Machine Learning Algorithms

Content

- ML Algorithms
- Supervised and Unsupervised Learning
- The Bias-Variance Trade-Off
- Overfitting and Underfitting
- Parametric and Nonparametric ML Algorithms
- Example: Algorithms in Python

Machine Learning Workflow

What do you mean by

Input

Cat

Training

AI Models

Learning

Output

Cats' Features:

- Fur Color
- Eye Color
- Ear Shape
- Tail Length
- Whiskers

Doraemon's Features:

- Blue Body
- Bell Around Neck
- Pocket on Stomach
- Round Face
- Propeller Hat

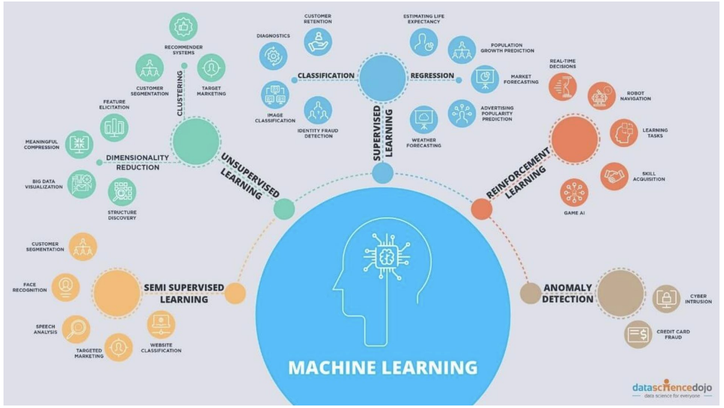
Lucky Cats' Features:

- Raised Paw
- Coin or Fish
- Base or Platform
- Expression
- Color

ML Algorithms

What is a ML algorithm?

- A machine learning algorithm is a set of rules or processes used by an AI system to perform tasks.
- These algorithms allow computers to understand patterns and make predictions based on data without explicit programming.
- They serve as the foundation for modern artificial intelligence and find applications in areas like
  - image recognition
  - speech processing
  - recommendation systems
  - fraud detection
  - autonomous vehicles



https://laconicml.com/wp-content/uploads/2020/07/List-of-Top-5-Powerful-Machine-Learning-Algorithms-That-Will-Solve-99-of-Your-Problems.jpg

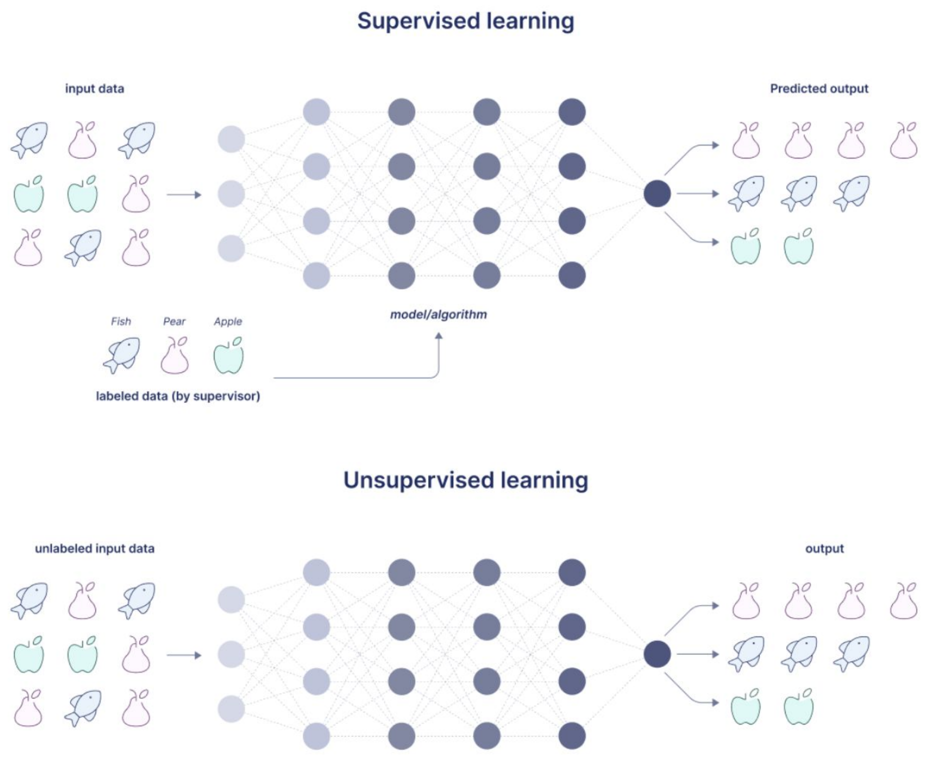
ML Algorithms

What is the category of a ML algorithm?

- **Supervised Learning**
  In supervised learning, an algorithm learns from labeled training data to make predictions or decisions without explicit programming.
  - Regression
  - Classification
- **Unsupervised Learning**
  Unlike supervised learning, where labeled data guides the model, unsupervised learning uses unlabeled data. The goal is to explore the data and uncover hidden patterns, trends, and relationships.
  - Clustering
  - Association (We have learned on week 3's lecture)
  - Dimensionality Reduction
- **Reinforcement Learning (RL)**
  RL focuses on how an intelligent agent should take actions in a dynamic environment to maximize cumulative reward. Unlike supervised learning, it doesn't rely on labeled input/output pairs or explicit corrections for sub-optimal actions. Instead, it balances exploration (uncharted territory) and exploitation (current knowledge) to achieve long-term rewards, even with incomplete or delayed feedback.

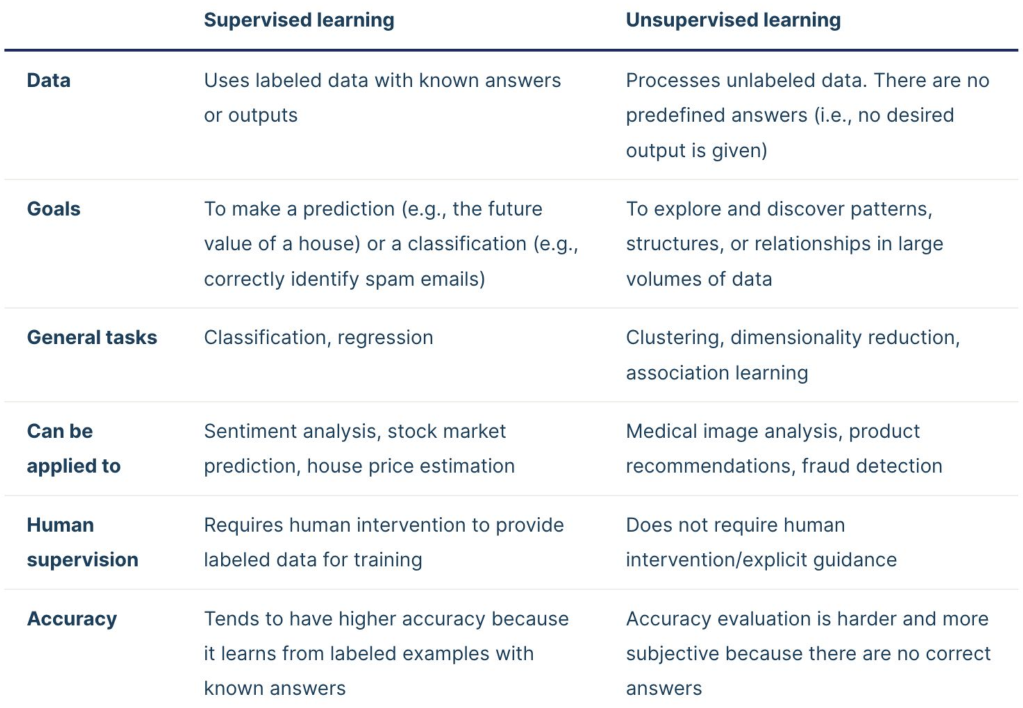
Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning**



