**MACHINE LEARNING**

**Discuss working progress of Machine learning. l2**

### AN). Working Process of Machine Learning (Short Overview)

1. **Problem Definition**: Identify the problem and objectives.
2. **Data Collection**: Gather relevant data from various sources.
3. **Data Preprocessing**:
   * **Cleaning**: Remove noise and handle missing values.
   * **Transformation**: Normalize or standardize data.
   * **Feature Engineering**: Create or modify features for better model performance.
4. **Data Splitting**: Divide data into training, validation, and test sets.
5. **Model Selection**: Choose an appropriate algorithm (e.g., KNN, SVM).
6. **Model Training**: Train the model on the training dataset.
7. **Model Evaluation**: Assess performance using metrics like accuracy or F1-score on the validation set.
8. **Hyperparameter Tuning**: Optimize model parameters for improved performance.
9. **Final Testing**: Evaluate the model on the test dataset.
10. **Deployment**: Integrate the model into a real-world application.
11. **Monitoring and Maintenance**: Continuously monitor and update the model as needed.

**Discuss some real time examples of Machine learning. l2**

### AN). ****Real-Time Examples of Machine Learning with Definitions****

1. **Recommendation Systems**: Algorithms that suggest content/products based on user preferences and behavior.

**Example**: Netflix and Amazon suggest content/products based on user behavior.

1. **Image Recognition**: Technology that identifies and classifies objects or faces in images.

**Example**: Google Photos classifies images and detects faces/objects.

1. **Natural Language Processing (NLP)**: Techniques that enable computers to understand and respond to human language.

**Example**: Chatbots understand and respond to user queries.

1. **Spam Detection**: Systems that filter out unwanted or harmful emails using pattern recognition.

**Example**: Gmail filters out spam emails using machine learning.

1. **Fraud Detection**: Machine learning models that identify suspicious financial transactions in real-time.

**Example**: Banks identify suspicious transactions in real-time.

1. **Autonomous Vehicles**: Self-driving technology that uses sensors and algorithms to navigate and make decisions.

**Example**: Tesla uses machine learning for object recognition in self-driving cars.

1. **Predictive Maintenance**: Tools that forecast equipment failures based on data analysis from sensors.

**Example**: Manufacturers predict equipment failures using sensor data.

1. **Health Diagnosis**: AI systems that analyze medical images to detect diseases early.

**Example**: Medical imaging applications detect diseases like cancer from scans.

1. **Stock Market Prediction**: Algorithms that analyze historical data to forecast stock price movements.

**Example**: Financial firms use algorithms to forecast stock prices.

1. **Personal Assistants**: Voice-activated technologies that perform tasks and answer questions using natural language processing.

**Example**: Siri and Alexa improve voice recognition through machine learning.

**2. Discuss limitations of KNN with suitable example. l2**

### AN). ****Limitations of K-Nearest Neighbors (KNN) with Example Dataset****

#### ****Example Dataset****: Iris Dataset

| **Sepal Length** | **Sepal Width** | **Petal Length** | **Petal Width** | **Species** |
| --- | --- | --- | --- | --- |
| 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| 7.0 | 3.2 | 4.7 | 1.4 | Versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | Versicolor |
| 6.3 | 3.3 | 6.0 | 2.5 | Virginica |
| 5.8 | 2.7 | 5.1 | 1.9 | Virginica |
| 8.0 | 4.0 | 6.0 | 2.0 | Virginica |
| 5.5 | 2.4 | 3.8 | 1.1 | Versicolor |

### ****Limitations Illustrated****

1. **Computationally Intensive**:
   * For a new data point, KNN calculates the distance to all existing data points. If the dataset has thousands of instances, this can be slow.
2. **Curse of Dimensionality**:
   * As features (e.g., Sepal Length, Sepal Width) increase, the concept of "nearness" becomes less reliable. If additional features (like color) were added, the distances may not accurately reflect similarity.
3. **Sensitive to Noise**:
   * Consider if a noisy data point (e.g., a flower with incorrect measurements: 8.0, 5.0, 7.0, 2.0) is included. This can mislead KNN predictions by affecting the nearest neighbors.
4. **Choice of K**:
   * If K=1, a single noisy point could misclassify a flower. If K=3, it might average out to the wrong class in the presence of outliers.
5. **Memory Usage**:
   * KNN requires storing all data points. With large datasets, this could lead to high memory consumption.
6. **Imbalanced Datasets**:
   * If the dataset had many more Setosa than Virginica instances, KNN may frequently predict Setosa for new points, failing to recognize the minority class.

