REGRESSION

By

Riyaz

Regression

- Regression is a technique to model the relationship between variables.
- It predicts **continuous outcomes** based on input (independent) features.
- Commonly used in **machine learning**, **statistics**, and **data analysis**.
- The main goal is to **understand** and **predict** the behavior of a target variable.
- Example: Predicting house price based on size and location.

Regression

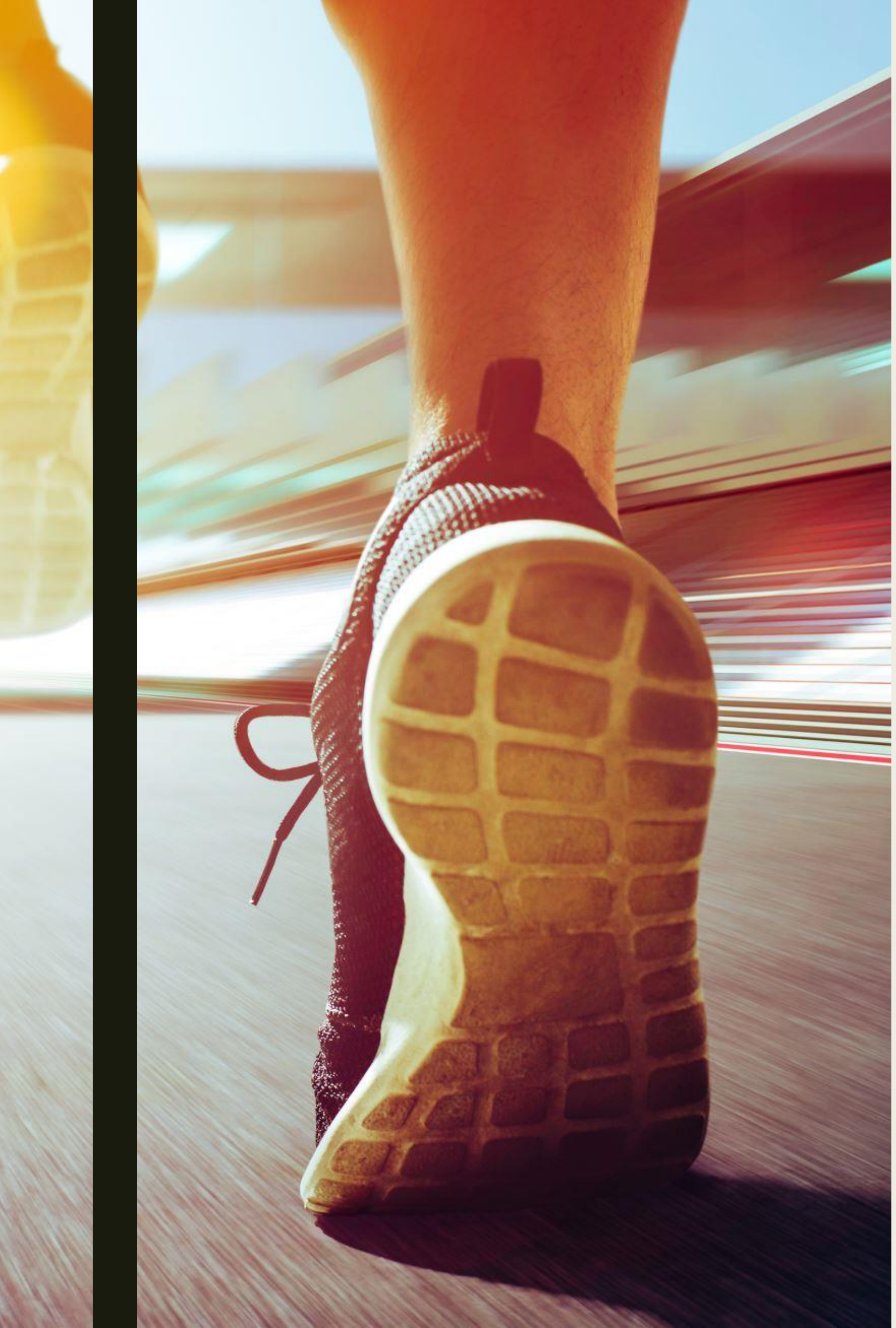
- The term *regression* refers to a method used to determine the relationship between variables.
- In machine learning and statistical modeling, this relationship helps predict the outcome of future events.
- Regression involves a set of techniques used to predict a response variable (also known as the dependent, criterion, or outcome variable) using one or more predictor variables (also called independent or explanatory variables).
- It can identify which explanatory variables are related to the response variable, describe the nature of those relationships, and provide an equation to make future predictions.

