# Linear Regression

Minor in AI, IIT Ropar

2nd April, 2025

## Introduction

Linear regression is a fundamental supervised learning technique used to predict a continuous output variable ($Y$) based on one or more input variables ($X$). It is akin to teaching a child how to recognize patterns in labeled data, much like learning to identify fruits in a supermarket. This case study explores linear regression concepts, its mathematical foundation, and practical implementation using Python.

## The Curious Kid and Linear Regression

Imagine teaching a child about the relationship between age and height. You provide labeled examples: "At age 1, height is 75 cm; at age 2, height is 85 cm." The child learns that as age increases, so does height. This mirrors the essence of linear regression: finding the best-fit line that explains the relationship between input ($X$) and output ($Y$).

## Core Concepts

### Mathematical Foundation

The linear regression model is represented as:

$$Y = MX + B$$

where:

- $M$: Slope of the line (rate of change of $Y$ with respect to $X$),
- $B$: Intercept (value of $Y$ when $X = 0$).

### Model Training

The goal is to minimize the error between actual ($Y$) and predicted ($\hat{Y}$) values by adjusting $M$ and $B$.

### Visualization

Scatter plots display data points. The regression line represents the relationship learned by the model.

### Handling Outliers

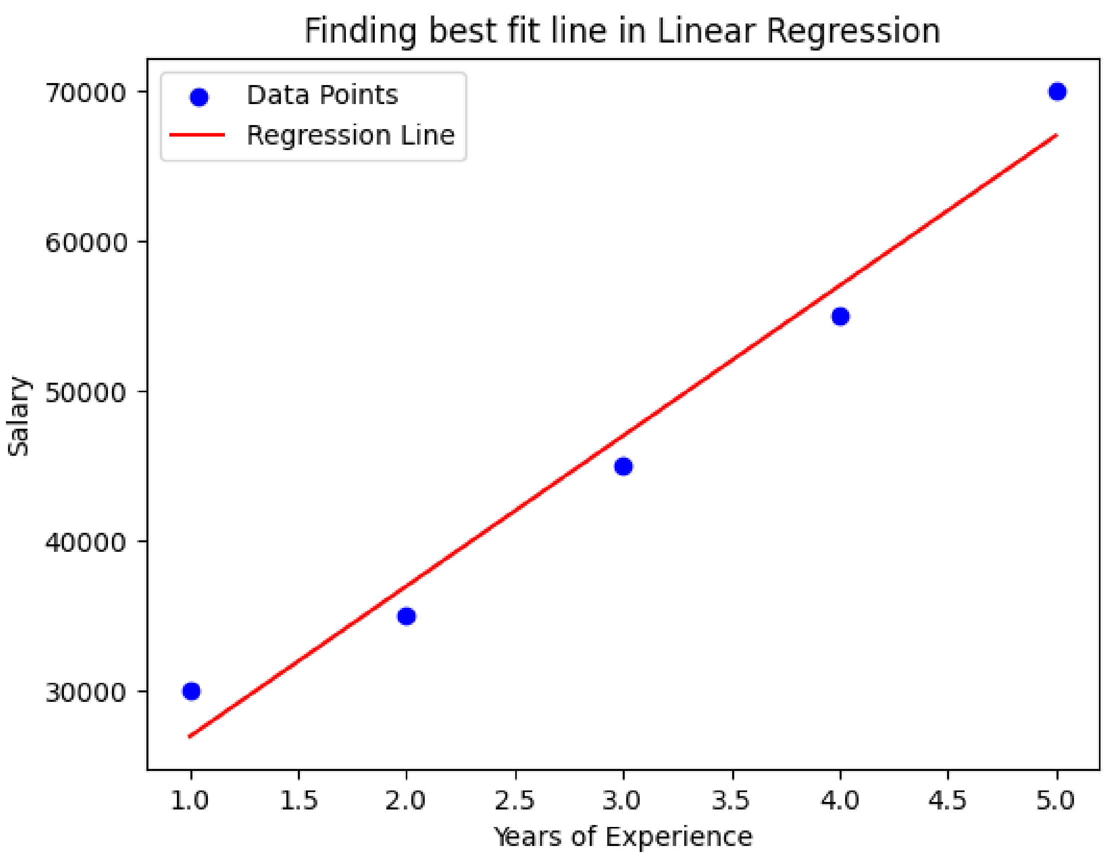
Outliers can distort the slope and intercept, leading to inaccurate predictions. Tools like box plots help identify such anomalies.

