Class 9 - Intro to Data Science

MISE Summer Programming Camp 2023

Motivation

Why Data Science?

- Data is continuously being collected all the time, both about individuals and collective entities
- This is extremely helpful for making informed decisions about the world
- Data science offers the tools to extract this information from data

What is Data?

Data is any information that can be collected, stored, and analyzed

- Temperature measurements
- Age of a collection of patients
- Price of an item over time
- GPS coordinates

A list of **floats**

- Subscription status of users
- Binary classification (loan/no loan)

A list of **booleans**

- Product ratings
- Website traffic
- Product inventory
- Exam scores

A list of **ints**

What is Data Science

- Data science is an interdisciplinary field that combines **statistics**, **mathematics**, **programming**, and domain expertise to extract knowledge and insights from data
- It involves various processes, including **data collection**, **cleaning**, **analysis**, and visualization.
- Data scientists use techniques such as **machine learning**, **data mining**, and **statistical modeling**
- The goal of data science is to derive **actionable insights** and **drive data-informed decision-making**

How does Data Science work?

- **Data Collection**: Gathering relevant and reliable data from various sources, such as databases, APIs, surveys, and sensors.
- **Data Cleaning and Preprocessing**: Ensuring the data is accurate, consistent, and free from errors or missing values. This step involves data validation, transformation, and handling outliers.
- **Exploratory Data Analysis (EDA)**: Understanding the data through visualizations, descriptive statistics, and uncovering patterns or relationships.
- **Machine Learning**: Utilizing algorithms to build models that can learn from the data and make predictions or classifications.
- **Data Visualization**: Presenting data and insights in a visual format, such as charts, graphs, and interactive dashboards, to facilitate understanding and communication.

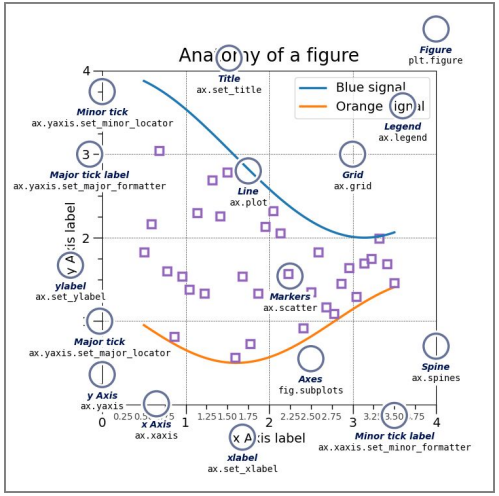
Where is Data Science used?

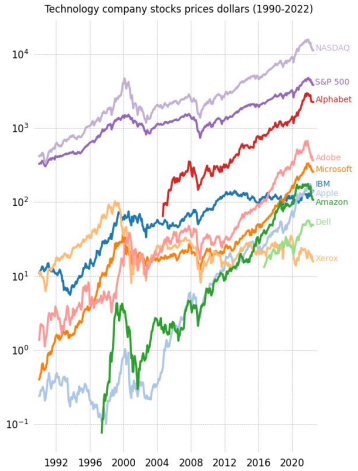
- **Finance**: Fraud detection, risk assessment, algorithmic trading.
- **Healthcare**: Disease prediction, personalized medicine, electronic health records analysis.
- **Marketing**: Customer segmentation, targeted advertising, sentiment analysis.
- **Transportation**: Route optimization, demand forecasting, autonomous vehicles.
- **Retail**: Recommender systems, inventory management, pricing optimization.
- **Manufacturing**: Predictive maintenance, quality control, supply chain optimization.

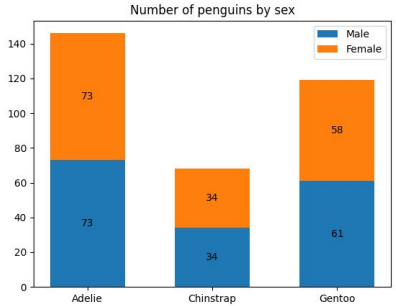
Visualizing Data

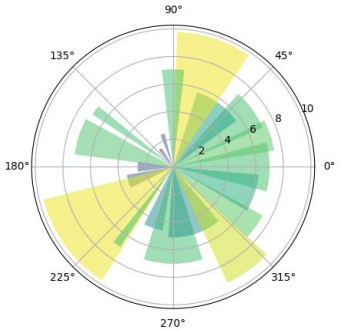
Examples of visualizations

- Various types of plots and charts are used to visualize different aspects of data, including trends, patterns, distributions, and relationships