Supervised and Unsupervised Learning

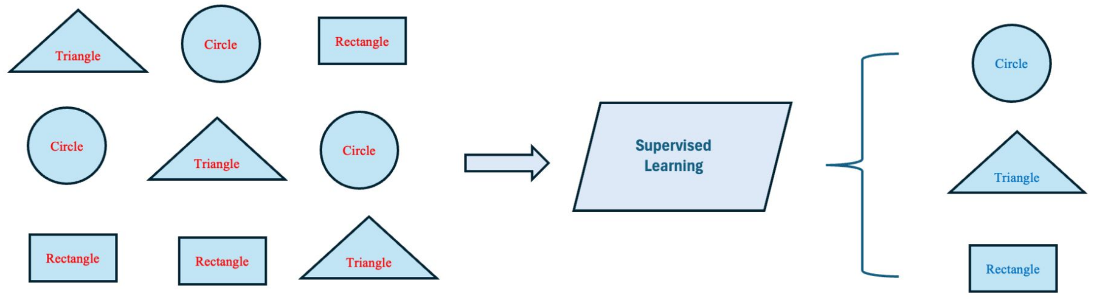
- **Supervised Learning vs. Unsupervised Learning**

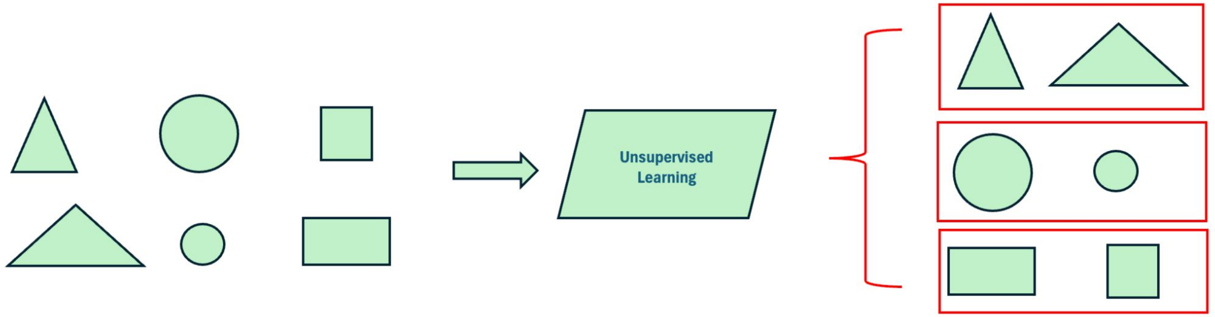


	Supervised learning	Unsupervised learning
Data	Uses labeled data with known answers or outputs	Processes unlabeled data. There are no predefined answers (i.e., no desired output is given)
Goals	To make a prediction (e.g., the future value of a house) or a classification (e.g., correctly identify spam emails)	To explore and discover patterns, structures, or relationships in large volumes of data
General tasks	Classification, regression	Clustering, dimensionality reduction, association learning
Can be applied to	Sentiment analysis, stock market prediction, house price estimation	Medical image analysis, product recommendations, fraud detection
Human supervision	Requires human intervention to provide labeled data for training	Does not require human intervention/explicit guidance
Accuracy	Tends to have higher accuracy because it learns from labeled examples with known answers	Accuracy evaluation is harder and more subjective because there are no correct answers

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Data & Goals**



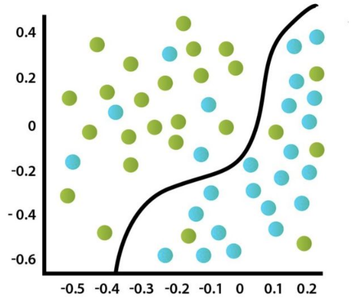


Supervised and Unsupervised Learning

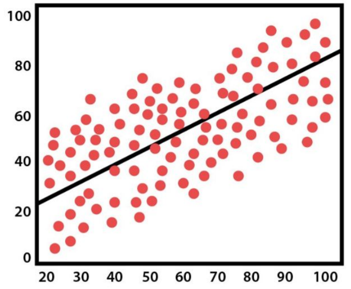
- **Supervised Learning vs. Unsupervised Learning**
  - Methods
    - Supervised Learning
      - Classification
      - Regression
    - Unsupervised Learning
      - Clustering
      - Dimensionality Reduction
      - Association Learning

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Methods**
  - Supervised Learning
    - **Classification**
      - Classification is used when our goal is to categorize data into predefined classes or labels. It’s like sorting things into different buckets based on their characteristics.
      - Algorithms
        - Logistic Regression
        - Decision Trees
        - Random Forests
        - Support Vector Machines (SVM)
    - **Regression**
      - Regression models predict continuous values (real numbers). They’re ideal for problems where the outcome is not discrete but lies on a continuous scale.
      - Algorithms
        - Linear Regression
        - Polynomial Regression
        - Support Vector Regression (SVR)



CLASSIFICATION



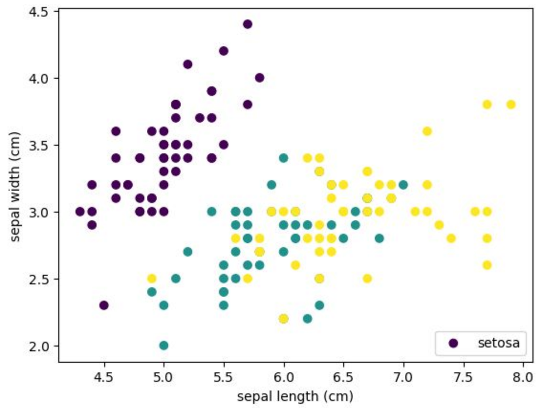
REGRESSION

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Methods**
  - Unsupervised Learning
    - **Clustering**
      - Clustering aims to group similar data points together based on their features. It helps discover inherent structures within the data.
      - Algorithms
        - K-Means
        - Hierarchical Clustering
        - DBSCAN
        - Gaussian Mixture Models (GMM)
    - **Dimensionality Reduction**
      - Reducing the number of features while preserving essential information. It helps visualize and analyze high-dimensional data.
      - Algorithms
        - **PCA** (Principal Component Analysis)
        - Autoencoders
    - **Association Learning**
      - Identifying interesting relationships or patterns among items in transactional data (e.g., market basket analysis).
      - Examples: **Market Basket Analysis**, Recommendation Systems

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - *https://en.wikipedia.org/wiki/Iris_flower_data_set*
    - **Description**:
      - The Iris dataset contains measurements of sepal and petal lengths for three species of iris flowers: **Setosa**, **Versicolor**, and **Virginica**.
      - Each sample has four features: Sepal Length, Sepal Width, Petal Length, and Petal Width.
      - It's a multi-class classification dataset with a total of 150 samples (50 samples per class).
    - **Attributes**:
      - Sepal Length
      - Sepal Width
      - Petal Length
      - Petal Width
    - **Species**:
      - Setosa
      - Versicolor
      - Virginica