## Implementation in Python

### Example 1: Predicting Salary Based on Experience

The following Python code demonstrates how to train a linear regression model using Scikit-learn:

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Input (Years of Experience) and Output (Salary)
X = [[1], [2], [3], [4], [5]]
Y = [30000, 35000, 45000, 55000, 70000]

# Train the model
model = LinearRegression().fit(X, Y)

# Visualization
plt.scatter(X, Y, color='blue', label="Data Points")
plt.plot(X, model.predict(X), color='red', label="Regression Line")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.title("Salary Prediction using Linear Regression")
plt.show()
```

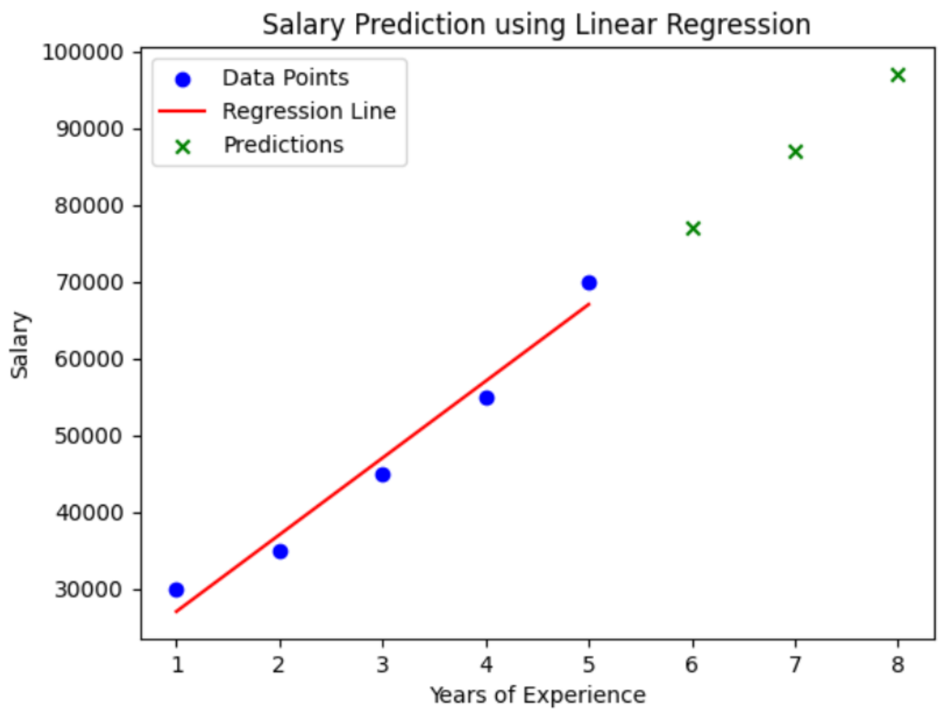



The plot illustrates the linear relationship between years of experience and salary.

### Example 2: Predicting Outputs for New Inputs

The following Python code predicts salary for new inputs:

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Input (Years of Experience) and Output (Salary)
X = [[1], [2], [3], [4], [5]]
Y = [30000, 35000, 45000, 55000, 70000]

# Train the model
model = LinearRegression().fit(X, Y)

# Visualization
plt.scatter(X, Y, color='blue', label="Data Points")
plt.plot(X, model.predict(X), color='red', label="Regression Line")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.title("Finding best fit line in Linear Regression")

# Predicting for new inputs
new_inputs = [[6], [7], [8]]
predictions = model.predict(new_inputs)

# Plotting for new inputs
plt.scatter(new_inputs, predictions, color='green', marker='x', label="Predictions")
plt.legend()
plt.show()

for inp, pred in zip(new_inputs, predictions):
    print(f"For input {inp[0]}, predicted output is {pred:.2f}")
```




```
For input 6, predicted output is 77000.00
For input 7, predicted output is 87000.00
For input 8, predicted output is 97000.00
```

This demonstrates how predictions can be made for unseen data after training, i.e. after the best fit line has been created by our linear regression model.