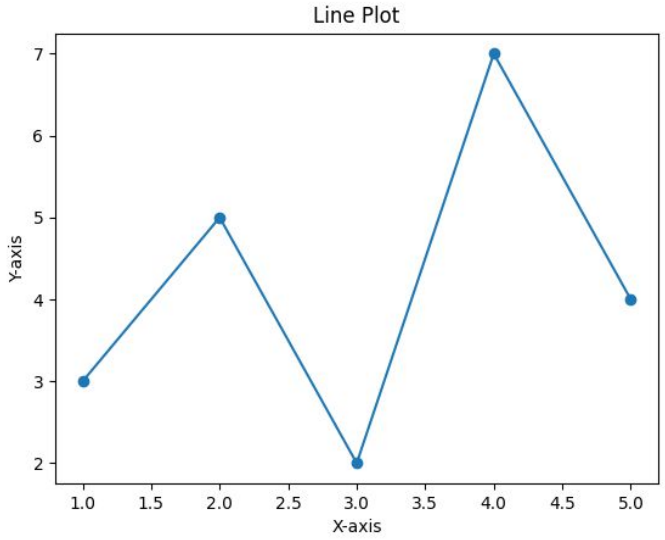






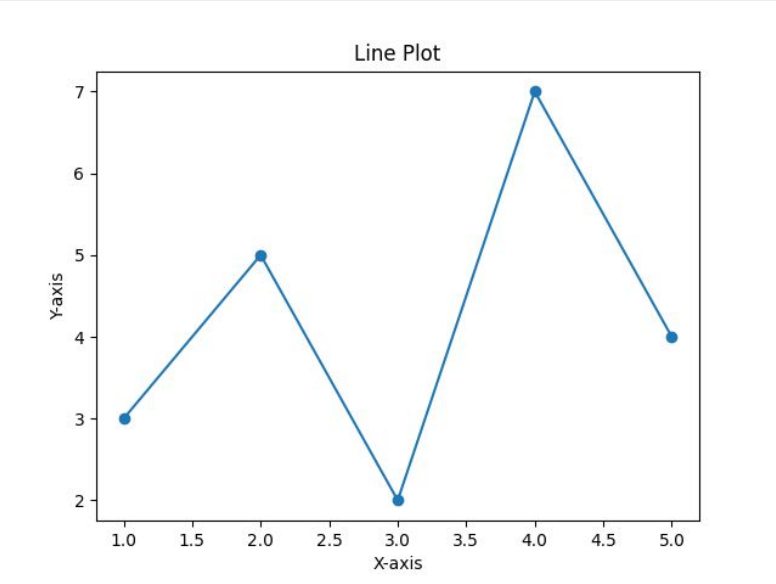
In Python: Matplotlib

- Matplotlib is a widely used data visualization **library** in Python
- It provides a comprehensive set of tools for creating various types of plots, charts, and graphs



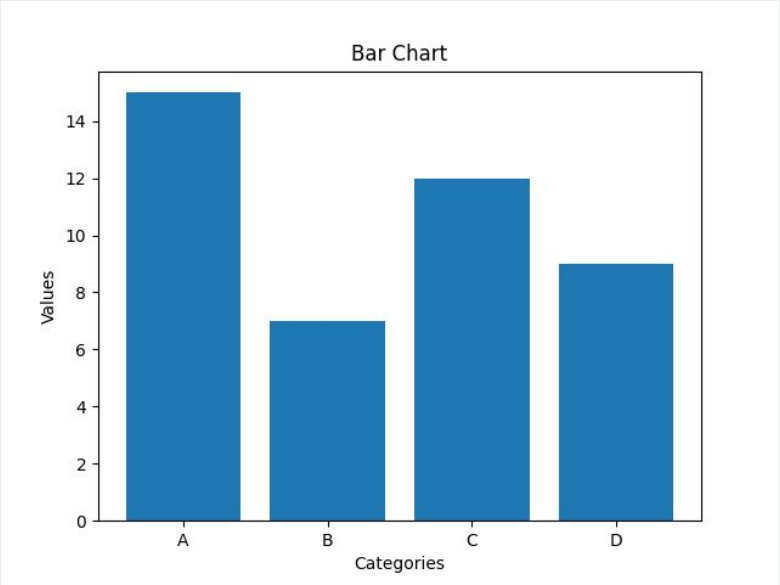
Note: we won't go into how to install any libraries today

