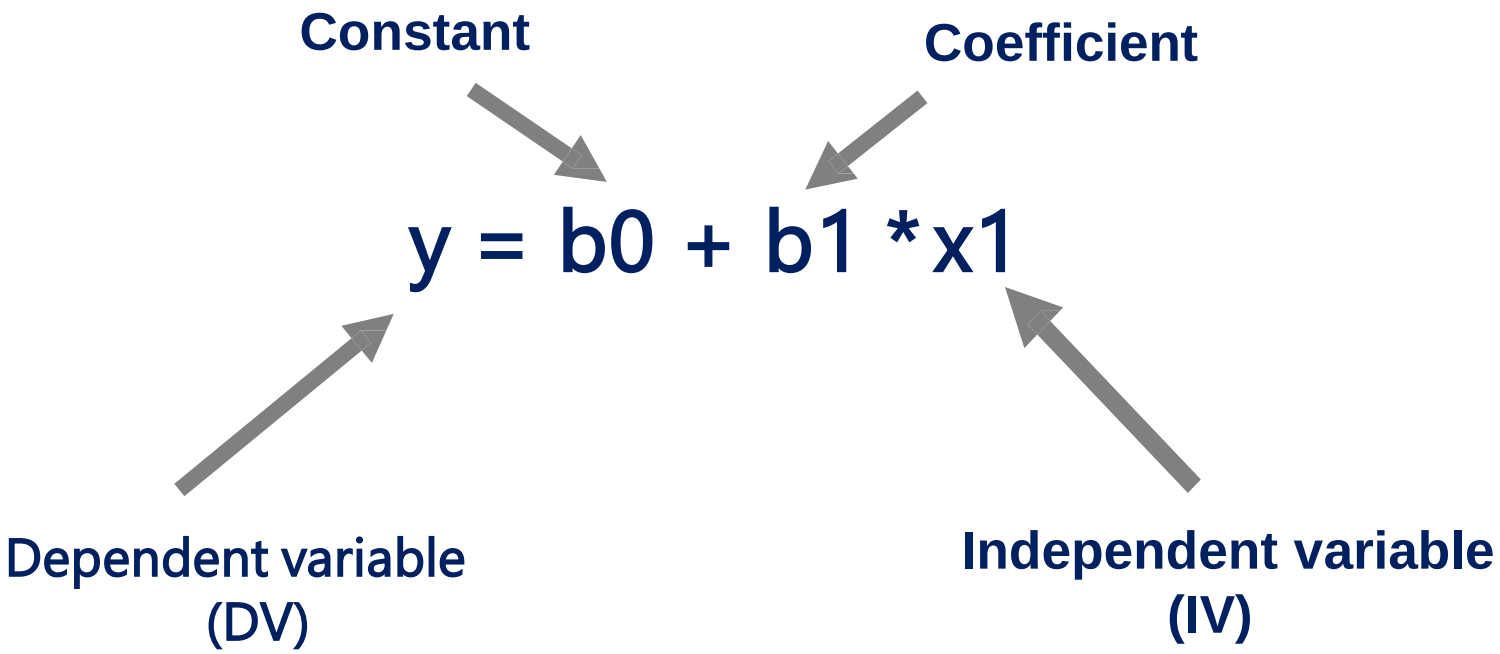
# EC3357:Machine Learning

## Lecture 3b: Multi-Linear Regression

# Simple Linear Regression

- Simple regression problem (a single x and a single y), the form of the model would be:




- **b0** (y-intercept) and **b1** (slope) are the coefficients whose values represent the accuracy of predicted values with the actual values.

# Motivation

- Single regression (i.e., with one IV) allows us to study the relationship between two variables only.
- However, in reality, we do not believe that only a single variable explains all the variation of the dependent variable.
- For example, in the scenario of IQ and income, we do not expect IQ only to explain income, but we expect that there are also other variables, such as years of education, to explain income.
- Hence, to make the model more realistic, it makes sense to include multiple independent variables in the regression.

