Introduction to Artificial Intelligence using Python

Week 3 of 4

Anass B. El-Yaagoubi

STAT - CEMSE, King Abdullah University of Science and Technology

KAUST Academy: AI Summer School

Thuwal, July 16th, 2023

Course Overview

- Goal of this course is to introduce you to artificial intelligence (AI).
- Learn how to implement various AI algorithms in Python.
- Four weeks program.
  1. Python programming
  2. Data wrangling and visualization
  3. Machine learning
  4. Neural networks and deep learning
- AI project in parallel the course

Overview of week 3:

I What is Machine Learning?

II Linear vs Logistic regression

III Introduction to Clustering

IV Classification algorithms I

V Classification algorithms II

Day 1: What is Machine Learning?

- What does it mean to learn for a machine?
- Supervised vs Unsupervised Learning.
- Regression vs Classification problems.
- MSE, Accuracy, Precision and Recall.
- Training and testing datasets.

What does it mean to learn for a machine?

- Machines learn by acquiring knowledge or skills from data or experience.
- Learning enables machines to generalize from examples and make predictions or decisions on new data.
- Machine learning algorithms detect patterns in data to make informed decisions or predictions.
- Learning involves optimizing model parameters based on feedback or evaluation metrics.
- Learning for machines can be supervised, unsupervised, or reinforcement-based.

What does it mean to learn for a machine?

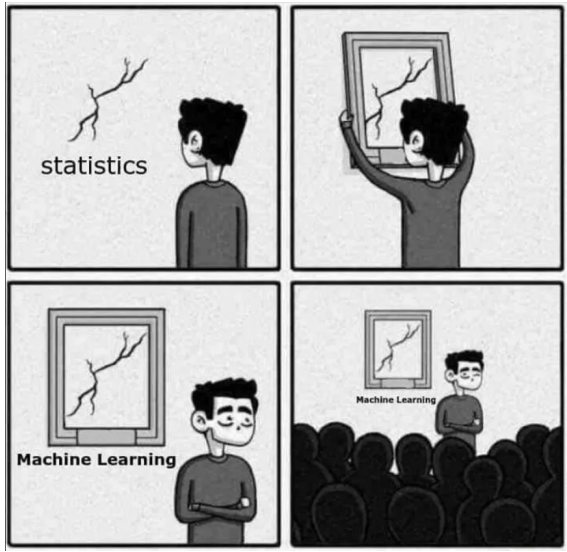

Fig. 1: Machine learning in a nutshell.

Supervised vs Unsupervised Learning.

- **Supervised Learning**: Uses labeled data to train a model and predict unseen inputs.
  - Requires labeled data.
  - Focuses on prediction.
  - Evaluation using metrics like accuracy or mean squared error.
  - Examples: Classification, Regression.

- **Unsupervised Learning**: Uses unlabeled data to discover patterns and relationships in the data.
  - Works with unlabeled data.
  - Focuses on discovering hidden patterns.
  - Evaluation can be subjective.
  - Examples: Clustering, Dimensionality Reduction.

Supervised vs Unsupervised Learning.

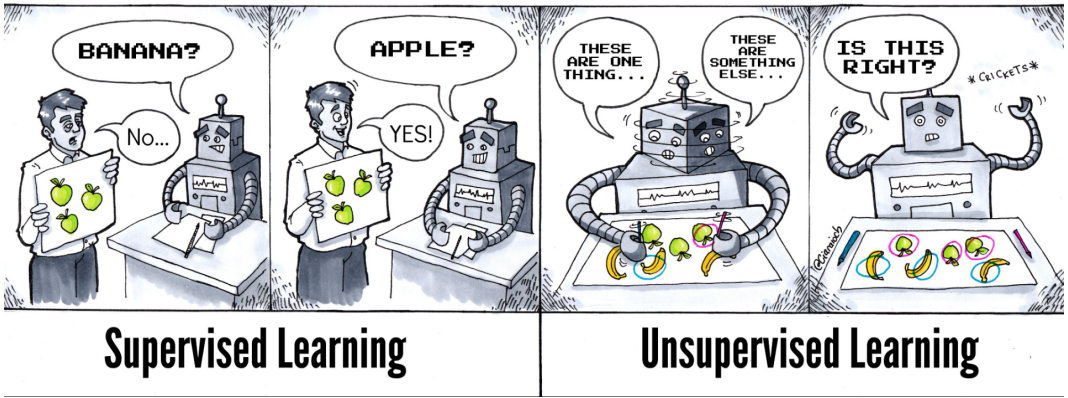

Fig. 2: Supervised vs Unsupervised Learning in a nutshell.

