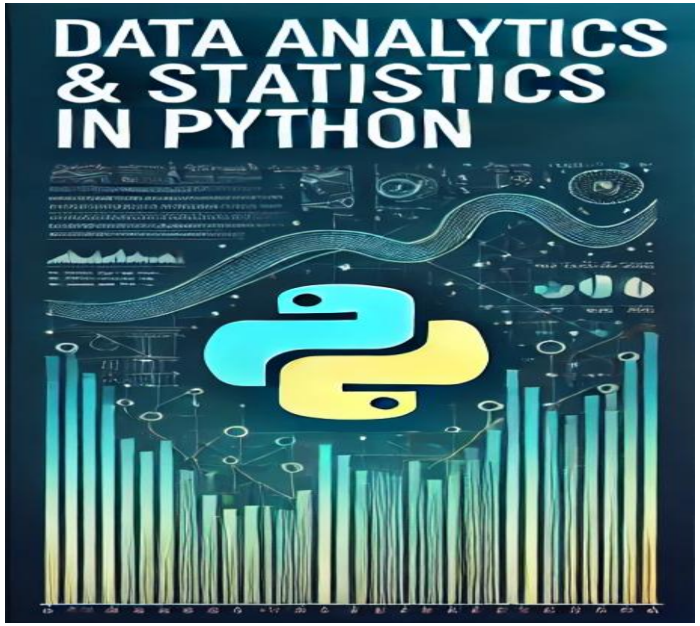
# Data Analytics & Statistics in Python
# Session 5: Relationships Between Variables




Metropolia
University of Applied Sciences

***Learning data-driven decision-making with Python***

**Instructor:** Hamed Ahmadinia, Ph.D.

**Email:** hamed.ahmadinia@metropolia.fi

# Concepts of Today

- **Key Topics:**
  1. Relationships Between Variables
  2. Covariance (How two variables move together)
  3. Correlation (How strong the relationship is)
  4. Linear Regression (Predicting outcomes based on input variables)

# Understanding Covariance

- **What is Covariance?**
  - **Covariance** shows how two variables move together.
    - **Positive Covariance**: Both variables increase or decrease together.
    - **Negative Covariance**: When one increases, the other decreases.
    - **Zero Covariance**: No relationship between the variables.
- **Example:**
  - **Temperature and ice cream sales**: High positive covariance (both increase in summer).
  - **Temperature and hot drink sales**: Likely negative covariance (hot drinks decrease as temperatures rise).

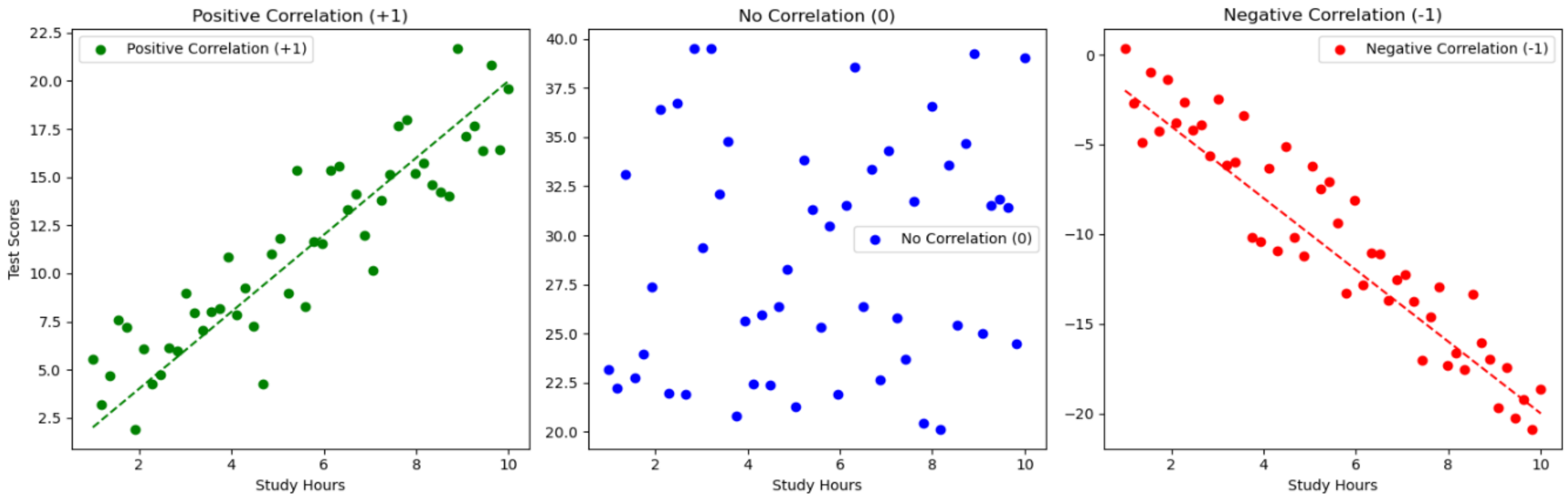
# Calculate covariance - NumPy

```python
import numpy as np

# Sample data
x = [1, 2, 3, 4, 5]  # Example variable X
y = [2, 4, 6, 8, 10]  # Example variable Y

# Calculate covariance matrix
cov_matrix = np.cov(x, y)

# Extract covariance between X and Y
cov_xy = cov_matrix[0, 1]  # [0, 1] gives covariance of X and Y
print(f"Covariance between X and Y: {cov_xy:.2f}")
```

```
Covariance between X and Y: 5.00
```