Why Learn Regression? (Importance)

- Forms the **foundation of many machine learning models**.
- Used in **real-world decisions** – finance, marketing, healthcare, etc.
- Helps make **data-driven predictions** and forecasts.
- Aids in **understanding relationships** between features and outcomes.
- Boosts your ability to **analyze trends** and make smarter business decisions.

Example - Regression in Exercise Physiology

- An exercise physiologist might use regression analysis to develop an equation for predicting the number of calories a person is expected to burn while exercising on a treadmill.
- The **response variable** is the number of calories burned (calculated from the amount of oxygen consumed), and the **predictor variables** might include:-
  - *Duration of exercise (in minutes)*
  - *Percentage of time spent at the target heart rate*
  - *Average speed (in mph)*
  - *Age (in years)*
  - *Gender*
  - *Body Mass Index (BMI)*



Theoretical Perspective of the Analysis

- From a theoretical point of view, the analysis can help answer questions such as:-
  - ***What is the relationship between exercise duration and calories burned?***
    For example, does exercise have a diminishing impact on calorie burn after a certain duration?
  - ***How does effort factor in?***
    Consider variables like the percentage of time spent at the target heart rate or the average walking speed.
  - ***Are these relationships consistent across different groups?***
    For instance, do they vary between young and old, male and female, or individuals with different body compositions (e.g., heavy vs. slim)?

Using Regression to Answer Real-World Questions

- From a practical point of view, the analysis can help answer questions such as:-

  - ***How many calories can a 30-year-old man with a BMI of 28.7 expect to burn*** *if he walks for 45 minutes at an average speed of 4 miles per hour and stays within his target heart rate 80% of the time?*

  - ***What is the minimum number of variables needed*** *to accurately predict the number of calories a person will burn while walking?*

  - ***How accurate will the prediction be*** *on average?*

  - *Regression analysis plays a central role in both modern statistics and machine learning. Effective regression analysis is a* ***comprehensive, iterative process*** *that requires not just technical steps, but also a good deal of skill and judgment.*

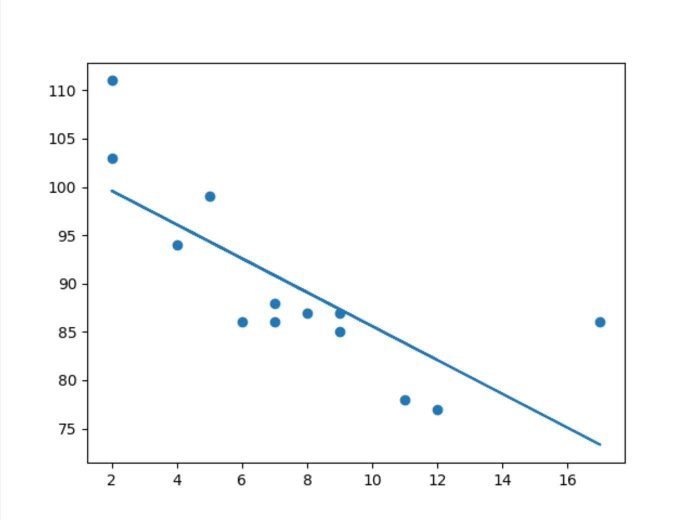
Types of Regression

- **Linear Regression** – Predicts using a straight-line relationship.
- **Multiple Linear Regression** – Uses multiple input variables.
- **Polynomial Regression** – Models curved, non-linear relationships.
- **Logistic Regression** – For binary classification (e.g., yes/no outcomes).

Linear Regression

- Linear regression is a fundamental statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a straight line through the data points.

- This line, often referred to as the regression line or line of best fit, represents the equation:-
- The regression line can be used to predict future values based on new input data.
- In machine learning, linear regression is widely used for forecasting, trend analysis, and as a baseline model for many real-world problems due to its simplicity and interpretability.



Linear Regression

- $y = \beta_0 + \beta_1 x + \varepsilon$

where:-

- - y is the predicted value (dependent variable)
- - x is the input (independent variable)
- - β_0 is the intercept
- - β_1 is the slope (coefficient)
- - ε is the error term

How Does it Work?
Using Linear Regression in Python

Python provides built-in methods to identify relationships between data points and to draw a **linear regression line**—without needing to manually apply the mathematical formula.

In the example below:

- *The **x-axis** represents **age***
- *The **y-axis** represents **speed***
- *We recorded the age and speed of **13 cars** as they passed through a tollbooth.*

- Can we use this data to find a pattern and predict outcomes?

How Does it Work?

- Example

Start by drawing a scatter plot:

```
import matplotlib.pyplot as plt
```

X= Age and Y = speed

```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```

How Does it Work?

- Import scipy and draw the line of Linear Regression:

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err =
stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

Example Explained

- Import the modules you need.

