Responsible AI (AI354): Lab Assignment 2
Linear Regression Analysis

Devesh Singh Chauhan (I23MA002)

January 18, 2026

1 Question 1: Impact of Shuffling

1.1 Methodology

The goal of this experiment was to see if the way we organize our data before feeding it into the model changes the results.

First, we took the salary dataset and split it into two parts: 70% was used to train the model, and the remaining 30% was hidden away to test it later.

To test the impact of shuffling, we didn't just split the data once. We repeated the process five separate times. In each "run," we shuffled the data in a different random order before splitting it. This meant that different data points ended up in the test set each time. For every run, we calculated the Mean Squared Error (MSE) to see how much the error rate fluctuated.

1.2 Results

As shown in the graph below, the error rate was not constant. It changed significantly depending on which specific data points happened to fall into the test set.

This happens because our dataset is quite small (only 30 rows). In a small dataset, a single "unusual" data point (like someone with low experience but a very high salary) can skew the results heavily if it ends up in the test set versus the training set.

Figure 1: Fluctuation of MSE across 5 different random shuffles.

2 Question 2: Impact of Normalization

2.1 Methodology

In this second experiment, we wanted to see if the "scale" or "units" of the numbers mattered to the Linear Regression model. For example, does the model get confused because "Salary" (e.g., 40,000) is a much larger number than "Years of Experience" (e.g., 2.5)?

We used an 80:20 split for this task. We then trained three separate models:

1. **Raw Data:** We fed the numbers in exactly as they appeared in the CSV file.
2. **Min-Max Scaling:** We mathematically squeezed the "Years of Experience" so all values fell strictly between 0 and 1.
3. **Standard Normalization:** We adjusted the values so they had an average (mean) of 0 and a standard deviation of 1.

2.2 Results

After testing all three models, we found that the Mean Squared Error was identical for all of them (MSE $\approx 49.8 \times 10^6$).

This proves that for a standard Linear Regression model, normalization is not necessary for accuracy. The math behind the model automatically adjusts the "slope" (m) to compensate for the size of the input numbers. If we shrink the inputs, the model simply makes the slope steeper to arrive at the same prediction.

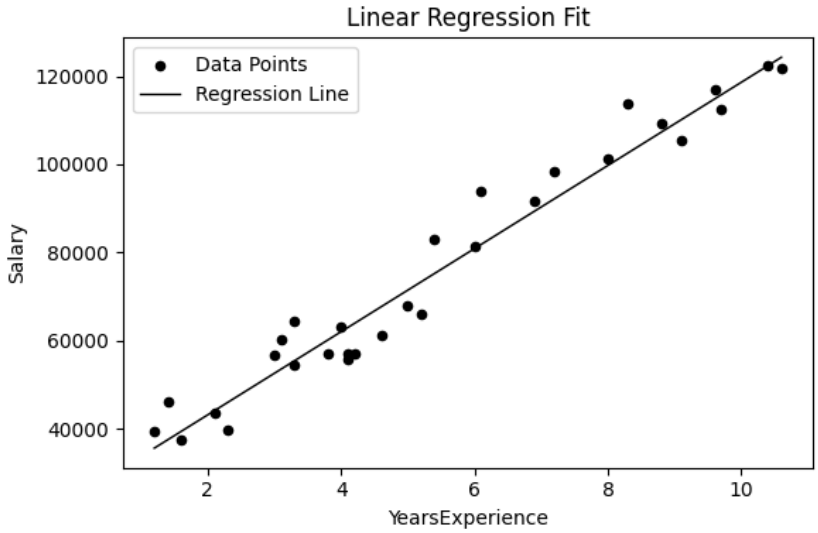

Figure 2: The Linear Regression fit used for the calculation.

3 Appendix: Python Code

3.1 Code for Question 1 (Shuffling)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('Salary_dataset.csv')
X = df[['YearsExperience']]
y = df['Salary']

# We loop through 5 different "seeds" to simulate 5 different shuffles
seeds = [10, 20, 30, 40, 50]
for seed in seeds:
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=seed, shuffle=True
    )
    model = LinearRegression()
    model.fit(X_train, y_train)
    mse = mean_squared_error(y_test, model.predict(X_test))
    print(f"Seed␣{seed}␣MSE:␣{mse}")
```

3.2 Code for Question 2 (Normalization)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler, StandardScaler

df = pd.read_csv('Salary_dataset.csv')
X = df[['YearsExperience']]
y = df['Salary']

# Fixed split so we can compare apples to apples
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 1. No Normalization
model = LinearRegression()
model.fit(X_train, y_train)
print(f"MSE␣None:␣{mean_squared_error(y_test,␣model.predict(X_test))}")

# 2. Min-Max Scaling
scaler = MinMaxScaler()
X_tr_mm = scaler.fit_transform(X_train)
X_te_mm = scaler.transform(X_test)
model.fit(X_tr_mm, y_train)
print(f"MSE␣MinMax:␣{mean_squared_error(y_test,␣model.predict(X_te_mm))}")

# 3. Standard Normalization
scaler = StandardScaler()
X_tr_std = scaler.fit_transform(X_train)
X_te_std = scaler.transform(X_test)
model.fit(X_tr_std, y_train)
print(f"MSE␣Std:␣{mean_squared_error(y_test,␣model.predict(X_te_std))}")
```