# Understanding Correlation

- **Definition:** Correlation measures the strength and direction of the relationship between two variables.
- **Range of Correlation Coefficient (r):**
  - **+1**: Strong positive correlation (as one increases, the other increases).
  - **0**: No correlation.
  - **-1**: Strong negative correlation (as one increases, the other decreases).




**Example**:

**Scenario**: Imagine studying the relationship between **daily study hours** (X) and **test scores** (Y) among students.

**+1 Correlation**: Students who study more hours consistently get higher scores (perfect upward trend).

**0 Correlation**: No connection between study time and scores (scattered points with no pattern).

**-1 Correlation**: Students who study more somehow perform worse in tests (perfect downward trend).

# Different Correlation Metrics

- **Pearson Correlation**: Measures linear relationships (sensitive to outliers).
- **Spearman Rank Correlation**: Works with ranked data and non-linear relationships.
- **Kendall's Tau**: Preferred when the dataset is small and ordinal.

| Correlation | When to use | Example |
|---|---|---|
| Pearson | Continuous, normally distributed data | Test relationship between height and weight |
| Spearman | Non-linear, ranked data | Examining customer satisfaction ratings |
| Kendall | Small datasets | Comparing student rankings across two exams |

# Calculate Correlation - NumPy

```python
import pandas as pd
import numpy as np

# Sample data: Study Hours vs Test Scores
data = {
    'Study Hours': [1, 2, 3, 4, 5, 6, 7, 8, 9],
    'Test Scores': [10, 20, 25, 40, 50, 55, 70, 80, 90]
}

df = pd.DataFrame(data)

# Calculate correlations
pearson_corr = df.corr(method='pearson')['Study Hours']['Test Scores']
spearman_corr = df.corr(method='spearman')['Study Hours']['Test Scores']
kendall_corr = df.corr(method='kendall')['Study Hours']['Test Scores']

print(f"Pearson Correlation: {pearson_corr:.2f}")
print(f"Spearman Correlation: {spearman_corr:.2f}")
print(f"Kendall Correlation: {kendall_corr:.2f}")
```

# Linear Regression

- **Linear regression:** helps predict an outcome ($y$) based on one or more input variables ($x$). It tries to fit a straight line or a curve to the data to explain the relationship between variables.
  - **Example:**
    - **Scenario:** You want to predict your monthly salary based on the number of sales made.
    - **Formula**: $y = b_0 + b_1 \cdot x$
      - $b_0$ (Intercept): Base salary when no sales are made (e.g., €1000).
      - $b_1$ (Slope): Salary increase for each sale (e.g., €50 per sale).

- If you make 10 sales:
- Salary: $y = 1000 + 50 \cdot 10$

(Your salary is €1500 for 10 sales.)