# Examples

The following are situations where we can use multiple regression:

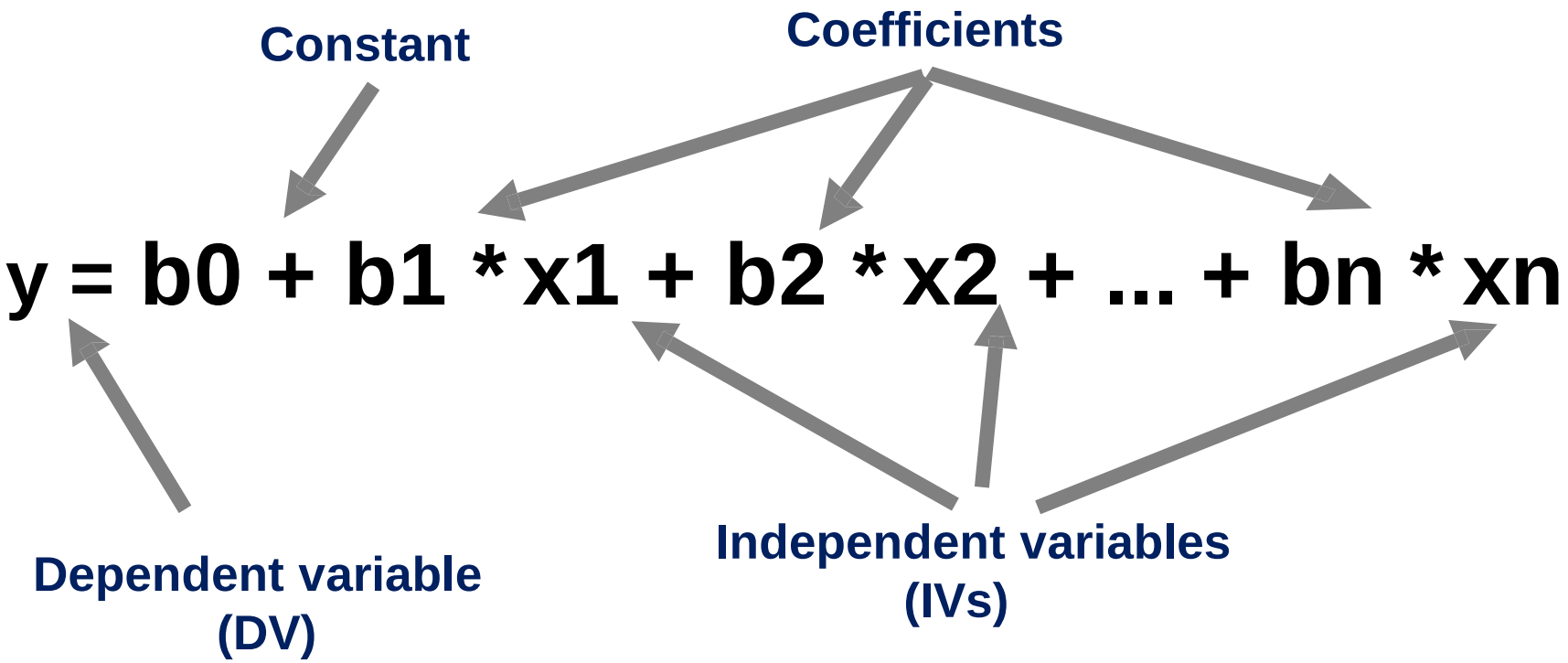
- Testing if IQ and level of education affect income (IQ and years of education are the IV and income is the DV).
- Testing if study time and pre-test scores affect final grades (DV is final grades, and study time and pre-test scores are the IV).
- Testing if exercise and amount of salt in the diet affect blood pressure (exercise and salt are the IV and blood pressure is the DV).

# More than one prediction attribute

- X1, X2
- For example,
  - X1=‘years of experience’
  - X2=‘age’
  - Y=‘salary’
- Equation:

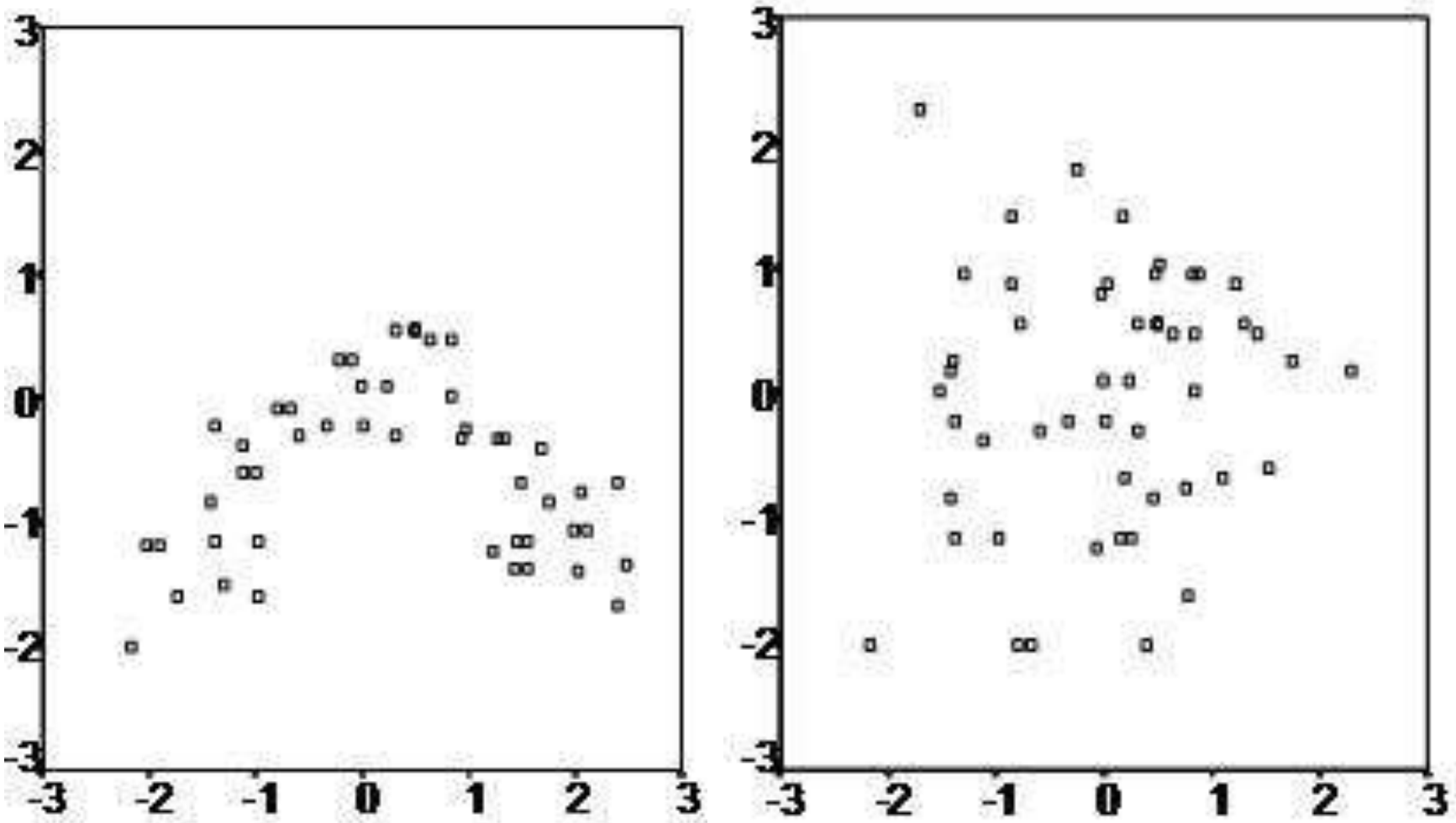
$$Y = \alpha + \beta_1 x_1 + \beta_2 x_2$$

# Multi Linear Regression



- Multiple linear regression requires at least two independent variables, which can be nominal, ordinal, or interval/ratio level variables.
- A rule of thumb for the sample size is that regression analysis requires at least 20 cases per independent variable in the analysis.
- **Multiple linear regression requires the relationship between the independent and dependent variables to be linear.**
- The linearity assumption can best be tested with scatterplots. The following two examples depict a curvilinear relationship (left) and a linear relationship (right).
- **Multiple linear regression assumes that there is no multicollinearity in the data.**
- **Multicollinearity occurs when the independent variables are too highly correlated with each other.**

curvilinear relationship (left) and a linear relationship (right).

