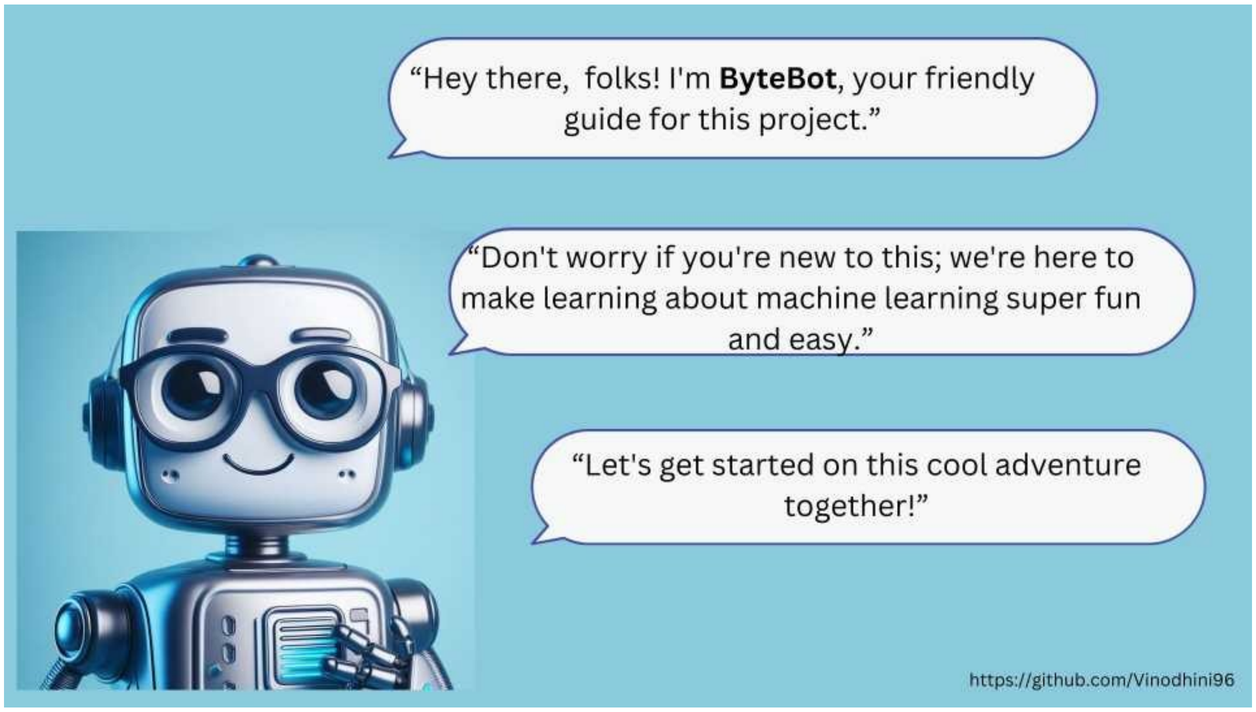
Introduction to Machine Learning:

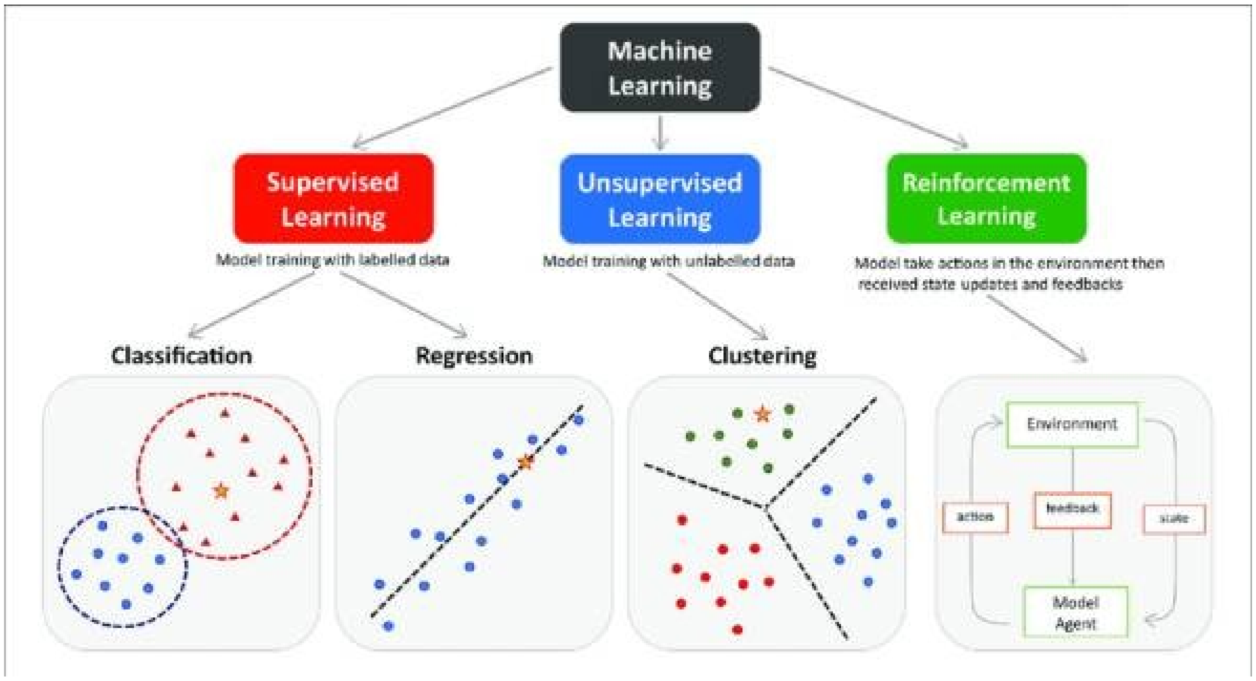
by

Vinodhini Rajamanickam


"Hey there, folks! I'm **ByteBot**, your friendly guide for this project."
"Don't worry if you're new to this; we're here to make learning about machine learning super fun and easy."
"Let's get started on this cool adventure together!"
https://github.com/Vinodhini96

Chapter 1: What is Machine Learning?

Machine Learning (ML) is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computer systems to learn and make predictions or decisions without being explicitly programmed. In essence, it's about teaching computers to learn from data and improve their performance on specific tasks over time.



Machine learning relies on data as its primary source of knowledge. This data can be of various types, including `text`, `images`, `numbers`, and more.

Machine learning algorithms use data to identify patterns, relationships, and trends. These algorithms learn from data by adjusting their internal parameters to optimize their performance.

Once a machine learning model has been trained on data, it can make predictions or decisions based on new, unseen data. These predictions can range from classifying objects in images to forecasting future values.

A good machine learning model should not only perform well on the data it was trained on but also generalize its knowledge to new, unseen data. Generalization ensures that the model is useful in practical applications.

Machine learning encompasses a wide range of algorithms and models, such as `decision trees`, `neural networks`, `support vector machines`, and more. The choice of algorithm depends on the specific problem and dataset.

The performance of a machine learning model is assessed using various metrics, depending on the type of problem. Common evaluation metrics include `accuracy`, `precision`, `recall`, `F1-score`, `mean squared error (MSE)`, and `R-squared`.

Chapter 2: Applications of Machine Learning:

Machine Learning (ML) has found applications across a wide range of industries and domains. Its ability to analyze data, identify patterns, and make predictions or decisions without explicit programming has led to numerous practical uses. Here are some notable applications of machine learning:

1.Image and Video Analysis:

- **Object Recognition**: ML models can recognize and classify objects in images and videos, which is used in autonomous vehicles, security systems, and image search engines.
- **Facial Recognition**: Facial recognition technology is used for authentication, surveillance, and tagging in social media.

2.Manufacturing and Industry:

- **Predictive Maintenance:** ML can predict equipment failures and schedule maintenance, reducing downtime and costs.
- **Quality Control:** ML models inspect products for defects in real-time during manufacturing.

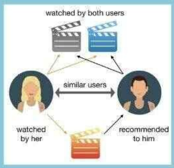
Applications of Machine Learning

3.Autonomous Vehicles:

- **Self-Driving Cars:** ML algorithms process sensor data from vehicles to navigate, make driving decisions, and avoid accidents.
- **Traffic Management:** ML can optimize traffic flow and reduce congestion in smart cities.

4. Recommendation Systems:

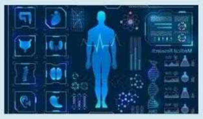
- **Content Recommendations**: Online platforms like Netflix and Amazon use ML to suggest movies, products, or content based on user preferences and behavior.
- **Personalization**: ML tailors user experiences on websites and apps by showing relevant content and ads.



https://github.com/Vinodhini96

5. Healthcare and Medical Diagnosis:

- **Disease Diagnosis:** ML is used to analyze medical images (X-rays, MRIs, CT scans) for disease detection. For example, in detecting cancerous tumors.
- **Drug Discovery:** ML models help identify potential drug candidates by analyzing chemical properties and interactions.
- **Patient Monitoring:** ML can monitor patients' health, detect anomalies, and predict disease outbreaks.



6.Environmental Monitoring:

- **Climate Modeling:** ML assists in climate prediction and modeling by analyzing vast datasets of weather and environmental data.

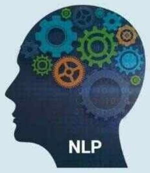
Applications of Machine Learning

7.Energy Management:

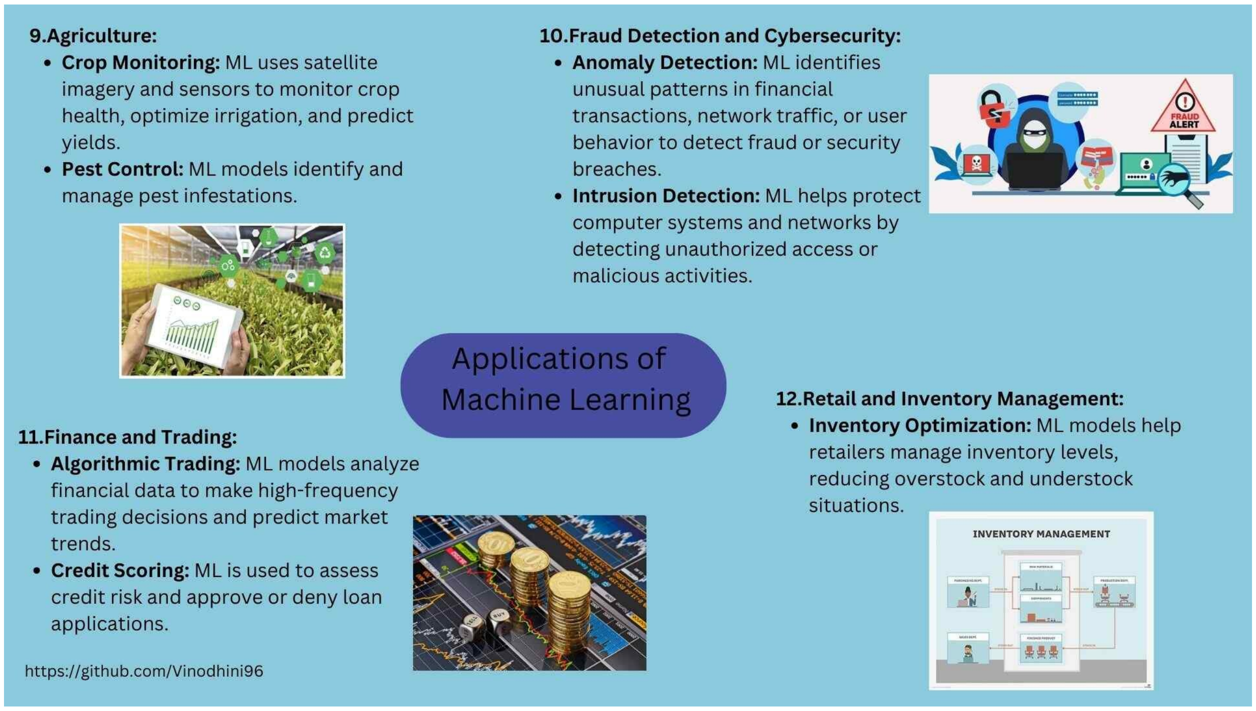
- **Energy Forecasting:** ML predicts energy demand and optimizes energy distribution, contributing to efficient energy use and sustainability.

8. Natural Language Processing (NLP):

- **Sentiment Analysis:** ML models can analyze text data to determine sentiment, making it valuable for gauging public opinion on social media or customer reviews.
- **Language Translation:** NLP models are used in translation services like Google Translate to translate text between languages.
- **Chatbots:** ML-powered chatbots provide automated customer support and assist with inquiries on websites and messaging platforms.



https://github.com/Vinodhini96



9.Agriculture:

- **Crop Monitoring:** ML uses satellite imagery and sensors to monitor crop health, optimize irrigation, and predict yields.
- **Pest Control:** ML models identify and manage pest infestations.

10.Fraud Detection and Cybersecurity:

- **Anomaly Detection:** ML identifies unusual patterns in financial transactions, network traffic, or user behavior to detect fraud or security breaches.
- **Intrusion Detection:** ML helps protect computer systems and networks by detecting unauthorized access or malicious activities.

Applications of Machine Learning

11.Finance and Trading:

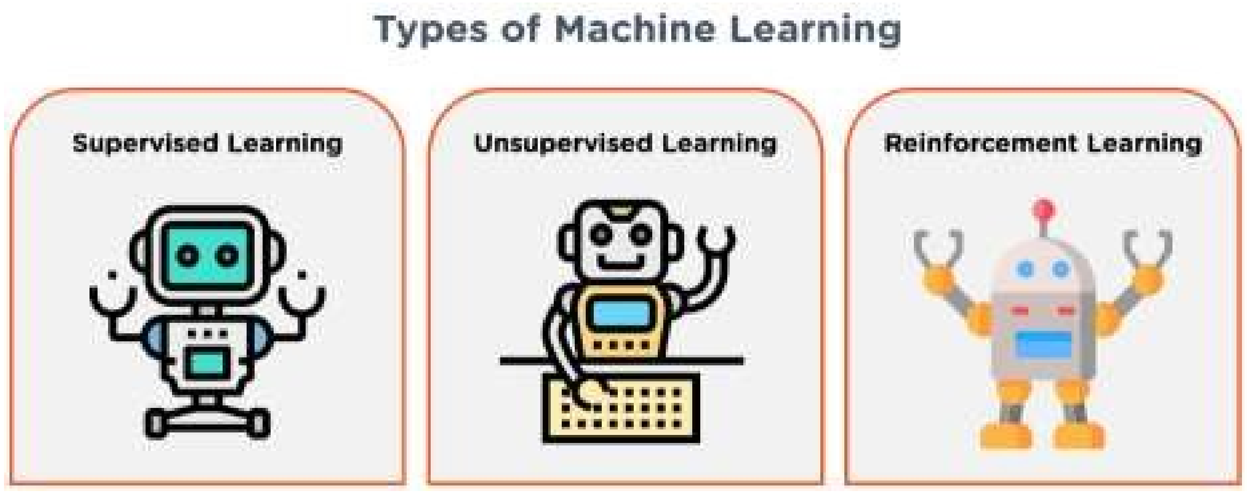
- **Algorithmic Trading:** ML models analyze financial data to make high-frequency trading decisions and predict market trends.
- **Credit Scoring:** ML is used to assess credit risk and approve or deny loan applications.

12.Retail and Inventory Management:

- **Inventory Optimization:** ML models help retailers manage inventory levels, reducing overstock and understock situations.

https://github.com/Vinodhini96

Chapter 3: Types of machine learning



1. Supervised Learning:

n supervised learning, the algorithm learns from a labeled dataset, where each input data point has a corresponding output or target value. The goal is to learn a mapping from input features to the correct output.

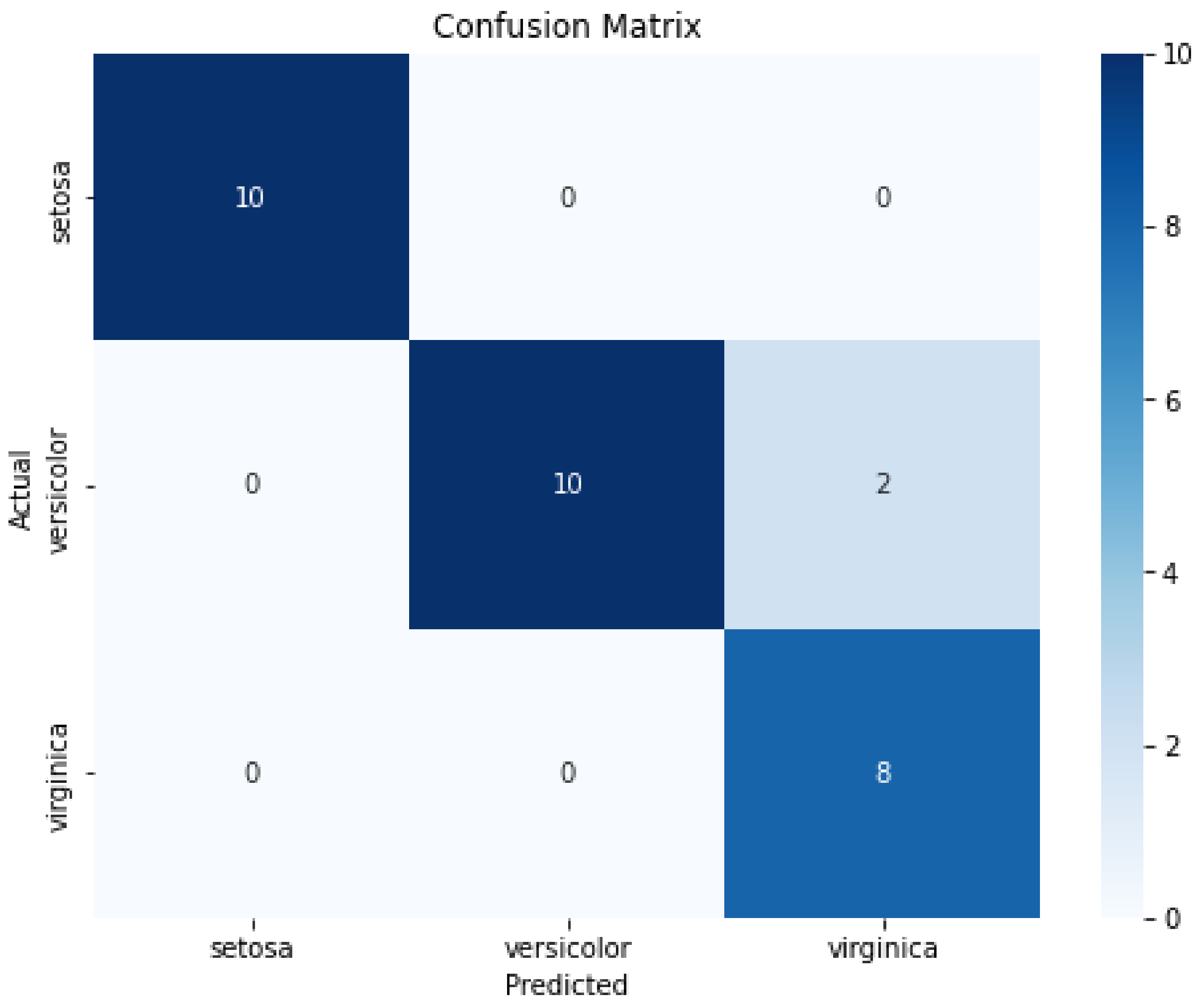
Example: Image classification, where the algorithm is trained to recognize objects in images and classify them into predefined categories.

Example Code (Python - Classification):

```python
from sklearn import datasets, model_selection
from sklearn.neighbors import KNeighborsClassifier

# Load the Iris dataset
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
model_selection.train_test_split(iris.data, iris.target,
test_size=0.2)

# Create a k-nearest neighbors classifier
knn = KNeighborsClassifier(n_neighbors=3)

# Train the model
knn.fit(X_train, y_train)

# Make predictions on test data
y_pred = knn.predict(X_test)
y_pred
```

```
array([2, 1, 1, 1, 1, 2, 1, 0, 0, 2, 0, 0, 0, 2, 2, 2, 1, 1, 2, 0, 0,
1,
       0, 2, 1, 2, 0, 0, 1, 2])
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

# Create a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a classification report
class_report = classification_report(y_test, y_pred,
target_names=iris.target_names)

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Print the classification report
print("Classification Report:\n", class_report)
```



```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.83      0.91        12
   virginica       0.80      1.00      0.89         8

    accuracy                           0.93        30
   macro avg       0.93      0.94      0.93        30
weighted avg       0.95      0.93      0.93        30
```

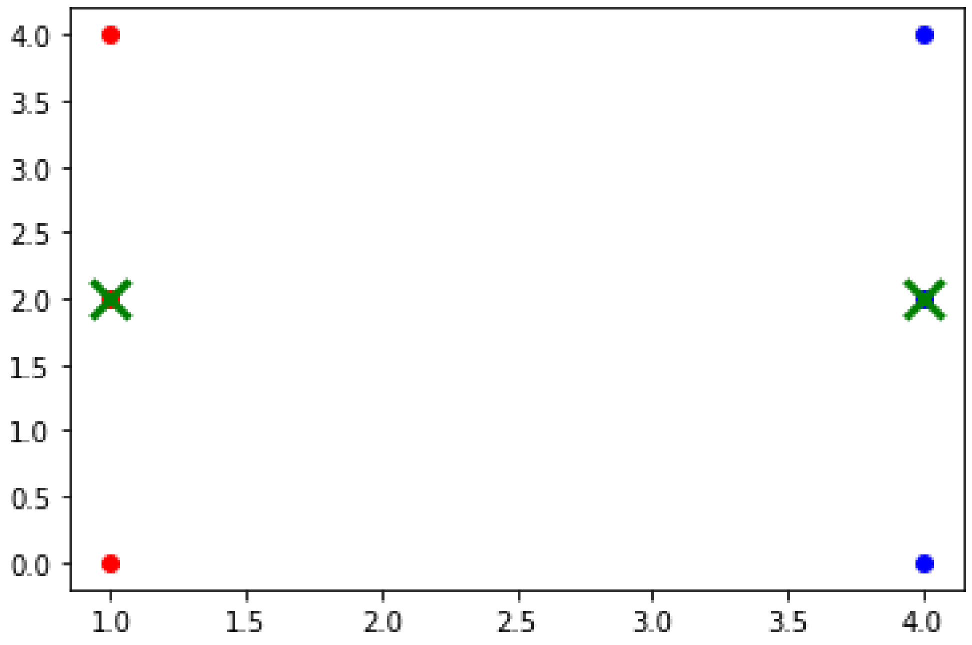
2. Unsupervised Learning:

`Unsupervised learning` involves training an algorithm on an unlabeled dataset, where there are no predefined output values. The goal is to discover patterns, relationships, or structure within the data.

