Linear Regression:
A Step-by-Step Tutorial

Linear regression is a statistical method used to model the relationship between a dependent variable (output) and one or more independent variables (inputs). In simple terms, it finds the best-fitting straight line through the data points, which can be used to make predictions. The objective of linear regression is to determine the line that minimizes the differences (errors) between the predicted values and the actual values.

The goal of linear regression is to find a straight line that best fits a given set of data points. This line is used to predict an output (dependent variable) based on an input (independent variable). In linear regression, we assume a *linear relationship* between the input and output.

The Equation of the line y = ax+b

Where:

- y is the predicted output (in this case, Disease Progression),

- x is the input (in this case, BMI),

- a is the *slope* of the line (how much *y* changes when x increases by 1),

- b is the *intercept* (the value of (y) when (x = 0).

The goal of linear regression is to *find the best values of a and b* such that the line fits the data as closely as possible, meaning it minimizes the error between predicted and actual values.

2. *Data Splitting Process*

Data:

We start with a dataset containing 20 data points. Each data point includes a *BMI value* (input) and a corresponding *Disease Progression value* (output).

BMI (X values): [22, 25, 28, 21, 29, 23, 27, 26, 30, 24, 31, 33, 35, 36, 34, 32, 40, 38, 37, 39]

Disease Progression (Y values): [45, 48, 51, 44, 54, 47, 50, 49, 55, 46, 57, 60, 62, 64, 61, 59, 70, 68, 65, 72]

Splitting the Data:

To build a reliable model, we split the data into two sets:

- Training Set: Used to "train" the model and learn the relationship between BMI and Disease Progression.

- Testing Set: Used to test the model and check how well it predicts new data.

We split the data so that:

- The first 15 data points are used for training.

- The last 5 data points are used for testing.

Training Data:

- **Training Data (BMI):** [22, 25, 28, 21, 29, 23, 27, 26, 30, 24, 31, 33, 35, 36, 34]

- **Training Data (Disease Progression)**:[45, 48, 51, 44, 54, 47, 50, 49, 55, 46, 57, 60, 62, 64, 61]

Testing Data:

- **Testing Data (BMI):** [32, 40, 38, 37, 39]

- **Testing Data (Disease Progression):** [59, 70, 68, 65, 72]

3. Training Process: How the Model Learns the Equation

Train the Linear Regression Model

Now we'll use the training data (BMI and Disease Progression) to train the linear regression model. The model will try to find a line (equation) that best fits the training data.

Training Data:

- **X_train (BMI):** `[22, 25, 28, 21, 29, 23, 27, 26, 30, 24, 31, 33, 35, 36, 34]`
- **y_train (Disease Progression):** `[45, 48, 51, 44, 54, 47, 50, 49, 55, 46, 57, 60, 62, 64, 61]`

The model tries to fit a line to these points. The equation of the line is typically of the form:

Disease Progression=a×BMI+b Where:

- a is the slope (how much disease progression increases for each unit increase in BMI).
- b is the intercept (the disease progression when BMI is 0).

The model needs to learn the relationship between BMI and Disease Progression by finding the best-fitting line. This is done using a method called *Gradient Descent*.

Step 1: Start with Initial Guess for the Line

The model begins by guessing initial values for the *slope* a and *intercept* b. These might be random values, but let's assume: a = 0, b = 0

This means the model initially predicts a flat line at y = 0 for all BMI values, which is clearly not a good fit.

Step 2: Calculate the Errors

The model calculates the *error* (difference between the actual value and predicted value) for each data point. It does this for all 15 training points.

For example, for the first training point (BMI = 22, Disease Progression = 45):

Predicted Disease Progression = a * 22 + b = 0 *22+0 = 0

Error=Actual Disease Progression−Predicted Disease Progression=45−0=45

The error is 45 for this point. The model does this for **all 15 training points** and then calculates the **Sum of Squared Errors** (SSE) to ensure all errors (both positive and negative) contribute equally.

Error for each point = Actual value - Predicted value

Squared error = $(\text{Error})^2$

Sum of Squared Errors (SSE) = Add up all the squared errors.

For example, after calculating all errors, the SSE might look something like this:

SSE=45^2+… (for all 15 points)

Step 3: Adjust the Line (Optimize aaa and bbb)

At this point, the model sees that its prediction is way off because the error is large (the SSE is big). So, the model needs to **adjust** the slope aaa and intercept bbb to reduce the error.

Gradient Descent: How Adjustments Work

The model uses a method called **Gradient Descent** to make small adjustments to aaa and bbb to gradually make the line fit better.