## How Changing Slope (M) and Intercept (B) Affects the Regression Line

The slope ($M$) and intercept ($B$) fundamentally shape a regression line's behavior. Here's how they influence predictions:

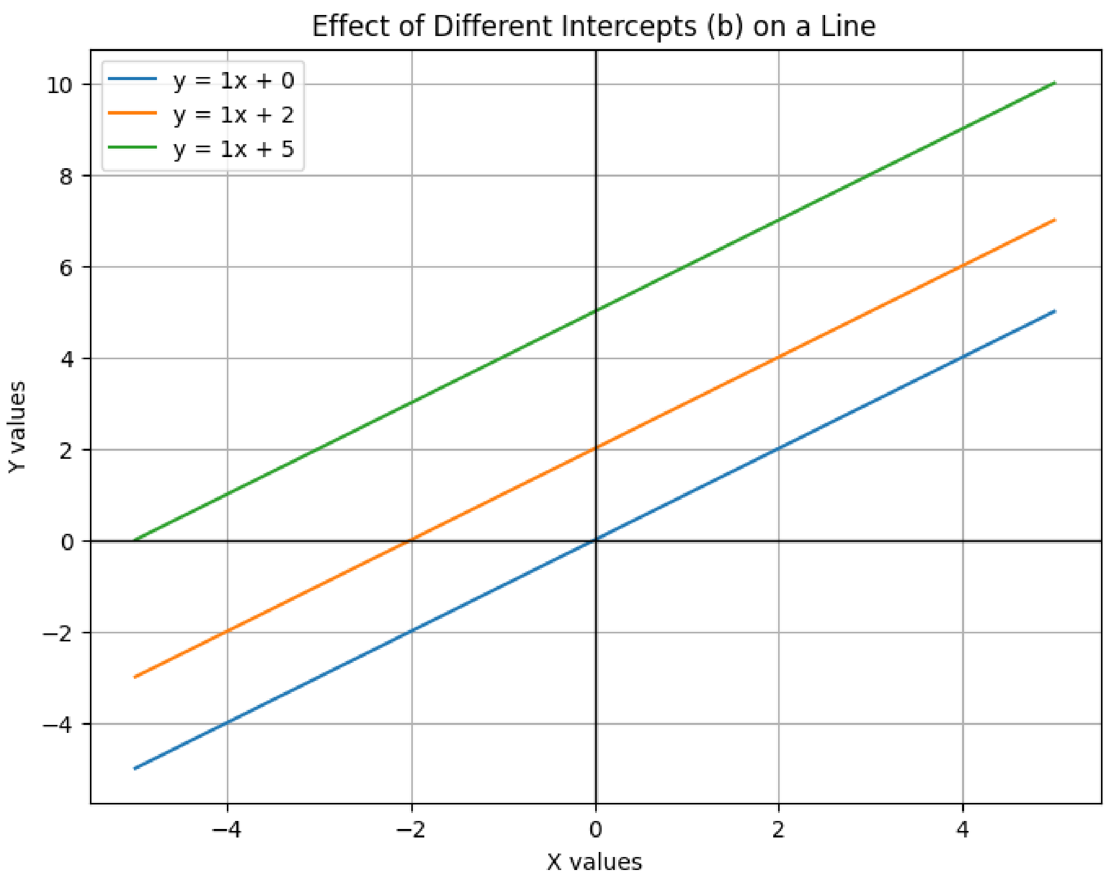
### 1. Impact of Intercept (B)

- The intercept determines where the line crosses the Y-axis (when $X = 0$).
- Increasing $B$ shifts the entire line **vertically upward** without changing its steepness.
- Decreasing $B$ shifts the line **downward**, maintaining the same rate of change.

The following Python code visualizes how changes in $B$ affect the regression line:

```
import numpy as np
import matplotlib.pyplot as plt

# Define different values of b (intercept) with a fixed m
b_values = [0, 2, 5]  # Different intercepts
m = 1  # Fixed slope
X = np.linspace(-5, 5, 100)  # Generate values from -5 to 5
```

```
# Plot different intercepts
plt.figure(figsize=(8,6))
for b in b_values:
    Y = m * X + b
    plt.plot(X, Y, label=f"y = {m}x + {b}")

# Highlight origin and axes
plt.axhline(0, color='black', linewidth=1)
plt.axvline(0, color='black', linewidth=1)

# Labels and legend
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Effect of Different Intercepts (b) on a Line")
plt.legend()
plt.grid()
plt.show()
```