Example: Clustering customer data to identify distinct customer segments based on their purchasing behavior.

Example Code (Clustering):

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Generate sample data
data = [[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]]
kmeans = KMeans(n_clusters=2)  # Specify the number of clusters
kmeans.fit(data)

# Get cluster labels and centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Visualize the clusters
for i in range(len(data)):
    color = 'r' if labels[i] == 0 else 'b'
    plt.scatter(data[i][0], data[i][1], c=color)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200,
linewidths=3, color='g')
plt.show()
```



3. Reinforcement Learning:

Reinforcement learning is about training agents to make sequences of decisions in an environment to maximize a reward signal. Agents learn from the consequences of their actions, and the goal is to find an optimal policy that leads to the highest cumulative reward.

Example:training an AI agent to navigate a grid world while maximizing its cumulative rewards over time using Q-learning.

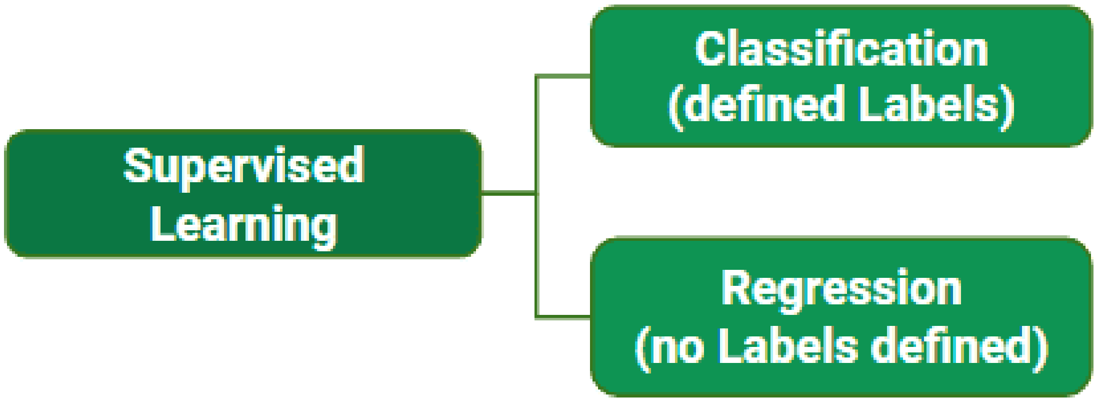
Example Code (Reinforcement Learning):

```python
import numpy as np

# Define the grid world environment
grid_world = [
    ['S', 'F', 'F', 'X'],
    ['F', 'X', 'F', 'X'],
    ['F', 'F', 'F', 'F'],
    ['X', 'F', 'F', 'G']
]

# Define the actions (up, down, left, right)
actions = [(0, -1), (0, 1), (-1, 0), (1, 0)]
num_actions = len(actions)

# Define the Q-table as a dictionary
Q = {}

# Hyperparameters
learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.1
num_episodes = 1000

# Initialize the Q-table with zeros
for i in range(len(grid_world)):
    for j in range(len(grid_world[i])):
        state = (i, j)
        if grid_world[i][j] != 'X':
            Q[state] = np.zeros(num_actions)

# Training loop
for episode in range(num_episodes):
    state = (0, 0)  # Start from the top-left corner
    done = False
    total_reward = 0

    while not done:
        if np.random.rand() < epsilon:
            action = np.random.choice(num_actions)  # Exploration
        else:
            action = np.argmax(Q[state])  # Exploitation

        dx, dy = actions[action]
        next_state = (state[0] + dx, state[1] + dy)

        # Check if the agent has reached the goal or encountered an
```

```
obstacle
        if next_state[0] < 0 or next_state[0] >= len(grid_world) \
                or next_state[1] < 0 or next_state[1] >=
len(grid_world[0]) \
                or grid_world[next_state[0]][next_state[1]] == 'X':
            # Invalid move, stay in the current state
            next_state = state

        # Check if the agent has reached the goal
        if grid_world[next_state[0]][next_state[1]] == 'G':
            reward = 1.0
            done = True
        else:
            reward = 0.0

        # Update the Q-value using the Q-learning formula
        Q[state][action] = (1 - learning_rate) * Q[state][action] + \
                           learning_rate * (reward + discount_factor *
np.max(Q[next_state]))

        total_reward += reward
        state = next_state

    print(f"Episode {episode}: Total Reward = {total_reward}")

# Evaluate the trained policy (follow the Q-table to reach the goal)
state = (0, 0)
path = [state]

while grid_world[state[0]][state[1]] != 'G':
    action = np.argmax(Q[state])
    dx, dy = actions[action]
    next_state = (state[0] + dx, state[1] + dy)
    path.append(next_state)
    state = next_state

print("Optimal Path:")
for i, state in enumerate(path):
    print(f"Step {i}: {state}")
```