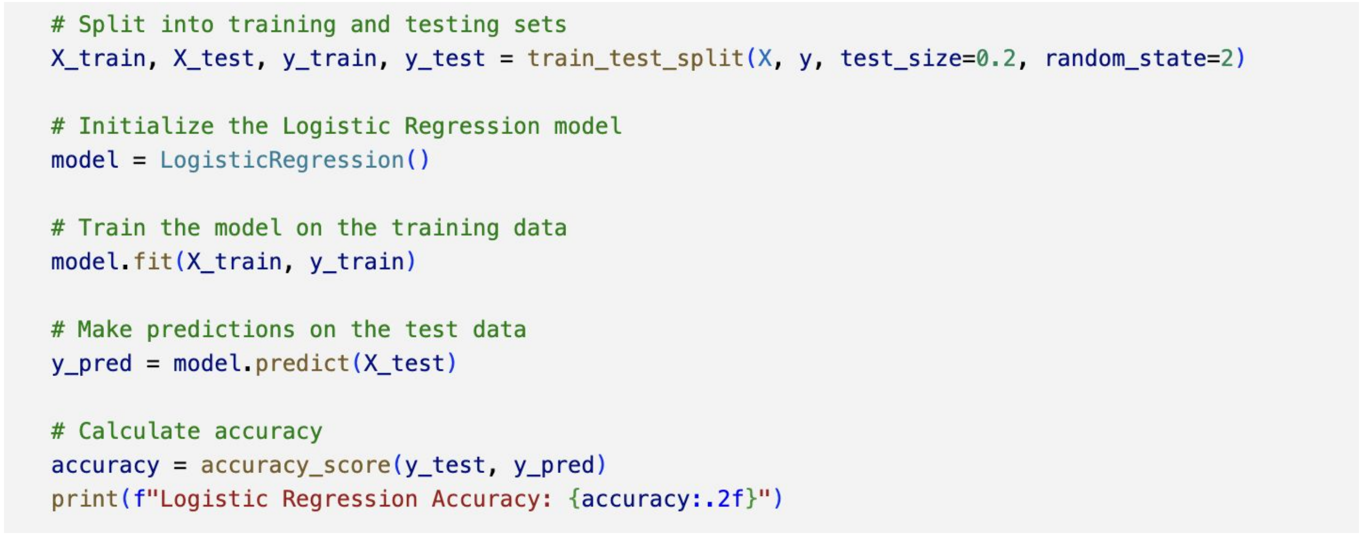
Iris virginica

Iris versicolor

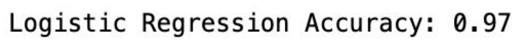
Iris setosa

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
    - Iris flower data set
        - Logistic Regression



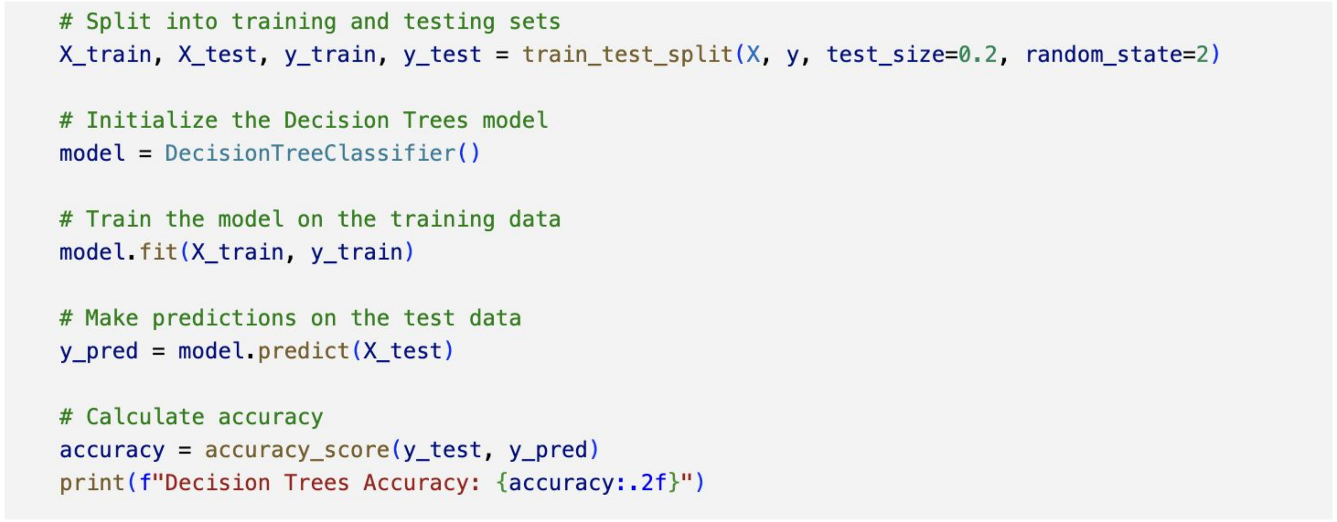
```
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# Initialize the Logistic Regression model
model = LogisticRegression()

# 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 accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")
```



```
Logistic Regression Accuracy: 0.97
```

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - Decision Trees (branch figure)

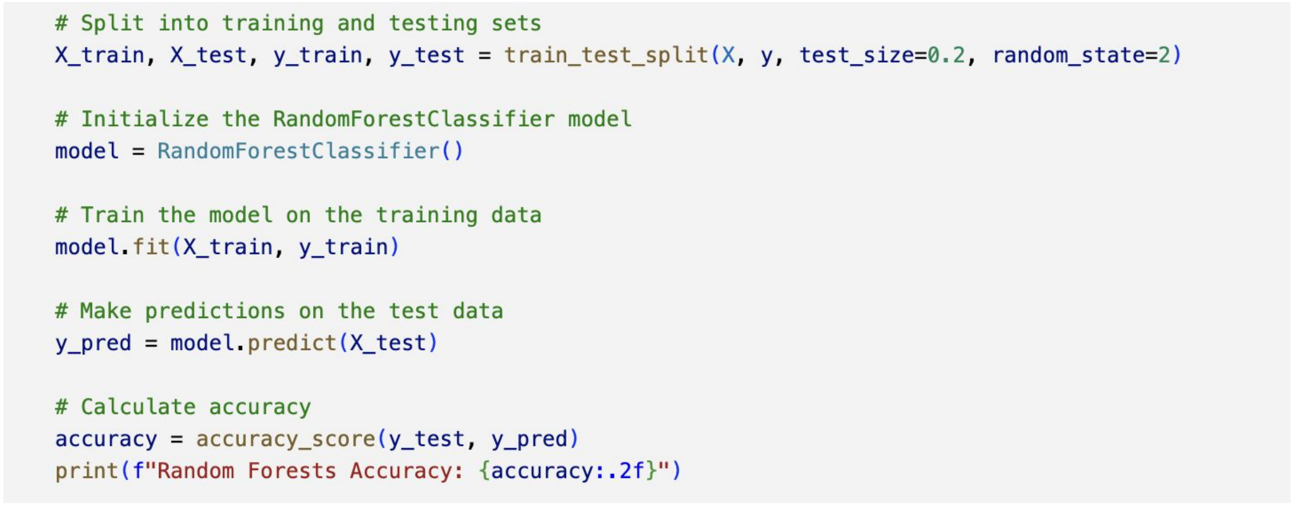


```
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# Initialize the Decision Trees model
model = DecisionTreeClassifier()

# 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 accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Trees Accuracy: {accuracy:.2f}")
```

```
Decision Trees Accuracy: 0.93
```