Here's how Gradient Descent works in simple terms:

1. The model calculates the **direction** in which it needs to adjust a and b to reduce the error. It does this by looking at how the error changes when a or b change slightly.

2. The model makes a **small adjustment** to a and b. This is called a **learning step**. If increasing a decreases the error, the model will increase a by a small amount. Similarly, it adjusts b.
3. The model recalculates the errors with the new a and b values.
4. The model repeats this process many times, adjusting a and b a little bit each time to gradually reduce the error.

For example, after a few steps, the model might adjust the line to:

$a=1, b=10$

Now, the predicted line is:

$$\text{Predicted Disease Progression} = 1 \times \text{BMI} + 10$$

Let's calculate the new prediction for the first training point (BMI = 22):

$$\text{Predicted Disease Progression} = 1 \times 22 + 10 = 32$$

The new error for this point is:

$$\text{Error} = 45 - 32 = 13$$

This is better than the original error of 45, so the line is improving.

The error is now smaller, so the line is improving.

Step 4: Repeat the Process

The model continues adjusting the slope a and intercept b many times, gradually reducing the error. Each time, the model recalculates the total error (SSE) and adjusts the line accordingly.

Let's say after many iterations, the model adjusts the values of a and b as follows:

$a=1.5, \ b=10$

This gives the equation: $\text{Disease Progression} = 1.5 \times \text{BMI} + 10$

Now, for the first training point (BMI = 22), the predicted value is:

$$\text{Predicted Disease Progression} = 1.5 \times 22 + 10 = 43$$

The error is:

$$\text{Error} = 45 - 43 = 2$$

This error is much smaller than before, and when the model does this for all the training points, the overall error (SSE) becomes much smaller than the original error.

The model keeps adjusting a and b over many iterations, each time reducing the error a little more. Eventually, after many adjustments, it finds the **best-fitting line**.

Step 5: Arriving at the Final Line

After the model has gone through many iterations, adjusting a and b each time to reduce the error, it finally arrives at the **best-fitting line**. This is the line that has the **smallest possible Sum of Squared Errors (SSE)**.

For our example, the final line might be:

Disease Progression=1.5×BMI+10

This is the equation the model finds after training, and it represents the best fit for the training data. This line minimizes the error between the predicted and actual disease progression values.

This is the equation the model finds that minimizes the error between predicted and actual values.

Why Does the Model Use Gradient Descent?

- **Gradient Descent** is like the model's "trial and error" process. It tries different values for a and b, checks if the error goes down, and then adjusts the values to make the error smaller.
- By using small steps to adjust a and b, Gradient Descent ensures that the model slowly moves towards the best-fitting line, reducing the error with each step.

---

4. Testing the Model

Now that we have a trained model, we can use it to predict the disease progression for the **testing data** (the BMI values the model has not seen yet).

- **Testing Data (BMI):** `[32, 40, 38, 37, 39]`

We will use the model's equation Disease Progression=1.5×BMI+10 to make predictions:

- For BMI = 32:
  Predicted Progression=1.5×32+10=58
- For BMI = 40:
  Predicted Progression=1.5×40+10=70
- For BMI = 38:
  Predicted Progression=1.5×38+10=67
- For BMI = 37:
  Predicted Progression=1.5×37+10=65.5

- For BMI = 39:
  Predicted Progression=1.5×39+10=68.5

The predicted disease progression values are:

- **Predicted Progression**: `[58, 70, 67, 65.5, 68.5]`

5. Comparing Predictions to Actual Values

We now compare the predicted disease progression values with the actual values from the testing data:

- **Actual Progression**: `[59, 70, 68, 65, 72]`
- **Predicted Progression**: `[58, 70, 67, 65.5, 68.5]`

You can see that the predicted values are quite close to the actual values, which means our model is doing well.

6. Evaluating the Model with R^2 Score

The **R^2 score** (also called the coefficient of determination) measures how well the model's predictions match the actual values. It tells us what percentage of the variance in the target (disease progression) is explained by the model.

The formula for R^2 is:

$R^2 = 1 - \text{Sum of Squared Errors of Prediction} / \text{Total Sum of Squares of the Actual Data}$

Where:

Sum of Squared Errors of Prediction: Measures the difference between the actual and predicted values.

Total Sum of Squares: Measures how much the actual values vary.

