NON-PARAMETRIC

Bachelor of Technology
Computer Science and Engineering

Submitted By

NAME – ABHIRUP BAG
ROLL NUMBER – 13000122082
DEPARTMENT – CSE(B)
SEMESTER – 6
PAPER – PATTERN
RECOGNITION (PEC-IT602D)

March 2025

Techno Main
EM-4/1, Sector-V, Salt Lake
Kolkata- 700091
West Bengal
India

Table of Contents

1. Introduction 3

2. Characteristics of Non-Parametric Methods 3

3. Applications of Non-Parametric Methods 3

4. Advantages of Non-Parametric Methods 4

5. Disadvantages of Non-Parametric Methods 5

6. Common non-parametric methods 5

7. K-Nearest Neighbours (KNN) 5

A. How KNN Works: 6

B. Choosing the Optimal Value of k 7

C. How KNN algorithm Works: 7

D. Distance Metrics Used in KNN 8

E. Applications of KNN 8

F. Advantages & Disadvantages 9

G. KNN Implementation in Python (Using Scikit-Learn) 9

8. Support Vector Machine (SVM) Algorithm 10

A. Key Concepts of SVM 10

B. Types of SVM 11

C. How does Support Vector Machine Algorithm Work? 11

D. How does SVM classify the data? 12

E. What to do if data are not linearly separable? 13

F. Advantages of SVM 14

G. Disadvantages of SVM 14

H. Implementing SVM Algorithm in Python 14

9. Conclusion 17

10. References 17

1. Introduction

Non-parametric methods are statistical procedures that do not make assumptions about the underlying distribution of the population under study. These methods are commonly referred to as "distribution-free" because they make no assumptions regarding the distribution's form.

The primary notion behind the parametric method is that no parameters are assumed for the provided population or the population under study. In truth, the procedures are not dependent on the population. There is no predetermined set of parameters provided here, nor is any type of distribution (normal distribution, etc.) available for use. This is why nonparametric approaches are often known as distribution-free methods.

2. Characteristics of Non-Parametric Methods

A. **No Fixed Parameter Assumptions –** Unlike parametric models that have a predefined structure, non-parametric models adapt to the data without a fixed number of parameters.

B. **Flexibility –** These methods can model complex relationships that parametric models might miss.

C. **Data-Driven Approach –** They rely on the structure and distribution of the observed data rather than a theoretical model.

D. **Computational Complexity –** Many non-parametric methods require more computational power, especially for large datasets.

3. Applications of Non-Parametric Methods

A. Non-Parametric Statistics

Non-parametric statistical methods are used when data do not meet the assumptions of parametric tests, such as normality. Examples include:

- **Mann-Whitney U Test:** A substitute for the t-test when comparing two independent samples.

- **Wilcoxon Signed-Rank Test:** A non-parametric alternative to the paired t-test.
- **Kruskal-Wallis Test:** An alternative to one-way ANOVA for comparing more than two groups.
- **Kolmogorov-Smirnov Test:** Used to compare two distributions without assuming normality.

B. Non-Parametric Machine Learning Models

In machine learning, non-parametric models adjust their complexity based on the data rather than having a fixed structure. Examples include:

- **k-Nearest Neighbours (k-NN):** A classification and regression method that makes predictions based on the closest data points.
- **Decision Trees:** Used in classification and regression, splitting data based on feature values without assuming a specific functional form.
- **Random Forests:** An ensemble learning method that builds multiple decision trees for better accuracy.
- **Support Vector Machines (SVM) with Kernel Trick:** A method that transforms data into higher dimensions to identify patterns.

4. Advantages of Non-Parametric Methods

A. Robust to outliers: Non-parametric methods are not affected by outliers in the data, making them more reliable in situations where the data is noisy.

B. Widely applicable: Non-parametric methods can be used with a variety of data types, including ordinal, nominal, and continuous data.

C. Easy to implement: Non-parametric methods are often computationally simple and easy to implement, making them suitable for a wide range of users.