Regression vs Classification Problems

- **Regression**: Predicts continuous numerical values.
  - Utilizes algorithms like Linear Regression, Polynomial Regression.
  - Objective: Minimize the difference between predicted and actual values.
  - Examples: Predicting house prices, estimating sales revenue.
  - $\{(X_i, Y_i)\}_{i=1}^n, \quad X_i \in \mathbb{R}^n, \quad Y_i \in \mathbb{R}$

- **Classification**: Predicts discrete categories or classes.
  - Utilizes algorithms like Logistic Regression, Decision Trees.
  - Objective: Assign the correct class label to each input.
  - Examples: Email spam detection, image classification.
  - $\{(X_i, Y_i)\}_{i=1}^n, \quad X_i \in \mathbb{R}^n, \quad Y_i \in \{0, 1, \ldots, C-1\}$

MSE, Accuracy, Precision, and Recall

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values in regression problems.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Accuracy**: Measures the proportion of correctly classified instances in classification problems.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: Measures the proportion of true positive predictions among the predicted positive instances.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: Measures the proportion of true positive predictions among the actual positive instances.

$$Recall = \frac{TP}{TP + FN}$$

Training and Testing Datasets

- **Data Split**:
  - The training dataset typically comprises 80% of the data.
  - The testing dataset typically comprises 20% of the data.
  - This split helps evaluate the model's performance on unseen data.

- **Training Dataset**: Used to train the machine learning model.
  - Contains labeled data with input features ($\mathbf{X}$) and corresponding target labels ($\mathbf{y}$).
  - Used to estimate model parameters and learn patterns.

- **Testing Dataset**: Used to evaluate the performance of the trained model.
  - Contains unlabeled data with input features ($\mathbf{X}$).
  - Model predictions are compared against the true target labels to assess performance metrics.
  - Ensures the model generalizes well to unseen data.

Day 2: Linear vs Logistic regression

- What is linear regression?
- MSE as a loss function
- How can we learn the model parameters?
- What is logistic regression?
- Log likelihood as a loss function
- How can we learn the model parameters?

Linear Regression

- **Simplicity**: Straightforward to understand and implement.
- **Interpretability**: Coefficients provide clear interpretations of feature importance.
- **Efficiency**: Relatively low computational complexity.
- **Baseline Model**: Serves as a baseline to compare with advanced algorithms.

Linear Regression

$$
\begin{aligned}
y_1 &= x_{1,1}\theta_1 + x_{1,2}\theta_2 + \cdots + x_{1,p}\theta_p + \epsilon_1 \\
y_2 &= x_{2,1}\theta_1 + x_{2,2}\theta_2 + \cdots + x_{2,p}\theta_p + \epsilon_2 \\
&\vdots \\
y_n &= x_{n,1}\theta_1 + x_{n,2}\theta_2 + \cdots + x_{n,p}\theta_p + \epsilon_n
\end{aligned}
$$

Linear Regression

Notation:

$$\mathrm{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathrm{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \quad \mathrm{E} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

This notation is used to represent the linear regression model components:

- Y: The $n \times 1$ vector of target variable (class) values.
- X: The $n \times p$ matrix of feature values.
- θ: The $p \times 1$ vector of model parameters (coefficients).
- E: The $n \times 1$ vector of error terms (residuals).