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - Random Forests

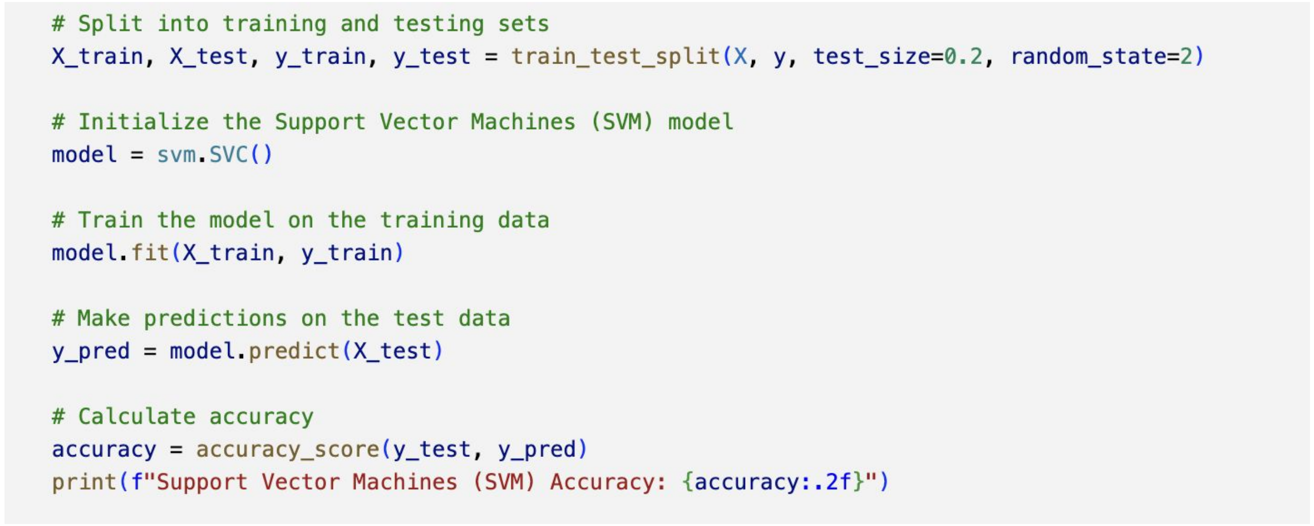


```python
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# Initialize the RandomForestClassifier model
model = RandomForestClassifier()

# 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 accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forests Accuracy: {accuracy:.2f}")
```

```
Random Forests Accuracy: 0.97
```

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - Support Vector Machines (SVM)

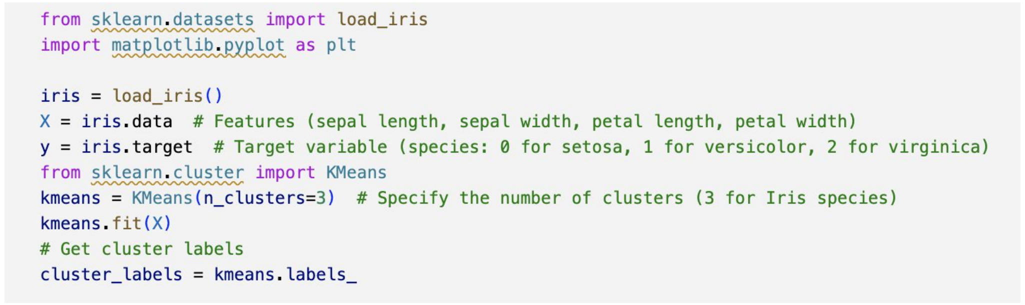


```python
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# Initialize the Support Vector Machines (SVM) model
model = svm.SVC()

# 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 accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Support Vector Machines (SVM) Accuracy: {accuracy:.2f}")
```

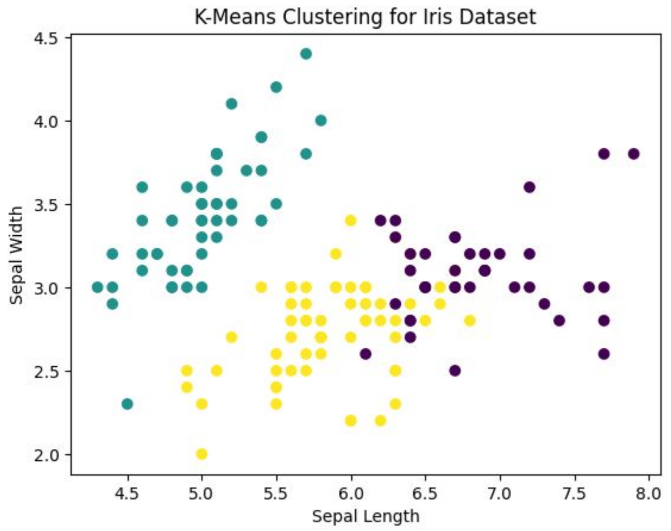
```
Support Vector Machines (SVM) Accuracy: 0.97
```

Supervised and Unsupervised Learning

- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - Clustering



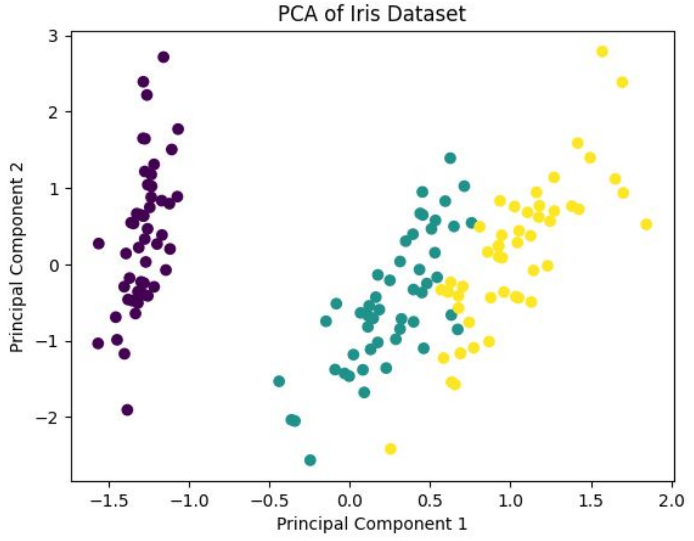
```
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data  # Features (sepal length, sepal width, petal length, petal width)
y = iris.target  # Target variable (species: 0 for setosa, 1 for versicolor, 2 for virginica)
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)  # Specify the number of clusters (3 for Iris species)
kmeans.fit(X)
# Get cluster labels
cluster_labels = kmeans.labels_
```



Supervised and Unsupervised Learning

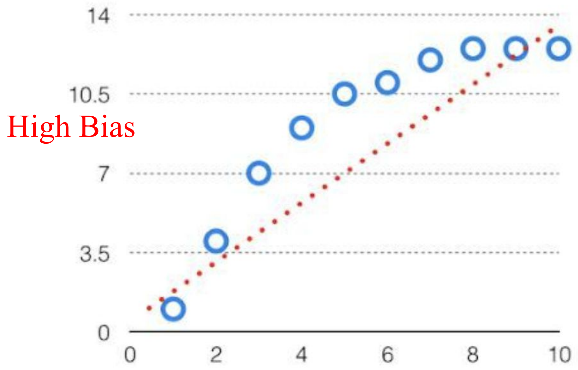
- **Supervised Learning vs. Unsupervised Learning: Example in Python**
  - Iris flower data set
    - PCA (Principal Component Analysis)

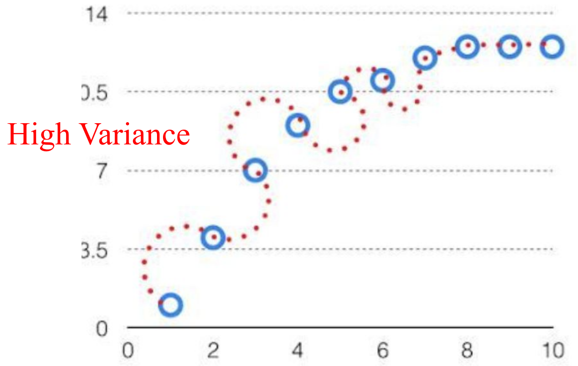
```python
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data
# Apply PCA
pca = PCA(n_components=2, whiten=True)  # Specify the number of components (2 in this case)
X_pca = pca.fit_transform(X)
# Visualize the results
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=iris.target, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.show()
```



The Bias-Variance Trade-Off

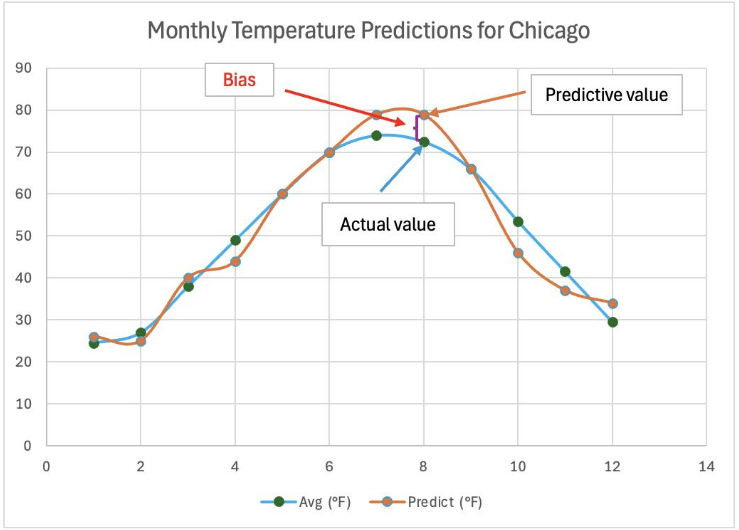
- Trains on input variables (X) to predict values (Y) close to actual values.
- The difference between predicted, and actual values is the error used for evaluation.
- Types of Errors
  - **Bias Error**: Systematic error due to incorrect assumptions about data.
  - **Variance Error**: Variability in model predictions across different datasets.
  - **Noise**: Irreducible error that cannot be eliminated.
- Bias-Variance Tradeoff
  - Balancing bias and variance is crucial for model selection.
  - Techniques aim to minimize both bias and variance.