```
Episode 0: Total Reward = 1.0
Episode 1: Total Reward = 1.0
Episode 2: Total Reward = 1.0
Episode 3: Total Reward = 1.0
Episode 4: Total Reward = 1.0
Episode 5: Total Reward = 1.0
Episode 6: Total Reward = 1.0
Episode 7: Total Reward = 1.0
Episode 8: Total Reward = 1.0
Episode 9: Total Reward = 1.0
Episode 10: Total Reward = 1.0
Episode 11: Total Reward = 1.0
Episode 12: Total Reward = 1.0
Episode 13: Total Reward = 1.0
Episode 14: Total Reward = 1.0
Episode 15: Total Reward = 1.0
Episode 16: Total Reward = 1.0
Episode 17: Total Reward = 1.0
Episode 18: Total Reward = 1.0
Episode 19: Total Reward = 1.0
Episode 20: Total Reward = 1.0
Episode 21: Total Reward = 1.0
Episode 22: Total Reward = 1.0
Episode 23: Total Reward = 1.0
Episode 24: Total Reward = 1.0
Episode 25: Total Reward = 1.0
Episode 26: Total Reward = 1.0
Episode 27: Total Reward = 1.0
Episode 28: Total Reward = 1.0
Episode 29: Total Reward = 1.0
Episode 30: Total Reward = 1.0
Episode 31: Total Reward = 1.0
Episode 32: Total Reward = 1.0
Episode 33: Total Reward = 1.0
Episode 34: Total Reward = 1.0
Episode 35: Total Reward = 1.0
Episode 36: Total Reward = 1.0
Episode 37: Total Reward = 1.0
Episode 38: Total Reward = 1.0
Episode 39: Total Reward = 1.0
Episode 40: Total Reward = 1.0
Episode 41: Total Reward = 1.0
Episode 42: Total Reward = 1.0
Episode 43: Total Reward = 1.0
Episode 44: Total Reward = 1.0
Episode 45: Total Reward = 1.0
Episode 46: Total Reward = 1.0
Episode 47: Total Reward = 1.0
Episode 48: Total Reward = 1.0
Episode 49: Total Reward = 1.0
Episode 50: Total Reward = 1.0
Episode 51: Total Reward = 1.0
Episode 52: Total Reward = 1.0
Episode 53: Total Reward = 1.0
Episode 54: Total Reward = 1.0
Episode 55: Total Reward = 1.0
Episode 56: Total Reward = 1.0
Episode 57: Total Reward = 1.0
Episode 58: Total Reward = 1.0
Episode 59: Total Reward = 1.0
Episode 60: Total Reward = 1.0
Episode 61: Total Reward = 1.0
Episode 62: Total Reward = 1.0
Episode 63: Total Reward = 1.0
Episode 64: Total Reward = 1.0
Episode 65: Total Reward = 1.0
Episode 66: Total Reward = 1.0
Episode 67: Total Reward = 1.0
Episode 68: Total Reward = 1.0
Episode 69: Total Reward = 1.0
Episode 70: Total Reward = 1.0
Episode 71: Total Reward = 1.0
Episode 72: Total Reward = 1.0
Episode 73: Total Reward = 1.0
Episode 74: Total Reward = 1.0
Episode 75: Total Reward = 1.0
Episode 76: Total Reward = 1.0
Episode 77: Total Reward = 1.0
Episode 78: Total Reward = 1.0
Episode 79: Total Reward = 1.0
Episode 80: Total Reward = 1.0
Episode 81: Total Reward = 1.0
Episode 82: Total Reward = 1.0
Episode 83: Total Reward = 1.0
Episode 84: Total Reward = 1.0
Episode 85: Total Reward = 1.0
Episode 86: Total Reward = 1.0
Episode 87: Total Reward = 1.0
Episode 88: Total Reward = 1.0
Episode 89: Total Reward = 1.0
Episode 90: Total Reward = 1.0
Episode 91: Total Reward = 1.0
Episode 92: Total Reward = 1.0
Episode 93: Total Reward = 1.0
Episode 94: Total Reward = 1.0
Episode 95: Total Reward = 1.0
Episode 96: Total Reward = 1.0
Episode 97: Total Reward = 1.0
Episode 98: Total Reward = 1.0
Episode 99: Total Reward = 1.0
Episode 100: Total Reward = 1.0
Episode 101: Total Reward = 1.0
Episode 102: Total Reward = 1.0
Episode 103: Total Reward = 1.0
Episode 104: Total Reward = 1.0
Episode 105: Total Reward = 1.0
Episode 106: Total Reward = 1.0
Episode 107: Total Reward = 1.0
Episode 108: Total Reward = 1.0
Episode 109: Total Reward = 1.0
Episode 110: Total Reward = 1.0
Episode 111: Total Reward = 1.0
Episode 112: Total Reward = 1.0
Episode 113: Total Reward = 1.0
Episode 114: Total Reward = 1.0
Episode 115: Total Reward = 1.0
Episode 116: Total Reward = 1.0
Episode 117: Total Reward = 1.0
Episode 118: Total Reward = 1.0
Episode 119: Total Reward = 1.0
Episode 120: Total Reward = 1.0
Episode 121: Total Reward = 1.0
Episode 122: Total Reward = 1.0
Episode 123: Total Reward = 1.0
Episode 124: Total Reward = 1.0
Episode 125: Total Reward = 1.0
Episode 126: Total Reward = 1.0
Episode 127: Total Reward = 1.0
Episode 128: Total Reward = 1.0
Episode 129: Total Reward = 1.0
Episode 130: Total Reward = 1.0
Episode 131: Total Reward = 1.0
Episode 132: Total Reward = 1.0
Episode 133: Total Reward = 1.0
Episode 134: Total Reward = 1.0
Episode 135: Total Reward = 1.0
Episode 136: Total Reward = 1.0
Episode 137: Total Reward = 1.0
Episode 138: Total Reward = 1.0
Episode 139: Total Reward = 1.0
Episode 140: Total Reward = 1.0
Episode 141: Total Reward = 1.0
Episode 142: Total Reward = 1.0
Episode 143: Total Reward = 1.0
Episode 144: Total Reward = 1.0
Episode 145: Total Reward = 1.0
Episode 146: Total Reward = 1.0
Episode 147: Total Reward = 1.0
Episode 148: Total Reward = 1.0
Episode 149: Total Reward = 1.0
Episode 150: Total Reward = 1.0
Episode 151: Total Reward = 1.0
Episode 152: Total Reward = 1.0
Episode 153: Total Reward = 1.0
Episode 154: Total Reward = 1.0
Episode 155: Total Reward = 1.0
Episode 156: Total Reward = 1.0
Episode 157: Total Reward = 1.0
Episode 158: Total Reward = 1.0
Episode 159: Total Reward = 1.0
Episode 160: Total Reward = 1.0
Episode 161: Total Reward = 1.0
Episode 162: Total Reward = 1.0
Episode 163: Total Reward = 1.0
Episode 164: Total Reward = 1.0
Episode 165: Total Reward = 1.0
Episode 166: Total Reward = 1.0
Episode 167: Total Reward = 1.0
Episode 168: Total Reward = 1.0
Episode 169: Total Reward = 1.0
Episode 170: Total Reward = 1.0
Episode 171: Total Reward = 1.0
Episode 172: Total Reward = 1.0
Episode 173: Total Reward = 1.0
Episode 174: Total Reward = 1.0
Episode 175: Total Reward = 1.0
Episode 176: Total Reward = 1.0
Episode 177: Total Reward = 1.0
Episode 178: Total Reward = 1.0
Episode 179: Total Reward = 1.0
Episode 180: Total Reward = 1.0
Episode 181: Total Reward = 1.0
Episode 182: Total Reward = 1.0
Episode 183: Total Reward = 1.0
Episode 184: Total Reward = 1.0
Episode 185: Total Reward = 1.0
Episode 186: Total Reward = 1.0
Episode 187: Total Reward = 1.0
Episode 188: Total Reward = 1.0
Episode 189: Total Reward = 1.0
Episode 190: Total Reward = 1.0
Episode 191: Total Reward = 1.0
Episode 192: Total Reward = 1.0
Episode 193: Total Reward = 1.0
Episode 194: Total Reward = 1.0
Episode 195: Total Reward = 1.0
Episode 196: Total Reward = 1.0
Episode 197: Total Reward = 1.0
Episode 198: Total Reward = 1.0
Episode 199: Total Reward = 1.0
Episode 200: Total Reward = 1.0
Episode 201: Total Reward = 1.0
Episode 202: Total Reward = 1.0
Episode 203: Total Reward = 1.0
Episode 204: Total Reward = 1.0
Episode 205: Total Reward = 1.0
Episode 206: Total Reward = 1.0
Episode 207: Total Reward = 1.0
Episode 208: Total Reward = 1.0
Episode 209: Total Reward = 1.0
Episode 210: Total Reward = 1.0
Episode 211: Total Reward = 1.0
Episode 212: Total Reward = 1.0
Episode 213: Total Reward = 1.0
Episode 214: Total Reward = 1.0
Episode 215: Total Reward = 1.0
Episode 216: Total Reward = 1.0
Episode 217: Total Reward = 1.0
Episode 218: Total Reward = 1.0
Episode 219: Total Reward = 1.0
Episode 220: Total Reward = 1.0
Episode 221: Total Reward = 1.0
Episode 222: Total Reward = 1.0
Episode 223: Total Reward = 1.0
Episode 224: Total Reward = 1.0
Episode 225: Total Reward = 1.0
Episode 226: Total Reward = 1.0
Episode 227: Total Reward = 1.0
Episode 228: Total Reward = 1.0
Episode 229: Total Reward = 1.0
Episode 230: Total Reward = 1.0
Episode 231: Total Reward = 1.0
Episode 232: Total Reward = 1.0
Episode 233: Total Reward = 1.0
Episode 234: Total Reward = 1.0
Episode 235: Total Reward = 1.0
Episode 236: Total Reward = 1.0
Episode 237: Total Reward = 1.0
Episode 238: Total Reward = 1.0
Episode 239: Total Reward = 1.0
Episode 240: Total Reward = 1.0
Episode 241: Total Reward = 1.0
Episode 242: Total Reward = 1.0
Episode 243: Total Reward = 1.0
Episode 244: Total Reward = 1.0
Episode 245: Total Reward = 1.0
Episode 246: Total Reward = 1.0
Episode 247: Total Reward = 1.0
Episode 248: Total Reward = 1.0
Episode 249: Total Reward = 1.0
Episode 250: Total Reward = 1.0
Episode 251: Total Reward = 1.0
Episode 252: Total Reward = 1.0
Episode 253: Total Reward = 1.0
Episode 254: Total Reward = 1.0
Episode 255: Total Reward = 1.0
Episode 256: Total Reward = 1.0
Episode 257: Total Reward = 1.0
Episode 258: Total Reward = 1.0
Episode 259: Total Reward = 1.0
Episode 260: Total Reward = 1.0
Episode 261: Total Reward = 1.0
Episode 262: Total Reward = 1.0
Episode 263: Total Reward = 1.0
Episode 264: Total Reward = 1.0
Episode 265: Total Reward = 1.0
Episode 266: Total Reward = 1.0
Episode 267: Total Reward = 1.0
Episode 268: Total Reward = 1.0
Episode 269: Total Reward = 1.0
Episode 270: Total Reward = 1.0
Episode 271: Total Reward = 1.0
Episode 272: Total Reward = 1.0
Episode 273: Total Reward = 1.0
Episode 274: Total Reward = 1.0
Episode 275: Total Reward = 1.0
Episode 276: Total Reward = 1.0
Episode 277: Total Reward = 1.0
Episode 278: Total Reward = 1.0
Episode 279: Total Reward = 1.0
Episode 280: Total Reward = 1.0
Episode 281: Total Reward = 1.0
Episode 282: Total Reward = 1.0
Episode 283: Total Reward = 1.0
Episode 284: Total Reward = 1.0
Episode 285: Total Reward = 1.0
Episode 286: Total Reward = 1.0
Episode 287: Total Reward = 1.0
Episode 288: Total Reward = 1.0
Episode 289: Total Reward = 1.0
Episode 290: Total Reward = 1.0
Episode 291: Total Reward = 1.0
Episode 292: Total Reward = 1.0
Episode 293: Total Reward = 1.0
Episode 294: Total Reward = 1.0
Episode 295: Total Reward = 1.0
Episode 296: Total Reward = 1.0
Episode 297: Total Reward = 1.0
Episode 298: Total Reward = 1.0
Episode 299: Total Reward = 1.0
Episode 300: Total Reward = 1.0
Episode 301: Total Reward = 1.0
Episode 302: Total Reward = 1.0
Episode 303: Total Reward = 1.0
Episode 304: Total Reward = 1.0
Episode 305: Total Reward = 1.0
Episode 306: Total Reward = 1.0
Episode 307: Total Reward = 1.0
Episode 308: Total Reward = 1.0
Episode 309: Total Reward = 1.0
Episode 310: Total Reward = 1.0
Episode 311: Total Reward = 1.0
Episode 312: Total Reward = 1.0
Episode 313: Total Reward = 1.0
Episode 314: Total Reward = 1.0
Episode 315: Total Reward = 1.0
Episode 316: Total Reward = 1.0
Episode 317: Total Reward = 1.0
Episode 318: Total Reward = 1.0
Episode 319: Total Reward = 1.0
Episode 320: Total Reward = 1.0
Episode 321: Total Reward = 1.0
Episode 322: Total Reward = 1.0
Episode 323: Total Reward = 1.0
Episode 324: Total Reward = 1.0
Episode 325: Total Reward = 1.0
Episode 326: Total Reward = 1.0
Episode 327: Total Reward = 1.0
Episode 328: Total Reward = 1.0
Episode 329: Total Reward = 1.0
Episode 330: Total Reward = 1.0
Episode 331: Total Reward = 1.0
Episode 332: Total Reward = 1.0
Episode 333: Total Reward = 1.0
Episode 334: Total Reward = 1.0
Episode 335: Total Reward = 1.0
Episode 336: Total Reward = 1.0
Episode 337: Total Reward = 1.0
Episode 338: Total Reward = 1.0
Episode 339: Total Reward = 1.0
Episode 340: Total Reward = 1.0
Episode 341: Total Reward = 1.0
Episode 342: Total Reward = 1.0
Episode 343: Total Reward = 1.0
Episode 344: Total Reward = 1.0
Episode 345: Total Reward = 1.0
Episode 346: Total Reward = 1.0
Episode 347: Total Reward = 1.0
Episode 348: Total Reward = 1.0
Episode 349: Total Reward = 1.0
Episode 350: Total Reward = 1.0
Episode 351: Total Reward = 1.0
Episode 352: Total Reward = 1.0
Episode 353: Total Reward = 1.0
Episode 354: Total Reward = 1.0
Episode 355: Total Reward = 1.0
Episode 356: Total Reward = 1.0
Episode 357: Total Reward = 1.0
Episode 358: Total Reward = 1.0
Episode 359: Total Reward = 1.0
Episode 360: Total Reward = 1.0
Episode 361: Total Reward = 1.0
Episode 362: Total Reward = 1.0
Episode 363: Total Reward = 1.0
Episode 364: Total Reward = 1.0
Episode 365: Total Reward = 1.0
Episode 366: Total Reward = 1.0
Episode 367: Total Reward = 1.0
Episode 368: Total Reward = 1.0
Episode 369: Total Reward = 1.0
Episode 370: Total Reward = 1.0
Episode 371: Total Reward = 1.0
Episode 372: Total Reward = 1.0
Episode 373: Total Reward = 1.0
Episode 374: Total Reward = 1.0
Episode 375: Total Reward = 1.0
Episode 376: Total Reward = 1.0
Episode 377: Total Reward = 1.0
Episode 378: Total Reward = 1.0
Episode 379: Total Reward = 1.0
Episode 380: Total Reward = 1.0
Episode 381: Total Reward = 1.0
Episode 382: Total Reward = 1.0
Episode 383: Total Reward = 1.0
Episode 384: Total Reward = 1.0
Episode 385: Total Reward = 1.0
Episode 386: Total Reward = 1.0
Episode 387: Total Reward = 1.0
Episode 388: Total Reward = 1.0
Episode 389: Total Reward = 1.0
Episode 390: Total Reward = 1.0
Episode 391: Total Reward = 1.0
Episode 392: Total Reward = 1.0
Episode 393: Total Reward = 1.0
Episode 394: Total Reward = 1.0
Episode 395: Total Reward = 1.0
Episode 396: Total Reward = 1.0
Episode 397: Total Reward = 1.0
Episode 398: Total Reward = 1.0
Episode 399: Total Reward = 1.0
Episode 400: Total Reward = 1.0
Episode 401: Total Reward = 1.0
Episode 402: Total Reward = 1.0
Episode 403: Total Reward = 1.0
Episode 404: Total Reward = 1.0
Episode 405: Total Reward = 1.0
Episode 406: Total Reward = 1.0
Episode 407: Total Reward = 1.0
Episode 408: Total Reward = 1.0
Episode 409: Total Reward = 1.0
Episode 410: Total Reward = 1.0
Episode 411: Total Reward = 1.0
Episode 412: Total Reward = 1.0
Episode 413: Total Reward = 1.0
Episode 414: Total Reward = 1.0
Episode 415: Total Reward = 1.0
Episode 416: Total Reward = 1.0
Episode 417: Total Reward = 1.0
Episode 418: Total Reward = 1.0
Episode 419: Total Reward = 1.0
Episode 420: Total Reward = 1.0
Episode 421: Total Reward = 1.0
Episode 422: Total Reward = 1.0
Episode 423: Total Reward = 1.0
Episode 424: Total Reward = 1.0
Episode 425: Total Reward = 1.0
Episode 426: Total Reward = 1.0
Episode 427: Total Reward = 1.0
Episode 428: Total Reward = 1.0
Episode 429: Total Reward = 1.0
Episode 430: Total Reward = 1.0
Episode 431: Total Reward = 1.0
Episode 432: Total Reward = 1.0
Episode 433: Total Reward = 1.0
Episode 434: Total Reward = 1.0
Episode 435: Total Reward = 1.0
Episode 436: Total Reward = 1.0
Episode 437: Total Reward = 1.0
Episode 438: Total Reward = 1.0
Episode 439: Total Reward = 1.0
Episode 440: Total Reward = 1.0
Episode 441: Total Reward = 1.0
Episode 442: Total Reward = 1.0
Episode 443: Total Reward = 1.0
Episode 444: Total Reward = 1.0
Episode 445: Total Reward = 1.0
Episode 446: Total Reward = 1.0
Episode 447: Total Reward = 1.0
Episode 448: Total Reward = 1.0
Episode 449: Total Reward = 1.0
Episode 450: Total Reward = 1.0
Episode 451: Total Reward = 1.0
Episode 452: Total Reward = 1.0
Episode 453: Total Reward = 1.0
Episode 454: Total Reward = 1.0
Episode 455: Total Reward = 1.0
Episode 456: Total Reward = 1.0
Episode 457: Total Reward = 1.0
Episode 458: Total Reward = 1.0
Episode 459: Total Reward = 1.0
Episode 460: Total Reward = 1.0
Episode 461: Total Reward = 1.0
Episode 462: Total Reward = 1.0
Episode 463: Total Reward = 1.0
Episode 464: Total Reward = 1.0
Episode 465: Total Reward = 1.0
Episode 466: Total Reward = 1.0
Episode 467: Total Reward = 1.0
Episode 468: Total Reward = 1.0
Episode 469: Total Reward = 1.0
Episode 470: Total Reward = 1.0
Episode 471: Total Reward = 1.0
Episode 472: Total Reward = 1.0
Episode 473: Total Reward = 1.0
Episode 474: Total Reward = 1.0
Episode 475: Total Reward = 1.0
Episode 476: Total Reward = 1.0
Episode 477: Total Reward = 1.0
Episode 478: Total Reward = 1.0
Episode 479: Total Reward = 1.0
Episode 480: Total Reward = 1.0
Episode 481: Total Reward = 1.0
Episode 482: Total Reward = 1.0
Episode 483: Total Reward = 1.0
Episode 484: Total Reward = 1.0
Episode 485: Total Reward = 1.0
Episode 486: Total Reward = 1.0
Episode 487: Total Reward = 1.0
Episode 488: Total Reward = 1.0
Episode 489: Total Reward = 1.0
Episode 490: Total Reward = 1.0
Episode 491: Total Reward = 1.0
Episode 492: Total Reward = 1.0
Episode 493: Total Reward = 1.0
Episode 494: Total Reward = 1.0
Episode 495: Total Reward = 1.0
Episode 496: Total Reward = 1.0
Episode 497: Total Reward = 1.0
Episode 498: Total Reward = 1.0
Episode 499: Total Reward = 1.0
Episode 500: Total Reward = 1.0
Episode 501: Total Reward = 1.0
Episode 502: Total Reward = 1.0
Episode 503: Total Reward = 1.0
Episode 504: Total Reward = 1.0
Episode 505: Total Reward = 1.0
Episode 506: Total Reward = 1.0
Episode 507: Total Reward = 1.0
Episode 508: Total Reward = 1.0
Episode 509: Total Reward = 1.0
Episode 510: Total Reward = 1.0
Episode 511: Total Reward = 1.0
Episode 512: Total Reward = 1.0
Episode 513: Total Reward = 1.0
Episode 514: Total Reward = 1.0
Episode 515: Total Reward = 1.0
Episode 516: Total Reward = 1.0
Episode 517: Total Reward = 1.0
Episode 518: Total Reward = 1.0
Episode 519: Total Reward = 1.0
Episode 520: Total Reward = 1.0
Episode 521: Total Reward = 1.0
Episode 522: Total Reward = 1.0
Episode 523: Total Reward = 1.0
Episode 524: Total Reward = 1.0
Episode 525: Total Reward = 1.0
Episode 526: Total Reward = 1.0
Episode 527: Total Reward = 1.0
Episode 528: Total Reward = 1.0
Episode 529: Total Reward = 1.0
Episode 530: Total Reward = 1.0
Episode 531: Total Reward = 1.0
Episode 532: Total Reward = 1.0
Episode 533: Total Reward = 1.0
Episode 534: Total Reward = 1.0
Episode 535: Total Reward = 1.0
Episode 536: Total Reward = 1.0
Episode 537: Total Reward = 1.0
Episode 538: Total Reward = 1.0
Episode 539: Total Reward = 1.0
Episode 540: Total Reward = 1.0
Episode 541: Total Reward = 1.0
Episode 542: Total Reward = 1.0
Episode 543: Total Reward = 1.0
Episode 544: Total Reward = 1.0
Episode 545: Total Reward = 1.0
Episode 546: Total Reward = 1.0
Episode 547: Total Reward = 1.0
Episode 548: Total Reward = 1.0
Episode 549: Total Reward = 1.0
Episode 550: Total Reward = 1.0
Episode 551: Total Reward = 1.0
Episode 552: Total Reward = 1.0
Episode 553: Total Reward = 1.0
Episode 554: Total Reward = 1.0
Episode 555: Total Reward = 1.0
Episode 556: Total Reward = 1.0
Episode 557: Total Reward = 1.0
Episode 558: Total Reward = 1.0
Episode 559: Total Reward = 1.0
Episode 560: Total Reward = 1.0
Episode 561: Total Reward = 1.0
Episode 562: Total Reward = 1.0
Episode 563: Total Reward = 1.0
Episode 564: Total Reward = 1.0
Episode 565: Total Reward = 1.0
Episode 566: Total Reward = 1.0
Episode 567: Total Reward = 1.0
Episode 568: Total Reward = 1.0
Episode 569: Total Reward = 1.0
Episode 570: Total Reward = 1.0
Episode 571: Total Reward = 1.0
Episode 572: Total Reward = 1.0
Episode 573: Total Reward = 1.0
Episode 574: Total Reward = 1.0
Episode 575: Total Reward = 1.0
Episode 576: Total Reward = 1.0
Episode 577: Total Reward = 1.0
Episode 578: Total Reward = 1.0
Episode 579: Total Reward = 1.0
Episode 580: Total Reward = 1.0
Episode 581: Total Reward = 1.0
Episode 582: Total Reward = 1.0
Episode 583: Total Reward = 1.0
Episode 584: Total Reward = 1.0
Episode 585: Total Reward = 1.0
Episode 586: Total Reward = 1.0
Episode 587: Total Reward = 1.0
Episode 588: Total Reward = 1.0
Episode 589: Total Reward = 1.0
Episode 590: Total Reward = 1.0
Episode 591: Total Reward = 1.0
Episode 592: Total Reward = 1.0
Episode 593: Total Reward = 1.0
Episode 594: Total Reward = 1.0
Episode 595: Total Reward = 1.0
Episode 596: Total Reward = 1.0
Episode 597: Total Reward = 1.0
Episode 598: Total Reward = 1.0
Episode 599: Total Reward = 1.0
Episode 600: Total Reward = 1.0
Episode 601: Total Reward = 1.0
Episode 602: Total Reward = 1.0
Episode 603: Total Reward = 1.0
Episode 604: Total Reward = 1.0
Episode 605: Total Reward = 1.0
Episode 606: Total Reward = 1.0
Episode 607: Total Reward = 1.0
Episode 608: Total Reward = 1.0
Episode 609: Total Reward = 1.0
Episode 610: Total Reward = 1.0
Episode 611: Total Reward = 1.0
Episode 612: Total Reward = 1.0
Episode 613: Total Reward = 1.0
Episode 614: Total Reward = 1.0
Episode 615: Total Reward = 1.0
Episode 616: Total Reward = 1.0
Episode 617: Total Reward = 1.0
Episode 618: Total Reward = 1.0
Episode 619: Total Reward = 1.0
Episode 620: Total Reward = 1.0
Episode 621: Total Reward = 1.0
Episode 622: Total Reward = 1.0
Episode 623: Total Reward = 1.0
Episode 624: Total Reward = 1.0
Episode 625: Total Reward = 1.0
Episode 626: Total Reward = 1.0
Episode 627: Total Reward = 1.0
Episode 628: Total Reward = 1.0
Episode 629: Total Reward = 1.0
Episode 630: Total Reward = 1.0
Episode 631: Total Reward = 1.0
Episode 632: Total Reward = 1.0
Episode 633: Total Reward = 1.0
Episode 634: Total Reward = 1.0
Episode 635: Total Reward = 1.0
Episode 636: Total Reward = 1.0
Episode 637: Total Reward = 1.0
Episode 638: Total Reward = 1.0
Episode 639: Total Reward = 1.0
Episode 640: Total Reward = 1.0
Episode 641: Total Reward = 1.0
Episode 642: Total Reward = 1.0
Episode 643: Total Reward = 1.0
Episode 644: Total Reward = 1.0
Episode 645: Total Reward = 1.0
Episode 646: Total Reward = 1.0
Episode 647: Total Reward = 1.0
Episode 648: Total Reward = 1.0
Episode 649: Total Reward = 1.0
Episode 650: Total Reward = 1.0
Episode 651: Total Reward = 1.0
Episode 652: Total Reward = 1.0
Episode 653: Total Reward = 1.0
Episode 654: Total Reward = 1.0
Episode 655: Total Reward = 1.0
Episode 656: Total Reward = 1.0
Episode 657: Total Reward = 1.0
Episode 658: Total Reward = 1.0
Episode 659: Total Reward = 1.0
Episode 660: Total Reward = 1.0
Episode 661: Total Reward = 1.0
Episode 662: Total Reward = 1.0
Episode 663: Total Reward = 1.0
Episode 664: Total Reward = 1.0
Episode 665: Total Reward = 1.0
Episode 666: Total Reward = 1.0
Episode 667: Total Reward = 1.0
Episode 668: Total Reward = 1.0
Episode 669: Total Reward = 1.0
Episode 670: Total Reward = 1.0
Episode 671: Total Reward = 1.0
Episode 672: Total Reward = 1.0
Episode 673: Total Reward = 1.0
Episode 674: Total Reward = 1.0
Episode 675: Total Reward = 1.0
Episode 676: Total Reward = 1.0
Episode 677: Total Reward = 1.0
Episode 678: Total Reward = 1.0
Episode 679: Total Reward = 1.0
Episode 680: Total Reward = 1.0
Episode 681: Total Reward = 1.0
Episode 682: Total Reward = 1.0
Episode 683: Total Reward = 1.0
Episode 684: Total Reward = 1.0
Episode 685: Total Reward = 1.0
Episode 686: Total Reward = 1.0
Episode 687: Total Reward = 1.0
Episode 688: Total Reward = 1.0
Episode 689: Total Reward = 1.0
Episode 690: Total Reward = 1.0
Episode 691: Total Reward = 1.0
Episode 692: Total Reward = 1.0
Episode 693: Total Reward = 1.0
Episode 694: Total Reward = 1.0
Episode 695: Total Reward = 1.0
Episode 696: Total Reward = 1.0
Episode 697: Total Reward = 1.0
Episode 698: Total Reward = 1.0
Episode 699: Total Reward = 1.0
Episode 700: Total Reward = 1.0
Episode 701: Total Reward = 1.0
Episode 702: Total Reward = 1.0
Episode 703: Total Reward = 1.0
Episode 704: Total Reward = 1.0
Episode 705: Total Reward = 1.0
Episode 706: Total Reward = 1.0
Episode 707: Total Reward = 1.0
Episode 708: Total Reward = 1.0
Episode 709: Total Reward = 1.0
Episode 710: Total Reward = 1.0
Episode 711: Total Reward = 1.0
Episode 712: Total Reward = 1.0
Episode 713: Total Reward = 1.0
Episode 714: Total Reward = 1.0
Episode 715: Total Reward = 1.0
Episode 716: Total Reward = 1.0
Episode 717: Total Reward = 1.0
Episode 718: Total Reward = 1.0
Episode 719: Total Reward = 1.0
Episode 720: Total Reward = 1.0
Episode 721: Total Reward = 1.0
Episode 722: Total Reward = 1.0
Episode 723: Total Reward = 1.0
Episode 724: Total Reward = 1.0
Episode 725: Total Reward = 1.0
Episode 726: Total Reward = 1.0
Episode 727: Total Reward = 1.0
Episode 728: Total Reward = 1.0
Episode 729: Total Reward = 1.0
Episode 730: Total Reward = 1.0
Episode 731: Total Reward = 1.0
Episode 732: Total Reward = 1.0
Episode 733: Total Reward = 1.0
Episode 734: Total Reward = 1.0
Episode 735: Total Reward = 1.0
Episode 736: Total Reward = 1.0
Episode 737: Total Reward = 1.0
Episode 738: Total Reward = 1.0
Episode 739: Total Reward = 1.0
Episode 740: Total Reward = 1.0
Episode 741: Total Reward = 1.0
Episode 742: Total Reward = 1.0
Episode 743: Total Reward = 1.0
Episode 744: Total Reward = 1.0
Episode 745: Total Reward = 1.0
Episode 746: Total Reward = 1.0
Episode 747: Total Reward = 1.0
Episode 748: Total Reward = 1.0
Episode 749: Total Reward = 1.0
Episode 750: Total Reward = 1.0
Episode 751: Total Reward = 1.0
Episode 752: Total Reward = 1.0
Episode 753: Total Reward = 1.0
Episode 754: Total Reward = 1.0
Episode 755: Total Reward = 1.0
Episode 756: Total Reward = 1.0
Episode 757: Total Reward = 1.0
Episode 758: Total Reward = 1.0
Episode 759: Total Reward = 1.0
Episode 760: Total Reward = 1.0
Episode 761: Total Reward = 1.0
Episode 762: Total Reward = 1.0
Episode 763: Total Reward = 1.0
Episode 764: Total Reward = 1.0
Episode 765: Total Reward = 1.0
Episode 766: Total Reward = 1.0
Episode 767: Total Reward = 1.0
Episode 768: Total Reward = 1.0
Episode 769: Total Reward = 1.0
Episode 770: Total Reward = 1.0
Episode 771: Total Reward = 1.0
Episode 772: Total Reward = 1.0
Episode 773: Total Reward = 1.0
Episode 774: Total Reward = 1.0
Episode 775: Total Reward = 1.0
Episode 776: Total Reward = 1.0
Episode 777: Total Reward = 1.0
Episode 778: Total Reward = 1.0
Episode 779: Total Reward = 1.0
Episode 780: Total Reward = 1.0
Episode 781: Total Reward = 1.0
Episode 782: Total Reward = 1.0
Episode 783: Total Reward = 1.0
Episode 784: Total Reward = 1.0
Episode 785: Total Reward = 1.0
Episode 786: Total Reward = 1.0
Episode 787: Total Reward = 1.0
Episode 788: Total Reward = 1.0
Episode 789: Total Reward = 1.0
Episode 790: Total Reward = 1.0
Episode 791: Total Reward = 1.0
Episode 792: Total Reward = 1.0
Episode 793: Total Reward = 1.0
Episode 794: Total Reward = 1.0
Episode 795: Total Reward = 1.0
Episode 796: Total Reward = 1.0
Episode 797: Total Reward = 1.0
Episode 798: Total Reward = 1.0
Episode 799: Total Reward = 1.0
Episode 800: Total Reward = 1.0
Episode 801: Total Reward = 1.0
Episode 802: Total Reward = 1.0
Episode 803: Total Reward = 1.0
Episode 804: Total Reward = 1.0
Episode 805: Total Reward = 1.0
Episode 806: Total Reward = 1.0
Episode 807: Total Reward = 1.0
Episode 808: Total Reward = 1.0
Episode 809: Total Reward = 1.0
Episode 810: Total Reward = 1.0
Episode 811: Total Reward = 1.0
Episode 812: Total Reward = 1.0
Episode 813: Total Reward = 1.0
Episode 814: Total Reward = 1.0
Episode 815: Total Reward = 1.0
Episode 816: Total Reward = 1.0
Episode 817: Total Reward = 1.0
Episode 818: Total Reward = 1.0
Episode 819: Total Reward = 1.0
Episode 820: Total Reward = 1.0
Episode 821: Total Reward = 1.0
Episode 822: Total Reward = 1.0
Episode 823: Total Reward = 1.0
Episode 824: Total Reward = 1.0
Episode 825: Total Reward = 1.0
Episode 826: Total Reward = 1.0
Episode 827: Total Reward = 1.0
Episode 828: Total Reward = 1.0
Episode 829: Total Reward = 1.0
Episode 830: Total Reward = 1.0
Episode 831: Total Reward = 1.0
Episode 832: Total Reward = 1.0
Episode 833: Total Reward = 1.0
Episode 834: Total Reward = 1.0
Episode 835: Total Reward = 1.0
Episode 836: Total Reward = 1.0
Episode 837: Total Reward = 1.0
Episode 838: Total Reward = 1.0
Episode 839: Total Reward = 1.0
Episode 840: Total Reward = 1.0
Episode 841: Total Reward = 1.0
Episode 842: Total Reward = 1.0
Episode 843: Total Reward = 1.0
Episode 844: Total Reward = 1.0
Episode 845: Total Reward = 1.0
Episode 846: Total Reward = 1.0
Episode 847: Total Reward = 1.0
Episode 848: Total Reward = 1.0
Episode 849: Total Reward = 1.0
Episode 850: Total Reward = 1.0
Episode 851: Total Reward = 1.0
Episode 852: Total Reward = 1.0
Episode 853: Total Reward = 1.0
Episode 854: Total Reward = 1.0
Episode 855: Total Reward = 1.0
Episode 856: Total Reward = 1.0
Episode 857: Total Reward = 1.0
Episode 858: Total Reward = 1.0
Episode 859: Total Reward = 1.0
Episode 860: Total Reward = 1.0
Episode 861: Total Reward = 1.0
Episode 862: Total Reward = 1.0
Episode 863: Total Reward = 1.0
Episode 864: Total Reward = 1.0
Episode 865: Total Reward = 1.0
Episode 866: Total Reward = 1.0
Episode 867: Total Reward = 1.0
Episode 868: Total Reward = 1.0
Episode 869: Total Reward = 1.0
Episode 870: Total Reward = 1.0
Episode 871: Total Reward = 1.0
Episode 872: Total Reward = 1.0
Episode 873: Total Reward = 1.0
Episode 874: Total Reward = 1.0
Episode 875: Total Reward = 1.0
Episode 876: Total Reward = 1.0
Episode 877: Total Reward = 1.0
Episode 878: Total Reward = 1.0
Episode 879: Total Reward = 1.0
Episode 880: Total Reward = 1.0
Episode 881: Total Reward = 1.0
Episode 882: Total Reward = 1.0
Episode 883: Total Reward = 1.0
Episode 884: Total Reward = 1.0
Episode 885: Total Reward = 1.0
Episode 886: Total Reward = 1.0
Episode 887: Total Reward = 1.0
Episode 888: Total Reward = 1.0
Episode 889: Total Reward = 1.0
Episode 890: Total Reward = 1.0
Episode 891: Total Reward = 1.0
Episode 892: Total Reward = 1.0
Episode 893: Total Reward = 1.0
Episode 894: Total Reward = 1.0
Episode 895: Total Reward = 1.0
Episode 896: Total Reward = 1.0
Episode 897: Total Reward = 1.0
Episode 898: Total Reward = 1.0
Episode 899: Total Reward = 1.0
Episode 900: Total Reward = 1.0
Episode 901: Total Reward = 1.0
Episode 902: Total Reward = 1.0
Episode 903: Total Reward = 1.0
Episode 904: Total Reward = 1.0
Episode 905: Total Reward = 1.0
Episode 906: Total Reward = 1.0
Episode 907: Total Reward = 1.0
Episode 908: Total Reward = 1.0
Episode 909: Total Reward = 1.0
Episode 910: Total Reward = 1.0
Episode 911: Total Reward = 1.0
Episode 912: Total Reward = 1.0
Episode 913: Total Reward = 1.0
Episode 914: Total Reward = 1.0
Episode 915: Total Reward = 1.0
Episode 916: Total Reward = 1.0
Episode 917: Total Reward = 1.0
Episode 918: Total Reward = 1.0
Episode 919: Total Reward = 1.0
Episode 920: Total Reward = 1.0
Episode 921: Total Reward = 1.0
Episode 922: Total Reward = 1.0
Episode 923: Total Reward = 1.0
Episode 924: Total Reward = 1.0
Episode 925: Total Reward = 1.0
Episode 926: Total Reward = 1.0
Episode 927: Total Reward = 1.0
Episode 928: Total Reward = 1.0
Episode 929: Total Reward = 1.0
Episode 930: Total Reward = 1.0
Episode 931: Total Reward = 1.0
Episode 932: Total Reward = 1.0
Episode 933: Total Reward = 1.0
Episode 934: Total Reward = 1.0
Episode 935: Total Reward = 1.0
Episode 936: Total Reward = 1.0
Episode 937: Total Reward = 1.0
Episode 938: Total Reward = 1.0
Episode 939: Total Reward = 1.0
Episode 940: Total Reward = 1.0
Episode 941: Total Reward = 1.0
Episode 942: Total Reward = 1.0
Episode 943: Total Reward = 1.0
Episode 944: Total Reward = 1.0
Episode 945: Total Reward = 1.0
Episode 946: Total Reward = 1.0
Episode 947: Total Reward = 1.0
Episode 948: Total Reward = 1.0
Episode 949: Total Reward = 1.0
Episode 950: Total Reward = 1.0
Episode 951: Total Reward = 1.0
Episode 952: Total Reward = 1.0
Episode 953: Total Reward = 1.0
Episode 954: Total Reward = 1.0
Episode 955: Total Reward = 1.0
Episode 956: Total Reward = 1.0
Episode 957: Total Reward = 1.0
Episode 958: Total Reward = 1.0
Episode 959: Total Reward = 1.0
Episode 960: Total Reward = 1.0
Episode 961: Total Reward = 1.0
Episode 962: Total Reward = 1.0
Episode 963: Total Reward = 1.0
Episode 964: Total Reward = 1.0
Episode 965: Total Reward = 1.0
Episode 966: Total Reward = 1.0
Episode 967: Total Reward = 1.0
Episode 968: Total Reward = 1.0
Episode 969: Total Reward = 1.0
Episode 970: Total Reward = 1.0
Episode 971: Total Reward = 1.0
Episode 972: Total Reward = 1.0
Episode 973: Total Reward = 1.0
Episode 974: Total Reward = 1.0
Episode 975: Total Reward = 1.0
Episode 976: Total Reward = 1.0
Episode 977: Total Reward = 1.0
Episode 978: Total Reward = 1.0
Episode 979: Total Reward = 1.0
Episode 980: Total Reward = 1.0
Episode 981: Total Reward = 1.0
Episode 982: Total Reward = 1.0
Episode 983: Total Reward = 1.0
Episode 984: Total Reward = 1.0
Episode 985: Total Reward = 1.0
Episode 986: Total Reward = 1.0
Episode 987: Total Reward = 1.0
Episode 988: Total Reward = 1.0
Episode 989: Total Reward = 1.0
Episode 990: Total Reward = 1.0
Episode 991: Total Reward = 1.0
Episode 992: Total Reward = 1.0
Episode 993: Total Reward = 1.0
Episode 994: Total Reward = 1.0
Episode 995: Total Reward = 1.0
Episode 996: Total Reward = 1.0
Episode 997: Total Reward = 1.0
Episode 998: Total Reward = 1.0
Episode 999: Total Reward = 1.0
Optimal Path:
Step 0: (0, 0)
Step 1: (1, 0)
Step 2: (2, 0)
Step 3: (2, 1)
Step 4: (3, 1)
Step 5: (3, 2)
Step 6: (3, 3)
```

Chapter 4: Types of Supervised Learning


Supervised Learning
Classification (defined Labels)
Regression (no Labels defined)

1. Supervised Learning: Classification

`Classification` is a type of supervised learning where the algorithm learns to categorize data into predefined classes or labels. It involves training a model to predict the class label of a given input based on its features.

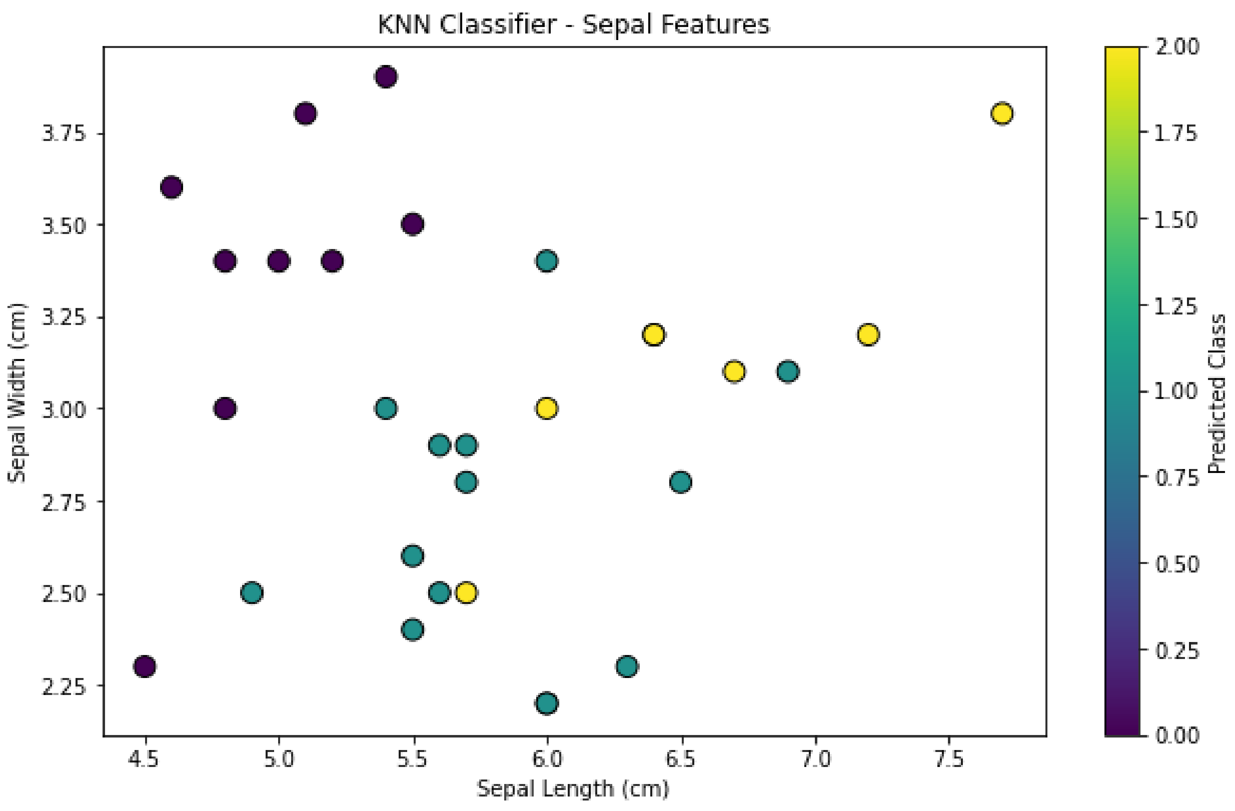
Example: Email spam detection, where the algorithm classifies emails as either "spam" or "not spam" based on their content.

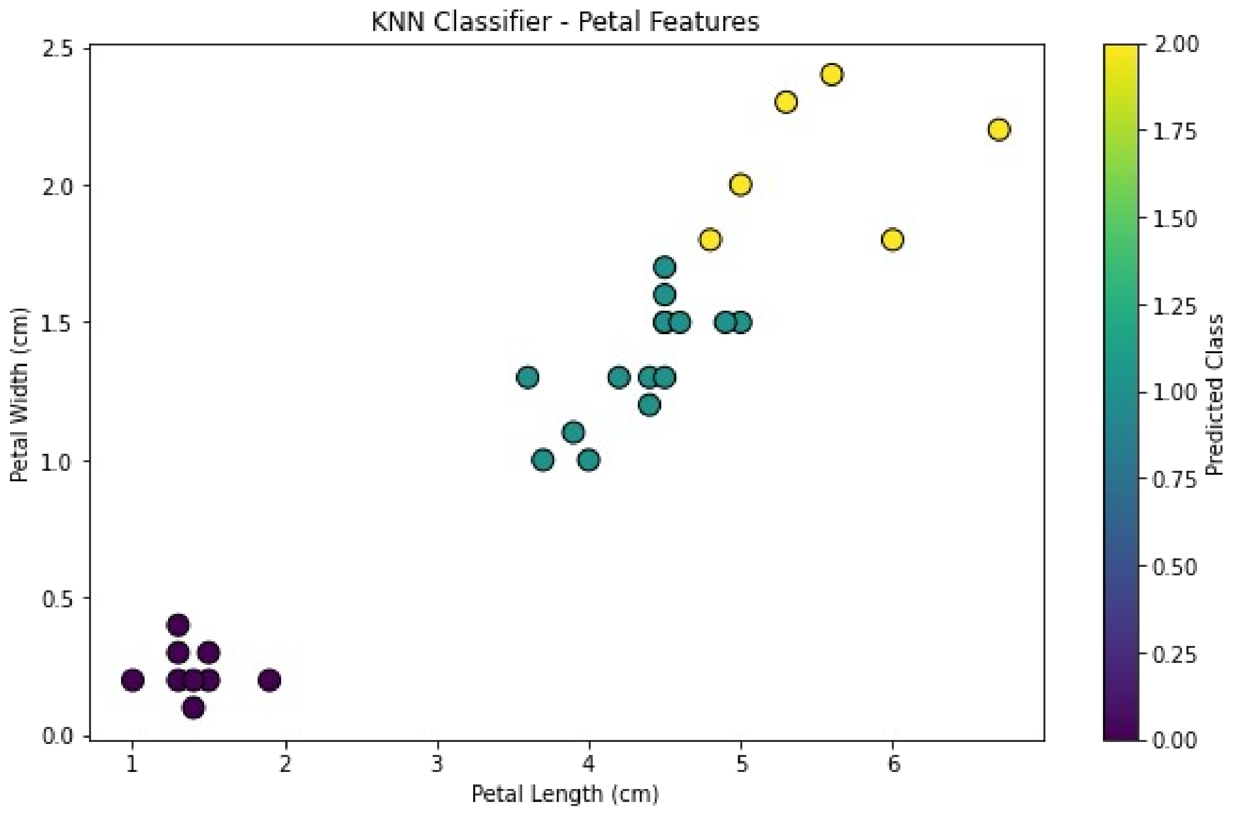
Example Code:

```python
from sklearn import datasets, model_selection
from sklearn.neighbors import KNeighborsClassifier

# Load a sample dataset (e.g., Iris dataset)
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
model_selection.train_test_split(iris.data, iris.target,
test_size=0.2)

# Create a k-nearest neighbors classifier
knn = KNeighborsClassifier(n_neighbors=3)

# Train the classifier on the training data
knn.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn.predict(X_test)
y_pred
```

```
array([2, 1, 2, 1, 0, 1, 0, 2, 1, 2, 0, 2, 1, 0, 1, 0, 1, 0, 1, 1, 1,
1,
       1, 1, 2, 1, 0, 0, 1, 0])
```

```python
import matplotlib.pyplot as plt

# Create a scatter plot for the first two features (sepal length and
sepal width)
plt.figure(figsize=(10, 6))
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='viridis',
marker='o', edgecolor='k', s=100)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('KNN Classifier - Sepal Features')
plt.colorbar(label='Predicted Class')
plt.show()

# Create a scatter plot for the next two features (petal length and
petal width)
plt.figure(figsize=(10, 6))
plt.scatter(X_test[:, 2], X_test[:, 3], c=y_pred, cmap='viridis',
marker='o', edgecolor='k', s=100)
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('KNN Classifier - Petal Features')
plt.colorbar(label='Predicted Class')
plt.show()
```


KNN Classifier - Sepal Features
Sepal Width (cm)
Sepal Length (cm)
Predicted Class
4.5
5.0
5.5
6.0
6.5
7.0
7.5
2.25
2.50
2.75
3.00
3.25
3.50
3.75
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


KNN Classifier - Petal Features
Petal Width (cm)
Petal Length (cm)
Predicted Class
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

2. Supervised Learning: Regression

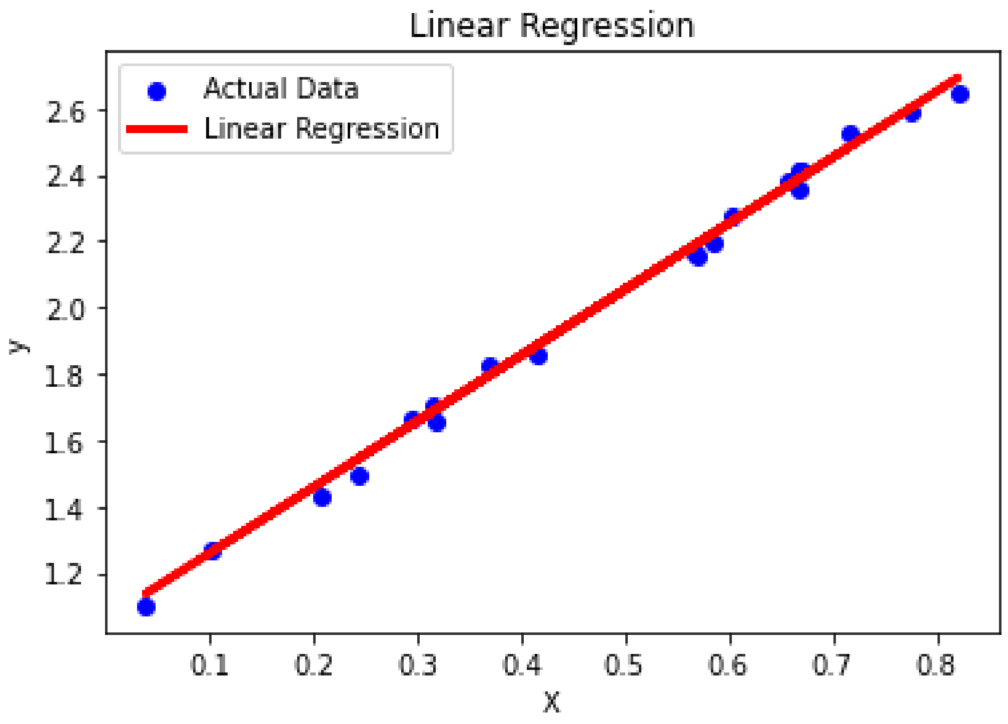
`Regression` is another type of supervised learning that deals with predicting continuous numerical values or outcomes. Instead of classes, regression models learn to map input features to a continuous target variable.

Example: Predicting house prices based on features such as square footage, number of bedrooms, and location.

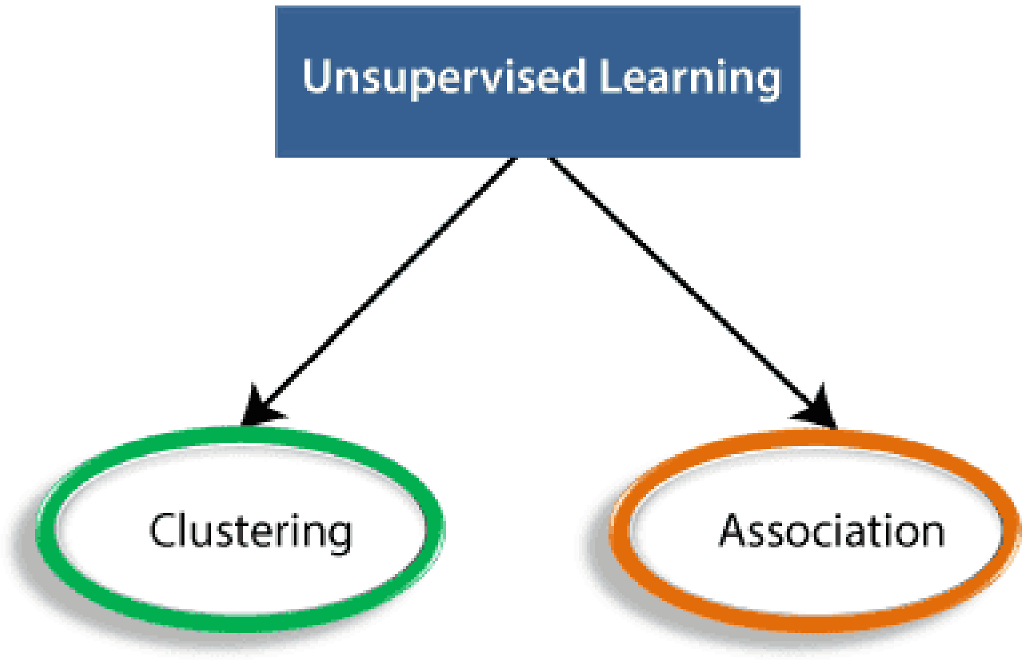
Example Code:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Generate synthetic data
np.random.seed(0)
X = np.random.rand(100, 1)  # Independent variable
y = 2 * X + 1 + 0.1 * np.random.rand(100, 1)  # Dependent variable
with some noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
y, test_size=0.2)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the Root Mean Square Error (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
y_pred
```

```
array([[2.38770167],
       [1.88462362],
       [2.25902634],
       [2.1942048 ],
       [2.38642051],
       [1.6487144 ],
       [2.19069603],
       [1.7931897 ],
       [1.13726865],
       [2.39412577],
       [2.69339757],
       [2.36564614],
       [2.48506718],
       [1.69219133],
       [2.22668098],
       [1.5457801 ],
       [1.68710612],
       [1.26238114],
       [1.47502264],
       [2.60032598]])

import matplotlib.pyplot as plt

# Scatter plot of the data points
plt.scatter(X_test, y_test, color='blue', label='Actual Data')

# Regression line
plt.plot(X_test, y_pred, color='red', linewidth=3, label='Linear
Regression')

# Labels and legend
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression')
plt.legend()

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



Chapter 5: Types of Unsupervised Learning



1. Unsupervised Learning: Clustering

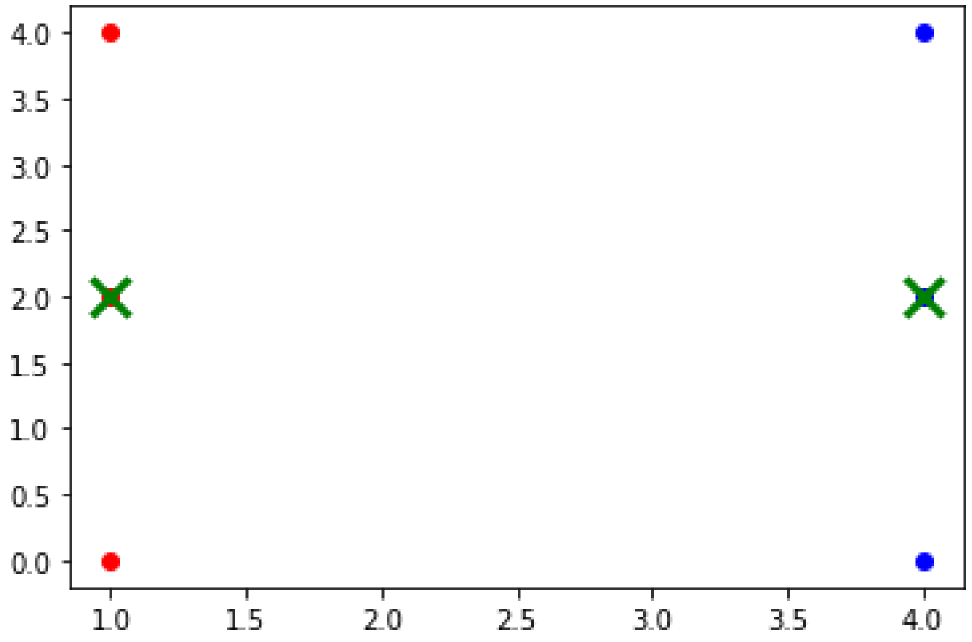
`Clustering` is a type of unsupervised learning where the algorithm groups similar data points together based on their features. The goal is to identify patterns or natural groupings within the data without prior knowledge of class labels.

Example: Customer segmentation, where customers are grouped into segments based on their purchasing behavior, allowing businesses to tailor marketing strategies.

Example Code:

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Generate sample data
data = [[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]]

# Specify the number of clusters (K)
kmeans = KMeans(n_clusters=2)

# Fit the model to the data
kmeans.fit(data)
```

```
# Get cluster labels and centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Visualize the clusters
for i in range(len(data)):
    color = 'r' if labels[i] == 0 else 'b'
    plt.scatter(data[i][0], data[i][1], c=color)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200,
linewidths=3, color='g')
plt.show()
```



2. Unsupervised Learning: Association

`Association rule mining` is another type of unsupervised learning where the algorithm discovers interesting relationships or associations between variables in a dataset. It identifies patterns such as "if X, then Y" or "X implies Y" in transactional data.

Example: Market basket analysis, where the algorithm identifies which products are frequently purchased together in a shopping cart.

Example Code:

```
from mlxtend.frequent_patterns import apriori, association_rules
import pandas as pd

# Create a sample transaction dataset
data = {'TransactionID': [1, 2, 3, 4, 5],
        'Items': [['apple', 'banana', 'cherry'],
                   ['apple', 'banana'],
                   ['apple', 'cherry'],
                   ['banana', 'cherry'],
                   ['apple', 'banana', 'cherry']]}

df = pd.DataFrame(data)

# Convert items to a binary format using one-hot encoding
df_encoded = df['Items'].str.join('|').str.get_dummies()

# Perform Apriori algorithm to find frequent itemsets with lower
min_support
frequent_itemsets = apriori(df_encoded, min_support=0.2,
use_colnames=True)

# Generate association rules with lower min_threshold
rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=0.8)

# Display association rules
rules
```

```
C:\Users\avino\anaconda3\lib\site-packages\mlxtend\frequent_patterns\
fpcommon.py:110: DeprecationWarning: DataFrames with non-bool types
result in worse computationalperformance and their support might be
discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

```
         antecedents       consequents  antecedent support  \
0           (banana)           (apple)                 0.8
1            (apple)          (banana)                 0.8
2            (apple)          (cherry)                 0.8
3           (cherry)           (apple)                 0.8
4           (banana)          (cherry)                 0.8
5           (cherry)          (banana)                 0.8
6    (banana, apple)          (cherry)                 0.6
7   (banana, cherry)           (apple)                 0.6
8    (apple, cherry)          (banana)                 0.6
9           (banana)   (apple, cherry)                 0.8
10           (apple)  (banana, cherry)                 0.8
11          (cherry)   (banana, apple)                 0.8

    consequent support  support  confidence      lift  leverage
conviction  \
0                  0.8      0.6    0.750000  0.937500     -0.04
0.8
1                  0.8      0.6    0.750000  0.937500     -0.04
0.8
2                  0.8      0.6    0.750000  0.937500     -0.04
0.8
```