Linear Regression

Linear Equation:

$$\mathrm{Y} = \mathrm{X}\theta + \mathrm{E}$$

Optimization Problem:

$$\min_{\theta} J(\theta) = \frac{1}{2n} \|\mathrm{Y} - \mathrm{X}\theta\|_2^2$$

Cost Function (MSE):

$$J(\theta) = \frac{1}{2n} \|\mathrm{Y} - \mathrm{X}\theta\|_2^2$$

Gradient Function:

$$\nabla J(\theta) = -\frac{1}{n} \mathrm{X}^T (\mathrm{Y} - \mathrm{X}\theta)$$

Linear Regression

Algorithm 1 Gradient Descent for Multivariate Linear Regression

Require: Training data: (X, y) where X is an $n \times p$ matrix and y is an $n \times 1$ vector
Require: Learning rate: α
Require: Number of iterations: N
Ensure: Model parameters: θ

1: Initialize parameters θ with small random values or zeros.
2: **for** $k = 0$ to $N - 1$ **do**
3: Calculate the gradient of the cost function: $\nabla J(\theta_k) = -\frac{1}{n} X^T (Y - X\theta_k)$.
4: Update the parameters: $\theta_{k+1} = \theta_k - \alpha \cdot \nabla J(\theta_k)$.
5: Calculate and save the cost function: $J(\theta_{k+1}) = \frac{1}{2n} ||Y - X\theta_{k+1}||_2^2$.
6: **end for**
7: Output the final parameter values θ.

Logistic Regression

- **Binary Classification**: Logistic regression is used for binary classification problems where the target variable can take only two possible classes.
- **Probabilistic Output**: Unlike linear regression, logistic regression outputs probabilities in the range [0, 1], representing the likelihood of the input belonging to a particular class.
- **Decision Boundary**: Logistic regression uses a decision boundary to separate the two classes based on the estimated probabilities.
- **Interpretability**: Coefficients still provide insights into feature importance, indicating how each feature influences the probability of a certain class.

Logistic Regression

Logistic Regression Equation:

$$
\begin{aligned}
p_1 &= \mathbb{P}(y_1 = 1 | x_{1,1}, x_{1,2}, \ldots, x_{1,p}) = \sigma(x_{1,1}\theta_1 + x_{1,2}\theta_2 + \cdots + x_{1,p}\theta_p + \epsilon_1) \\
p_2 &= \mathbb{P}(y_2 = 1 | x_{2,1}, x_{2,2}, \ldots, x_{2,p}) = \sigma(x_{2,1}\theta_1 + x_{2,2}\theta_2 + \cdots + x_{2,p}\theta_p + \epsilon_2) \\
&\vdots \\
p_n &= \mathbb{P}(y_n = 1 | x_{n,1}, x_{n,2}, \ldots, x_{n,p}) = \sigma(x_{n,1}\theta_1 + x_{n,2}\theta_2 + \cdots + x_{n,p}\theta_p + \epsilon_n)
\end{aligned}
$$

Where:

$$
\sigma(z) = \frac{1}{1 + e^{-z}}
$$

p_i is the probability of the i-th observation.
y_i is the class of of the i-th observation. Given its features $\mathbf{x}_i$, the logistic function $\sigma(z)$ maps the linear combination of features and coefficients to the range [0, 1], making it suitable for binary classification.

Logistic Regression

Log-Odds Ratio: The log-odds ratio, also known as the logit function, is expressed as a linear function of the features and coefficients in logistic regression. It is given by:

$$\sigma^{-1}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = x_{i,1}\theta_1 + x_{i,2}\theta_2 + \cdots + x_{i,p}\theta_p + \epsilon_i$$

The $\sigma^{-1}(.)$ is called the logit function and it maps the probabilities p_i to a continuous range from negative infinity to positive infinity.

Logistic Regression

Notation:

$$\mathrm{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathrm{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \quad \mathrm{E} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

This notation is used to represent the logistic regression model components:

- Y: The $n \times 1$ vector of target variable (class) values.
- X: The $n \times p$ matrix of feature values.
- θ: The $p \times 1$ vector of model parameters (coefficients).
- E: The $n \times 1$ vector of error terms (residuals).