# Multivariate Regression

- **Multivariate Linear Regression:** When you predict $y$ using more than one input variable:
  - **Example:** Predicting house prices: $y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots + b_n \cdot x_n$
  - $x_1$ : Size of the house
  - $x_2$ : Number of rooms
  - $x_3$ : Location

*(Each feature affects the price differently. A bigger house might increase the price, while location could have an even bigger effect.)*

# Evaluating Model Performance

- **Train, Validation, and Test Sets:**
  1. **Train Set**: Used to train the model (majority of data, 60–80%).
  2. **Validation Set**: Used to tune the model (optional for linear regression).
  3. **Test Set**: Used to evaluate final performance (10–20%).

- **Metrics for Evaluation**:

1. **MSE (Mean Squared Error)**: Average squared error.
2. $R^2$ **Score**: Measures how well the model explains the variance:

$$R^2 = 1 - \frac{\text{Sum of Squared Residuals}}{Total\ Sum\ of\ Squares}$$

   1. $R^2 =$ 1: Perfect fit
   2. $R^2 =$ 0 : No correlation

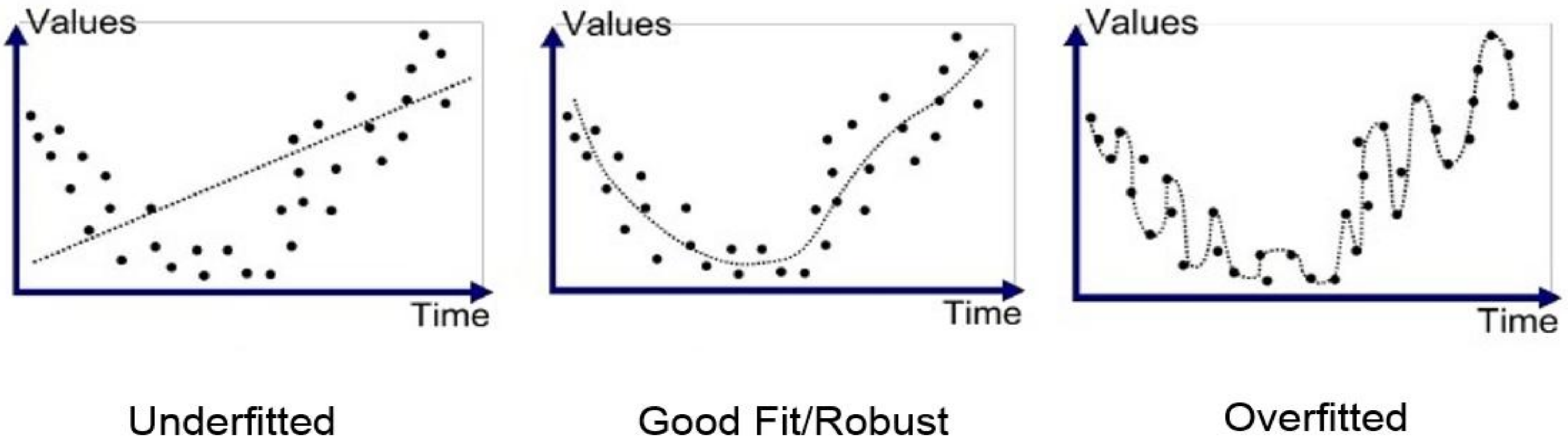
- **Key Point:** A high difference in MSE between training and test sets indicates **overfitting**.

# Loss Function and Minimization

- Measures how far the model's predictions are from the actual values.
- Think of it as the "average error" of the model.

- Example: Suppose you predict students' scores based on study hours:
  - **Actual scores**: 80, 85, 90
  - **Predicted scores**: 78, 86, 88
- **Errors (differences)**:
  - $(80 - 78)^2 = 4$
  - $(85 - 86)^2 = 1$
  - $(90 - 88)^2 = 4$
- **Mean Squared Error (MSE):** $MSE = \frac{4+1+4}{3} = 3$
- Minimize MSE → Find the line that makes the errors as small as possible!

# Loss Function and Minimization




- **Underfitting (Left Plot):** The model is too simple and doesn't capture patterns.
- **Proper Fit (Middle Plot):** The model captures the general trend correctly.
- **Overfitting (Right Plot):** The model is too complex and starts fitting noise instead of actual patterns.