```
3                    0.8       0.6    0.750000  0.937500     -0.04
0.8
4                    0.8       0.6    0.750000  0.937500     -0.04
0.8
5                    0.8       0.6    0.750000  0.937500     -0.04
0.8
6                    0.8       0.4    0.666667  0.833333     -0.08
0.6
7                    0.8       0.4    0.666667  0.833333     -0.08
0.6
8                    0.8       0.4    0.666667  0.833333     -0.08
0.6
9                    0.6       0.4    0.500000  0.833333     -0.08
0.8
10                   0.6       0.4    0.500000  0.833333     -0.08
0.8
11                   0.6       0.4    0.500000  0.833333     -0.08
0.8

    zhangs_metric
0       -0.250000
1       -0.250000
2       -0.250000
3       -0.250000
4       -0.250000
5       -0.250000
6       -0.333333
7       -0.333333
8       -0.333333
9       -0.500000
10      -0.500000
11      -0.500000
```

Chapter 6: Multiple Regression (Non-Linear):

`Multiple regression` is a statistical method used in machine learning that extends simple linear regression to model the relationship between a dependent variable (target) and two or more independent variables (features or predictors). In `multiple regression`, you can have both `linear` and `non-linear` relationships between the independent variables and the dependent variable.

Explanation:

1. `Multiple Regression:` In multiple regression, the goal is to create a model that predicts a continuous target variable based on multiple input features. The model assumes a linear relationship between the features and the target variable.

2. `Non-Linear Relationships:` While multiple regression typically assumes linear relationships, it can also handle non-linear relationships when appropriate transformations of the independent variables are included in the model.

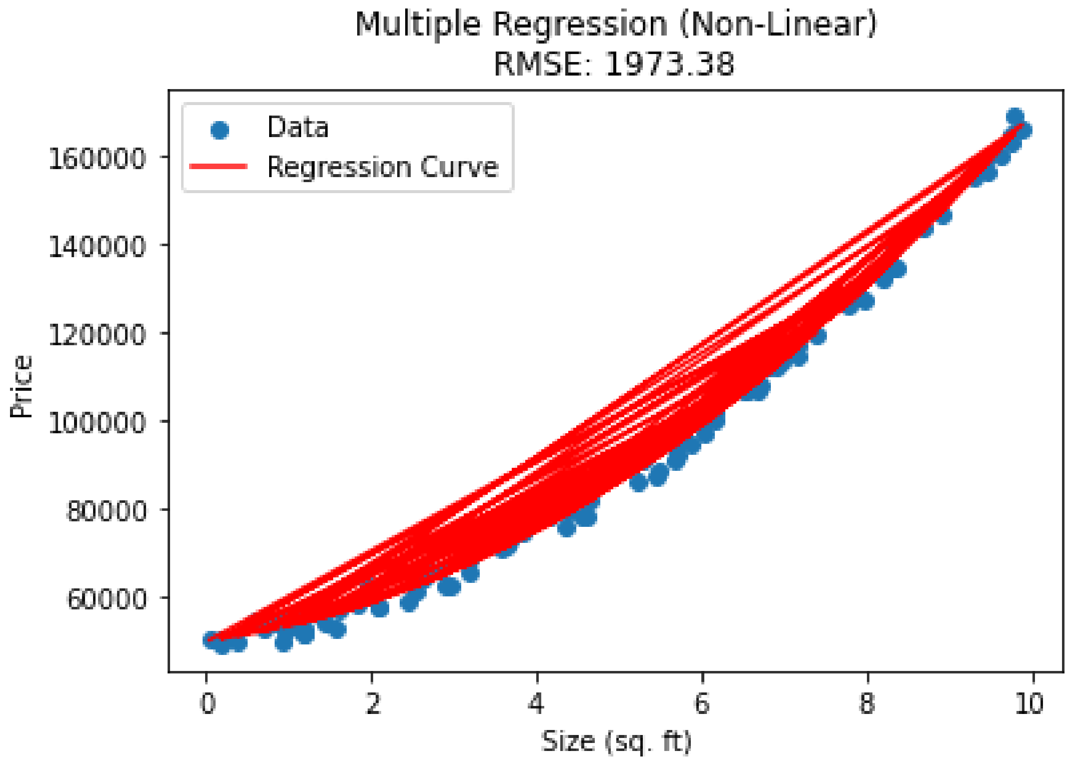
Example:

Let's consider an example where we want to predict the price of a house based on two independent variables: the size of the house (in square feet) and the age of the house (in years). In this case, the relationship between the size and price of the house may be linear, but the relationship between the age and price may not be linear. We can handle this non-linear relationship by including polynomial terms in the regression model.

Example Code (Multiple Regression with Non-Linear Relationship):

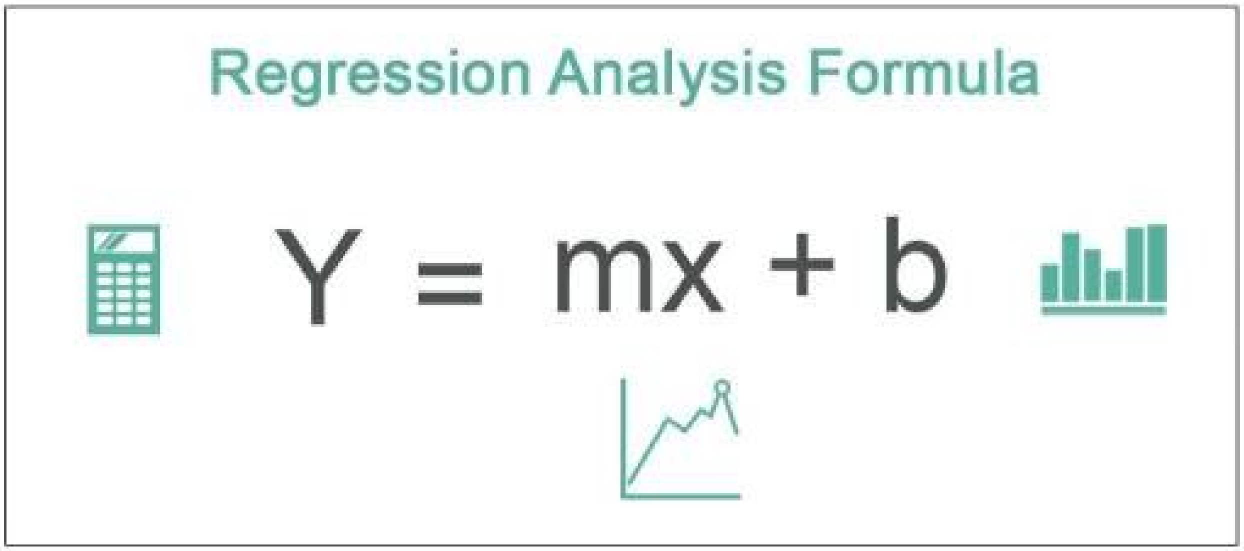
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.random.rand(100, 1) * 10  # House size (in square feet)
y = 50000 + 2000 * X + 1000 * X**2 + np.random.randn(100, 1) * 2000  #
House price

# Create a DataFrame
data = pd.DataFrame({'Size': X.flatten(), 'Price': y.flatten()})

# Extract features and target
X = data[['Size']]
y = data['Price']

# Add polynomial features (in this case, up to degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Create a multiple regression model
model = LinearRegression()

# Fit the model to the data
model.fit(X_poly, y)

# Make predictions
y_pred = model.predict(X_poly)

# Calculate the Root Mean Square Error (RMSE)
rmse = np.sqrt(mean_squared_error(y, y_pred))

# Convert Pandas Series to NumPy arrays for plotting
```

```
X_values = X['Size'].values
y_values = y.values
y_pred_values = y_pred.flatten()

# Plot the data and regression curve
plt.scatter(X_values, y_values, label='Data')
plt.plot(X_values, y_pred_values, color='red', label='Regression
Curve')
plt.xlabel('Size (sq. ft)')
plt.ylabel('Price')
plt.legend()
plt.title(f'Multiple Regression (Non-Linear)\nRMSE: {rmse:.2f}')
plt.show()
```



Chapter 7: Regression Analysis in Machine Learning:

`Regression analysis` is a fundamental technique in machine learning and statistics used to model the relationship between a dependent variable (also known as the target or response variable) and one or more independent variables (predictors or features). The primary objective of regression analysis is to understand and predict the value of the dependent variable based on the values of the independent variables. Regression models come in various forms and can be used for both simple and complex predictive tasks.



key concepts and components related to regression analysis in machine learning:

- This is the variable we aim to predict or explain. It's the outcome or target variable that we want our regression model to estimate.
- These are the variables that influence or explain changes in the dependent variable. They are also referred to as predictors or features.

Multicollinearity occurs when two or more independent variables in a regression model are highly correlated. This can lead to problems in interpreting the model because it becomes challenging to separate the individual effects of the correlated variables. It can also make the model unstable and less reliable.

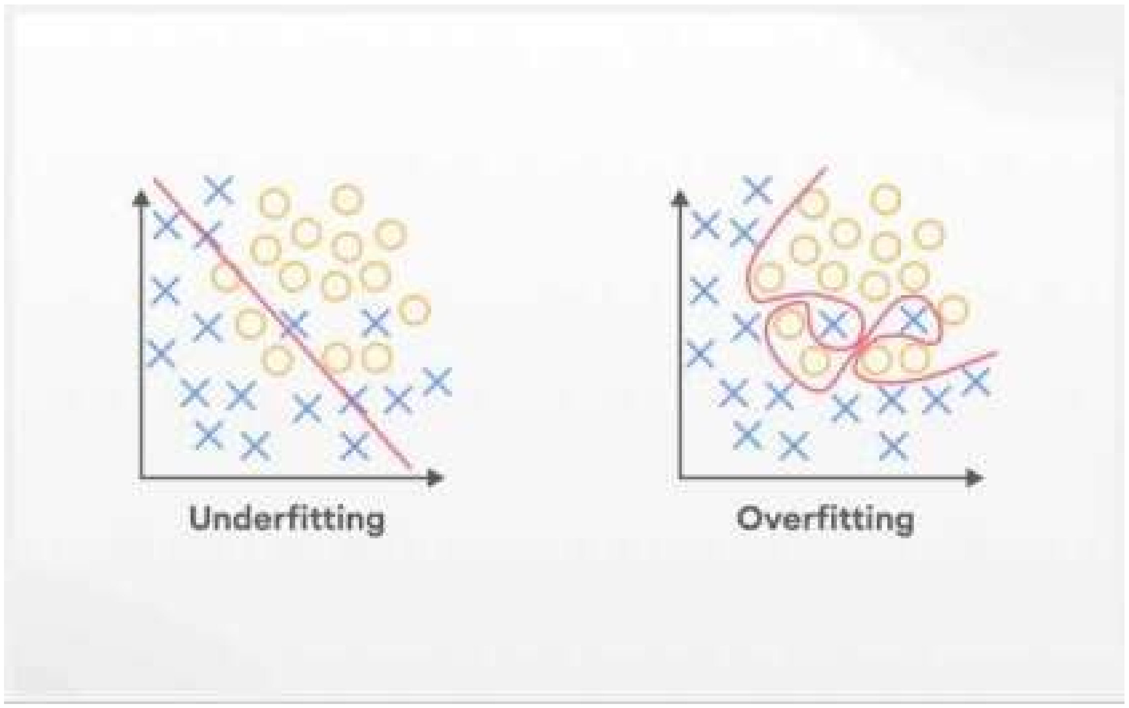
Feature engineering is the process of creating new features or modifying existing ones in a dataset to improve the performance of machine learning models. It involves selecting, transforming, or creating features that are most relevant to the problem and can help the model make better predictions.

Cross-validation is a technique used to assess the performance of a machine learning model. It involves dividing the data into multiple subsets (folds), training the model on different subsets, and evaluating its performance on the remaining data. Cross-validation helps estimate how well a model will generalize to unseen data.

Regularization is a technique used to prevent overfitting in machine learning models. It involves adding a penalty term to the model's objective function that discourages large coefficients. Common regularization methods include L1 (Lasso) and L2 (Ridge) regularization, which help control the complexity of the model.

Model interpretability refers to the ability to understand and explain the predictions made by a machine learning model. Interpretable models are easier to analyze and trust. Complex models, like deep neural networks, often lack interpretability, while simpler models like linear regression are more interpretable.

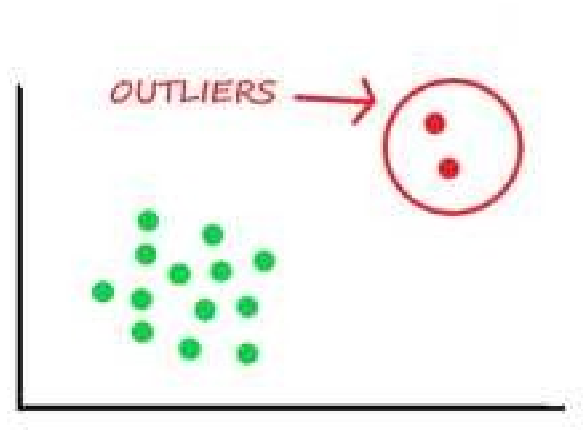
- Underfitting happens when a machine learning model is too simple to capture the underlying patterns in the data. It occurs when the model is not complex enough to represent the relationships between variables accurately. An underfit model performs poorly on both training and testing data.
- Overfitting occurs when a machine learning model is too complex and fits the training data too closely, capturing noise or random fluctuations in the data. As a result, the model performs well on the training data but poorly on unseen data because it fails to generalize. Regularization techniques are often used to combat overfitting.



Outliers are data points that significantly differ from the majority of the data in a dataset. They are often seen as anomalies or unusual observations that don't conform to the expected pattern. Here are some key points about outliers:

- `Identification`: Outliers can be identified through various methods, such as statistical tests, data visualization, or domain knowledge. Common techniques include box plots, scatter plots, and Z-scores.
- `Causes`: Outliers can arise due to various reasons, including data entry errors, measurement errors, natural variability, or genuine extreme observations.
- `Impact`: Outliers can have a significant impact on statistical analyses and machine learning models. They can skew summary statistics, affect the accuracy of predictive models, and lead to incorrect conclusions.

- `Treatment`: Depending on the context and cause, outliers can be handled in different ways. Options include removing them, transforming the data, or using robust statistical methods that are less sensitive to outliers.
- `Visualization`: Data visualization is a powerful tool for identifying outliers. Box plots, scatter plots, and histograms can reveal the presence of extreme values.
- `Domain Knowledge`: Sometimes, outliers are not errors but valuable insights. In such cases, domain knowledge is crucial to understanding why these outliers exist and what they represent.
- `Multivariate Outliers`: Outliers can also exist in multidimensional data, where they may not be obvious in individual dimensions but become apparent when considering multiple dimensions together.



Chapter 8: Types of Regression

There are several types of regression techniques used in machine learning and statistics, each tailored to different types of data and modeling objectives. Here are some common types of regression:

- `Simple Linear Regression`: Models the relationship between a single independent variable and a continuous dependent variable with a linear equation (a straight line).
- `Multiple Linear Regression`: Extends simple linear regression to include two or more independent variables to predict a continuous dependent variable.

- Used for binary classification tasks, where the dependent variable is binary (e.g., 0 or 1).
- Models the relationship between independent variables and the log-odds of the binary outcome.

- Models non-linear relationships between the independent and dependent variables by including polynomial terms of the independent variables.
- Utilizes support vector machines to predict continuous values by finding a hyperplane that best fits the data while minimizing errors.
- Uses decision trees to model relationships between independent variables and continuous dependent variables.
- An ensemble technique that combines multiple decision trees to improve predictive accuracy.
- A form of linear regression with L2 regularization to prevent overfitting by adding a penalty term to the loss function.
- Another form of linear regression with L1 regularization, which can lead to feature selection by driving some coefficients to zero.
- Combines L1 (Lasso) and L2 (Ridge) regularization to balance their effects.
- Applies Bayesian statistical techniques to regression models, allowing for uncertainty estimates in predictions.
- Used when the dependent variable follows a Poisson distribution, often in count data or event prediction.
- Models relationships between categorical variables by taking the natural logarithm of the dependent variable.
- Estimates conditional quantiles of the dependent variable rather than its mean.
- Robust regression technique that is resistant to outliers by iteratively fitting models to subsets of data.
- General category for regression techniques that model non-linear relationships using various mathematical functions, such as exponential or sigmoid functions.
- Combines ridge regression with kernel methods to handle non-linear data transformations.
- A feature selection technique that selects the most significant features for the regression model.
- Extends regression models to predict multiple dependent variables simultaneously.

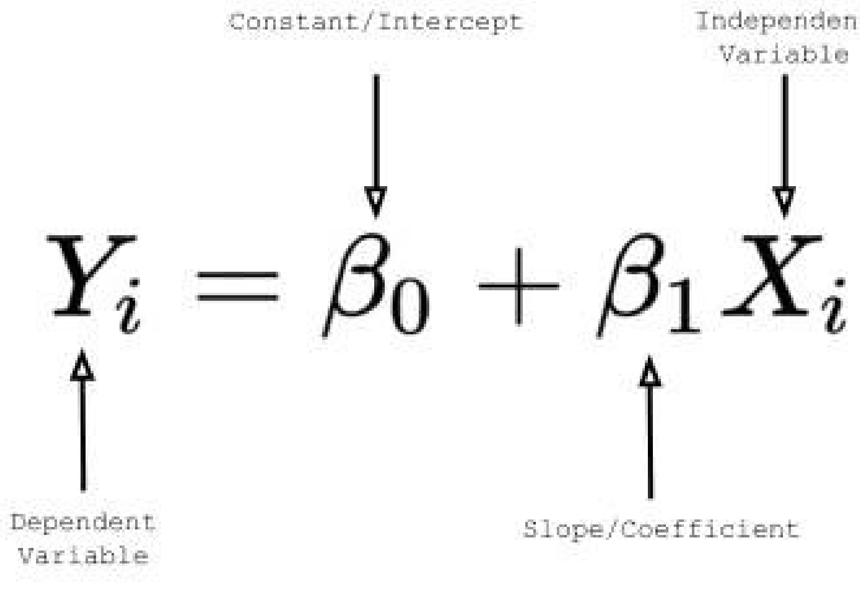
Chapter 9: Linear Regression

`Linear regression` is a simple and widely used statistical technique that models the relationship between a dependent variable (target) and one or more independent variables (predictors or features) using a linear equation. It assumes that this relationship can be represented by a straight line.

Types of Linear Regression:

1.Simple Linear Regression:

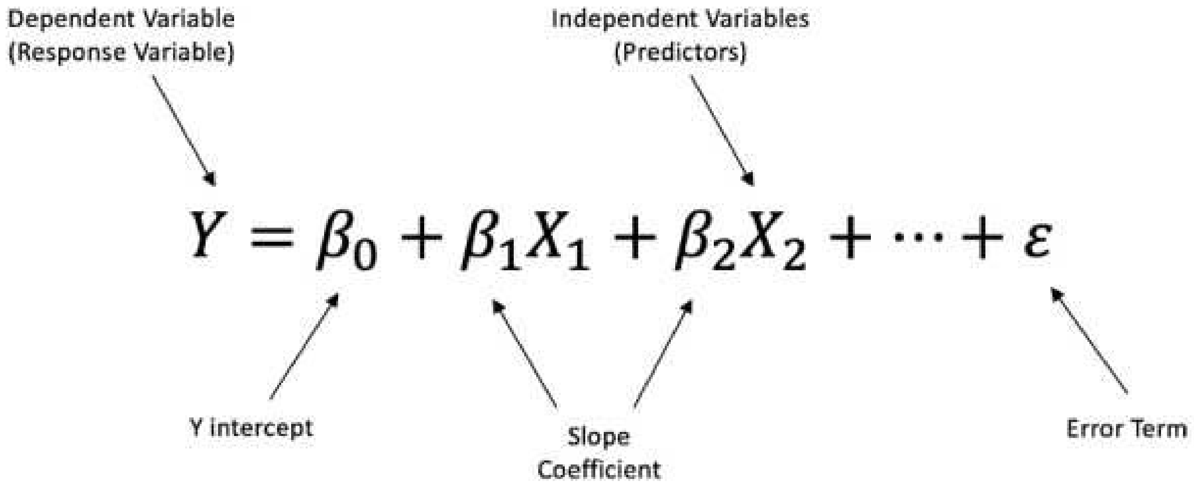
- In simple linear regression, there is only one independent variable that is used to predict a single dependent variable.
- The linear relationship between the independent and dependent variables is represented by a straight line equation:



2.Multiple Linear Regression:

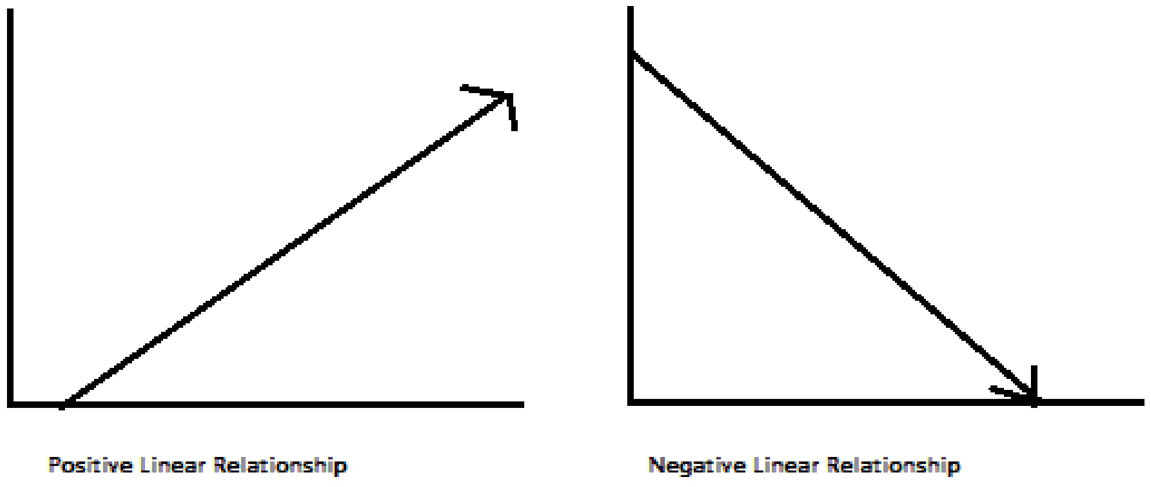
- Multiple linear regression extends simple linear regression to include two or more independent variables to predict a single dependent variable.

- The linear relationship is represented as:


Dependent Variable (Response Variable)
Independent Variables (Predictors)
$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \varepsilon$
Y intercept
Slope Coefficient
Error Term

3. Linear Regression Line:

The linear regression line represents the relationship between the independent and dependent variables in a linear regression model. It is a straight line that best fits the data points. The line is defined by its slope (b) and intercept (a).

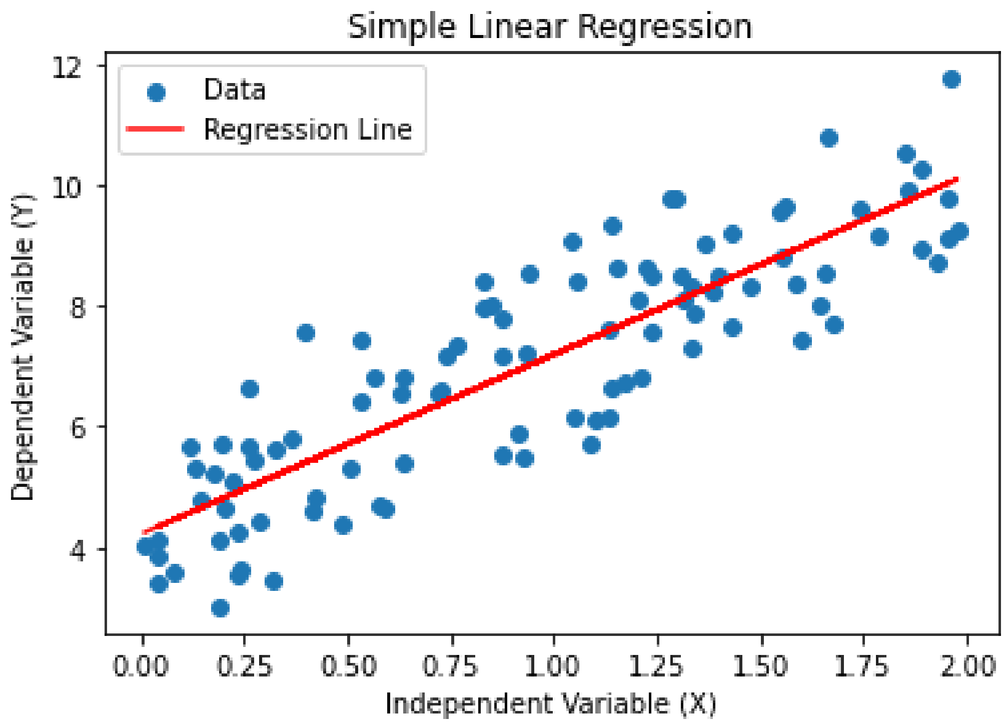


- When the slope (b) of the linear regression line is positive, it indicates a positive linear relationship between the independent and dependent variables.
- This means that as the independent variable (X) increases, the dependent variable (Y) also increases.
- For example, if we are modeling the relationship between study hours and exam scores, a positive linear relationship implies that more study hours are associated with higher exam scores.

- When the slope (b) of the linear regression line is negative, it indicates a negative linear relationship between the independent and dependent variables.
- This means that as the independent variable (X) increases, the dependent variable (Y) decreases.
- For example, in the context of cost and quantity, a negative linear relationship implies that as the quantity of a product increases, the cost per unit decreases.