Let's calculate R^2 for our example:

Actual Progression: `[59, 70, 68, 65, 72]`

Predicted Progression: `[58, 70, 67, 65.5, 68.5]`

1. **Calculate the errors** (difference between actual and predicted values):

   $\text{Errors} = [59-58, 70-70, 68-67, 65-65.5, 72-68.5] = [1, 0, 1, -0.5, 3.5]$

2. **Square the errors** and sum them up:

   $\text{Sum of Squared Errors} = 1^2 + 0^2 + 1^2 + (-0.5)^2 + 3.5^2 = 1 + 0 + 1 + 0.25 + 12.25 = 14.5$

3. **Calculate the total sum of squares** (how much the actual values vary from their mean):
    - Mean of actual values = (59+70+68+65+72) / 5 = 66.8
    - Differences from mean: $[59-66.8, 70-66.8, 68-66.8, 65-66.8, 72-66.8] = [-7.8, 3.2, 1.2, -1.8, 5.2]$
    - Square these differences and sum them:
      Total Sum of Squares $= (-7.8)^2 + 3.2^2 + 1.2^2 + (-1.8)^2 + 5.2^2 = 60.84 + 10.24 + 1.44 + 3.24 + 27.04 = 102.8$
4. **Calculate R^2**:

$R^2 = 1 - 14.5/102.8 = 1 - 0.141 = 0.859$

The **R^2 score is 0.859**, which means that about **85.9%** of the variation in the disease progression is explained by the model.

Why R^2 is Important:

- R^2 gives us an idea of how well the model is performing.
- An R^2 score close to 1 means the model's predictions are very accurate.
- An R^2 score close to 0 means the model is not doing a good job.

In our case, an R^2 score of 0.859 shows that the model explains most of the variation in disease progression based on BMI, which means it is a good fit!

1. Goal of Linear Regression: Find the best-fitting line that describes the relationship between BMI and Disease Progression.
    a. The model starts with random guesses for the slope (a) and intercept (b).
    b. It calculates the errors (differences between predicted and actual values) for the training data.

2. Data Splitting: Split data into training and testing sets to evaluate the model's performance.

3. Training Process: Use Gradient Descent to adjust the slope a and intercept b to minimize the error.

    a. We started with 20 data points and split them into 15 for training and 5 for testing.

    b. It adjusts a and b using Gradient Descent to reduce the error step by step.
    c. After many iterations, the model finds the line that best fits the data, minimizing the error.
    d. This equation is then used to make predictions for new, unseen data.

5. Testing Process: Use the learned equation to predict Disease Progression for new BMI values.
    - We tested the model on new BMI values and found that it made good predictions.

5. Comparing Predictions to Actual Values: Check how well the model's predictions match the actual data.

6. Evaluating the Model with R^2 Score: Assess the model's accuracy using the R^2 score.

The R^2 score showed that our model was able to explain 85.9% of the variation in disease progression, indicating it was a successful linear regression model.

This process demonstrates how linear regression works and how it is used to make predictions based on a simple linear relationship between input and output.

Python Implementation

```
# pip install scikit-learn


# Importing necessary libraries
import matplotlib.pyplot as plt  # For plotting
import numpy as np  # For numerical operations

# Importing required modules from sklearn
from sklearn import datasets, linear_model  # Datasets and linear regression model
from sklearn.metrics import mean_squared_error, r2_score  # To evaluate the model

# Load the diabetes dataset (available by default in scikit-learn)
# The dataset has 10 physiological variables (features) measured on 442 patients
# It is used for regression tasks to predict the disease progression after one year.
X_diabetes, y_diabetes = datasets.load_diabetes(return_X_y=True)

# Selecting only one feature (e.g., BMI or Body Mass Index, which is the 3rd feature in this case)
# `np.newaxis` ensures the data is reshaped to a 2D array with one column.
X_selected_feature = X_diabetes[:, np.newaxis, 2]

# Split the data into training and testing sets
# Training set includes all but the last 20 samples, testing set includes the last 20 samples
X_train = X_selected_feature[:-20]  # First 422 data points for training
X_test = X_selected_feature[-20:]   # Last 20 data points for testing
```

```
# Split the target values (disease progression) into training and testing sets
y_train = y_diabetes[:-20]  # First 422 target values for training
y_test = y_diabetes[-20:]   # Last 20 target values for testing

# Create an object for the Linear Regression model
linear_regression_model = linear_model.LinearRegression()

# Train the linear regression model using the training set (fit the model)
linear_regression_model.fit(X_train, y_train)

# Use the trained model to predict disease progression for the test set
y_predicted = linear_regression_model.predict(X_test)

# Displaying model results
# Displaying the coefficients (slope) of the linear regression line
print(f"Coefficients (Slope of the regression line): {linear_regression_model.coef_}")
# Calculating and displaying the Mean Squared Error (MSE), a measure of model accuracy
mse = mean_squared_error(y_test, y_predicted)
print(f"Mean Squared Error (MSE): {mse:.2f}")
# Calculating and displaying the R^2 score, which indicates the proportion of variance explained by the model
r2 = r2_score(y_test, y_predicted)
print(f"R^2 Score (Coefficient of Determination): {r2:.2f}")

# Plotting the data points (actual values) and the regression line (predicted values)
plt.scatter(X_test, y_test, color="black", label='Actual data')  # Plot actual data points
plt.plot(X_test, y_predicted, color="blue", linewidth=2, label='Linear regression line')  # Plot regression line
plt.xlabel("Selected Feature (BMI)")
```

```
plt.ylabel("Disease Progression")

plt.title("Linear Regression on Diabetes Dataset")

plt.legend()


# Display the plot

plt.show()
```