Example: Line Plot



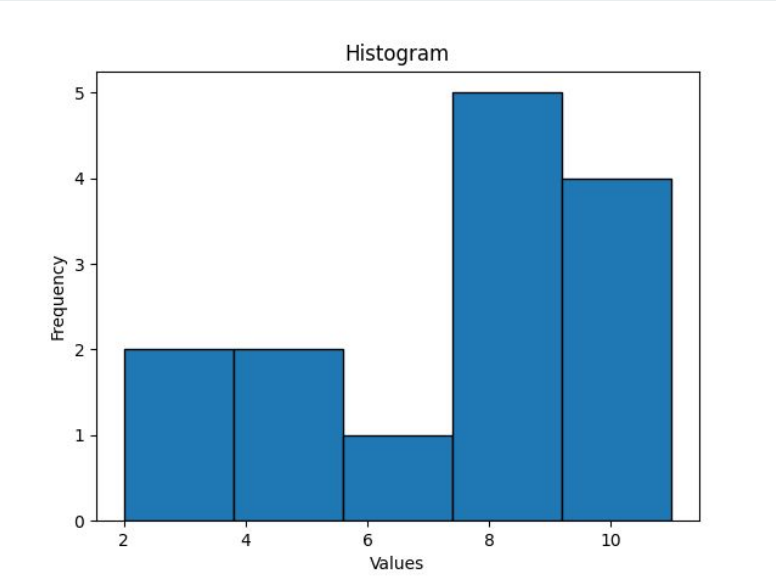

```python
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y = [3, 5, 2, 7, 4]

# Create a line plot
plt.plot(x, y, marker='o')

# Customize labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot')

# Display the plot
plt.show()
```

Example: Bar Chart




```python
import matplotlib.pyplot as plt

# Data
categories = ['A', 'B', 'C', 'D']
values = [15, 7, 12, 9]

# Create a bar chart
plt.bar(categories, values)

# Customize labels and title
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart')

# Display the plot
plt.show()
```

Example: Bar Chart



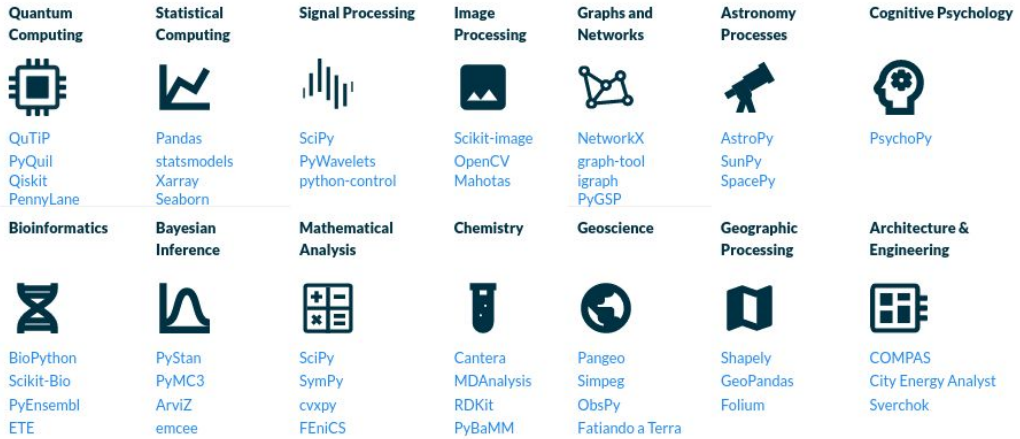

```python
import matplotlib.pyplot as plt

# Data
values = [2, 3, 5, 5, 7, 8, 8, 8, 9, 9, 10, 11, 11, 11]

# Create a histogram
plt.hist(values, bins=5, edgecolor='black')

# Customize labels and title
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Histogram')

# Display the plot
plt.show()
```

Analyzing Data

The numpy library

- NumPy (Numerical Python) is a library for scientific computing
- Powerful tools for working with arrays and performing mathematical operations efficiently

NumPy



Basic Data Analysis

- **Descriptive Statistics**: statistical measures such as mean, median, standard deviation, variance, and percentiles.
- **Array Operations**: sorting, filtering, searching, and element-wise computations.
- **Linear Algebra**: matrix multiplication, decomposition, and solving linear equations.