```
import matplotlib.pyplot as plt
from scipy import stats
```

- Create the arrays that represent the values of the x and y axis:

```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
```

- Execute a method that returns some important key values of Linear Regression:

```
slope, intercept, r, p, std_err =
stats.linregress(x, y)
```

- Create a function that uses the slope and intercept values to return a new value. This new value represents where on the y-axis the corresponding x value will be placed:

```
def myfunc(x):
  return slope * x + intercept
```

Example Explained

- Run each value of the x array through the function. This will result in a new array with new values for the y-axis:

```
mymodel = list(map(myfunc, x))
```

- Draw the original scatter plot:

```
plt.scatter(x, y)
```

- Draw the line of linear regression:

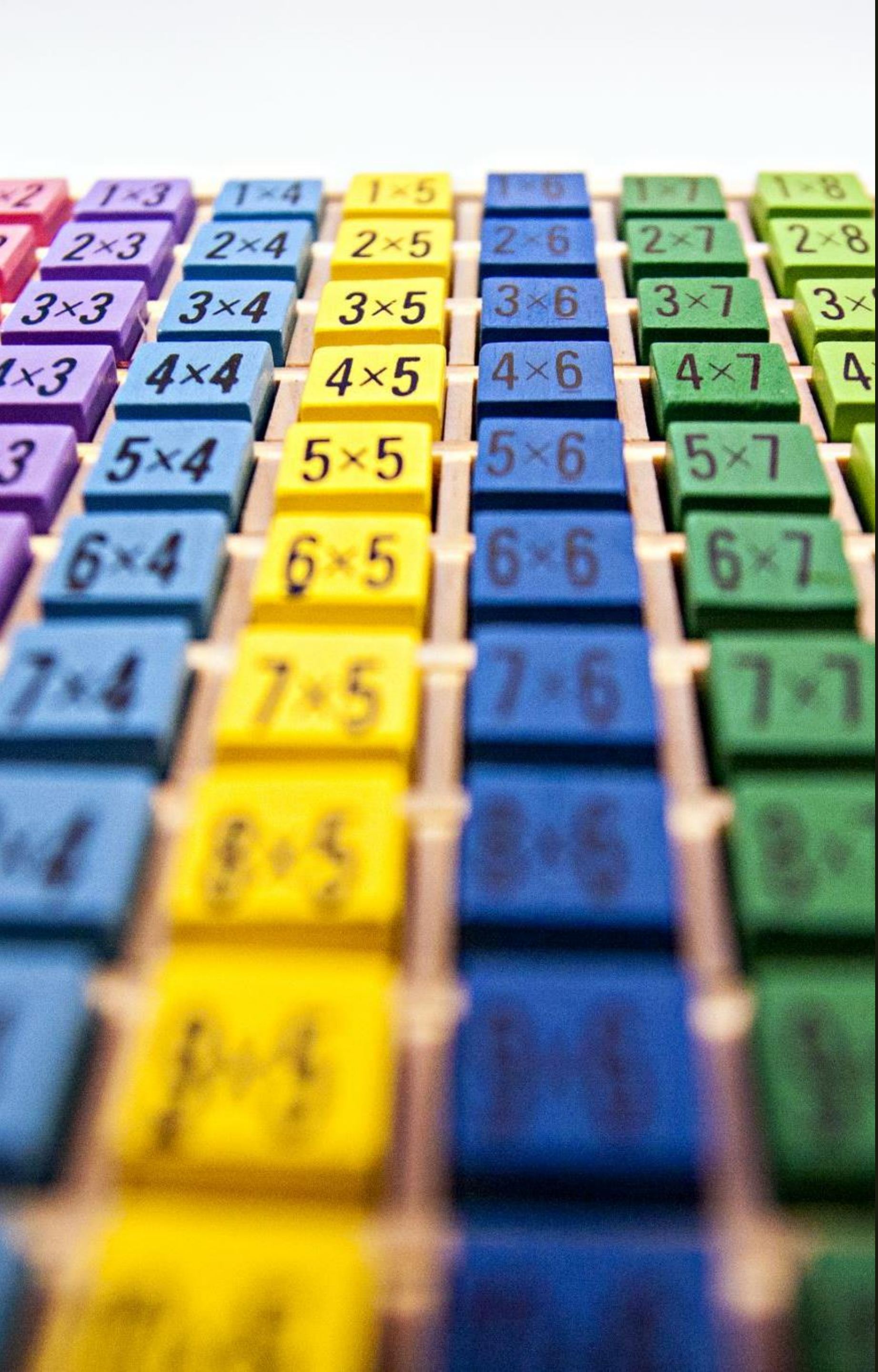
```
plt.plot(x, mymodel)
```

- Display the diagram