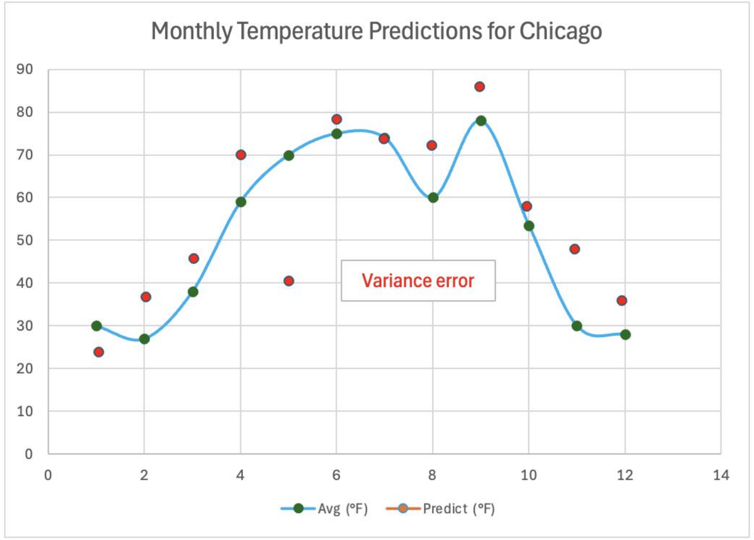




The Bias-Variance Trade-Off

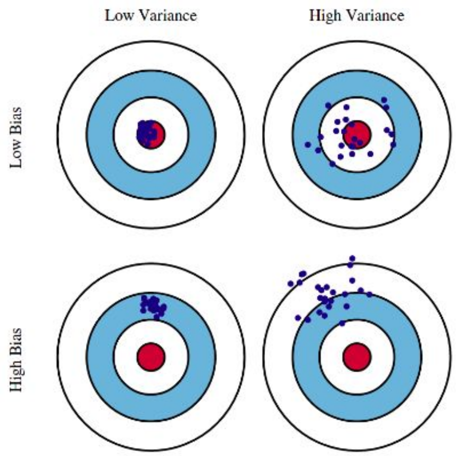
- Example: Bias and Variance





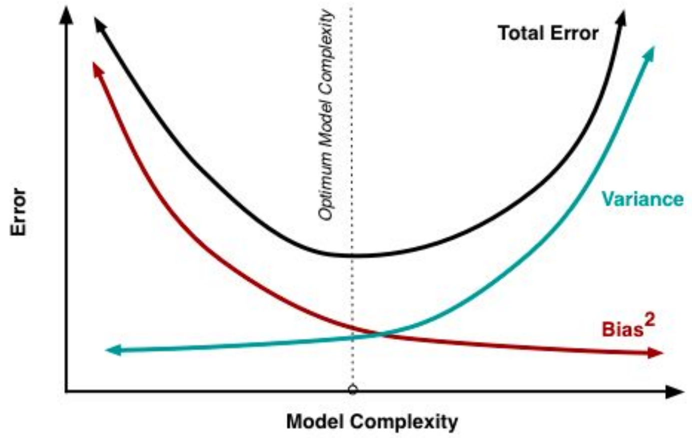
The Bias-Variance Trade-Off

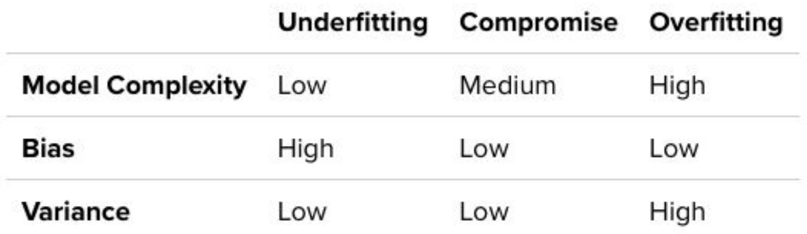
- **Different combinations of bias and variance in machine learning models**
  - Low bias, low variance: ideal model
    A machine learning model with low bias and low variance is considered ideal but is not often the case in the machine learning practice, so we can speak of “reasonable bias” and “reasonable variance”.
  - Low bias, high variance: results in **overfitting**
    This combination results in inconsistent predictions that are accurate on average. It occurs when a model has too many parameters and fits too closely to the training data.
  - High bias, low variance: results in **underfitting**
    Predictions are consistent but inaccurate on average in this scenario. This happens when the model doesn’t learn well from the training data or has too few parameters, leading to underfitting issues.
  - High bias, high variance: results in **inaccurate** predictions
    With both high bias and high variance, the predictions are both inconsistent and inaccurate on average.



The Bias-Variance Trade-Off

- **Why do we need to do Bias-Variance Trade-Off?**
  - Models that have high bias tend to have low variance.
  - However, models that have low bias tend to have high variance.

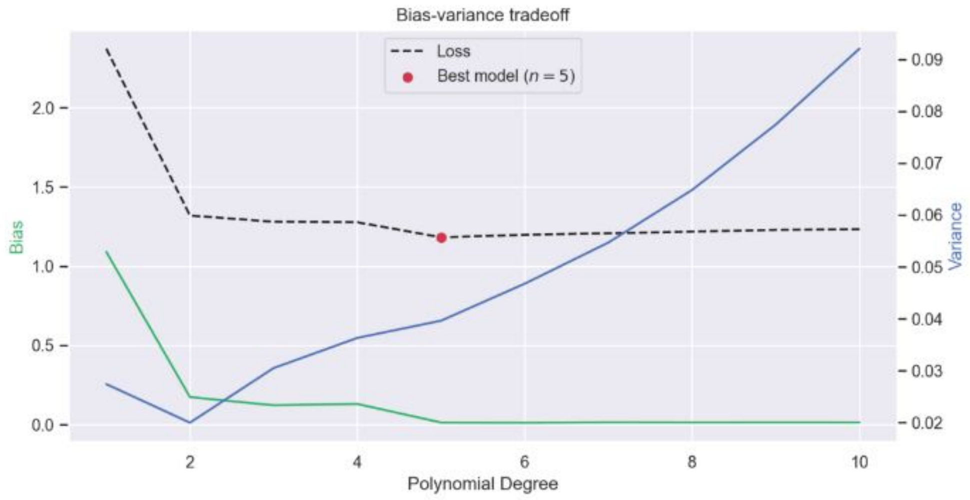




	Underfitting	Compromise	Overfitting
Model Complexity	Low	Medium	High
Bias	High	Low	Low
Variance	Low	Low	High

The Bias-Variance Trade-Off

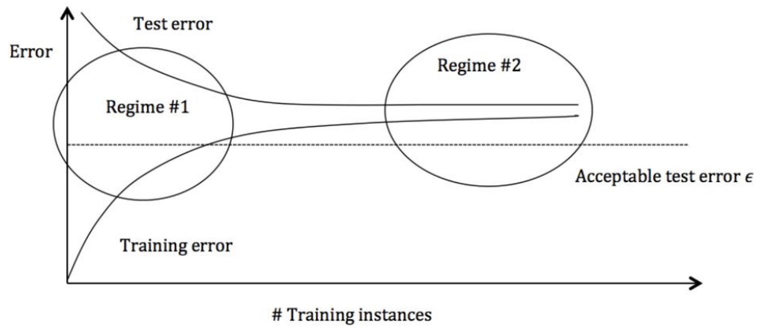
- **Example in Python**
  - Using Python to achieve a good bias-variance tradeoff in the model
  - See Notebook



The Bias-Variance Trade-Off

- **Detecting High Bias and High Variance**

If a classifier is under-performing (e.g. if the test or training error is too high), there are several ways to improve performance. To find out which of these many techniques is the right one for the situation, the first step is to determine the root of the problem.