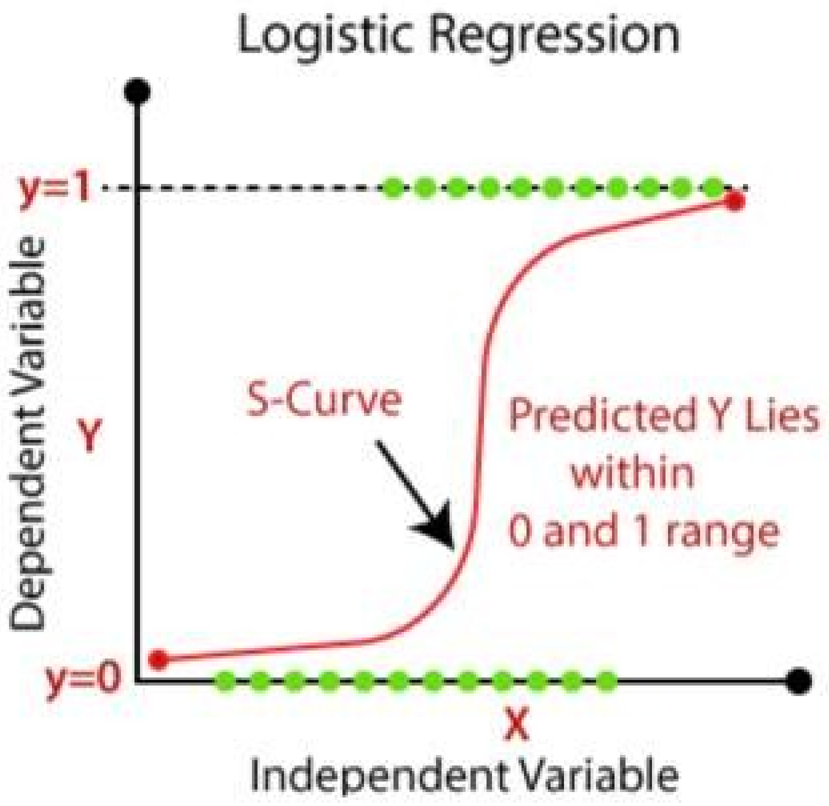
Example Code (Simple Linear Regression):

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

# Generate synthetic data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Create a linear regression model
model = LinearRegression()

# Fit the model to the data
model.fit(X, y)

# Get the intercept and slope of the regression line
intercept = model.intercept_
slope = model.coef_

# Plot the data and regression line
plt.scatter(X, y, label='Data')
plt.plot(X, intercept + slope * X, color='red', label='Regression
Line')
plt.xlabel('Independent Variable (X)')
plt.ylabel('Dependent Variable (Y)')
plt.legend()
plt.title('Simple Linear Regression')
plt.show()
```



Chapter 10: Logistic Regression:

`Logistic regression` is a statistical method used for binary classification tasks, where the dependent variable (target) is binary, meaning it has only two possible values (e.g., 0 or 1, True or False, Yes or No). Despite its name, logistic regression is a classification algorithm, not a regression algorithm like linear regression.



Explanation:

- `Binary Classification`: Logistic regression is well-suited for binary classification problems where the goal is to predict one of two possible outcomes based on one or more independent variables (features).

- `Logistic Function (Sigmoid)`: Logistic regression uses a logistic function (also known as the sigmoid function `(σ)`) to model the probability of the binary outcome. The logistic function maps any input to a value between `0` and `1`, representing the probability of belonging to the positive class.

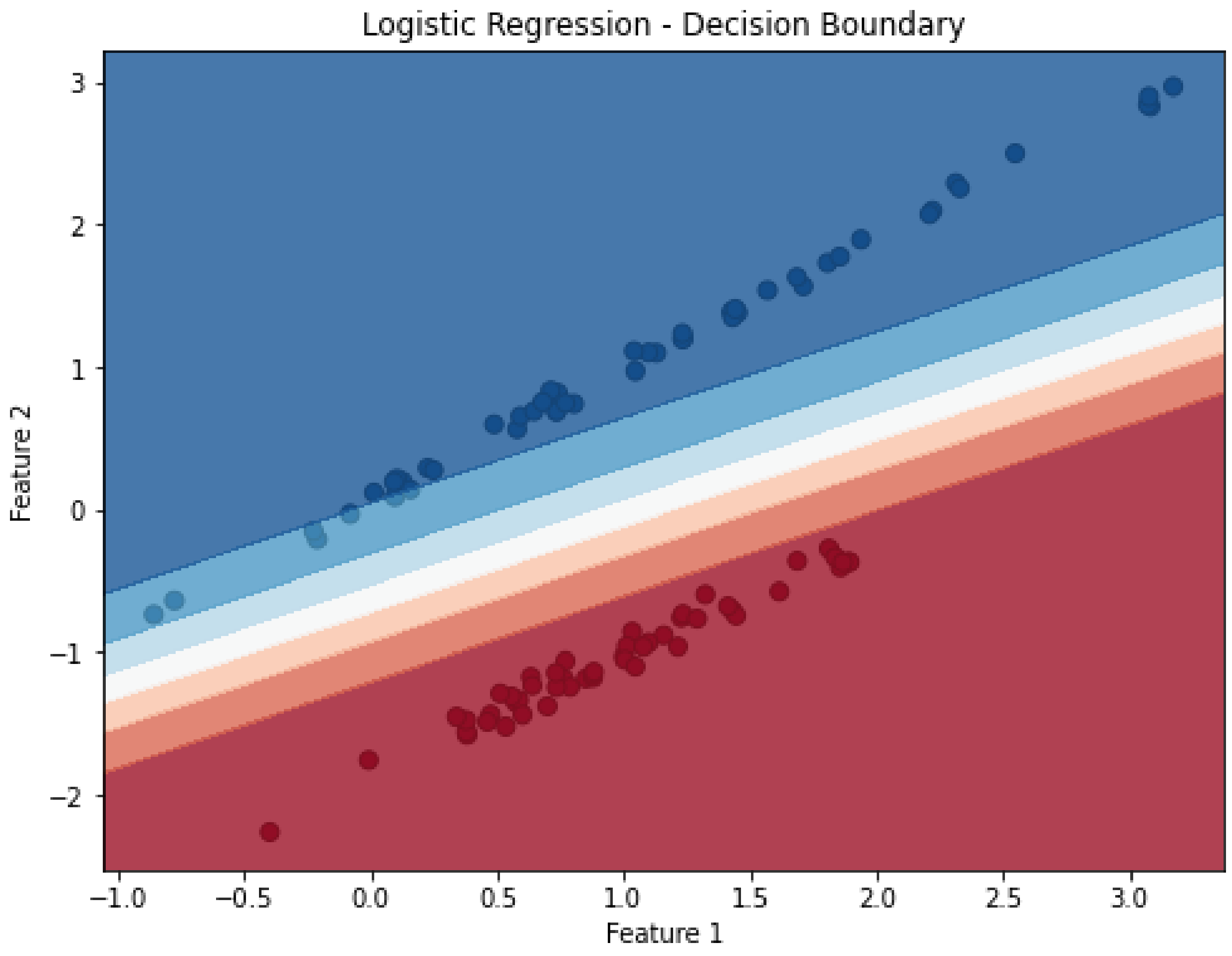
*`Mathematical Representation`: In logistic regression, the logistic function is applied to a linear combination of the independent variables:

$$P(Y=1)=\frac{1}{1+e^{-\left(\hat{a}+\sum_{i=1}^{k} X_i \hat{b}_i\right)}}$$

- P(Y=1) is the probability of the positive class (e.g., 1 or True).
- $X_1, X_2, \ldots, X_k$ are the independent variables.
- a is the intercept.

- $b_1, b_2, \ldots, b_k$ are the coefficients of the independent variables.
- `Decision Boundary`: Logistic regression finds the coefficients $b_1, b_2, \ldots, b_k$ that best fit the training data. It then uses a decision boundary (e.g.,P(Y=1)=0.5) to classify new data points.
- `Regularization`: Logistic regression can be regularized to prevent overfitting. Common regularization techniques include L1 (Lasso) and L2 (Ridge) regularization.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification

# Generate synthetic data for binary classification
X, y = make_classification(n_samples=100, n_features=2, n_classes=2,
n_clusters_per_class=1, n_redundant=0, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model to the data
model.fit(X, y)

# Plot decision boundary and data points
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdBu, marker='o', s=50,
edgecolor='k')
ax = plt.gca()
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()

# Generate grid points to plot decision boundary
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
np.linspace(y_min, y_max, 100))
Z = model.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
Z = Z.reshape(xx.shape)

# Plot decision boundary
plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=0.8)

plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Logistic Regression - Decision Boundary')
plt.show()
```



Chapter 11: Polynomial Regression:

`Polynomial regression` is a type of regression analysis used to model the relationship between a dependent variable and one or more independent variables by fitting a polynomial equation to the data. Unlike linear regression, which assumes a linear relationship between variables, polynomial regression can capture non-linear relationships.

Explanation:

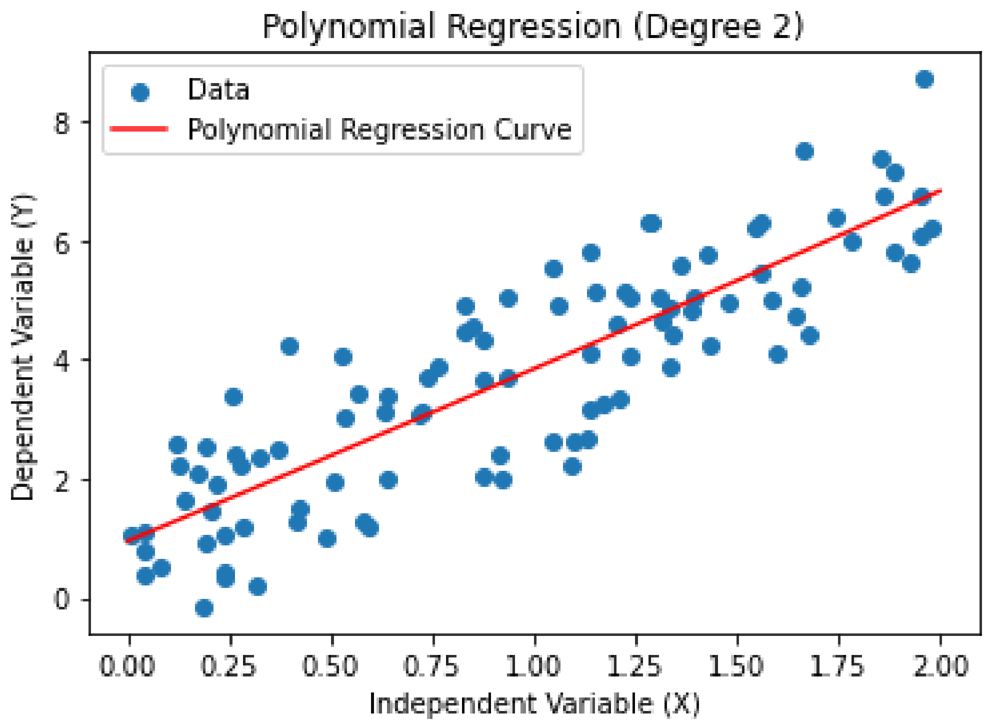
- `Polynomial Equation`: In polynomial regression, the relationship between the dependent variable `(Y)` and the independent variable `(X)` is modeled using a polynomial equation of a specified degree `(n)`:

$$Y = a_0 + a_1 X + a_2 X^2 + \ldots + a_n X^n$$

- `Y` is the dependent variable.
- `X` is the independent variable.
- $a_0, a_1, a_2, \ldots, a_n$ are coefficients that the algorithm estimates to best fit the data.

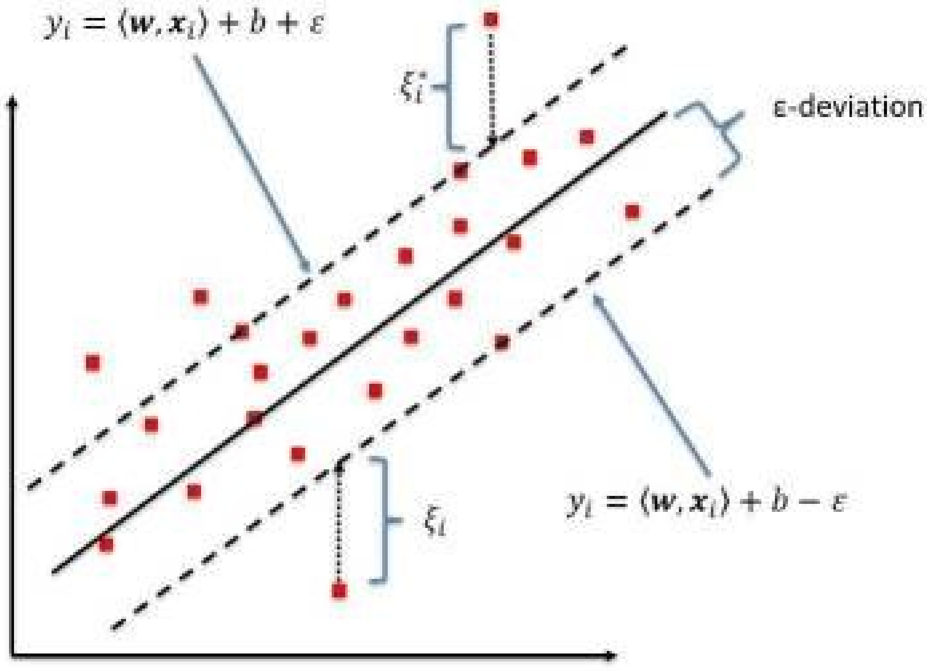
- `Degree of the Polynomial`: The degree `(n)` determines the complexity of the polynomial model. Higher degrees can capture more complex, non-linear relationships but may also lead to overfitting.
- `Overfitting`: Care should be taken when selecting the degree of the polynomial to avoid overfitting, where the model fits the training data noise rather than the underlying pattern.

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

# Generate synthetic data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 1 + 2 * X + 0.5 * X**2 + np.random.randn(100, 1)

# Create polynomial features up to degree 2
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)

# Create a linear regression model
model = LinearRegression()

# Fit the model to the polynomial features
model.fit(X_poly, y)

# Get the intercept and coefficients
intercept = model.intercept_
coefficients = model.coef_

# Plot the data and polynomial regression curve
plt.scatter(X, y, label='Data')
x_range = np.linspace(0, 2, 100).reshape(-1, 1)
x_range_poly = poly_features.transform(x_range)
y_pred = model.predict(x_range_poly)
plt.plot(x_range, y_pred, color='red', label='Polynomial Regression
Curve')
plt.xlabel('Independent Variable (X)')
plt.ylabel('Dependent Variable (Y)')
plt.legend()
plt.title('Polynomial Regression (Degree 2)')
plt.show()
```



Chapter 12: Support Vector Regression (SVR):

`Support Vector Regression (SVR)` is a regression technique that extends the concepts of support vector machines (SVMs) to solve regression problems. SVR is particularly useful when dealing with non-linear relationships between variables. It works by finding a `hyperplane` that best fits the data while allowing for a margin of error.



$y_i = \langle w, x_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle w, x_i \rangle + b - \varepsilon$

Explanation:

- `Hyperplane`: In SVR, the goal is to find a hyperplane that fits the data points as closely as possible while still maintaining a specified margin of error (often referred to as ε-tube or ε-insensitive zone). This margin of error allows some data points to fall outside the margin while minimizing the overall error.

- `Epsilon (ε)`: Epsilon is a user-defined parameter that determines the width of the ε-tube. It controls the trade-off between model complexity and fitting accuracy. Smaller values of ε result in a narrower margin and a more complex model, while larger values allow more data points to fall within the margin.

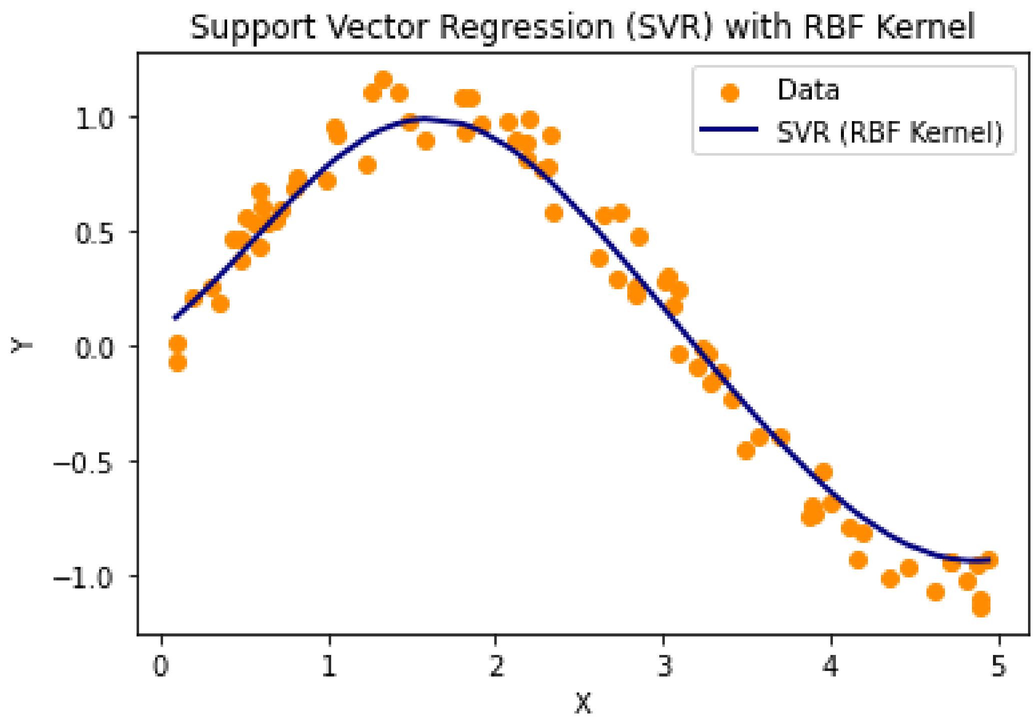
- `Kernel Trick`: SVR can handle non-linear relationships by applying the kernel trick, which transforms the original feature space into a higher-dimensional space, making it possible to find a hyperplane in the transformed space.

- `Loss Function`: The loss function in SVR aims to minimize the error between the predicted values and the actual values within the ε-tube while also maximizing the margin. Common loss functions include the ε-insensitive loss and quadratic loss.

- SVR supports different types of kernels for mapping data to a higher-dimensional space. Common kernels include:
- `Linear Kernel`: Suitable for linear relationships.
- `Polynomial Kernel`: Suitable for polynomial relationships.
- `Radial Basis Function (RBF) Kernel`: Suitable for non-linear relationships and is widely used.

Example Code (SVR using scikit-learn with RBF Kernel):

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Create SVR model with RBF kernel
model = SVR(kernel='rbf', C=1, epsilon=0.2)

# Fit the model to the data
model.fit(X, y)

# Predict using the trained SVR model
y_pred = model.predict(X)

# Plot the data points and SVR predictions
plt.scatter(X, y, color='darkorange', label='Data')
plt.plot(X, y_pred, color='navy', lw=2, label='SVR (RBF Kernel)')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Support Vector Regression (SVR) with RBF Kernel')
plt.legend()
plt.show()
```



Chapter 13: Decision Tree Regression:

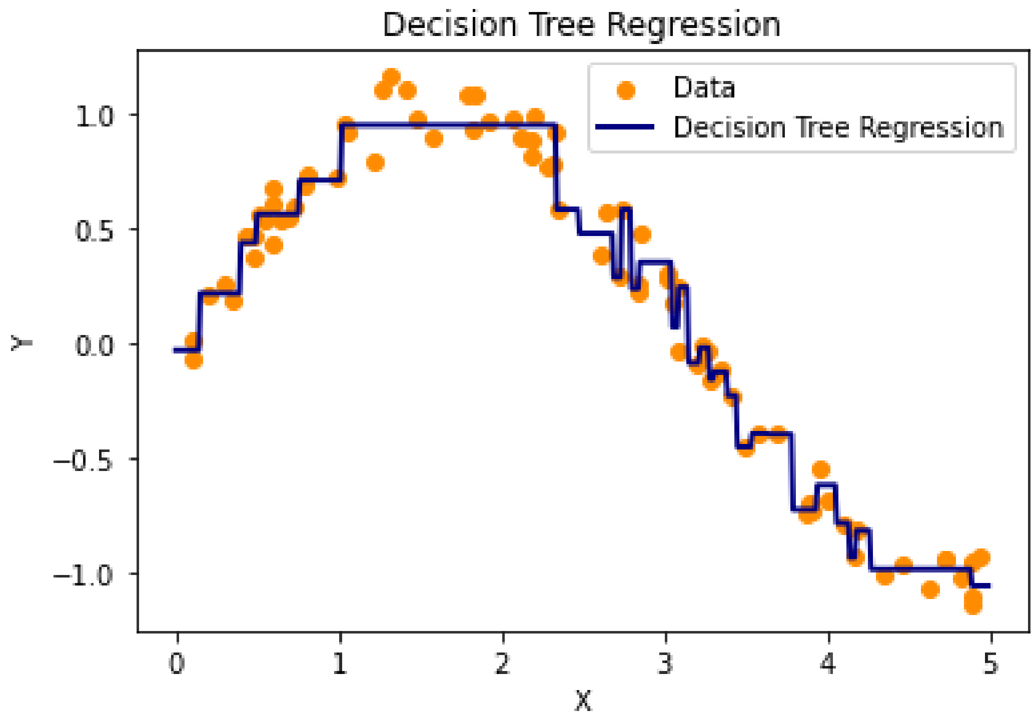
`Decision tree regression` is a machine learning technique used for regression tasks. It builds a tree-like model of decisions and their possible consequences, allowing you to make predictions based on the rules learned from the data. In the context of regression, decision trees are used to predict continuous numerical values.