```
plt.show()
```

Linear Regression Results Parameters

PARAMETER	DESCRIPTION
Coefficients	Estimated regression coefficients for each independent variable in the model
Standard Error	Measures the variability of the estimate of each coefficient
t-Statistic	Measures how many standard errors the coefficient estimate is away from zero
p-Value	Indicates the statistical significance of each coefficient estimate
R-squared	Measures the proportion of the variation in the dependent variable that is explained by the independent variables
Adjusted R-squared	Similar to R-squared, but adjusts for the number of independent variables in the model
F-statistic	Tests the overall significance of the model, i.e., whether any of the independent variables are useful
Residuals	Differences between the observed and predicted values of the dependent variable
Durbin-Watson Statistic	Measures the autocorrelation of the residuals, which should ideally be close to 2



SciPy Introduction

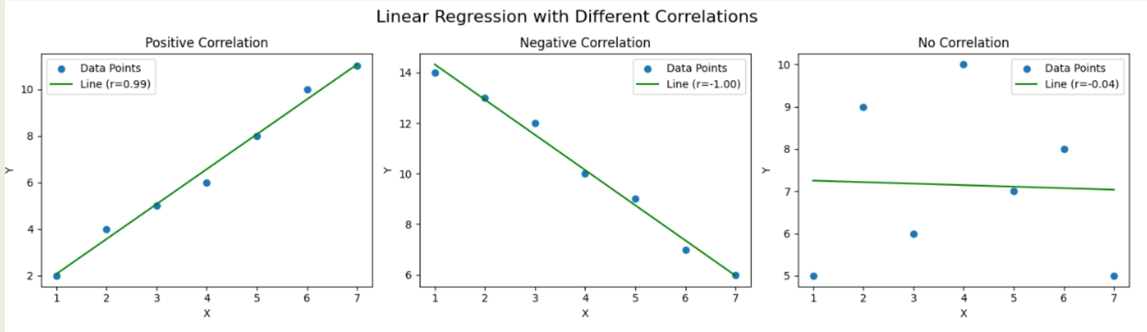
- **SciPy** (Scientific Python) is an open-source Python library used for **scientific and technical computing**.
- Built on top of **NumPy**, it adds powerful tools for tasks like **linear algebra, optimization, integration, and statistics**.
- Provides modules such as scipy.stats, scipy.optimize, scipy.integrate, and more.
- Commonly used in **data science, machine learning, engineering, and scientific research**.
- Helps simplify **complex mathematical computations** with ready-to-use functions and efficient algorithms.

Understanding the Correlation (r-value)

- It’s important to understand the relationship between **x-values** and **y-values** before using linear regression.
- If there is **no relationship**, linear regression cannot be used to make accurate predictions.
- This relationship is measured by the **correlation coefficient**, commonly denoted as **r**.
- The **r-value ranges from -1 to 1**:
  - **0** means no correlation
  - **1** or **-1** indicates a perfect correlation (positive or negative)
- Python, using the **SciPy module**, can calculate the r-value automatically—you just need to provide the x and y values.

Python Code

```python
# Import required libraries
import matplotlib.pyplot as plt  # For plotting graphs
from scipy import stats          # For performing linear regression

# Create 3 datasets with different types of correlation
datasets = {
    "Positive Correlation": {
        "x": [1, 2, 3, 4, 5, 6, 7],                 # Increasing x
        "y": [2, 4, 5, 6, 8, 10, 11]                # y increases with x
    },
    "Negative Correlation": {
        "x": [1, 2, 3, 4, 5, 6, 7],
        "y": [14, 13, 12, 10, 9, 7, 6]              # y decreases as x increases
    },
    "No Correlation": {
        "x": [1, 2, 3, 4, 5, 6, 7],
        "y": [5, 9, 6, 10, 7, 8, 5]                 # No clear pattern
    }
}

# Set up a figure with 3 subplots side by side
plt.figure(figsize=(15, 4))  # Create a figure window with width 15 and height 4

# Loop through each dataset
for i, (title, data) in enumerate(datasets.items(), 1):
    x = data["x"]  # Extract x values
    y = data["y"]  # Extract y values

    # Perform linear regression
    slope, intercept, r, p, std_err = stats.linregress(x, y)  # Compute regression stats

    # Calculate regression line values using y = slope * x + intercept
    regression_line = [slope * val + intercept for val in x]

    # Create subplot i (1 to 3)
    plt.subplot(1, 3, i)  # 1 row, 3 columns, position i
    plt.scatter(x, y, label="Data Points")  # Plot original data points
    plt.plot(x, regression_line, color='green', label=f"Line (r={r:.2f})")  # Plot regression line
    plt.title(title)  # Set subplot title (correlation type)
    plt.xlabel("X")  # X-axis label
    plt.ylabel("Y")  # Y-axis label
    plt.legend()  # Show legend

