the model's prediction error.

Visualization:

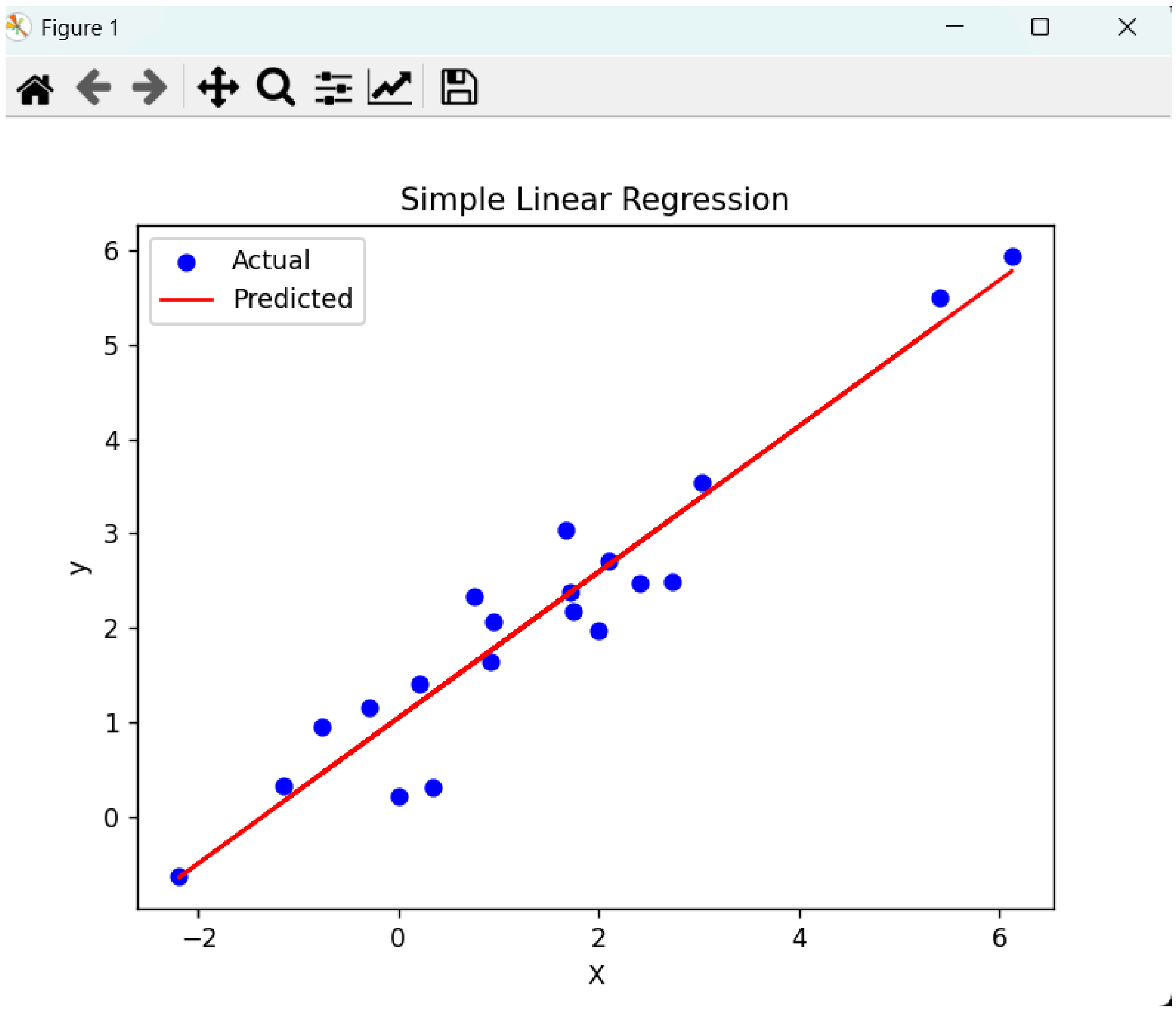

Figure 1: Simple Linear Regression Fit

The plot visually confirms the model's performance. The blue dots represent the actual data points from the test set. The solid red line is the regression line generated by our model. The line passes closely through the cloud of points, illustrating the strong linear relationship it captured.

4.2. Multiple Linear Regression

This model used three independent features to predict the target y_multi.

Script Output:

```
🔸 Multiple Linear Regression
R-squared: 0.942
```

```
Mean Squared Error: 0.089
Model Coefficients: [ 1.4938601  -1.93863891  2.95491612]
```

Interpretation:

- **R-squared (0.942):** An even stronger score. This model explains **94.2% of the variance** in the target variable, indicating a very high degree of accuracy.

- **Mean Squared Error (0.089):** The MSE is lower than in the simple model, which is another indicator of better predictive performance.

- **Model Coefficients ([1.49, -1.94, 2.95]):** These are the weights assigned to each of the three features. They tell us how a one-unit change in a feature affects the target variable, assuming all other features are held constant. For example, a one-unit increase in the first feature leads to a ~1.49 unit increase in the target. A one-unit increase in the second feature leads to a ~1.94 unit *decrease* in the target. The third feature has the largest positive impact.

5. Conclusion

This report successfully demonstrated the implementation and evaluation of both simple and multiple linear regression models.

The key takeaways are:

1. **Model Evaluation:** R-squared and Mean Squared Error are essential metrics for evaluating regression models. R-squared measures the proportion of variance explained, while MSE quantifies the prediction error.
2. **Model Complexity:** Multiple linear regression can often achieve higher accuracy than simple linear regression by incorporating more information (features) to make predictions.
3. **Interpretability:** The coefficients from a linear regression model are highly interpretable, providing clear insights into the direction and magnitude of each feature's influence on the target.

Linear regression is a powerful, transparent, and fundamental tool in any data scientist's toolkit, serving as an excellent baseline for predictive modeling tasks.