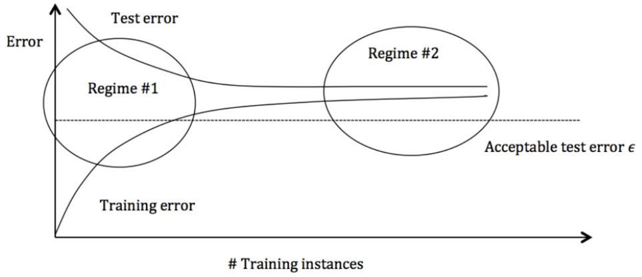


Lecture 12: Bias Variance Tradeoff. (n.d.). Www.cs.cornell.edu. https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

The graph above plots the training error and the test error and can be divided into two overarching regimes. In the first regime (on the left side of the graph), training error is below the desired error threshold (denoted by ϵ), but test error is significantly higher. In the second regime (on the right side of the graph), test error is remarkably close to training error, but both are above the desired tolerance of ϵ.

The Bias-Variance Trade-Off

- **Detecting High Bias and High Variance**
  - Regime 1 (High Variance)
  - In the first regime, the cause of the poor performance is high variance.
    - **Symptoms**
      - Training error is much lower than test error
      - Training error is lower than ϵϵ
      - Test error is above ϵ
    - **Remedies**
      - Add more training data
      - Reduce model complexity -- complex models are prone to high variance
      - Bagging (will be covered later in the course)
  - Regime 2 (High Bias)
  - Unlike the first regime, the second regime indicates high bias: the model being used is not robust enough to produce an accurate prediction.
    - **Symptoms**
      - Training error is higher than ϵ
      -
    - **Remedies**
      - Use more complex model (e.g. kernelize, use non-linear models)
      - Add features
      - Boosting (will be covered later in the course)



Overfitting and Underfitting

- **Overfitting**

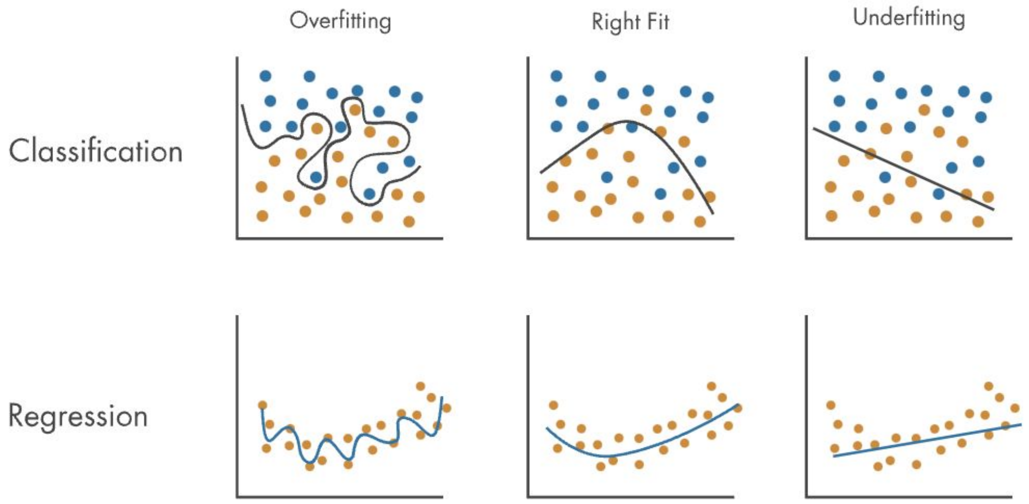
  Fitting the data too well

  - Features are noisy, uncorrelated to concept
  - Modeling process very sensitive (powerful)
  - Too much search

- **Underfitting**

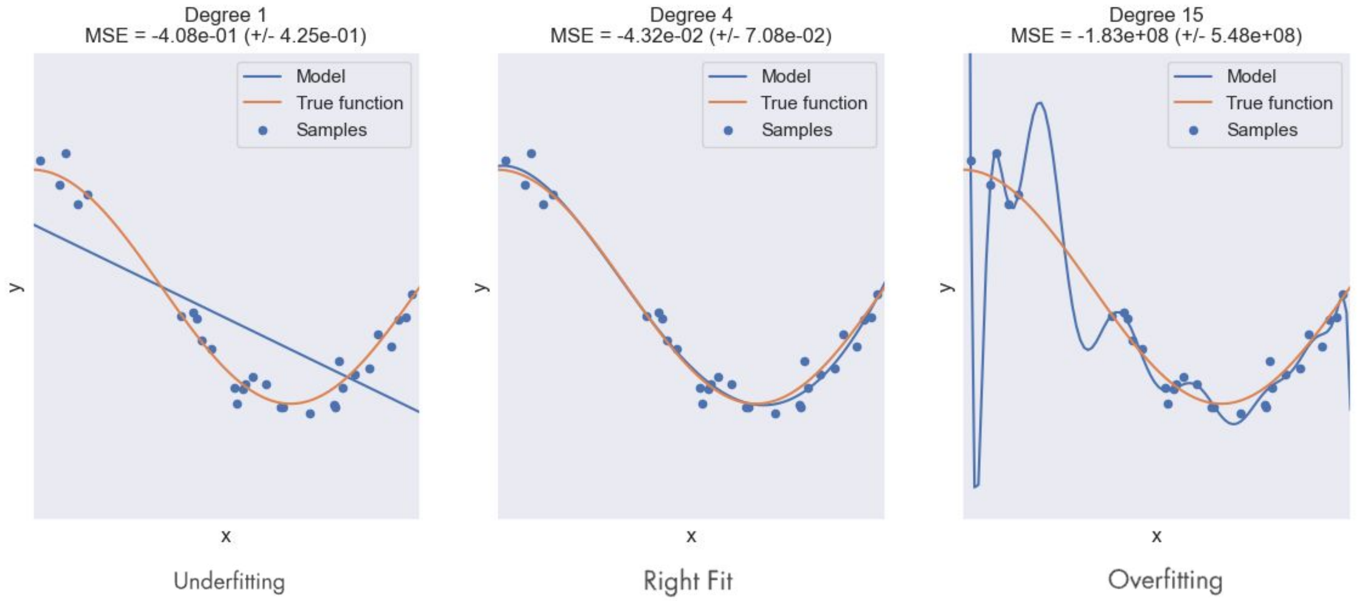
  Learning too little of the true concept

  - Features don't capture concept
  - Too much bias in model
  - Too little search to fit model



Overfitting and Underfitting

- **An example Python code that demonstrates overfitting and underfitting**



Overfitting and Underfitting

- **How to fix overfitting?**
  - Regularization:
    - Use L1 (Lasso) or L2 (Ridge) regularization to penalize large coefficients.
    - Regularization helps prevent the model from fitting noise in the training data.
  - Dropout:
    - Apply dropout layers during training in neural networks.
    - Dropout randomly deactivates some neurons, reducing over-reliance on specific features.
  - More Data:
    - Collect additional training data if possible.
    - A larger dataset can help the model generalize better.
  - Simpler Models:
    - Choose simpler model architectures with fewer parameters.
    - Avoid overly complex models that can memorize the training data.
  - Early Stopping:
    - Monitor the validation loss during training.
    - Stop training when the validation loss starts increasing (indicating overfitting).