# Adjust layout to prevent overlapping
plt.tight_layout()  # Automatically adjust spacing between plots

# Add a common title for the full figure
plt.suptitle("Linear Regression with Different Correlations", fontsize=16, y=1.05)

# Show the complete plot
plt.show()  # Display the figure
```



R for Relationship

- Example
- How well does my data fit in a linear regression?

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err =
stats.linregress(x, y)

print(r)
```

Predict Future Values

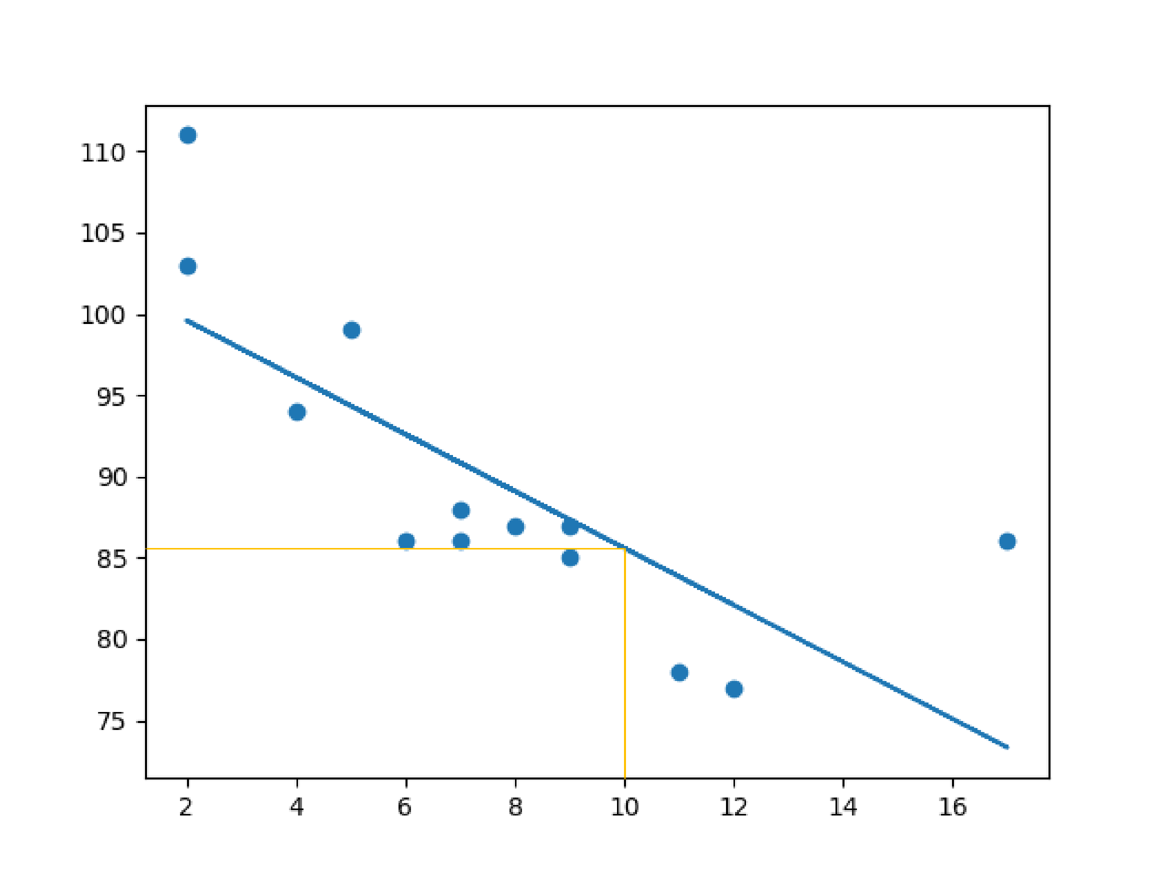
- Now we can use the information we have gathered to predict future values.
- Example: Let us try to predict the speed of a 10 years old car.
- To do so, we need the same myfunc() function from the previous example:

```
def myfunc(x):
  return slope * x + intercept
```