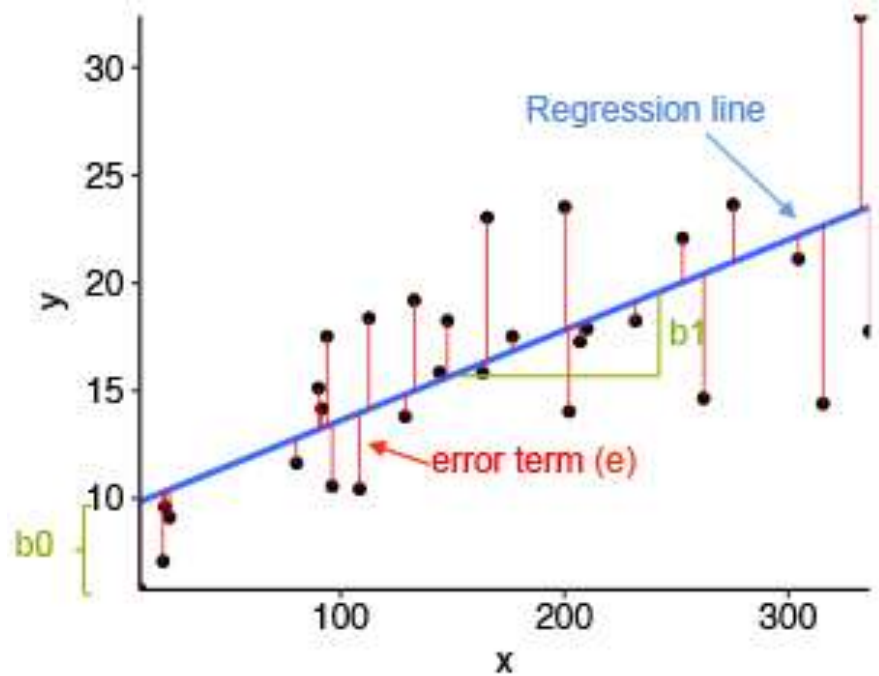


# Linear Regression

- Linear Regression is a statistical/machine learning technique that attempts to model the linear relationship between the independent predictor variables X and a dependent quantitative response variable Y. It is important that the predictor and response variables be numerical values. A general linear regression model can be represented mathematically as

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

# Example



# Example (Cont..)

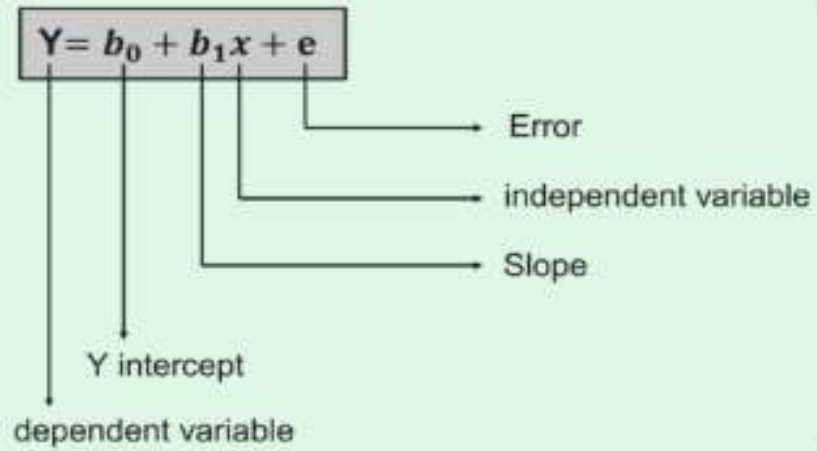
- In the previous diagram,
  - x is our independent variable which is plotted on the x-axis and y is the dependent variable which is plotted on the y-axis.
  - Black dots are the data points i.e the actual values.
  - $b_o$ is the intercept which is 10 and $b_1$ is the slope of the x variable.
  - The blue line is the best fit line predicted by the model i.e the predicted values lie on the blue line.
  - The vertical distance between the data point and the regression line is known as error or residual**.** Each data point has one residual and the sum of all the differences is known as the Sum of Residuals/Errors**.**