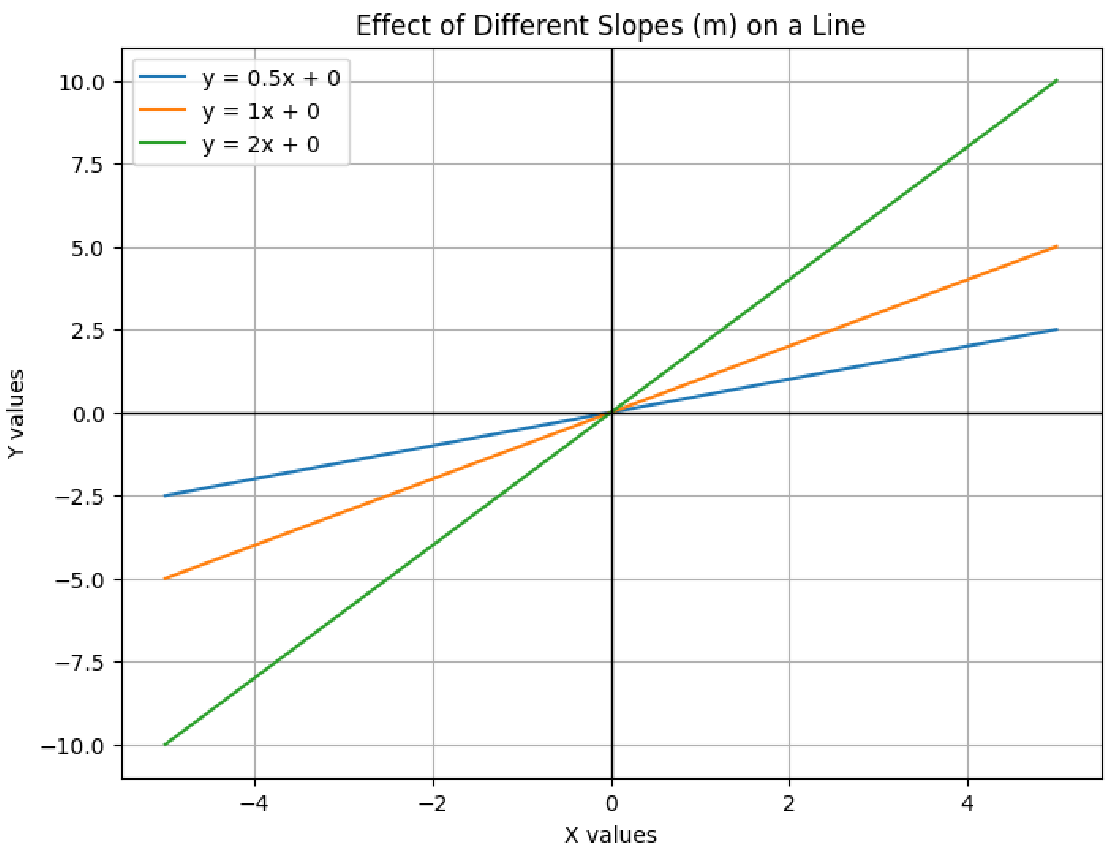

The plot shows that lines shift vertically while maintaining their slope when $B$ changes.

## 2. Impact of Slope (M)

- The slope defines the line's steepness and direction (positive/negative relationship).
- A larger $|M|$ creates a steeper line, indicating stronger rate of change.
- $M = 0$ produces a horizontal line (no relationship between $X$ and $Y$).

The following Python code visualizes how changes in $M$ affect the regression line:

```
import numpy as np
import matplotlib.pyplot as plt

# Define different values of m (slope) with a fixed b
m_values = [0.5, 1, 2]  # Different slopes
b = 0  # Fixed intercept
X = np.linspace(-5, 5, 100)

# Plot different slopes
plt.figure(figsize=(8,6))
for m in m_values:
    Y = m * X + b
    plt.plot(X, Y, label=f"y = {m}x + {b}")

# Highlight origin and axes
plt.axhline(0, color='black', linewidth=1)
plt.axvline(0, color='black', linewidth=1)

# Labels and legend
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Effect of Different Slopes (m) on a Line")
plt.legend()
plt.grid()
plt.show()
```




The plot shows that lines rotate while maintaining their intercept when $M$ changes.

## Real-World Analogy: Supervised Learning

Linear regression aligns with supervised learning principles:

- **Labeled Data**: The model learns from examples where both inputs ($X$) and outputs ($Y$) are known.
- **Prediction**: Once trained, the model predicts outputs for new inputs.

## Applications

Linear regression has widespread applications:

- Predicting house prices based on features like size or location.
- Forecasting sales trends over time.
- Estimating salaries based on years of experience.

## Conclusion

Linear regression bridges mathematical theory and practical application, offering insights into relationships within data. By understanding its fundamentals and implementing it step-by-step, we can solve real-world problems effectively.