Predict Future Values

- Example
- Predict the speed of a 10 years old car:-

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err =
stats.linregress(x, y)

def myfunc(x):
  return slope * x + intercept

speed = myfunc(10)

print(speed)
```



Use Case - Linear Regression

Use Case: Predicting House Prices

- **Inputs:** Square footage, number of rooms, age of house
- **Output:** Predicted price of the house
- Easy to interpret and visualize
- Assumes a linear relationship between features and price
- Good baseline model for continuous prediction tasks

Multiple Regression

- Multiple Regression is a method to model the relationship between one dependent variable and two or more independent variables.
- It extends simple linear regression to include multiple predictors for better accuracy.
- General equation:- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon$
- Used in fields like economics, healthcare, and marketing to analyze multiple influences.
- Helps answer questions like: 'How do age, income, and education affect spending habits?'

Use Case - Multiple Linear Regression

Use Case: Sales Forecasting for a Store

- **Inputs:** Advertising budget, number of staff, store size, location score
- **Output:** Predicted monthly sales
- Captures effects of multiple factors on sales
- More flexible than simple linear regression
- Useful for strategic business decisions

Multiple Linear Regression

- This helps improve prediction accuracy by including more influencing factors.
- Useful when a single variable doesn’t capture the whole picture.
- Example: Predicting car CO_2 emissions using both engine size and weight.
- Sample dataset includes car model, engine volume, weight, and CO_2 emissions.
- We aim to predict CO_2 emissions based on volume and weight.
- More variables lead to more reliable and realistic predictions.
- This is a classic use case of multiple regression in action.

Simple Linear vs Multiple Linear Regression

Criteria	Simple Linear Regression	Multiple Linear Regression
Definition	Linear relationship between dependent and independent variables	Linear relationship between dependent variable and multiple independent variables
Number of independent variables	1	2 or more
Purpose	Predicting the value of a dependent variable based on one independent variable	Predicting the value of a dependent variable based on multiple independent variables
Equation	y = mx + b	y = b0 + b1x1 + b2x2 + ... + bnxn
Model complexity	Less complex	More complex
Accuracy	Low accuracy for predicting complex outcomes	Higher accuracy for predicting complex outcomes
Use cases	Examining the relationship between two variables, such as income and spending	Predicting housing prices based on multiple variables like location, square footage, and number of bedrooms and bathrooms

How Does it Work?

- In Python we have modules that will do the work for us. Start by importing the Pandas module.

```
import pandas

df = pandas.read_csv("cars.csv")
```

Then make a list of the independent values and call this variable X.

Put the dependent values in a variable called y.

```
X = df[['Weight', 'Volume']]
y = df['CO2']
```

How Does it Work?

- In Python we have modules that will do the work for us. Start by importing the Pandas module.

```
import pandas

df = pandas.read_csv("cars.csv")
```

Then make a list of the independent values and call this variable X.

Put the dependent values in a variable called y.

```
X = df[['Weight', 'Volume']]
y = df['CO2']
```

It is common to name the list of independent values with a upper case X, and the list of dependent values with a lower-case y.

sklearn module

- We will use some methods from the sklearn module, so we will have to import that module as well:

```
from sklearn import linear_model
```

- From the sklearn module we will use the LinearRegression() method to create a linear regression object.
- This object has a method called fit() that takes the independent and dependent values as parameters and fills the regression object with data that describes the relationship:

```
regr = linear_model.LinearRegression()
regr.fit(X, y)
```

Now we have a regression object that are ready to predict CO2 values based on a car's weight and volume:

```
#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:
predictedCO2 = regr.predict([[2300, 1300]])
```

sklearn module

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm³:
predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
```

```
[107.2087328]
```

We have predicted that a car with 1.3 liter engine, and a weight of 2300 kg, will release approximately 107 grams of CO_2 for every kilometer it drives.

Coefficient

- The coefficient is a factor that describes the relationship with an unknown variable.

- Example: if x is a variable, then 2x is x two times. x is the unknown variable, and the number 2 is the coefficient.

- In this case, we can ask for the coefficient value of weight against CO_2, and for volume against CO_2.
- The answer(s) we get tells us what would happen if we increase, or decrease, one of the independent values.

Coefficient Example

- Print the coefficient values of the regression object:

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

print(regr.coef_)
```

Coefficient Example

- Print the coefficient values of the regression object:

Result:

[0.00755095 0.00780526]

Result Explained

The result array represents the coefficient values of weight and volume.
Weight: 0.00755095
Volume: 0.00780526

These values tell us that if the weight increase by 1kg, the CO2 emission increases by 0.00755095g.
And if the engine size (Volume) increases by 1 cm^3, the CO2 emission increases by 0.00780526 g.
I think that is a fair guess, but let test it!

Test

- Example
- Copy the example from before, but change the weight from 2300 to 3300:

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

predictedCO2 = regr.predict([[3300, 1300]])

print(predictedCO2)
```

```
[114.75968007]
```

We have predicted that a car with 1.3 liter engine, and a weight of 3300 kg, will release approximately 115 grams of CO_2 for every kilometer it drives.

Which shows that the coefficient of 0.00755095 is correct:

107.2087328 + (1000 * 0.00755095) = 114.75968

Polynomial Regression

- Polynomial Regression models the relationship as an nth-degree polynomial.
- Useful when data shows a non-linear trend that a straight line can't capture.
- Equation: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$
- Fits curves and complex patterns better than linear regression.
- Used in fields like economics, biology, and engineering.