# Assumptions of Linear Regression

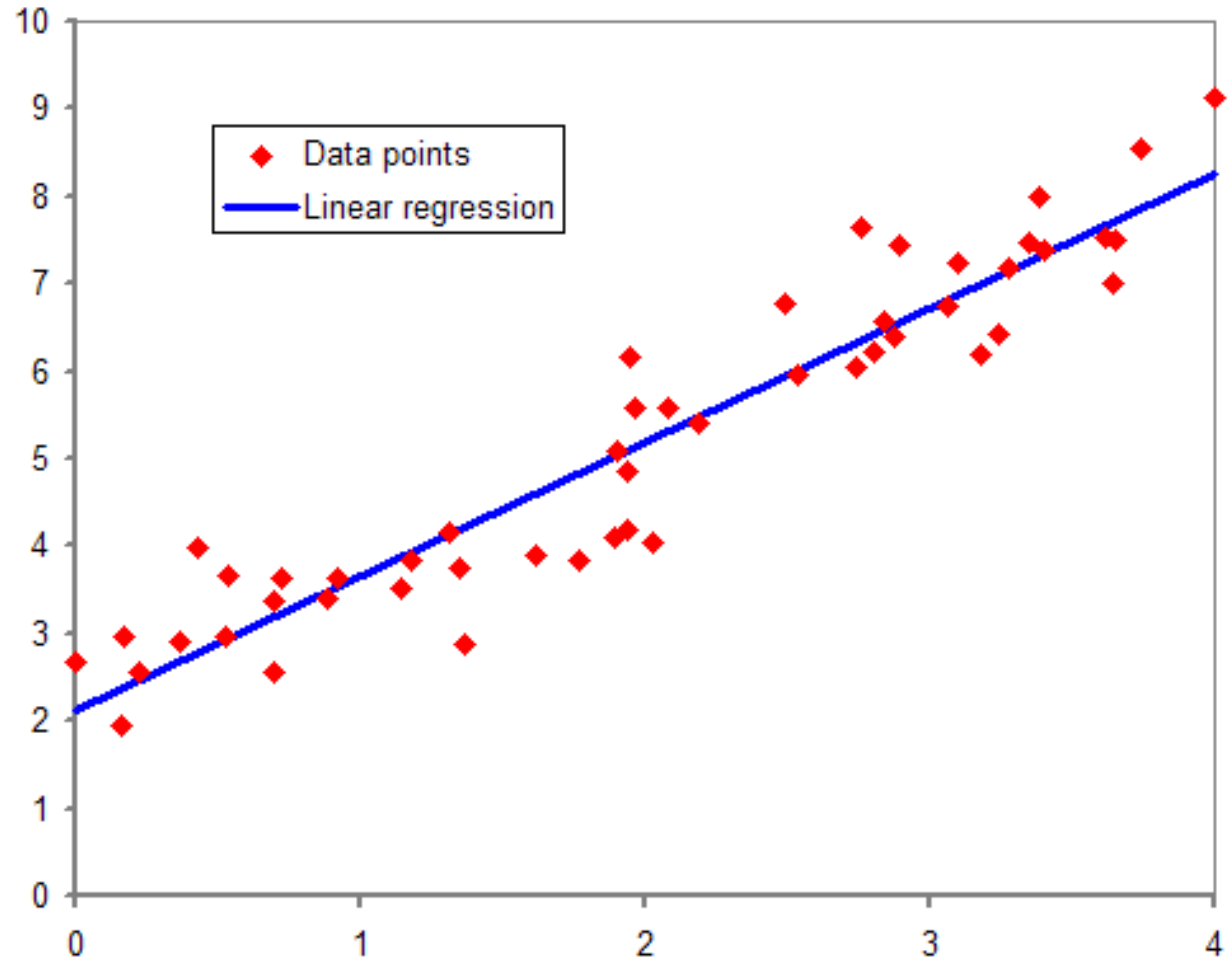
- Linear regression assumes that…
  - 1. The relationship between X and Y is linear
  - 2. Y is distributed normally at each value of X
  - 3. The variance of Y at every value of X is the same (homogeneity of variances)
  - 4. The observations are independent

# Linear Regression Terminologies: Cost Function

- The best fit line can be based on the linear equation given below.




- The dependent variable that is to be predicted is denoted by Y.
- A line that touches the y-axis is denoted by the intercept $b_0$.
- $b_1$ is the slope of the line, x represents the independent variables that determine the prediction of Y.
- The error in the resultant prediction is denoted by e.


Data points
Linear regression
10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4

- While training and building a regression model, it is these coefficients which are learned and fitted to training data. The aim of the training is to find the best fit line such that cost function is minimized. The cost function helps in measuring the error. During the training process, we try to minimize the error between actual and predicted values and thus minimizing the cost function.
- In the figure, the red points are the data points and the blue line is the predicted line for the training data. To get the predicted value, these data points are projected on to the line.