Overfitting and Underfitting

- **Some strategies to address underfitting**
  - Increase Training Data:
    - Collect more training examples to improve the model's ability to generalize.
    - A larger dataset can help the model learn better patterns and reduce underfitting.
  - Model Complexity:
    - Increase the complexity of the model architecture.
    - Use deeper neural networks or more complex algorithms to capture intricate relationships in the data.
  - Increase Model Parameters:
    - Add more parameters (such as hidden units in neural networks) to the model.
    - This allows the model to learn more flexible representations.
  - Extend Training Time:
    - Train the model for longer epochs.
    - Continue training until the cost function converges to a minimum.

Parametric and Nonparametric ML Algorithms

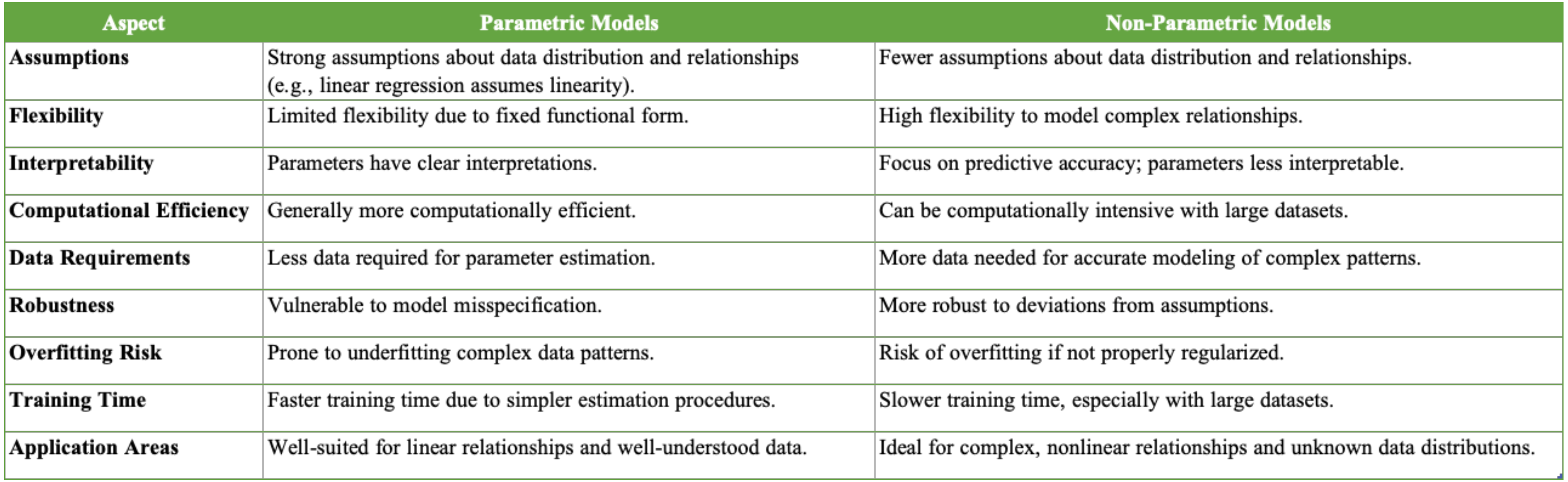
- **Parametric ML Algorithms vs. Nonparametric ML Algorithms**
  - **Parametric Algorithms**

    Parametric algorithms operate under predefined assumptions regarding the distribution and relationships in the underlying data, utilizing a fixed number of parameters to predict new data points based on these assumptions.
    - Example
      - Logistic Regression
      - Linear Discriminant Analysis
      - Perceptron
      - Naive Bayes
      - Simple Neural Networks
  - **Nonparametric Algorithms**

    Parametric algorithms operate under predefined assumptions regarding the distribution and relationships in the underlying data, utilizing a fixed number of parameters to predict new data points based on these assumptions.
    - Example
      - k-Nearest Neighbors
      - Decision Trees like CART and C4.5
      - Support Vector Machines (SVM)

Parametric and Nonparametric ML Algorithms

- **Parametric ML Algorithms vs. Nonparametric ML Algorithms**



Aspect	Parametric Models	Non-Parametric Models
Assumptions	Strong assumptions about data distribution and relationships (e.g., linear regression assumes linearity).	Fewer assumptions about data distribution and relationships.
Flexibility	Limited flexibility due to fixed functional form.	High flexibility to model complex relationships.
Interpretability	Parameters have clear interpretations.	Focus on predictive accuracy; parameters less interpretable.
Computational Efficiency	Generally more computationally efficient.	Can be computationally intensive with large datasets.
Data Requirements	Less data required for parameter estimation.	More data needed for accurate modeling of complex patterns.
Robustness	Vulnerable to model misspecification.	More robust to deviations from assumptions.
Overfitting Risk	Prone to underfitting complex data patterns.	Risk of overfitting if not properly regularized.
Training Time	Faster training time due to simpler estimation procedures.	Slower training time, especially with large datasets.
Application Areas	Well-suited for linear relationships and well-understood data.	Ideal for complex, nonlinear relationships and unknown data distributions.

Parametric and Nonparametric ML Algorithms

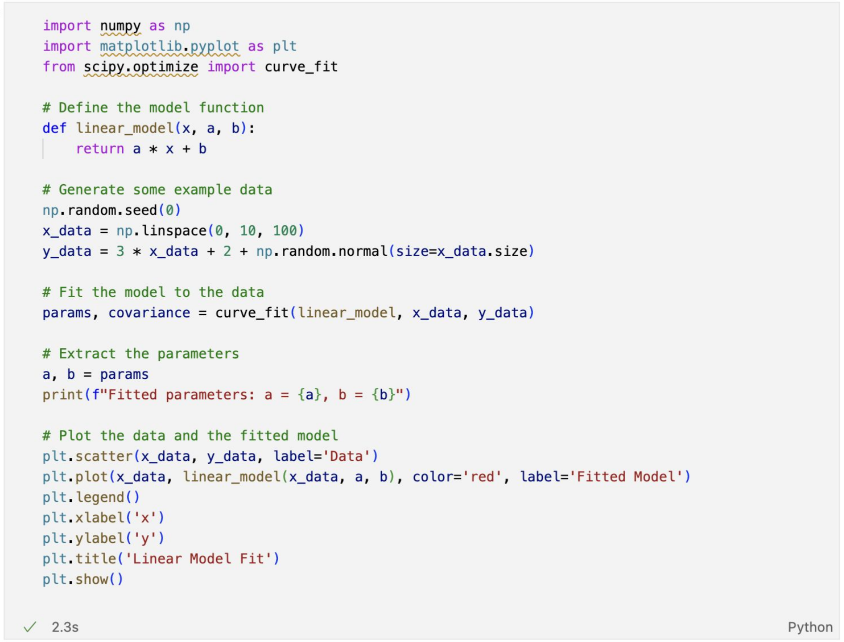
- **Nonparametric ML Algorithms Benefits & Limitations**
  - **Benefits**
    - **Flexibility**: Capable of fitting a large number of functional forms.
    - **Power**: No assumptions (or weak assumptions) about the underlying function.
    - **Performance**: Can result in higher performance models for prediction.
  - **Limitations**
    - **More data**: Require a lot more training data to estimate the mapping function.
    - **Slower**: A lot slower to train as they often have far more parameters to train.
    - **Overfitting**: More of a risk to overfit the training data and it is harder to explain why specific predictions are made.