Use Case - Polynomial Regression

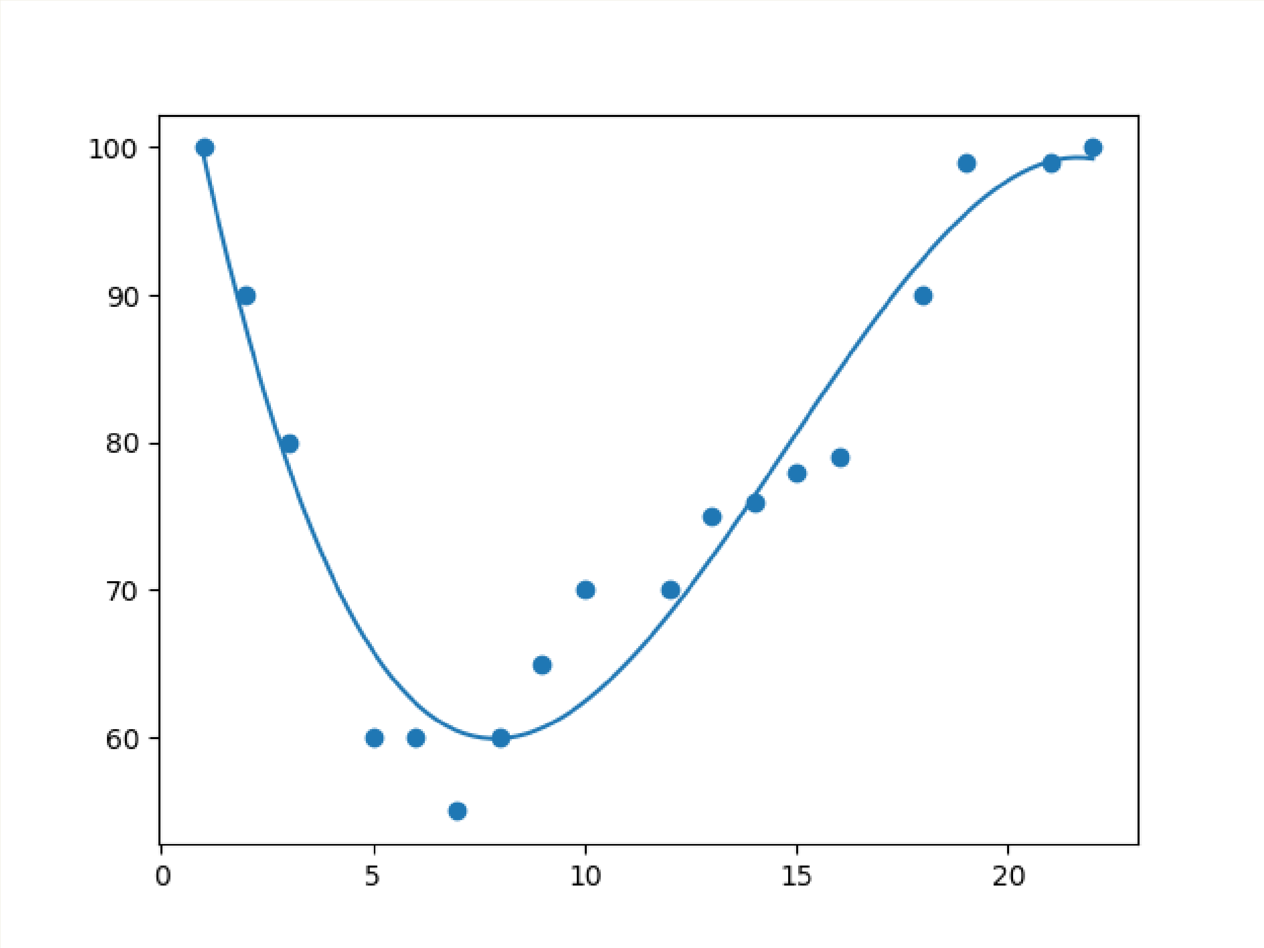
Use Case: Modeling Car Price Depreciation

- **Inputs:** Age of the car
- **Output:** Predicted resale value
- Captures the non-linear drop in value over time
- Fits curved trends better than linear models
- Often used in economics and asset management

Polynomial Regression

- If your data points clearly will not fit a linear regression (a straight line through all data points), it might be ideal for polynomial regression.

- Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.



How Does it Work?

- Python has methods for finding a relationship between data-points and to draw a line of polynomial regression.
- We will discuss how to use these methods instead of going through the mathematic formula.
- In the example below, we have registered 18 cars as they were passing a certain tollbooth.
- We have registered the car's speed, and the time of day (hour) the passing occurred.
- The x-axis represents the hours of the day and the y-axis represents the speed:

How Does it Work?