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

We choose the function above to minimize the error. We square the error difference and sum the error over all data points, the division between the total number of data points. Then, the produced value provides the averaged square error over all data points.

It is also known as MSE(Mean Squared Error), and we change the values of $b_0$ and $b_1$ so that the MSE value is settled at the minimum.

# Cost function

- A cost function, also known as a loss function or error function, is a mathematical function used to quantify the error or difference between the predicted values and the actual values in a machine learning model.
- The primary goal of training a machine learning model is to minimize the cost function, thereby improving the model's accuracy.
- The cost function helps to figure out the best possible values for a0 and a1, which provides the best fit line for the data points.

# Cost function

- **Mean Squared Error (MSE)** cost function is used, which is the average of squared error that occurred between the predicted values and actual values.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

*yi* is the ith observed value.

*ŷi* is the corresponding predicted value.

*n* = the number of observations.

# Gradient Descent

- The next important terminology to understand linear regression is **gradient descent.** It is a method of updating $b_0$ and $b_1$ values to reduce the MSE. The idea behind this is to keep iterating the $b_0$ and $b_1$ values until we reduce the MSE to the minimum.
- To update $b_0$ and $b_1$, we take gradients from the cost function. To find these gradients, we take partial derivatives with respect to $b_0$ and $b_1$. These partial derivatives are the gradients and are used to update the values of $b_0$ and $b_1$.

# Advantages And Disadvantages

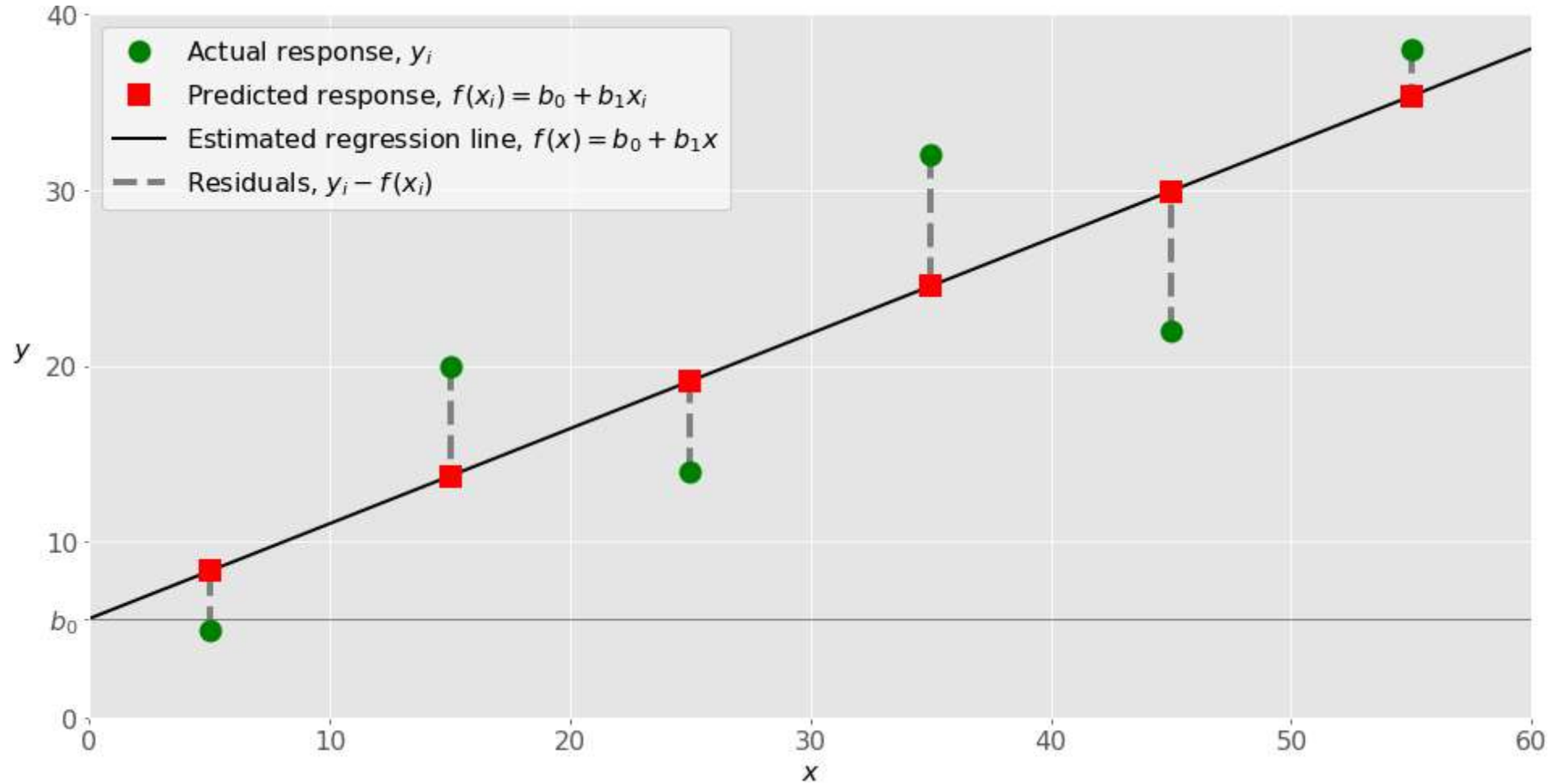
| Advantages | Disadvantages |
| --- | --- |
| Linear regression performs exceptionally well for linearly separable data | The assumption of linearity between dependent and independent variables |
| Easier to implement, interpret and efficient to train | It is often quite prone to noise and overfitting |
| It handles overfitting pretty well using dimensionally reduction techniques, regularization, and cross-validation | Linear regression is quite sensitive to outliers |
| One more advantage is the extrapolation beyond a specific data set | It is prone to multicollinearity |