# Linear Regression in Python

```
# 1. Import Libraries
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd

# 2. Sample Data (Study Hours vs Scores)
data = pd.DataFrame({'Hours': [1, 2, 3, 4, 5], 'Scores': [40, 50, 60, 70, 80]})
X = data[['Hours']]  # Feature: Study Hours
y = data['Scores']   # Target: Test Scores

# 3. Train-Test Split and Model Training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
model = LinearRegression().fit(X_train, y_train)

# 4. Make Predictions and Display Results
hours_to_predict = pd.DataFrame({'Hours': [3]})  # Format prediction input as a DataFrame
predicted_score = model.predict(hours_to_predict)[0]
print(f"Intercept: {model.intercept_:.2f}, Coefficient: {model.coef_[0]:.2f}")
print(f"Predicted Score for 3 hours of study: {predicted_score:.2f}")
```

```
Intercept: 30.00, Coefficient: 10.00
Predicted Score for 3 hours of study: 60.00
```

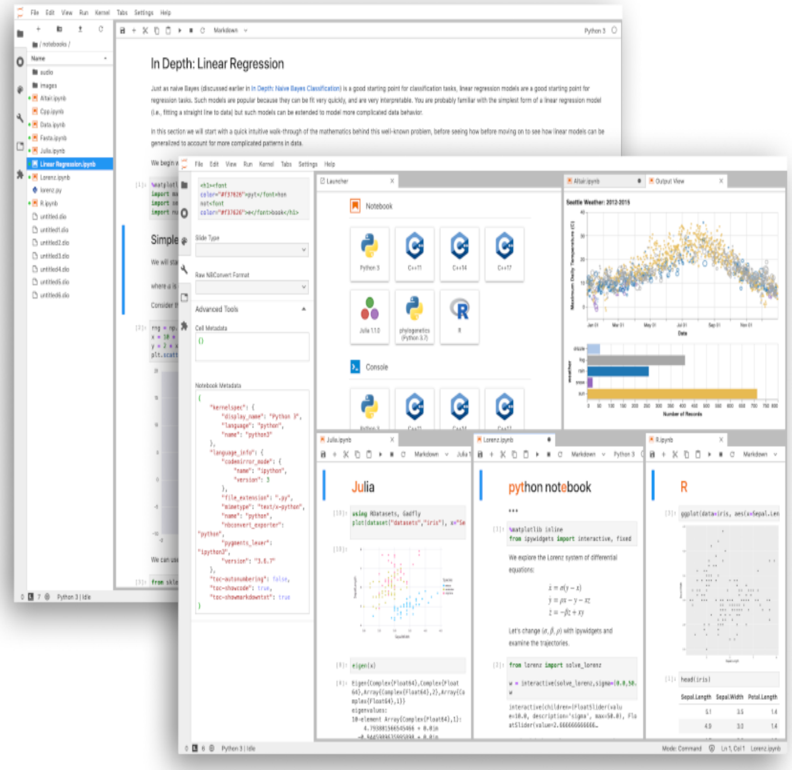
**Key Insights:**

- **Intercept**: Baseline prediction without features.
- **Coefficients**: How much the target variable changes for one unit increase in each feature.
- **R-squared ($R^2$)**: Shows how well the model fits the data (closer to 1 is better).
- **MSE (Mean Squared Error)**: Lower values indicate better predictions.

# Notebook Review

Walk through how to apply key Python concepts in a Jupyter Notebook:

- Covariance shows direction of relationships.
- Correlation shows the strength of relationships.
- Linear regression predicts based on a best-fit line



# Kahoot Quiz Time!

*Let's Test Our Knowledge!*

# Hands-on Exercise

**Form groups (2–3 members).**

- Download *Hands-on Exercise #5* from the course page.
- Complete the coding tasks and discuss your solutions.
- Don’t forget to add the names of your group members to the file.
- Submit your completed *Hands-on Exercise* to the course Moodle page or send it to the teacher's email address.

# Reference


- Vohra, M., & Patil, B. (2021). A Walk Through the World of Data Analytics. , 19-27. https://doi.org/10.4018/978-1-7998-3053-5.ch002.
- VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. O'Reilly Media. Available at https://jakevdp.github.io/PythonDataScienceHandbook/
- Severance, C. (2016). Python for everybody: Exploring data using Python 3. Charles Severance. Available at https://www.py4e.com/html3/
- McKinney, W. (2017). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter*. O'Reilly Media.