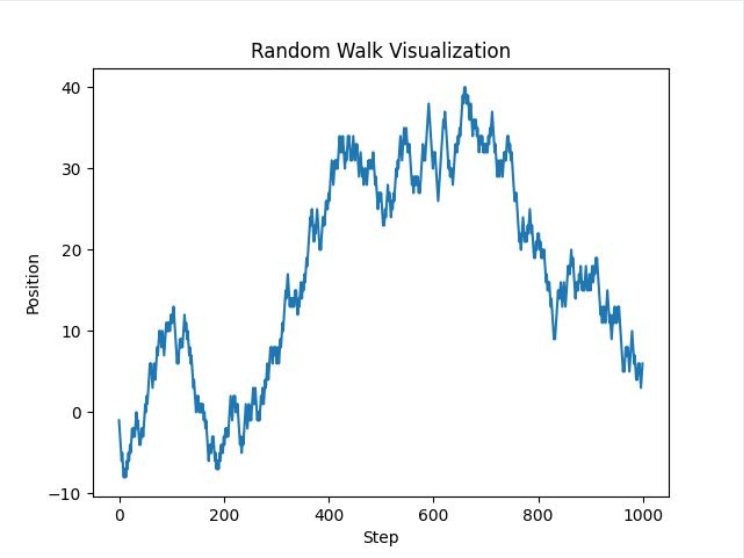
```python
import numpy as np

# Data
dataset = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])

# Calculate descriptive statistics
mean = np.mean(dataset)
median = np.median(dataset)
std_dev = np.std(dataset)
variance = np.var(dataset)

# Perform array operations
sorted_data = np.sort(dataset)
filtered_data = dataset[dataset > 5]
search_index = np.where(dataset == 3)

# Display the analysis results
print("Mean:", mean)
print("Median:", median)
print("Standard Deviation:", std_dev)
print("Variance:", variance)
print("Sorted Data:", sorted_data)
print("Filtered Data:", filtered_data)
print("Index of 3:", search_index)
```

Basic Data Synthesis

- **Random Number Generation**: generate random numbers from different distributions
- **Synthetic Dataset Creation**: create datasets with specific patterns, or characteristics.
- **Data Reshaping**: transforming data arrays to match specific dimensions

```python
import numpy as np

# Generate random numbers from a uniform distribution
uniform_numbers = np.random.random(size=10)

# Generate random integers
integer_numbers = np.random.randint(low=1, high=100, size=5)

# Generate random numbers from a normal distribution
mean = 0
std_dev = 1
normal_numbers = np.random.normal(loc=mean, scale=std_dev, size=10)

# Display the generated random numbers
print("Uniform Numbers:", uniform_numbers)
print("Integer Numbers:", integer_numbers)
print("Normal Numbers:", normal_numbers)
```

Example: Random Walk




```python
import numpy as np
import matplotlib.pyplot as plt

# Set the number of steps
num_steps = 1000

# Generate random steps
steps = np.random.choice([-1, 1], size=num_steps)

# Calculate the accumulated position
position = np.cumsum(steps)

# Plot the random walk
plt.plot(position)

# Customize the plot
plt.title('Random Walk Visualization')
plt.xlabel('Step')
plt.ylabel('Position')

# Display the plot
plt.show()
```

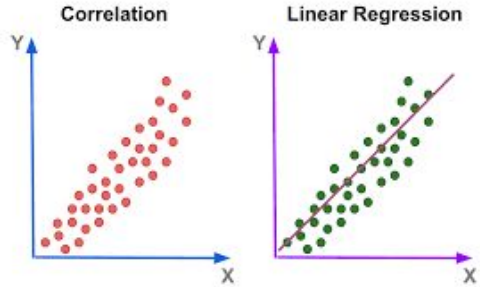
Predicting with Data

Prediction Models

- Mathematical algorithms that learn patterns from existing data to make predictions
- **Classification**: label data into discrete types, e.g. “Is this a fraudulent transaction?”
- **Clustering**: Group data together by common attributes, e.g. “Which of these people have similar health conditions?”
- **Regression**: estimating the relationships between dependent variables, e.g. “How much should this house cost?”