# Applications of Simple Linear Regression

- Marks scored by students based on number of hours studied (ideally)- Here marks scored in exams are independent and the number of hours studied is independent.

- Predicting crop yields based on the amount of rainfall- Yield is a dependent variable while the measure of precipitation is an independent variable.

- Predicting the Salary of a person based on years of experience- Therefore, Experience becomes the independent while Salary turns into the dependent variable.


Actual response, $y_i$
Predicted response, $f(x_i) = b_0 + b_1 x_i$
Estimated regression line, $f(x) = b_0 + b_1 x$
Residuals, $y_i - f(x_i)$
40
30
20
10
$b_0$
0
$y$
0
10
20
30
40
50
60
$x$

- When implementing simple linear regression, you typically start with a given set of input-output ($x$-$y$) pairs. These pairs are your observations, shown as green circles in the figure. For example, the leftmost observation has the input $x$ = 5 and the actual output, or response, $y$ = 5. The next one has $x$ = 15 and $y$ = 20, and so on.
- The estimated regression function, represented by the black line, has the equation $f(x) = b_0 + b_1x$. Your goal is to calculate the optimal values of the predicted weights $b_0$ and $b_1$ that minimize MSE and determine the estimated regression function.
- The value of $b_0$, also called the **intercept**, shows the point where the estimated regression line crosses the $y$ axis. It's the value of the estimated response $f(x)$ for $x$ = 0. The value of $b_1$ determines the **slope** of the estimated regression line.

- The predicted responses, shown as red squares, are the points on the regression line that correspond to the input values. For example, for the input $x$ = 5, the predicted response is $f(5)$ = 8.33, which the leftmost red square represents.
- The vertical dashed grey lines represent the residuals, which can be calculated as $y_i - f(\mathbf{x}_i) = y_i - b_0 - b_1 x_i$ for $i$ = 1, ..., $n$. They're the distances between the green circles and red squares. When you implement linear regression, you're actually trying to minimize these distances and make the red squares as close to the predefined green circles as possible.

- https://medium.com/machine-learning-with-python/multiple-linear-regression-implementation-in-python-2de9b303fc0c
- https://medium.com/the-code-monster/split-a-dataset-into-train-and-test-datasets-using-sk-learn-acc7fd1802e0

# Example-1

- Predicting sales based on the money spent on TV, Radio, and Newspaper for marketing. In this case, there are three independent variables, i.e., money spent on TV, Radio, and Newspaper for marketing, and one dependent variable, i.e., sales, that is the value to be predicted.
- Problem statement: Build a Multiple Linear Regression Model to predict sales based on the money spent on TV, Radio, and Newspaper for advertising.