Logistic Regression

Logistic Function:

$$p = \sigma(X\theta + E)$$

Optimization Problem:

$$\min_{\theta} J(\theta)$$

Cost Function (Negative Log-Likelihood):

$$J(\theta) = -\frac{1}{n}\left(Y^T \log(\sigma(X\theta)) + (\mathbb{1} - Y)^T \log(\mathbb{1} - \sigma(X\theta))\right)$$

Gradient Function:

$$\nabla J(\theta) = -\frac{1}{n} X^T (Y - \sigma(X\theta))$$

Logistic Regression

Algorithm 2 Gradient Descent for Logistic Regression

Require: Training data: (X, y) where X is an $n \times p$ matrix and y is an $n \times 1$ vector
Require: Learning rate: α
Require: Number of iterations: N
Ensure: Model parameters: θ

1: Initialize parameters θ with small random values or zeros.
2: **for** $k = 0$ to $N - 1$ **do**
3: Calculate the predicted probabilities using θ_k: $p = \sigma(X\theta_k)$.
4: Calculate the gradient of the cost function: $\nabla J(\theta_k) = -\frac{1}{n} X^T (Y - \sigma(X\theta_k))$.
5: Update the parameters: $\theta_{k+1} = \theta_k - \alpha \cdot \nabla J(\theta_k)$.
6: Calculate and save the cost function: $J(\theta_{k+1})$.
7: **end for**
8: Output the final parameter values θ.

Day 3: Introduction to Clustering

- What kind of learning is clustering?
- Why do we need to cluster data?
- k-Means algorithm
- HCA algorithm

Key Ideas Behind Clustering

- **Group Similar Data**: Clustering groups data points with similar characteristics.
- **Distance Metrics**: Clustering uses distance metrics to measure data similarity.
- **Unsupervised Learning**: Clustering does not require labeled data.
- **Number of Clusters**: The number of clusters is often predetermined or determined using evaluation metrics.
- **Pattern Discovery**: Clustering discovers underlying patterns and structures in the data.

Applications of Clustering

- **Marketing**: Customer segmentation for targeted strategies.
- **Image Segmentation**: Grouping pixels for object detection.
- **Anomaly Detection**: Identifying unusual patterns in data.
- **Recommendation Systems**: Personalized item/user grouping.
- **Bioinformatics**: Categorizing genes, proteins, patients.

k-Means Clustering

- **Objective**: Partition data into K clusters, minimizing the within-cluster sum of squares.
- **Algorithm**:
  1. Randomly initialize K centroids.
  2. Assign each data point to the nearest centroid.
  3. Update centroids as the mean of the points in each cluster.
  4. Repeat steps 2-3 until convergence.

k-Means Clustering

- **Distance Metric**: Euclidean distance is commonly used to measure point-to-centroid distance.
- **Optimization**: k-Means aims to minimize the Within-Cluster Sum of Squares (WCSS) as follows:

$$\text{WCSS} = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \text{centroid}_i\|^2$$

- **Initialization**: The algorithm's final result can be sensitive to initial centroid placement.
- **Elbow Method**: Select K by finding the "elbow" point in the WCSS plot.

k-Means Clustering with scikit-learn

```python
from sklearn.cluster import KMeans
# Create a KMeans object with the desired number of clusters
kmeans = KMeans(n_clusters=3)
# Fit the model to the data
kmeans.fit(X)
# Get the cluster centers and labels
ccs = kmeans.cluster_centers_
labels = kmeans.labels_
# Plot the data points with color-coded clusters
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(ccs[:, 0], ccs[:, 1], c='red', marker='X', s=200)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('k-Means Clustering')
plt.legend()
plt.show()
```

Advantages, Disadvantages, and Applications of k-Means Clustering:

- **Advantages**:
  - Fast and scalable for large datasets.
  - Simple and easy to implement.
  - Works well with well-separated clusters.
- **Disadvantages**:
  - Sensitive to initial centroids.
  - May converge to local optima.
  - Struggles with non-linear or overlapping clusters.
- **Applications**:
  - Customer Segmentation.
  - Image Compression.
  - Anomaly Detection.

Hierarchical Clustering Algorithm (HCA)

- **Objective**: HCA builds a hierarchy of clusters by recursively merging or splitting them.
- **Algorithm**:
  1. Assign each data point to its own cluster (initially N clusters).
  2. Compute the pairwise distance (e.g., Euclidean) between clusters.
  3. Merge the two closest clusters into a single cluster.
  4. Repeat steps 2-3 until all data points belong to a single cluster.