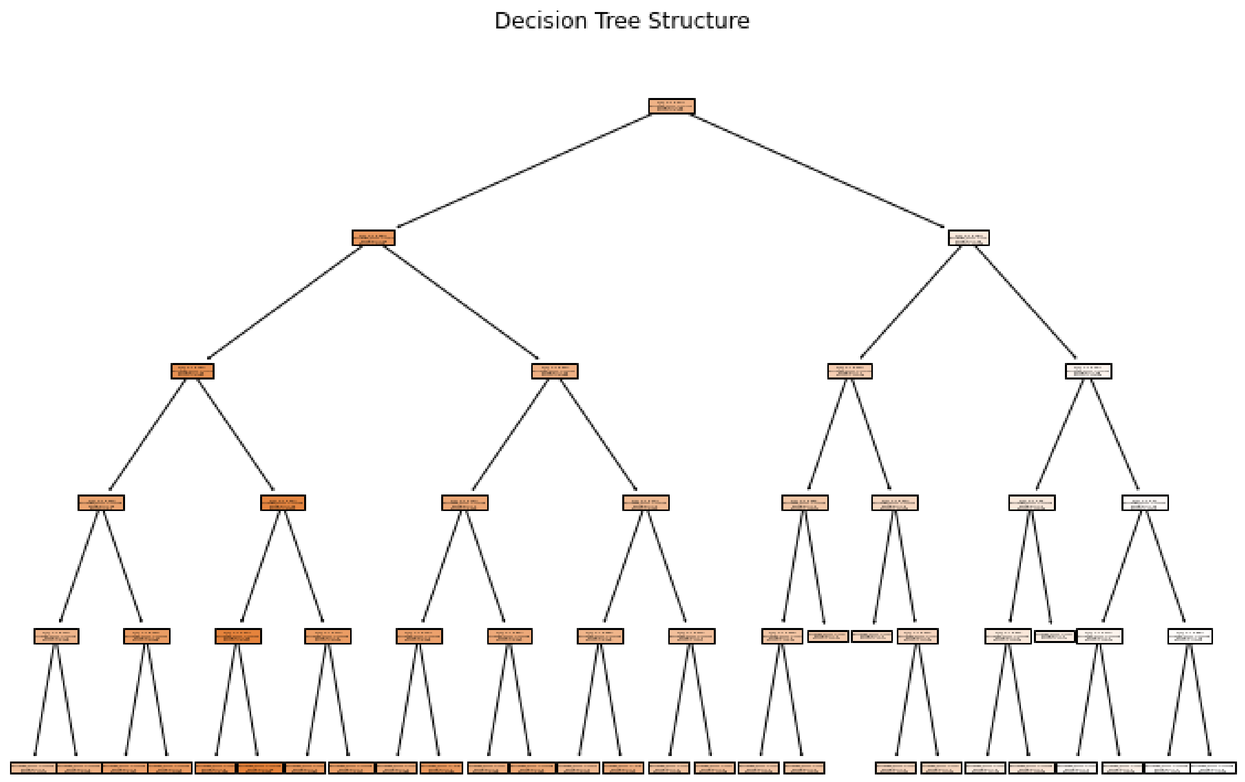
Explanation:

- `Tree Structure`: A decision tree is a hierarchical structure composed of nodes, branches, and leaves. Nodes represent decisions or tests on features, branches represent possible outcomes of the tests, and leaves represent the predicted values.
- `Splitting Criteria`: The tree-building process involves selecting the best feature and value to split the data at each node. Common splitting criteria include minimizing mean squared error, mean absolute error, or other regression-specific metrics.
- `Leaf Values`: In a decision tree regression model, each leaf node contains a predicted value for the target variable. This value is often the mean (or median) of the target values of the training data points that reach that leaf.
- `Tree Depth`: The depth of the tree determines how complex the model can be. A deeper tree may fit the training data more closely but could lead to overfitting.
- `Pruning`: Pruning is a technique used to reduce the depth of the tree by removing branches that do not significantly improve predictive performance on the validation or test data. Pruning helps prevent overfitting.
- `Decision Rules`: The decision tree can be interpreted as a set of rules. To make predictions, one starts at the root node and follows the decision rules based on the features until reaching a leaf node, where the predicted value is found.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree  # Import the plot_tree function

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Create a decision tree regressor
model = DecisionTreeRegressor(max_depth=5)

# Fit the model to the data
model.fit(X, y)
```

```
# Generate data for prediction
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]

# Predict using the trained decision tree regressor
y_pred = model.predict(X_test)

# Plot the data points and decision tree predictions
plt.scatter(X, y, color='darkorange', label='Data')
plt.plot(X_test, y_pred, color='navy', lw=2, label='Decision Tree
Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Decision Tree Regression')
plt.legend()

# Visualize the structure of the decision tree
plt.figure(figsize=(12, 8))
plot_tree(model, filled=True)
plt.title('Decision Tree Structure')
plt.show()
```





Chapter 14: Random Forest Regressor

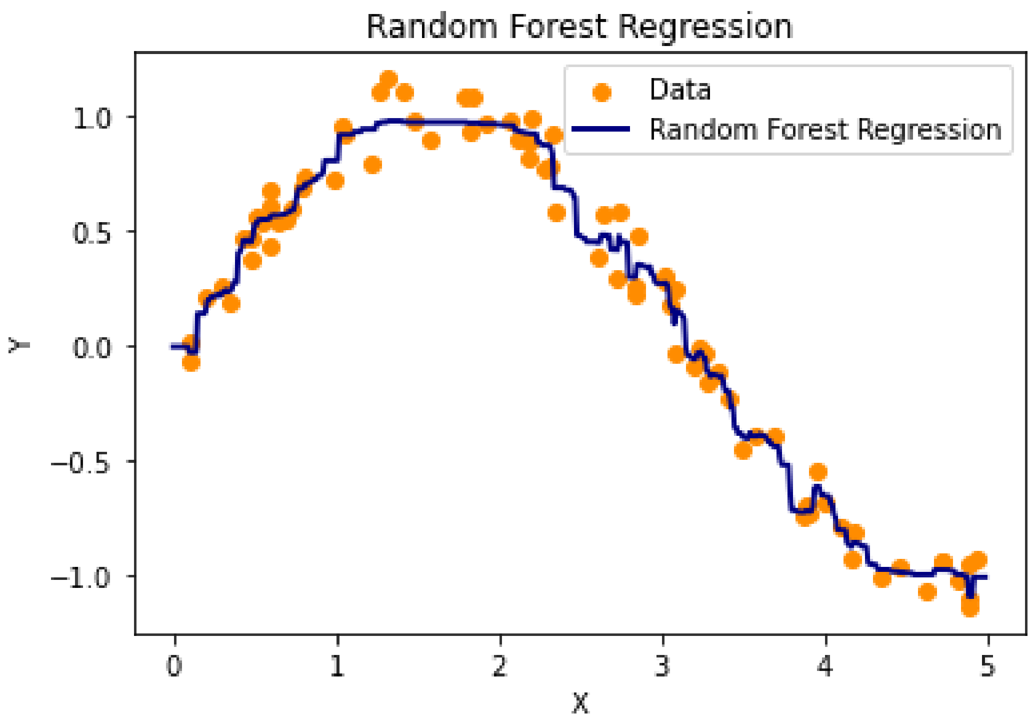
`Random Forest Regressor` is a machine learning technique that extends the idea of decision tree regression. It's an ensemble method that combines multiple decision trees to make more accurate predictions. `Random Forest Regressor` is particularly useful for handling complex, non-linear relationships in data and reducing overfitting.

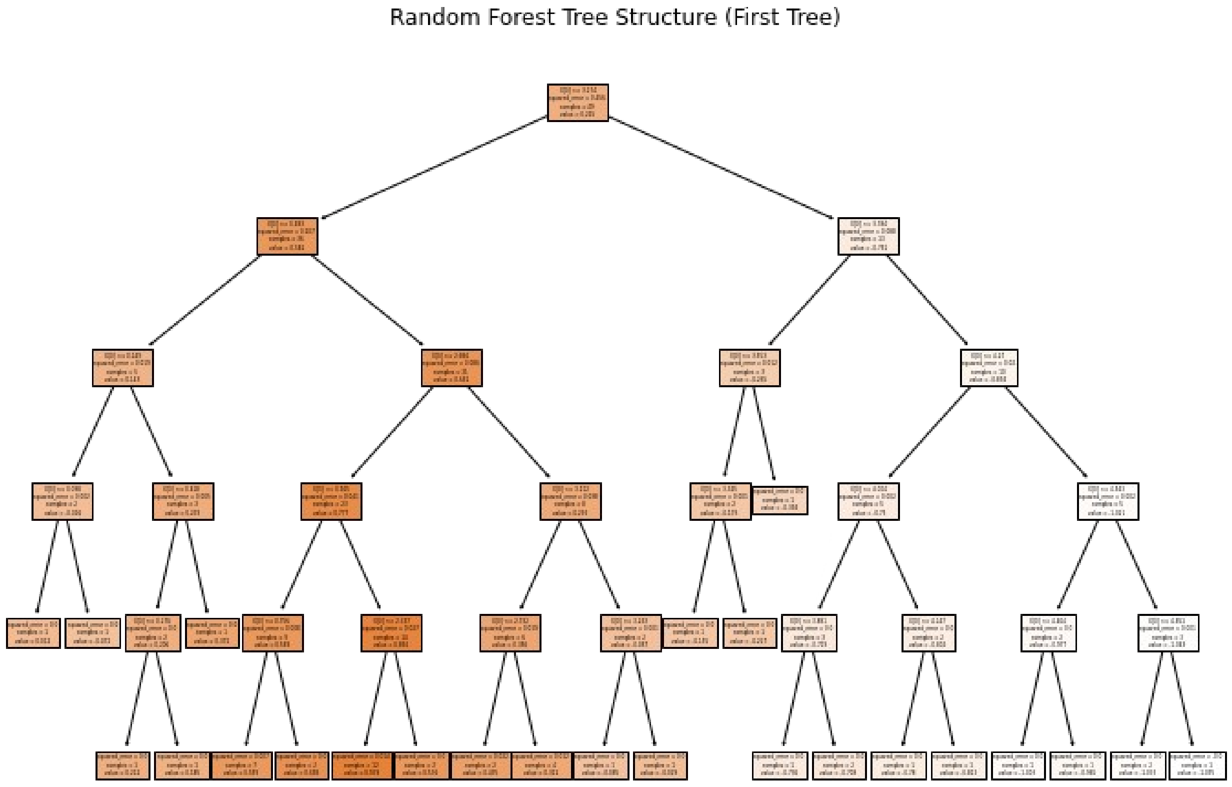
Explanation:

- `Ensemble Learning`: Random Forest is an ensemble learning method, which means it combines predictions from multiple individual models (in this case, decision trees) to make more accurate and robust predictions.
- `Random Subsampling`: Instead of using a single decision tree, Random Forest builds a collection of decision trees. Each tree is trained on a randomly selected subset of the data, called a bootstrap sample. This introduces diversity among the trees.
- `Random Feature Selection`: At each node of each decision tree, Random Forest considers a random subset of features for splitting. This further enhances the diversity of the trees and reduces overfitting.

- `Voting or Averaging`: For regression tasks, the predictions of individual decision trees are combined by taking their average. This ensemble approach typically results in more stable and accurate predictions.
- `Out-of-Bag (OOB) Error`: Random Forest can estimate its own performance during training using out-of-bag samples. This is data that was not included in the bootstrap sample for each tree. The OOB error provides an estimate of the model's accuracy without the need for a separate validation set.
- `Hyperparameters`: Random Forest has hyperparameters to control the number of trees in the ensemble, the depth of each tree, and other settings. Tuning these hyperparameters can optimize model performance.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import plot_tree  # Import the plot_tree function

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Create a random forest regressor with 100 trees
model = RandomForestRegressor(n_estimators=100, max_depth=5)

# Fit the model to the data
model.fit(X, y)

# Generate data for prediction
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]

# Predict using the trained random forest regressor
y_pred = model.predict(X_test)

# Plot the data points and random forest regression predictions
plt.scatter(X, y, color='darkorange', label='Data')
plt.plot(X_test, y_pred, color='navy', lw=2, label='Random Forest
Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Random Forest Regression')
plt.legend()

# Visualize the structure of the first tree in the Random Forest
plt.figure(figsize=(12, 8))
plot_tree(model.estimators_[0], filled=True)
plt.title('Random Forest Tree Structure (First Tree)')
plt.show()
```


Random Forest Regression
Data
Random Forest Regression
Y
X
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5


Random Forest Tree Structure (First Tree)

Chapter 15: Ridge Regression:

`Ridge Regression`, also known as `L2 regularization`, is a linear regression technique used to mitigate the problems of multicollinearity (high correlation between independent variables) and overfitting. It adds a penalty term to the linear regression objective function to constrain the magnitude of the coefficients.

Explanation:

- `Objective Function`: In linear regression, the objective is to minimize the sum of squared differences between the predicted values and the actual target values. In Ridge Regression, this objective function is modified to include a regularization term:

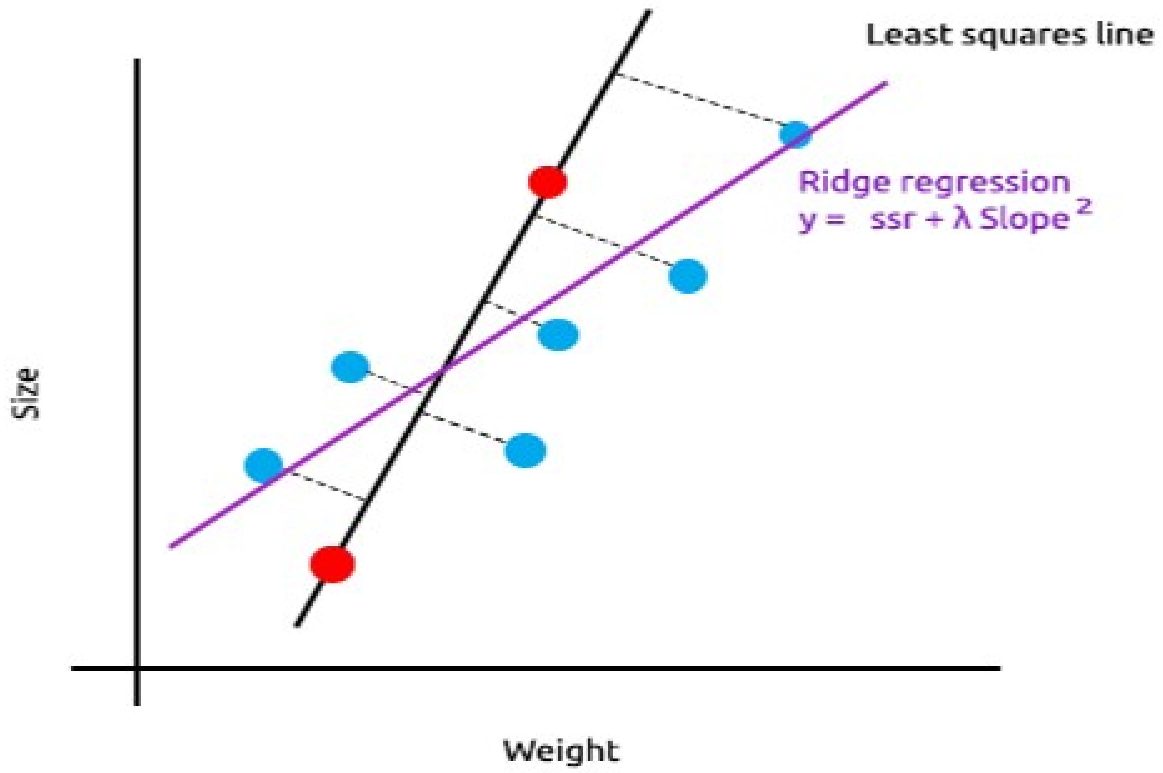
Objective Function ¿ $\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \alpha \sum_{j=1}^{p} \beta_j^2$

- y_i is the actual target value for the i-th data point.
- $\hat{y}_i$ is the predicted target value for the i-th data point.
- n is the number of data points.
- p is the number of features (independent variables).
- β_j represents the coefficients of the independent variables.
- a is the regularization parameter (also known as the regularization strength). It controls the trade-off between fitting the data well and keeping the coefficients small.

- `Regularization Term`: The regularization term, $\alpha \sum_{j=1}^{p} \beta_j^2$,penalizes large values of the coefficients. This means that Ridge Regression encourages the coefficients to be small and, if possible, close to zero.

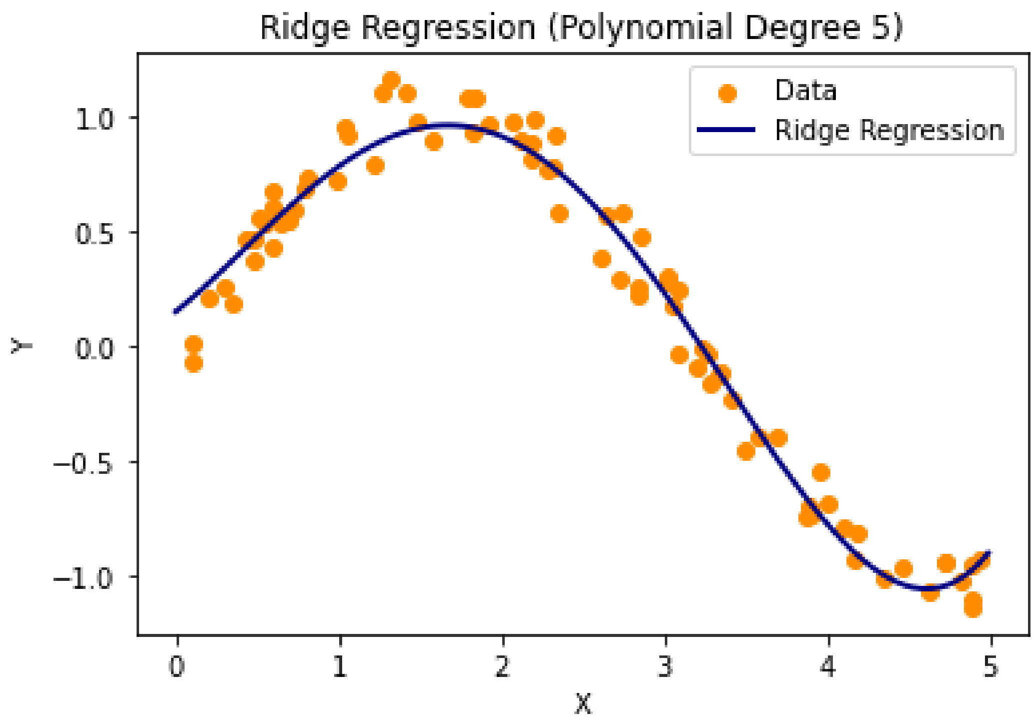
- `Multicollinearity`: Ridge Regression is particularly useful when multicollinearity is present in the data. Multicollinearity occurs when independent variables are highly correlated, making it difficult to determine their individual effects on the dependent variable. Ridge Regression helps by shrinking the coefficients of correlated variables.

- `Tuning the Regularization Parameter`: The choice of the α value is crucial. A smaller α allows the model to fit the data closely but may lead to overfitting. A larger α shrinks the coefficients more aggressively but may underfit the data. Cross-validation is often used to select an appropriate α value.



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Create a Ridge Regression model with a polynomial feature
transformation
degree = 5
model = make_pipeline(PolynomialFeatures(degree), Ridge(alpha=1.0))

# Fit the model to the data
model.fit(X, y)

# Generate data for prediction
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]

# Predict using the trained Ridge Regression model
y_pred = model.predict(X_test)
```

```
# Plot the data points and Ridge Regression predictions
plt.scatter(X, y, color='darkorange', label='Data')
plt.plot(X_test, y_pred, color='navy', lw=2, label='Ridge Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Ridge Regression (Polynomial Degree {})'.format(degree))
plt.legend()
plt.show()
```



Chapter 16: Lasso Regression:

`Lasso Regression`, short for `"Least Absolute Shrinkage and Selection Operator"`, is a linear regression technique that, like Ridge Regression, is used for mitigating overfitting and feature selection. Lasso adds a penalty term to the linear regression objective function to encourage sparse coefficients, effectively pushing some coefficients to become exactly zero.

Explanation:

- `Objective Function`: In Lasso Regression, the objective function is modified to include an L1 regularization term:

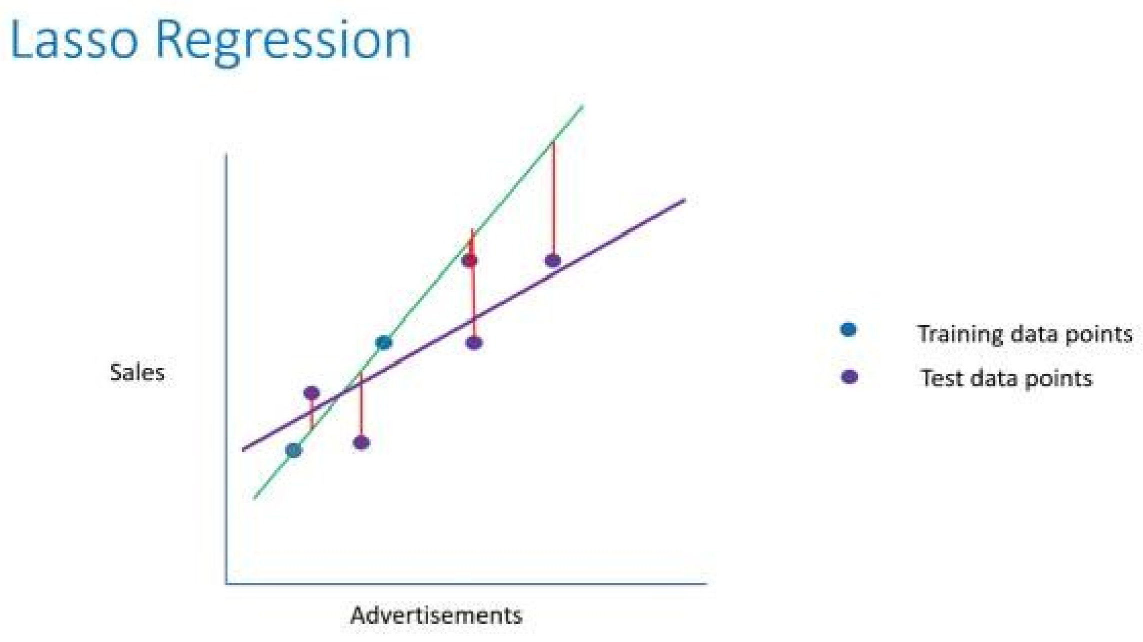
Objective Function ¿ $\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \alpha \sum_{j=1}^{p} \left|\beta_j\right|$

- y_i is the actual target value for the i-th data point.
- $\hat{y}_i$ is the predicted target value for the i-th data point.
- n is the number of data points.
- p is the number of features (independent variables).
- β_j represents the coefficients of the independent variables.
- α is the regularization parameter (also known as the regularization strength). It controls the trade-off between fitting the data well and keeping the coefficients small.

- `Regularization Term`: The L1 regularization term,$\alpha \sum_{j=1}^{p} |\beta_j|$,encourages sparsity in the coefficients. This means that Lasso Regression not only shrinks the coefficients but can also set some coefficients to exactly zero, effectively performing feature selection.

- `Feature Selection`: Lasso Regression is valuable for feature selection because it can identify and eliminate irrelevant or less important features from the model. This results in a simpler and more interpretable model.

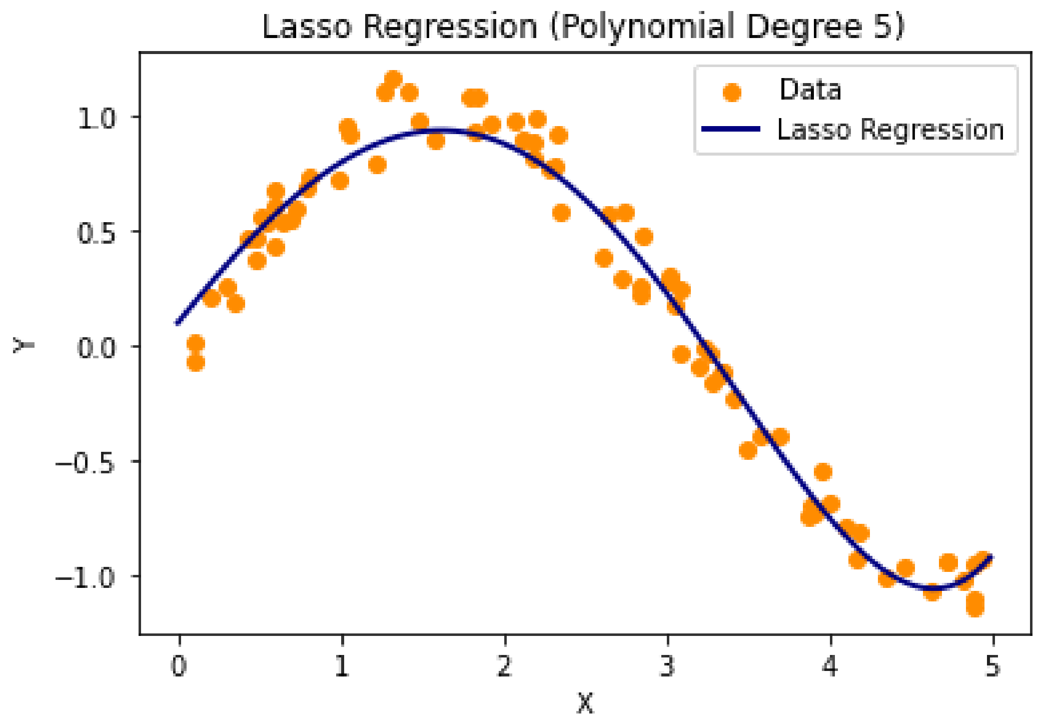
- `Tuning the Regularization Parameter`: As with Ridge Regression, the choice of the α value is essential in Lasso Regression. Smaller values of α allow the model to fit the data closely but may lead to overfitting. Larger values of α encourage more coefficients to become zero. Cross-validation is often used to select an appropriate α value.



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