Parametric and Nonparametric ML Algorithms

- **Parametric ML Algorithms Benefits & Limitations**
  - **Benefits**
    - **Simpler**: These methods are easier to understand and interpret results.
    - **Speed**: Parametric models are very fast to learn from data.
    - **Less Data**: They do not require as much training data and can work well even if the fit to the data is not perfect.
  - **Limitations**
    - **Constrained**: By choosing a functional form these methods are highly constrained to the specified form.
    - **Limited Complexity**: The methods are more suited to simpler problems.
    - **Poor Fit**: In practice the methods are unlikely to match the underlying mapping function.

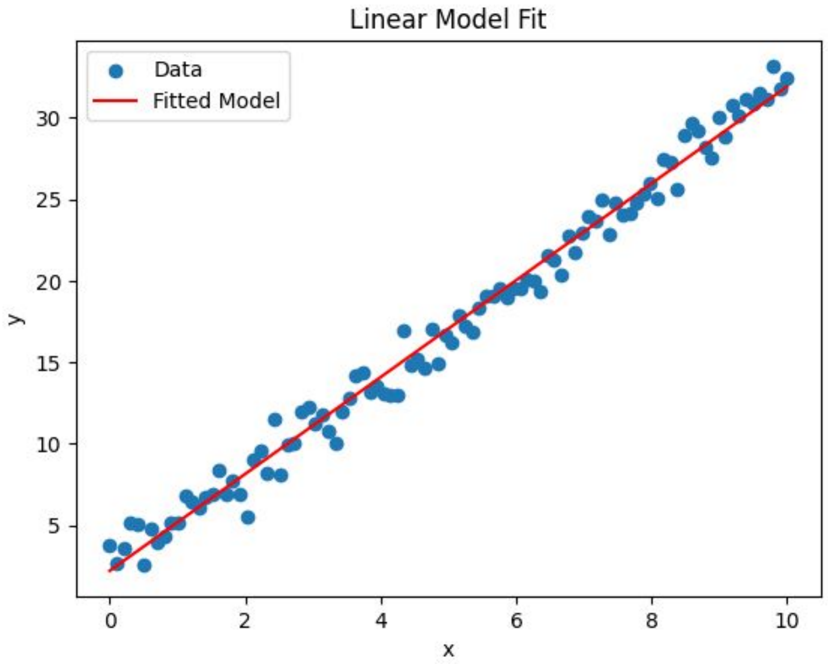
Parametric and Nonparametric ML Algorithms

- **Parametric ML Algorithms vs. Nonparametric ML Algorithms: Example in python**
  - Parametric algorithms



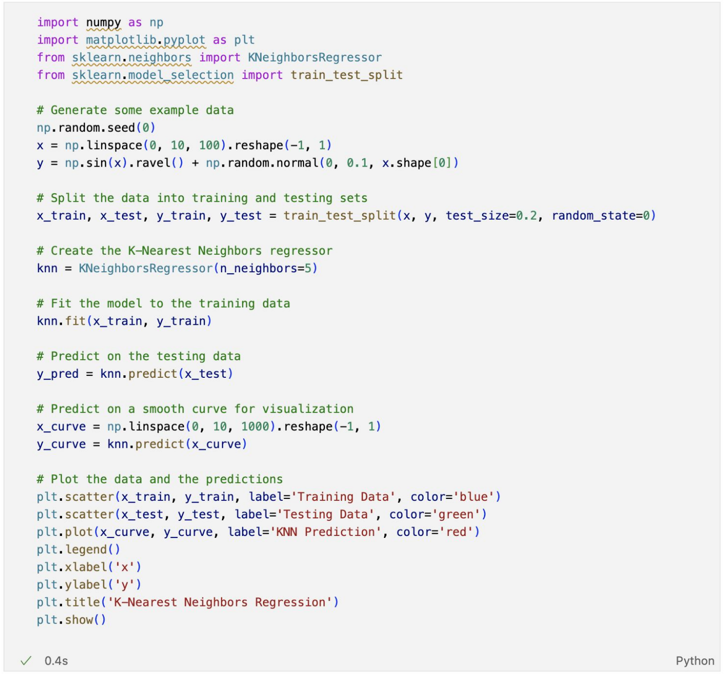
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Define the model function
def linear_model(x, a, b):
    return a * x + b

# Generate some example data
np.random.seed(0)
x_data = np.linspace(0, 10, 100)
y_data = 3 * x_data + 2 + np.random.normal(size=x_data.size)

# Fit the model to the data
params, covariance = curve_fit(linear_model, x_data, y_data)

# Extract the parameters
a, b = params
print(f"Fitted parameters: a = {a}, b = {b}")

# Plot the data and the fitted model
plt.scatter(x_data, y_data, label='Data')
plt.plot(x_data, linear_model(x_data, a, b), color='red', label='Fitted Model')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Model Fit')
plt.show()
```

```
Fitted parameters: a = 2.970267314589279, b = 2.208471442207131
```



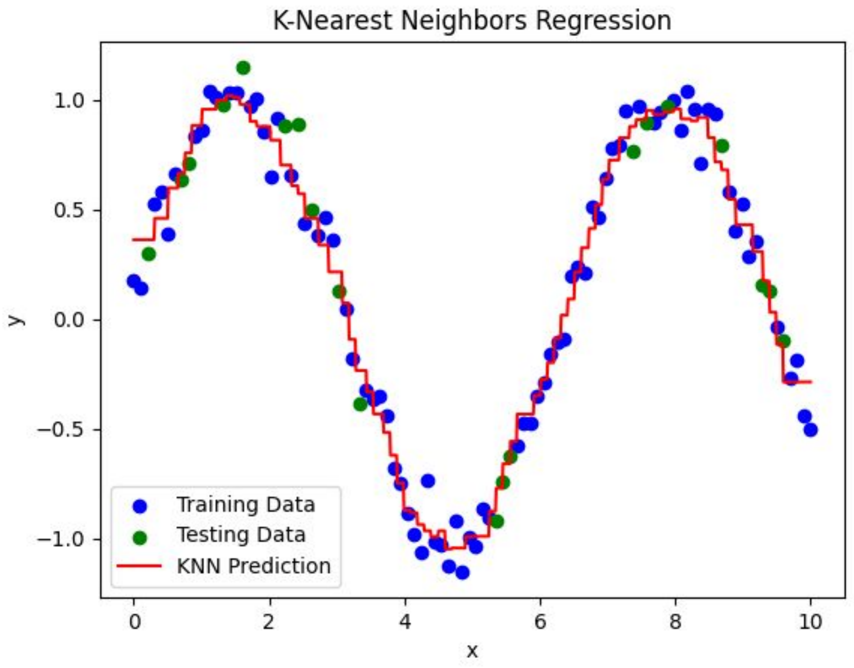
Parametric and Nonparametric ML Algorithms

- **Parametric ML Algorithms vs. Nonparametric ML Algorithms: Example in python**
  - Nonparametric algorithms



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split

# Generate some example data
np.random.seed(0)
x = np.linspace(0, 10, 100).reshape(-1, 1)
y = np.sin(x).ravel() + np.random.normal(0, 0.1, x.shape[0])

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

# Create the K-Nearest Neighbors regressor
knn = KNeighborsRegressor(n_neighbors=5)

# Fit the model to the training data
knn.fit(x_train, y_train)

# Predict on the testing data
y_pred = knn.predict(x_test)

# Predict on a smooth curve for visualization
x_curve = np.linspace(0, 10, 1000).reshape(-1, 1)
y_curve = knn.predict(x_curve)

# Plot the data and the predictions
plt.scatter(x_train, y_train, label='Training Data', color='blue')
plt.scatter(x_test, y_test, label='Testing Data', color='green')
plt.plot(x_curve, y_curve, label='KNN Prediction', color='red')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('K-Nearest Neighbors Regression')
plt.show()
```

✓ 0.4s Python



Example: ML Algorithms in Python

- **See Notebook**