- Example
- Start by drawing a scatter plot:-

```
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

plt.scatter(x, y)
plt.show()
```

How Does it Work?

- Example
- Start by drawing a scatter plot:

  Import numpy and matplotlib then draw the line of Polynomial Regression:

```
import numpy
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(1, 22, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```

R-Squared

- It is important to know how well the relationship between the values of the x- and y-axis is, if there are no relationship the polynomial regression can not be used to predict anything.
- The relationship is measured with a value called the r-squared.
- The r-squared value ranges from 0 to 1, where 0 means no relationship, and 1 means 100% related.
- Python and the Sklearn module will compute this value for you, all you have to do is feed it with the x and y arrays:

Example

- How well does my data fit in a polynomial regression?

```
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

print(r2_score(y, mymodel(x)))
```

The result 0.94 shows that there is a very good relationship, and we can use polynomial regression in future predictions.

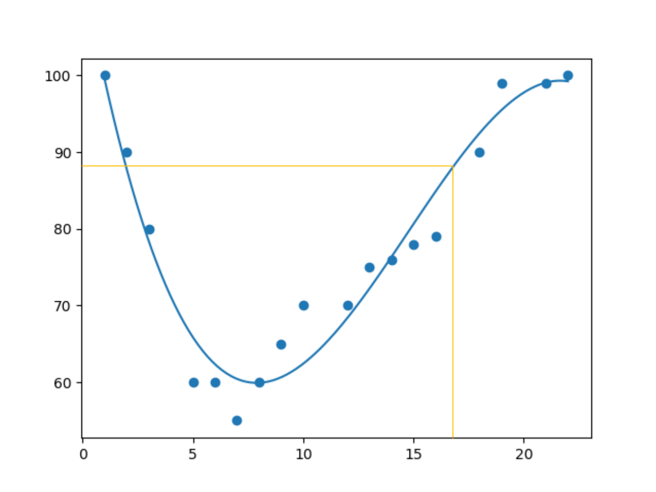
Predict Future Values

- Now we can use the information we have gathered to predict future values.
- Example: Let us try to predict the speed of a car that passes the tollbooth at around 17 P.M:-
- To do so, we need the same mymodel array from the example above:

Predict Future Values

- Example
- Predict the speed of a car passing at 17 P.M:

```
import numpy
from sklearn.metrics import r2_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

speed = mymodel(17)
print(speed)
```



The example predicted a speed to be 88.87, which we also could read from the diagram:

Logistic Regression

- Logistic Regression is a **classification algorithm** used to predict **binary outcomes** (e.g., Yes/No, 0/1).
- It estimates the **probability** that a given input belongs to a particular class.
- Instead of fitting a straight line, it uses a **sigmoid (S-shaped) curve** to map predictions to probabilities.
- Commonly used in **spam detection, customer churn, medical diagnosis**, and more.
- Output values are between **0 and 1**, which makes it suitable for **decision thresholds**.

Logistic regression

- It uses mathematics to find the relationships between two data factors.
- Based on these relationships, it can predict the value of one factor based on the other.
- The predictions from logistic regression usually have a finite number of outcomes, such as "yes" or "no."
- Logistic regression is commonly used in various fields, including medicine, economics, and marketing, to make predictions about binary outcomes.

Use Case - Logistic Regression

Use Case: Predicting Customer Churn

- **Inputs:** Tenure, usage behavior, support tickets, payment method
- **Output:** Probability of customer leaving (0 or 1)
- Though it's called regression, it performs classification
- Estimates probability using the logistic function
- Widely used in marketing and customer retention strategies

Logistic regression

Benefit	Description
Simplicity	Logistic regression models are simpler and require less mathematical complexity, making it easier to implement even without in-depth ML expertise.
Speed	Logistic regression models can process large volumes of data at high speed with less computational capacity, making them ideal for organizations starting with ML projects.
Flexibility	Logistic regression can be used to find answers to questions with two or more finite outcomes and to preprocess data for more accurate analysis with other ML techniques.
Visibility	Logistic regression analysis provides developers with greater visibility into internal software processes than other data analysis techniques, making troubleshooting and error correction easier with less complex calculations.

Linear regression vs logistic regression

Linear Regression	Logistic Regression
Used for continuous data	Used for categorical data
Outcome variable is continuous	Outcome variable is binary or nominal
Measures the strength of relationship between dependent and independent variables	Measures the probability of an event occurring
Example: predicting house prices based on square footage and number of rooms	Example: predicting whether a customer will churn or not based on their purchase history
Can be used for both simple and multiple regression analysis	Mainly used for binary logistic regression, but can also be used for multinomial and ordinal logistic regression