```
from sklearn.linear_model import Lasso
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# Generate synthetic data with a non-linear relationship
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Create a Lasso Regression model with a polynomial feature
transformation
degree = 5
model = make_pipeline(PolynomialFeatures(degree), Lasso(alpha=0.01))

# Fit the model to the data
model.fit(X, y)

# Generate data for prediction
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]

# Predict using the trained Lasso Regression model
y_pred = model.predict(X_test)

# Plot the data points and Lasso Regression predictions
plt.scatter(X, y, color='darkorange', label='Data')
plt.plot(X_test, y_pred, color='navy', lw=2, label='Lasso Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Lasso Regression (Polynomial Degree {})'.format(degree))
plt.legend()
plt.show()
```

```
C:\Users\avino\anaconda3\lib\site-packages\sklearn\linear_model\
_coordinate_descent.py:647: ConvergenceWarning: Objective did not
converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation.
Duality gap: 1.153e+00, tolerance: 3.764e-03
  model = cd_fast.enet_coordinate_descent(
```



Chapter 17: Model Performance Metrics:

When working with machine learning models, it's essential to evaluate their performance to assess how well they generalize to new, unseen data. Several commonly used performance metrics help you measure the quality of your models. Below are explanations of some key model performance metrics:

1. R-squared (R^2):

- R-squared, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable (target) that is explained by the independent variables (features) in a regression model.
- R-squared values range from 0 to 1, where 0 indicates that the model does not explain any variance, and 1 indicates a perfect fit.
- A higher R-squared value indicates a better fit of the model to the data.

Python Code (Calculating R-squared):

```
from sklearn.metrics import r2_score
y_true = [2, 4, 5, 4, 5]
y_pred = [2.2, 3.8, 4.7, 3.9, 5.1]
r_squared = r2_score(y_true, y_pred)
print(r_squared)

0.9683333333333334
```

2.Root Mean Square Error (RMSE):

- RMSE measures the average magnitude of the errors between predicted values and actual values in a regression model.
- RMSE is in the same units as the dependent variable, making it interpretable.
- Smaller RMSE values indicate better model performance.

Python Code (Calculating RMSE):

```
from sklearn.metrics import mean_squared_error
import math
y_true = [2, 4, 5, 4, 5]
y_pred = [2.2, 3.8, 4.7, 3.9, 5.1]
rmse = math.sqrt(mean_squared_error(y_true, y_pred))
print(rmse)

0.19493588689617924
```

3. Mean Absolute Error (MAE):

- MAE measures the average absolute difference between predicted values and actual values in a regression model.
- MAE is also in the same units as the dependent variable.
- Like RMSE, smaller MAE values indicate better model performance.

Python Code (Calculating MAE):

```
from sklearn.metrics import mean_absolute_error
y_true = [2, 4, 5, 4, 5]
y_pred = [2.2, 3.8, 4.7, 3.9, 5.1]
mae = mean_absolute_error(y_true, y_pred)
print(mae)

0.18
```

4. Mean Squared Error (MSE):

- MSE measures the average of the squared errors between predicted values and actual values in a regression model.
- MSE penalizes larger errors more heavily than MAE.
- Smaller MSE values indicate better model performance.

Python Code (Calculating MSE):

```
from sklearn.metrics import mean_squared_error
y_true = [2, 4, 5, 4, 5]
y_pred = [2.2, 3.8, 4.7, 3.9, 5.1]
mse = mean_squared_error(y_true, y_pred)
print(mse)

0.03799999999999999
```

5. Adjusted R-squared:

- Adjusted R-squared is a modified version of R-squared that accounts for the number of independent variables in a regression model.
- It penalizes the addition of irrelevant variables and rewards the inclusion of relevant variables.
- A higher adjusted R-squared suggests a more parsimonious model.

You typically calculate adjusted R-squared manually by considering the formula, but it's not available directly as a function in scikit-learn.

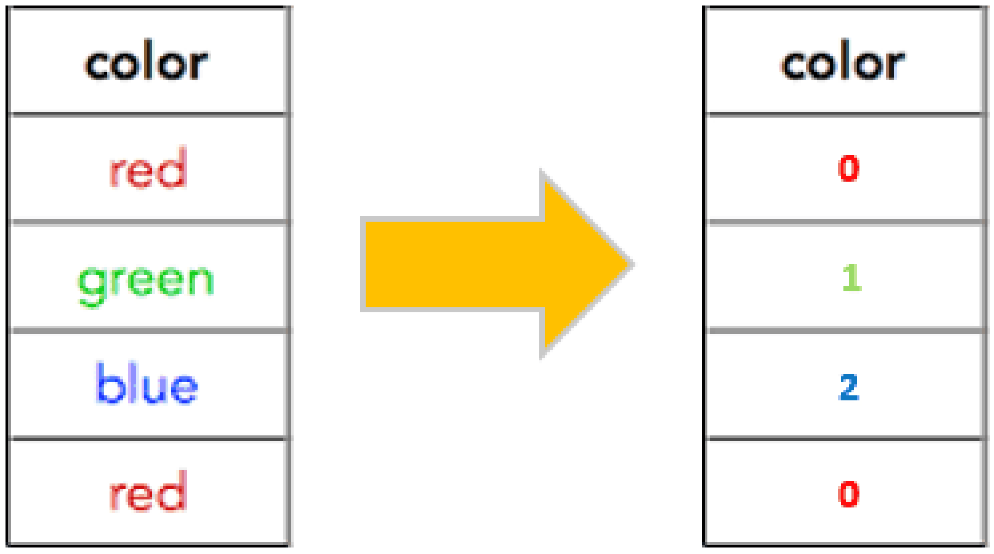
Chapter 18: Types of Encoding Techniques

`Encoding` techniques are essential in machine learning and data analysis when dealing with `categorical` data. Categorical data refers to data that consists of categories or labels rather than numerical values. Encoding transforms categorical data into a numerical format, allowing machine learning models to work with them effectively. Here are some common types of encoding techniques:

1. Label Encoding:

- Label encoding assigns a unique integer (or label) to each category in a categorical variable.
- It is suitable for ordinal categorical data, where the order of categories matters.
- Example: Converting ["Small", "Medium", "Large"] to [0, 1, 2].

Python Code (Label Encoding using scikit-learn):



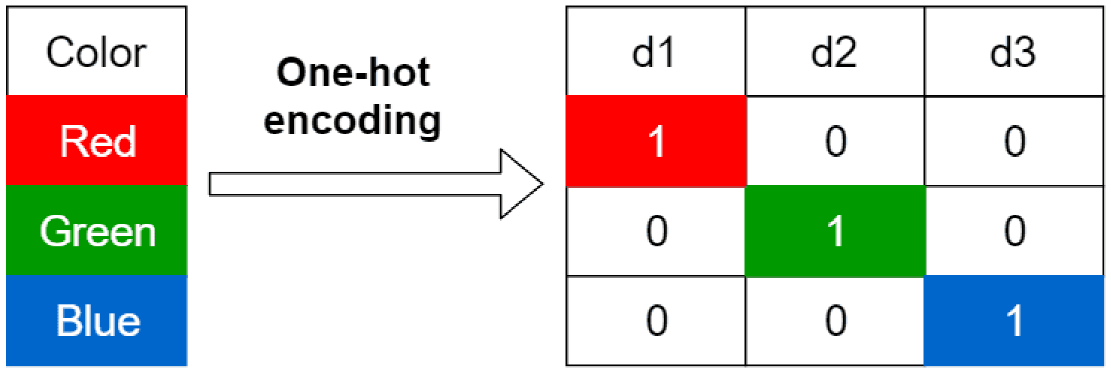
```python
from sklearn.preprocessing import LabelEncoder
```

```
# Create an instance of LabelEncoder
encoder = LabelEncoder()

# Encode the categorical labels
encoded_labels = encoder.fit_transform(["Small", "Medium", "Large"])

# Display the encoded labels
encoded_labels

array([2, 1, 0], dtype=int64)
```

2. One-Hot Encoding:

- One-hot encoding converts each category into a binary vector, where each category is represented by a unique binary digit (1 or 0).
- It is suitable for nominal categorical data, where there is no inherent order among categories.
- It prevents the model from assuming ordinal relationships between categories.
- Example: ["Red", "Green", "Blue"] might be encoded as [[1, 0, 0], [0, 1, 0], [0, 0, 1]].

example code(One-Hot Encoding using pandas):



```
import pandas as pd

# Create a DataFrame with a "Color" column containing categorical
values
data = pd.DataFrame({"Color": ["Red", "Green", "Blue"]})

# Apply one-hot encoding to the "Color" column
data_encoded = pd.get_dummies(data, columns=["Color"])

# Display the DataFrame with one-hot encoding
data_encoded
```

```
   Color_Blue  Color_Green  Color_Red
0       False        False       True
1       False         True      False
2        True        False      False
```

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Create a DataFrame with a "Color" column containing categorical
values
data = pd.DataFrame({"Color": ["Red", "Green", "Blue"]})

# Create an instance of the OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# Fit and transform the encoder on the "Color" column
data_encoded = encoder.fit_transform(data[["Color"]])

# Create a new DataFrame with the one-hot encoded columns
data_encoded_df = pd.DataFrame(data_encoded,
columns=encoder.get_feature_names_out(["Color"]))

# Display the DataFrame with one-hot encoding
data_encoded_df
```

```
   Color_Blue  Color_Green  Color_Red
0         0.0          0.0        1.0
1         0.0          1.0        0.0
2         1.0          0.0        0.0
```

3. Binary Encoding:

- Binary encoding combines aspects of label encoding and one-hot encoding.
- It first converts categories to numerical labels, then converts the labels to binary code, and each digit of the binary code becomes a separate feature.
- It reduces dimensionality compared to one-hot encoding while still handling nominal data.

Python Code (Binary Encoding using category_encoders library):

```
import pandas as pd
import category_encoders as ce

# Create a DataFrame with the "Color" column
data = pd.DataFrame({"Color": ["Red", "Green", "Blue"]})

# Create an instance of the BinaryEncoder
encoder = ce.BinaryEncoder(cols=["Color"])

# Fit and transform the encoder on the DataFrame
```

```
encoded_data = encoder.fit_transform(data)

# Display the encoded data
encoded_data

   Color_0  Color_1
0        0        1
1        1        0
2        1        1
```

4. Frequency Encoding:

- Frequency encoding replaces categories with their frequencies (counts) in the dataset.
- It can be useful when the frequency of occurrence of a category is informative for the problem.
- It may not be suitable for categories with very similar frequencies.

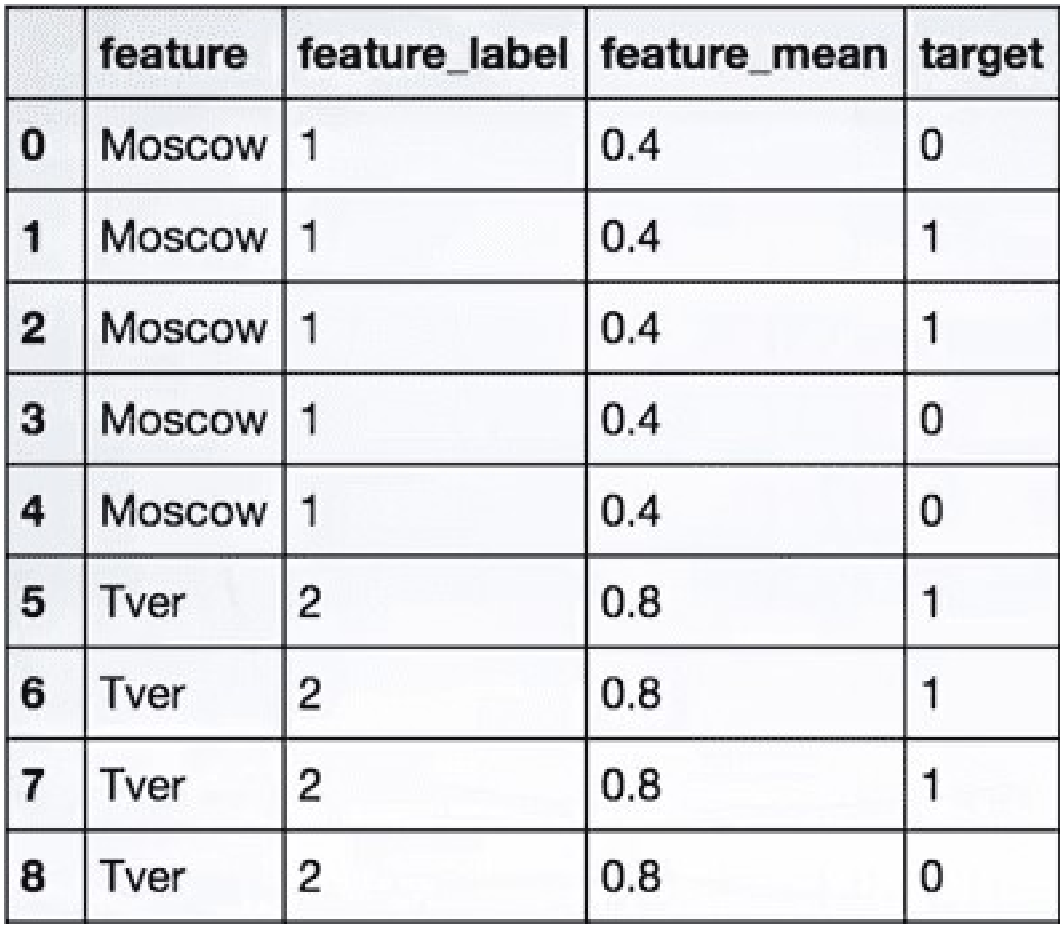
Python Code (Frequency Encoding using pandas):

```
import pandas as pd
data = pd.DataFrame({"Color": ["Red", "Green", "Blue", "Red",
"Green"]})
freq_encoding = data['Color'].value_counts(normalize=True).to_dict()
data['Color'] = data['Color'].map(freq_encoding)
data

   Color
0    0.4
1    0.4
2    0.2
3    0.4
4    0.4
```

5. Target Encoding (Mean Encoding):

- Target encoding uses the mean of the target variable for each category as the encoded value.
- It is particularly useful for classification tasks when dealing with high-cardinality categorical variables.
- It can lead to leakage if not used properly (e.g., target leakage), so care must be taken to avoid overfitting.

Python Code (Target Encoding using pandas):



	feature	feature_label	feature_mean	target
0	Moscow	1	0.4	0
1	Moscow	1	0.4	1
2	Moscow	1	0.4	1
3	Moscow	1	0.4	0
4	Moscow	1	0.4	0
5	Tver	2	0.8	1
6	Tver	2	0.8	1
7	Tver	2	0.8	1
8	Tver	2	0.8	0

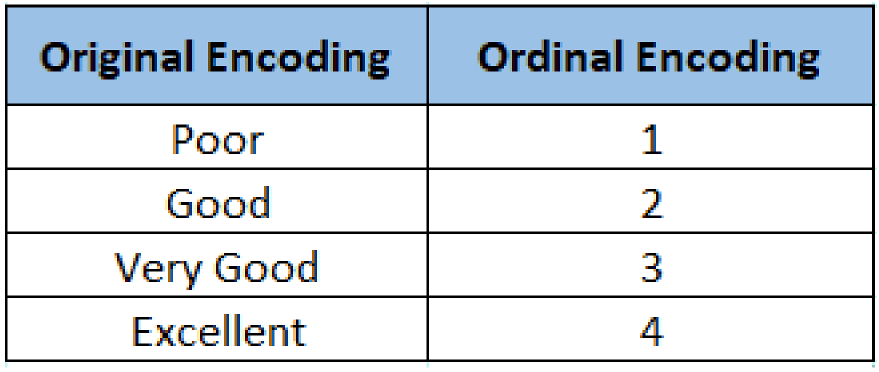
```
import pandas as pd
data = pd.DataFrame({"Category": ["A", "B", "A", "B", "B"], "Target":
[1, 0, 1, 0, 1]})
mean_encoding = data.groupby("Category")["Target"].mean().to_dict()
data["Category_Encoded"] = data["Category"].map(mean_encoding)
data
```

```
  Category  Target  Category_Encoded
0        A       1          1.000000
1        B       0          0.333333
2        A       1          1.000000
3        B       0          0.333333
4        B       1          0.333333
```

6. Ordinal Encoding:

- Ordinal encoding is used for ordinal categorical variables, where categories have a meaningful order.
- It assigns numerical values to categories based on their order, preserving the ordinal relationship.
- Example: ["Low", "Medium", "High"] might be encoded as [1, 2, 3].

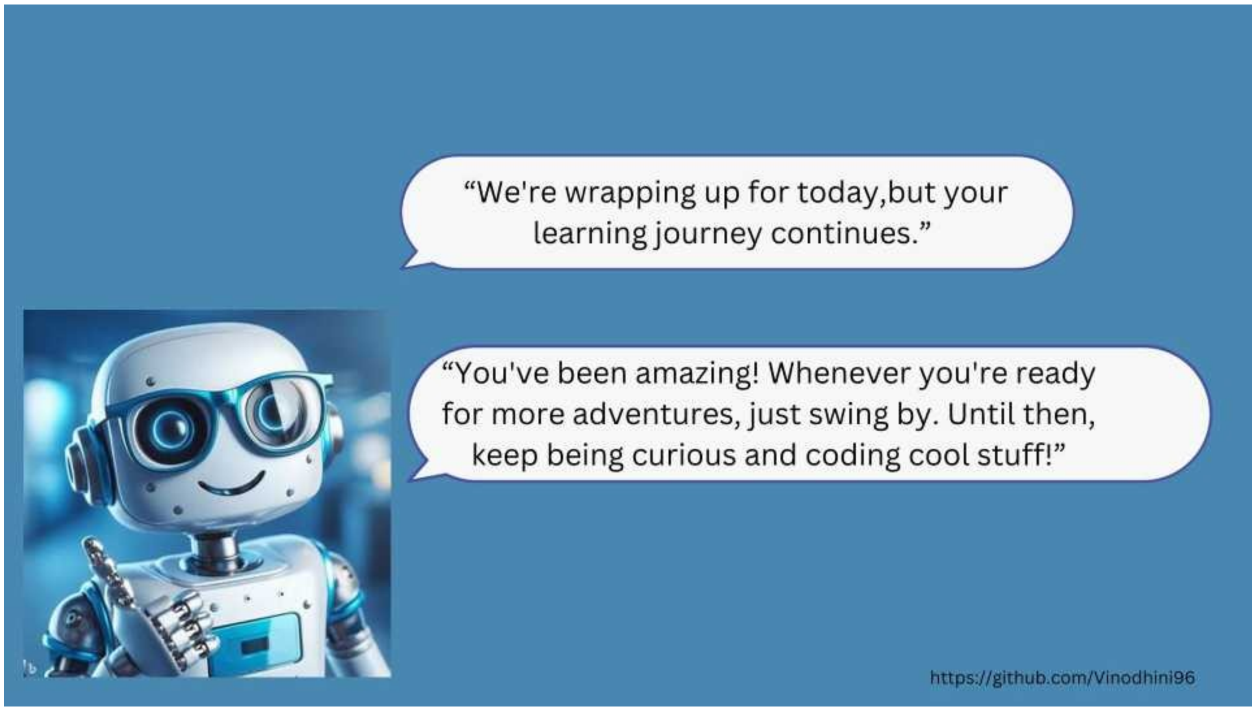
Python Code (Ordinal Encoding using a custom mapping dictionary):



Original Encoding	Ordinal Encoding
Poor	1
Good	2
Very Good	3
Excellent	4

```python
import pandas as pd

# Sample DataFrame with a "Category" column
data = pd.DataFrame({"Category": ["Low", "Medium", "High", "Low",
"High"]})

# Define the mapping dictionary
mapping = {"Low": 1, "Medium": 2, "High": 3}

# Apply ordinal encoding to the "Category" column in your DataFrame
data["ordinal_encoding"] = data["Category"].map(mapping)

# Display the DataFrame with the ordinal encoding
data
```

```
  Category  ordinal_encoding
0      Low                 1
1   Medium                 2
2     High                 3
3      Low                 1
4     High                 3
```


“We're wrapping up for today,but your learning journey continues.”
“You've been amazing! Whenever you're ready for more adventures, just swing by. Until then, keep being curious and coding cool stuff!”
https://github.com/Vinodhini96