Hierarchical Clustering Algorithm (HCA)

- **Linkage Methods**: HCA uses different methods to calculate distances between clusters:
  - Single Linkage: Shortest distance between any two points in the clusters.
  - Complete Linkage: Longest distance between any two points in the clusters.
  - Average Linkage: Average distance between all pairs of points in the clusters.
- **Dendrogram**: HCA visualizes the clustering hierarchy with a dendrogram.

HCA Implementation with scipy

```
from scipy.cluster.hierarchy import linkage, dendrogram

# Compute the linkage matrix using single linkage
linkage_matrix = linkage(X, method='single', metric='euclidean')

# Plot the dendrogram
plt.figure(figsize=(10, 6))
dendrogram(linkage_matrix)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()
```

HCA Implementation with scipy

```
from scipy.cluster.hierarchy import fcluster
# Cut the dendrogram to obtain clusters
distance_threshold = 0.5  # Set your desired distance threshold here
clusters = fcluster(
    linkage_matrix,
    distance_threshold,
    criterion='distance'
)
# Scatter plot the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap='viridis')
plt.title('Hierarchical Clustering Result')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```

Advantages, Disadvantages, and Applications of HCA:

- **Advantages**:
  - Flexibility to choose the number of clusters after dendrogram analysis.
  - No need to specify the number of clusters beforehand.
  - Captures hierarchical relationships between clusters.
- **Disadvantages**:
  - Computationally expensive for large datasets.
  - Sensitive to noise and outliers.
  - Not suitable for large datasets.
- **Applications**:
  - Genetics.
  - Image Segmentation.
  - Social Sciences.

Day 4: Classification algorithms I

- K Nearest Neighbors
- Decision trees

k-Nearest Neighbors (k-NNs)

- **Definition**: k-NN is a simple and popular algorithm for classification and regression.
- **Idea**: Classifies a data point by finding the majority class among its k-nearest neighbors.
- **Distance Metric**: Uses a distance metric (e.g., Euclidean distance) to measure similarity between data points.

k-Nearest Neighbors (k-NNs)

- **Choosing K**: The value of K determines the number of neighbors considered for classification.
- **Hyperparameter**: K is a hyperparameter tuned using cross-validation or other techniques.
- **Decision Boundary**: k-NN's boundary can be nonlinear, determined by data point distribution.

k-Nearest Neighbors (k-NN) Implementation with sklearn

```python
from sklearn.neighbors import KNeighborsClassifier

# Create k-NN classifier
k = 5
knn_classifier = KNeighborsClassifier(n_neighbors=k)

# Fit the model using training data
knn_classifier.fit(X_train, y_train)
# Predict the class labels for the test data
y_test_pred = knn_classifier.predict(X_test)

# Calculate the accuracy
accuracy = np.mean(y_test == y_test_pred)
print("Accuracy:", accuracy)
```

Advantages, Disadvantages, and Applications of k-NNs:

Advantages:

- Simple, intuitive, and easy to implement.
- Effective for a large number of classes and complex decision boundaries.
- Non-parametric nature allows flexibility with data.

Disadvantages:

- Computationally expensive for large datasets.
- Sensitive to irrelevant features, affecting accuracy.
- Optimal choice of neighbors (K) impacts performance.

Applications:

- Recommendation systems, collaborative filtering.
- Image recognition and object detection.
- Anomaly detection and outlier analysis.

Decision Trees (DT)

Definition:

- A decision tree is a supervised machine learning algorithm used for both classification and regression tasks.
- It models decisions as a tree-like structure where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label or a numerical value.

DTs Implementation with sklearn