# Importing the Libraries

```
#Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

# Reading the Dataset

```
#Reading the dataset
dataset = pd.read_csv("advertising.csv")
dataset.head()
```

| | TV | Radio | Newspaper | Sales |
|---|---|---|---|---|
| **0** | 230.1 | 37.8 | 69.2 | 22.1 |
| **1** | 44.5 | 39.3 | 45.1 | 10.4 |
| **2** | 17.2 | 45.9 | 69.3 | 12.0 |
| **3** | 151.5 | 41.3 | 58.5 | 16.5 |
| **4** | 180.8 | 10.8 | 58.4 | 17.9 |

**Equation:** Sales = β0 + (β1 * TV) + (β2 * Radio) + (β3 * Newspaper) + e

# Setting the values for independent (X) variable and dependent (Y) variable

```
#Setting the value for X and Y
x = dataset[['TV', 'Radio', 'Newspaper']]
y = dataset['Sales']
```

# Splitting the dataset into train and test set

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 100)
```

- from sklearn.model_selection import train_test_split: It is used for splitting data arrays into two subsets: for training data and testing data.
- test_size: This parameter specifies the size of the testing dataset. The default state suits the training size. It will be set to 0.25 if the training size is set to default.

# Implementing the linear model

```
#Fitting the Multiple Linear Regression model
mlr = LinearRegression()
mlr.fit(x_train, y_train)
```

To build a linear regression model, we need to create an instance of LinearRegression() class and use x_train, y_train to train the model using the fit() method of that class. Now, the variable mlr is an instance of the LinearRegression() class.

# Model Equation

```
#Intercept and Coefficient
print("Intercept: ", mlr.intercept_)
print("Coefficients:")
list(zip(x, mlr.coef_))
```

```
Intercept:  4.334595861728433


Coefficients:

[('TV', 0.05382910866725006),
 ('Radio', 0.11001224388558055),
 ('Newspaper', 0.006289950146130348)]
```

# Regression Equation

Sales = 4.3345+ (0.0538 * TV) + (1.1100* Radio) + (0.0062 *Newspaper) + e

# Prediction on the test set

```
#Prediction of test set
y_pred_mlr= mlr.predict(x_test)
#Predicted values
print("Prediction for test set: {}".format(y_pred_mlr))
```

```
Prediction for test set: [ 9.35221067 20.96344625 16.48851064 20.10971005 21.67148354 16.16054424
 13.5618056  15.39338129 20.81980757 21.00537077 12.29451311 20.70848608
  8.17367308 16.82471534 10.48954832  9.99530649 16.34698901 14.5758119
 17.23065133 12.56890735 18.55715915 12.12402775 20.43312609 17.78017811
 16.73623408 21.60387629 20.13532087 10.82559967 19.12782848 14.84537816
 13.13597397  9.07757918 12.07834143 16.62824427  8.41792841 14.0456697
  9.92050209 14.26101605 16.76262961 17.17185467 18.88797595 15.50165469
 15.78688377 16.86266686 13.03405813 10.47673934 10.6141644  20.85264977
 10.1517568   6.88471443 17.88702583 18.16013938 12.55907083 16.28189561
 18.98024679 11.33714913  5.91026916 10.06159509 17.62383031 13.19628335]
```

# Actual values and the predicted values

#Actual value and the predicted value

mlr_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred_mlr})

slr_diff.head()

| | Actual value | Predicted value |
|---|---|---|
| **126** | 6.6 | 9.352211 |
| **104** | 20.7 | 20.963446 |
| **99** | 17.2 | 16.488511 |
| **92** | 19.4 | 20.109710 |
| **111** | 21.8 | 21.671484 |

# Evaluating the Model

```
#Model Evaluation
from sklearn import metrics
meanAbErr = metrics.mean_absolute_error(y_test, y_pred_mlr)
meanSqErr = metrics.mean_squared_error(y_test, y_pred_mlr)
rootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_mlr))
print('R squared: {:.2f}'.format(mlr.score(x,y)*100))
print('Mean Absolute Error:', meanAbErr)
print('Mean Square Error:', meanSqErr)
print('Root Mean Square Error:', rootMeanSqErr)
```

# Evaluation Metrics

R squared: 90.11
Mean Absolute Error: 1.2278183566589418
Mean Square Error: 2.6360765623280673
Root Mean Square Error: 1.6235998775338913