Step 1: Splitting Data into Training and Testing Sets

```
python
Copy code
X_train = X_selected_feature[:-20]  # First 422 data points for training
X_test = X_selected_feature[-20:]   # Last 20 data points for testing
```

- **What are we doing here?** We are dividing the data into two parts: a **training set** and a **testing set**.
- **Why do we do this?** When we build a machine learning model, we want to "teach" the model by showing it some data (the training set). Then, we want to test the model's performance by giving it new data that it hasn't seen before (the testing set). This helps us understand how well the model will work in real life.
- **How does the code work?**
  - `X_selected_feature[:-20]`: This part takes **all the data except the last 20 samples**. The `[:-20]` means "take everything from the beginning until 20 from the end." This is our **training data**.
  - `X_selected_feature[-20:]`: This part takes **only the last 20 samples** of data. The `[-20:]` means "take the last 20 samples." This is our **testing data**.

Step 2: Splitting the Target Values into Training and Testing Sets

```
python
Copy code
y_train = y_diabetes[:-20]  # First 422 target values for training
y_test = y_diabetes[-20:]   # Last 20 target values for testing
```

- **What are the target values?** The target values (`y_diabetes`) are what we are trying to predict—in this case, it's the progression of diabetes in patients.
- **What are we doing here?** Just like we split the feature data in Step 1, we are splitting the target values (which represent the actual disease progression) into training and testing sets. This means that we will have some known values of the disease progression to compare against our model's predictions later.
- **How does the code work?**
  - `y_diabetes[:-20]`: This part takes all the target values **except the last 20 samples**. This becomes our training target data, which matches with the training features.

- `y_diabetes[-20:]`: This part takes only the **last 20 target values**, which corresponds to the testing set. These are the actual values of disease progression that we will compare against the model's predictions.

Step 3: Creating a Linear Regression Model

```
python
Copy code
linear_regression_model = linear_model.LinearRegression()
```

- **What is happening here?** We are creating a **linear regression model**. Think of it as building a machine that will learn the relationship between the input (BMI, in this case) and the output (disease progression).
- **What is linear regression?** Linear regression is a type of model that tries to find a straight line that best fits the data. This line helps us make predictions about new data. It looks for a pattern that says, "for every increase in BMI, how much does the disease progression change?"
- **Why do we need this model?** We need the model to help us predict the disease progression of new patients based on their BMI.

Step 4: Training the Linear Regression Model

```
python
Copy code
linear_regression_model.fit(X_train, y_train)
```

- **What does "training" mean?** Training means teaching the model to understand the relationship between the input (BMI) and the output (disease progression). The model will look at the training data and figure out a formula (a straight line) that fits the data well.
- **How does this work?** The `fit()` function tells the model to find the best line that connects the **training data (X_train)** to the **target values (y_train)**. In other words, it looks at the BMI data and disease progression to find a pattern.

Step 5: Using the Model to Make Predictions

```
python
Copy code
y_predicted = linear_regression_model.predict(X_test)
```

- **What are predictions?** Predictions are the model's guesses about what the disease progression will be based on new data it hasn't seen before (the testing set).
- **What is happening here?** Now that the model is trained, we give it the **testing data (X_test)**, which contains the BMI values of 20 patients the model has never seen before. The model will use what it learned from the training data to predict the disease progression for these patients.
- **Why do we make predictions?** We want to see how well our model can predict the actual disease progression. Later, we will compare these predictions (`y_predicted`) with the real values (`y_test`) to check how accurate the model is.