The simplest model: Linear Regression

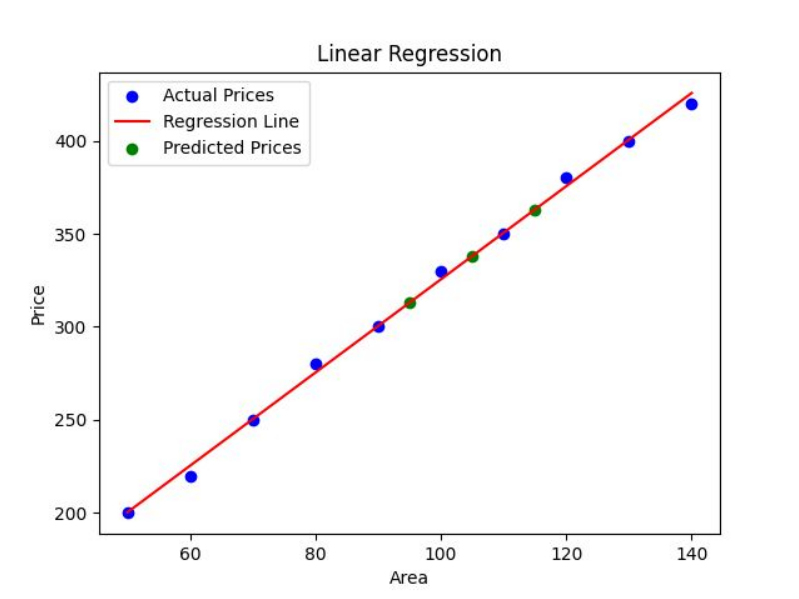
- Model linear relationship between variable (target) and independent variables (features)
- Assumes a linear equation of the form: y = mx + b, where y is the target variable, x is the feature, m is the slope, and b is the intercept
- The goal of is to find the best-fitting line that minimizes the difference between the predicted values and the actual values



Example: Linear Regression Equation

- Consider a simple example of predicting house prices based on the area of the house
- Model the relationship between house area (x) and house price (y): $y = mx + b$
- The slope (m) represents the change in house price for every unit increase in area, and the intercept (b) is the house price when the area is zero

Linear Regression in Python




```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Sample data
area = np.array([50, 60, 70, 80, 90, 100, 110, 120, 130, 140])
price = np.array([200, 220, 250, 280, 300, 330, 350, 380, 400, 420])

# Reshape the data
area = area.reshape((-1, 1))

# Create a linear regression model
model = LinearRegression()

# Fit the model to the data
model.fit(area, price)

# Predict house prices for new areas
new_areas = np.array([95, 105, 115]).reshape((-1, 1))
predicted_prices = model.predict(new_areas)

# Plot the data points and the regression line
plt.scatter(area, price, color='blue', label='Actual Prices')
plt.plot(area, model.predict(area), color='red', label='Regression Line')

# Plot the predicted prices for new areas
plt.scatter(new_areas, predicted_prices, color='green', label='Predicted Prices')

# Customize the plot
plt.xlabel('Area')
plt.ylabel('Price')
plt.title('Linear Regression')
plt.legend()

# Display the plot
plt.show()
```