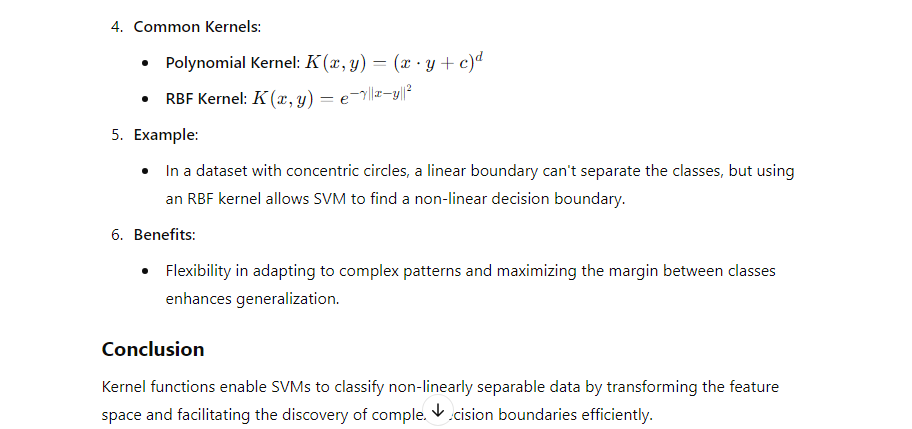
### ****Conclusion****

The Iris dataset serves as a good example to illustrate KNN's limitations. While it is simple and effective for small, well-distributed datasets, its performance can degrade with larger, high-dimensional, noisy, or imbalanced data.

**3. How do kernel functions enable SVMs to classify non-linearly separable data? L3**

### AN). Kernel Functions in SVMs for Non-Linearly Separable Data

1. **Purpose of SVMs**:
   * SVMs classify data by finding the optimal hyperplane to separate different classes.
2. **Challenge**:
   * Many datasets are not linearly separable, making it difficult to find a straight line (or hyperplane) for classification.
3. **Role of Kernel Functions**:
   * **Higher-Dimensional Mapping**: Kernels transform data into a higher-dimensional space where classes can be separated.
   * **Kernel Trick**: They compute inner products directly, avoiding the need for explicit transformations, which is efficient.



1. **Show how to partition a dataset into training and test sets using Scikit-learn. L3**

### AN). ****Partitioning a Dataset into Training and Test Sets Using Scikit-learn****

To partition a dataset into training and test sets in Python using the Scikit-learn library, you can use the train\_test\_split function. Here’s a step-by-step guide:

#### ****Step 1: Import Necessary Libraries****

#### import numpy as np

#### import pandas as pd

#### from sklearn.model\_selection import train\_test\_split

#### ****Step 2: Create or Load Your Dataset****

For demonstration, let's create a simple dataset:

# Example dataset

data = {

'Feature1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Feature2': [10, 9, 8, 7, 6, 5, 4, 3, 2, 1],

'Label': [0, 0, 0, 1, 1, 1, 0, 0, 1, 1]

}

# Convert to DataFrame

df = pd.DataFrame(data)

#### ****Step 3: Separate Features and Labels****

# Features and Labels

X = df[['Feature1', 'Feature2']] # Features

y = df['Label'] # Labels

#### ****Step 4: Split the Dataset****

Use train\_test\_split to partition the data:

# Split dataset into training and test sets (70% training, 30% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Output the shapes of the resulting sets

print("Training set shape:"***, X\_train.shape, y\_train.shape)***

print("Test set shape:", X\_test.shape, y\_test.shape)

#### ****Parameters Explained****:

* **test\_size**: Proportion of the dataset to include in the test split (e.g., 0.3 means 30% for testing).
* **random\_state**: Controls the shuffling applied to the data before splitting. Setting a seed ensures reproducibility.

#### ****Step 5: (Optional) View the Results****

# Display the training and test sets

print("Training Features:\n", X\_train)

print("Training Labels:\n", y\_train)

print("Test Features:\n", X\_test)

print("Test Labels:\n", y\_test)

**6). Demonstrate how to impute these values using the mean, median, and mode.L3**

### AN). ****Imputing Missing Values Using Mean, Median, and Mode****

To handle missing values in a dataset, we can use techniques like imputing with the **mean**, **median**, or **mode** using Scikit-learn and Pandas.

#### ****Step 1: Import Libraries****

import pandas as pd

from sklearn.impute import SimpleImputer

#### ****Step 2: Create a Dataset with Missing Values****

# Example dataset with missing values

data = {

'Feature1': [1, 2, None, 4, 5],

'Feature2': [10, None, 8, 7, None],

'Feature3': [None, 0, 1, 1, 0]

}

df = pd.DataFrame(data)

#### ****Step 3: Impute Using Mean, Median, and Mode****

1. **Impute with Mean**:

# Create the imputer object for mean

mean\_imputer = SimpleImputer(strategy='mean')

df\_mean\_imputed = pd.DataFrame(mean\_imputer.fit\_transform(df), columns=df.columns)

1. **Impute with Median**:

# Create the imputer object for median

median\_imputer = SimpleImputer(strategy='median')

df\_median\_imputed = pd.DataFrame(median\_imputer.fit\_transform(df), columns=df.columns)

1. **Impute with Mode**:

# Create the imputer object for mode

mode\_imputer = SimpleImputer(strategy='most\_frequent')

df\_mode\_imputed = pd.DataFrame(mode\_imputer.fit\_transform(df), columns=df.columns)

#### ****Step 4: View the Imputed Datasets****

print("Mean Imputed:\n", df\_mean\_imputed)

print("Median Imputed:\n", df\_median\_imputed)

print("Mode Imputed:\n", df\_mode\_imputed)

### ****Conclusion****

This demonstrates how to handle missing values by imputing them with the mean, median, or mode. These techniques help to fill gaps in the data for more consistent analysis or machine learning tasks