```python
from sklearn.tree import DecisionTreeClassifier

# Create Decision Tree classifier
dt_classifier = DecisionTreeClassifier()

# Fit the model using training data
dt_classifier.fit(X_train, y_train)

# Predict the class labels for the test data
y_test_pred = dt_classifier.predict(X_test)

# Calculate the accuracy
accuracy = np.mean(y_test == y_test_pred)
print("Accuracy:", accuracy)
```

Advantages, Disadvantages, and Applications of DTs:

Advantages:

- Easy to interpret and handle both numerical and categorical data.
- Non-parametric, no assumptions about data distribution needed.
- Performs automatic feature selection.

Disadvantages:

- Prone to overfitting, especially with deep trees and noisy data.
- Instability: Small data changes can lead to completely different trees.
- Greedy nature: Decisions are locally optimal but not always globally.

Applications:

- Classification and regression in medicine, finance, and marketing.
- Predicting house or stock prices.
- Feature engineering and ensemble methods like Random Forests.

Day 5: Classification algorithms II

- Support Vector Machines.
- Random Forests.

Support Vector Machines (SVMs)

Introduction:

- SVM is a powerful supervised learning algorithm used for classification and regression tasks.
- It finds the optimal hyperplane that best separates the data points into different classes.
- SVM is effective for both linearly separable and non-linearly separable data.

How do SVMs work?

Optimization Problem:

$$\text{Maximize } \frac{2}{\|\mathbf{w}\|}$$
$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

Meaning:

- The optimization problem seeks to find the maximum margin that separates the two classes represented by +1 and -1.
- The vector $\mathbf{w}$ is perpendicular to the decision boundary, and its magnitude represents the margin width.
- The equation $\mathbf{w} \cdot \mathbf{x} + b = 0$ defines the decision boundary that separates the two classes.

How do SVMs work?

Soft Margin and Kernel Trick:

- **Soft Margin**: Allows for misclassification to achieve a more flexible decision boundary.
- **Kernel Trick**: Transforms data into a higher-dimensional space to handle non-linear decision boundaries.

Optimization Problem with Soft Margin:

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i \text{ and } \xi_i \geq 0 \quad \forall i$$

Meaning:

- The parameter C controls margin-maximization vs. classification errors.
- The function $\phi(\mathbf{x})$ maps the data to a higher-dimensional space, allowing for non-linear decision boundaries.

SVM Implementation with sklearn

```
from sklearn.svm import SVC

# Create SVM classifier
svm_classifier = SVC(kernel='linear', C=1.0)

# Fit the model using training data
svm_classifier.fit(X_train, y_train)

# Predict the class labels for the test data
y_test_pred = svm_classifier.predict(X_test)

# Calculate the accuracy
accuracy = mp.mean(y_test == y_test_pred)
print("Accuracy:", accuracy)
```

Advantages, Disadvantages, and Applications of SVMs:

Advantages:

- Effective in high-dimensional spaces and with non-linear boundaries.
- Robust against overfitting, especially with the soft margin parameter.
- Versatile with various kernel functions to handle non-linear data.

Disadvantages:

- Computationally intensive for large datasets and complex models.
- Sensitive to the choice of hyperparameters, like the kernel and regularization parameter.
- Hard to interpret the decision boundaries for highly non-linear data.

Applications:

- Object detection in computer vision.
- Predictive maintenance for industrial equipment.
- Fraud detection in financial transactions.

Random Forests (RFs)

- Ensemble method for classification and regression.
- Builds multiple decision trees on random subsets of data and features.
- Combines tree predictions using majority vote or averaging for improved accuracy and generalization.

How do RFs Work?

- **Bootstrap Aggregating (Bagging)**: Train decision trees on random data subsets with replacement.
- **Random Feature Selection**: Use random subsets of features for each tree to introduce diversity.
- **Voting (Classification) or Averaging (Regression)**: Combine tree predictions using majority vote for classification tasks or averaging for regression tasks.

RFs Implementation with sklearn

```python
from sklearn.ensemble import RandomForestClassifier
# Create Random Forest classifier
rf_classifier = RandomForestClassifier(
    n_estimators=100,
    criterion='gini',
    max_depth=3
)
# Fit the model using training data
rf_classifier.fit(X_train, y_train)
# Predict the class labels for the test data
y_test_pred = rf_classifier.predict(X_test)
# Calculate the accuracy
accuracy = np.mean(y_test == y_test_pred)
print("Accuracy:", accuracy)
```

Advantages, Disadvantages, and Applications of RFs:

Advantages:

- High accuracy from an ensemble of diverse trees.
- Robust to overfitting and noisy data.
- Handles numerical and categorical data.

Disadvantages:

- Computationally more expensive than individual decision trees.
- Difficult to interpret the combined decision-making process.
- May not perform well on imbalanced datasets.

Applications: Feature ranking in various domains, image classification, predictive maintenance, and fraud detection.