Linear Regression in ML

**✅ Category:**

* Supervised Learning
* Regression Algorithm (predicts continuous values)

**🔍 What is Linear Regression?**

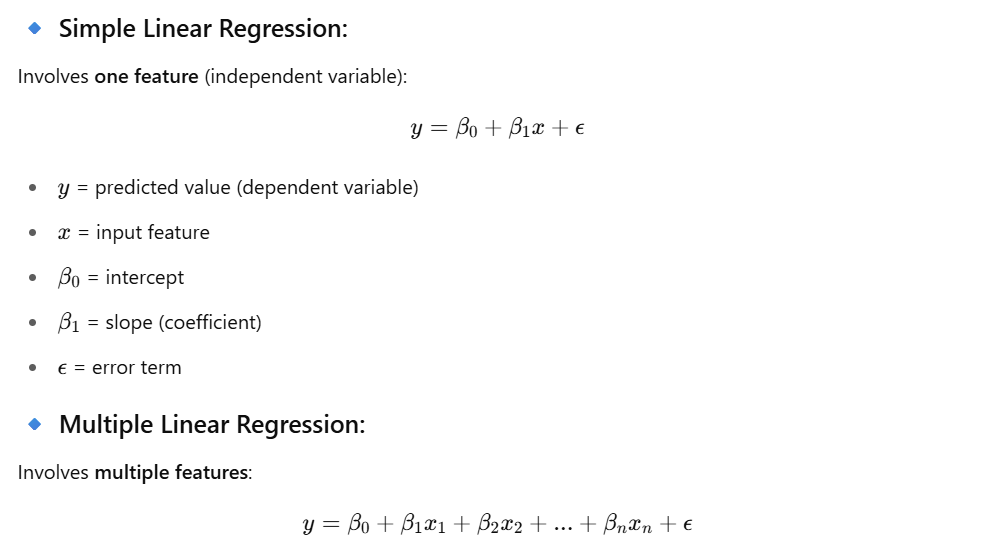
[**https://www.geeksforgeeks.org/linear-regression-python-implementation/**](https://www.geeksforgeeks.org/linear-regression-python-implementation/)

[**https://www.geeksforgeeks.org/ml-linear-regression/**](https://www.geeksforgeeks.org/ml-linear-regression/)

Linear Regression is a statistical method that models the **relationship between a dependent variable (target)** and one or more **independent variables (features)** using a **linear equation**.

**🔹 Simple Linear Regression:**

Bias variance trade off

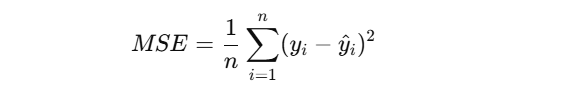


**🎯 Use Cases of Linear Regression**

| **Domain** | **Application** |
| --- | --- |
| Finance | Predicting stock prices, housing prices |
| Marketing | Sales forecasting, campaign ROI |
| Healthcare | Predicting disease progression (e.g., blood pressure) |
| HR/Analytics | Predicting employee salary or attrition risk |
| Sports | Player performance prediction |

**🛠️ How It Works (Training Process)**

1. **Input**: Historical data with features and labels
2. **Model**: Fit a line that minimizes the distance between the line and the actual data points
3. **Objective**: Minimize the **Mean Squared Error (MSE)**:



1. **Optimization**: Use **Gradient Descent** or **Normal Equation** to find optimal β\betaβ values

**📈 Assumptions of Linear Regression**

1. **Linearity**: The relationship between X and y is linear
2. **Independence**: Observations are independent
3. **Homoscedasticity**: Constant variance of errors
4. **Normality**: Errors are normally distributed
5. **No multicollinearity** (for multiple linear regression)

**Linear Regression Real Use Case**

**📘 Goal: Predict median house value based on average number of rooms per household.**

# 📌 Step 1: Import Libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

# 📌 Step 2: Load California Housing Dataset

data = fetch\_california\_housing()

X = data.data # Features

y = data.target # Target: Median house value

# Let's use only 1 feature for visualization: 'AveRooms' (index 3)

X = X[:, [3]] # Average number of rooms per household

# 📌 Step 3: Train/Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 📌 Step 4: Define and Train Linear Regression Model

model = make\_pipeline(

StandardScaler(), # Normalize input features

LinearRegression()

)

model.fit(X\_train, y\_train)

# 📌 Step 5: Predict on Test Data

y\_pred = model.predict(X\_test)

# 📌 Step 6: Evaluate Model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("✅ Mean Squared Error (MSE):", mse)

print("✅ R^2 Score:", r2)

# 📌 Step 7: Visualize Results

plt.figure(figsize=(10,6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual', alpha=0.5)

plt.plot(X\_test, y\_pred, color='red', label='Predicted Line', linewidth=2)

plt.xlabel('Average Rooms per Household')

plt.ylabel('Median House Value')

plt.title('Linear Regression - California Housing')

plt.legend()

plt.grid(True)

plt.show()

📊 Model Evaluation

| Metric | Meaning |
| --- | --- |
| MSE | Lower MSE means better prediction performance |
| R² Score | Closer to 1 is better (1 = perfect fit) |

🔁 Comparison with SVR

| Feature | Linear Regression | SVR (RBF Kernel) |
| --- | --- | --- |
| Model Type | Global linear fit | Localized, flexible fit |
| Works well on | Linear data | Nonlinear, noisy data |
| Easy to interpret | ✅ Yes | ❌ No |
| Handles outliers | ❌ Poorly | ✅ Better (with ε margin) |
| Performance | Fast, simple | Slower, more powerful |

✅ Next Steps

**✅ Pros and Cons**

**✅ Pros:**

* Simple and easy to understand
* Interpretable coefficients
* Fast to train
* Works well when assumptions are met

**❌ Cons:**

* Assumes linearity
* Sensitive to outliers
* Can underperform on complex datasets
* Requires assumptions to be met

**🔎 When to Use Linear Regression?**

* When the relationship between features and target is roughly linear
* When you need a quick, interpretable model
* When the dataset is small to medium size and clean