Case Study: Temperature Forecast

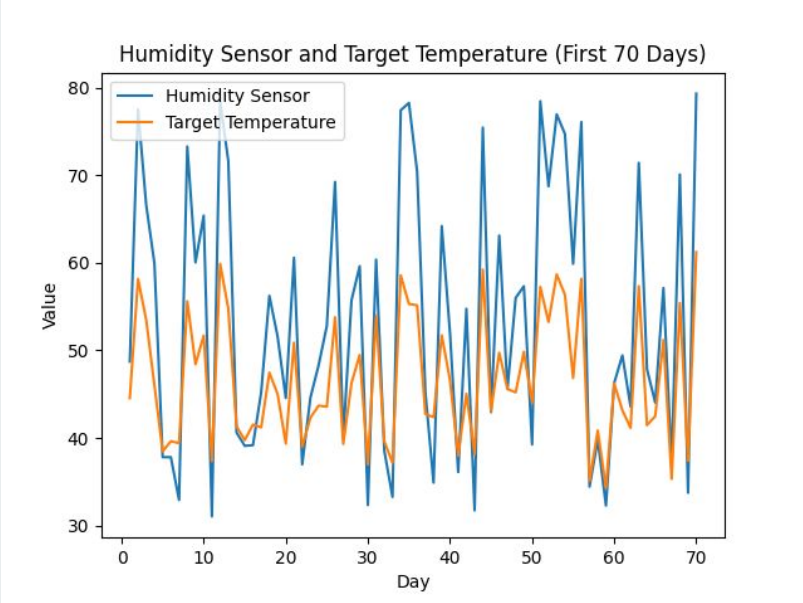
Forecasting temperature

1. **Introduction**:
   - **Objective**: Predicting temperature based on sensor measurements
   - **Dataset**: Generated data simulating temperature and humidity sensor readings
   - **Methodology**: Utilizing linear regression for temperature prediction
2. **Data Generation**:
   - **Humidity sensor data**: Randomly generated values for 100 days
   - **Target temperature**: Linearly dependent on humidity with Gaussian error
   - **Data Exploration**: Displaying the first 70 days of data
3. **Model Training**:
   - **Linear regression**: Training the model using the first 70 days of data
   - Learning the relationship between humidity and temperature
4. **Temperature Prediction**:
   - Applying the trained model to predict temperatures for the last 30 days
   - Comparing the predicted values with the true temperatures
5. **Visualization**:
   - Plotting the true temperature and predicted values for the last 30 days
   - Assessing the accuracy of the temperature predictions

Data Generation

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Generate random sensor data
np.random.seed(42)
num_days = 100
humidity_sensor = np.random.uniform(30, 80, num_days)
error = np.random.normal(0, 2, num_days)  # Gaussian error with mean 0 and standard
    deviation 2

# Generate target temperature data
slope = 0.5
intercept = 20
target_temperature = slope * humidity_sensor + intercept + error
```

Visualizing the Data

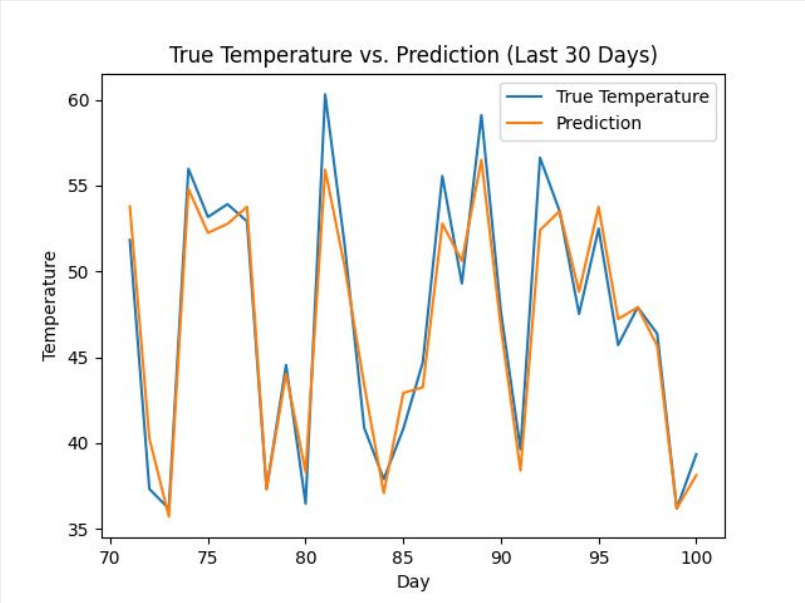



```python
# Display first 70 days of data
days = np.arange(1, 71)
plt.plot(days, humidity_sensor[:70], label='Humidity Sensor')
plt.plot(days, target_temperature[:70], label='Target Temperature')
plt.xlabel('Day')
plt.ylabel('Value')
plt.title('Humidity Sensor and Target Temperature (First 70 Days)')
plt.legend()
plt.show()
```

Model Training

```
26  # Data Analysis and Forecasting
27  X_train = humidity_sensor[:70].reshape(-1, 1)
28  y_train = target_temperature[:70]
29
30  model = LinearRegression()
31  model.fit(X_train, y_train)
32
33  # Predict temperature for the last 30 days
34  X_test = humidity_sensor[70:].reshape(-1, 1)
35  y_test = target_temperature[70:]
36  predicted_temperature = model.predict(X_test)
```

Visualizing the Results




```python
# Visualization of true temperature vs. prediction for the last 30 days
days = np.arange(71, 101)
plt.plot(days, y_test, label='True Temperature')
plt.plot(days, predicted_temperature, label='Prediction')
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.title('True Temperature vs. Prediction (Last 30 Days)')
plt.legend()
plt.show()

print("Predicted Temperature:", predicted_temperature)
```

Some parting thoughts

This is the end of the program!

Thank you all for participating!