5. Disadvantages of Non-Parametric Methods

A. **Less powerful:** When the assumptions of parametric methods are met, non-parametric tests are generally less powerful, meaning they are less likely to detect a real effect when it exists.
B. **May require larger sample sizes:** Non-parametric tests may require larger sample sizes than parametric tests to achieve the same level of power.
C. **Less information about the population:** Non-parametric methods provide less information about the population parameters than parametric methods.

6. Common non-parametric methods

A. **Decision Trees:** A tree-like model of decisions and their possible consequences, used for classification and regression tasks.
B. **Support Vector Machines (SVM):** A supervised learning model that finds the hyperplane that best divides a dataset into classes.
C. **K-Nearest Neighbours (KNN):** A simple, instance-based learning algorithm where an object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours.
D. **Neural Networks:** A series of algorithms that attempt to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

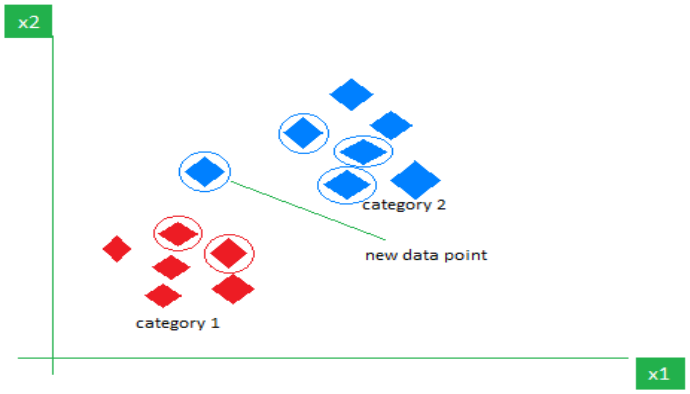
7. K-Nearest Neighbours (KNN)

K-Nearest Neighbors (KNN) is a non-parametric, **supervised** learning algorithm used for classification and regression tasks. It is based on the idea that data points with similar features tend to belong to the same category or have similar values. The algorithm classifies a new data point by considering the labels of its k nearest neighbors in the training set.

A. How KNN Works:

K-Nearest Neighbors is also called as a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification it performs an action on the dataset.

As an example, consider the following table of data points containing two features:



The new point is classified as **Category 2** because most of its closest neighbours are blue squares. KNN assigns the category based on the majority of nearby points.
The image shows how KNN predicts the category of a **new data point** based on its closest neighbours.

- The red diamonds represent Category 1 and the blue squares represent Category 2.
- The new data point checks its closest neighbours (circled points).
- Since the majority of its closest neighbours are blue squares (Category 2) KNN predicts the new data point belongs to Category 2.

In the **k-Nearest Neighbours (k-NN)** algorithm **k** is just a number that tells the algorithm how many nearby points (neighbours) to look at when it makes a decision.

Example:

Imagine you're deciding which fruit it is based on its shape and size. You compare it to fruits you already know.

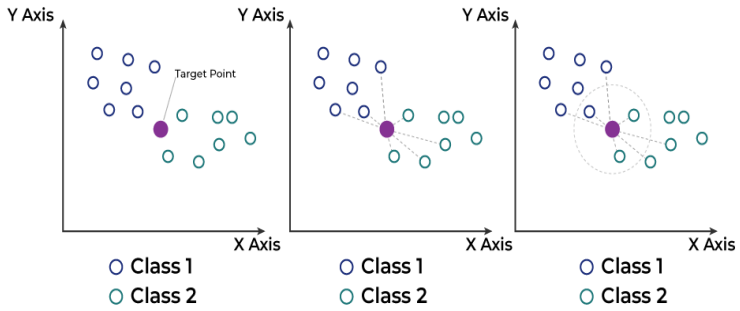
- If **k = 3**, the algorithm looks at the 3 closest fruits to the new one.
- If 2 of those 3 fruits are apples and 1 is a banana, the algorithm says the new fruit is an apple because most of its neighbours are apples.

B. Choosing the Optimal Value of k

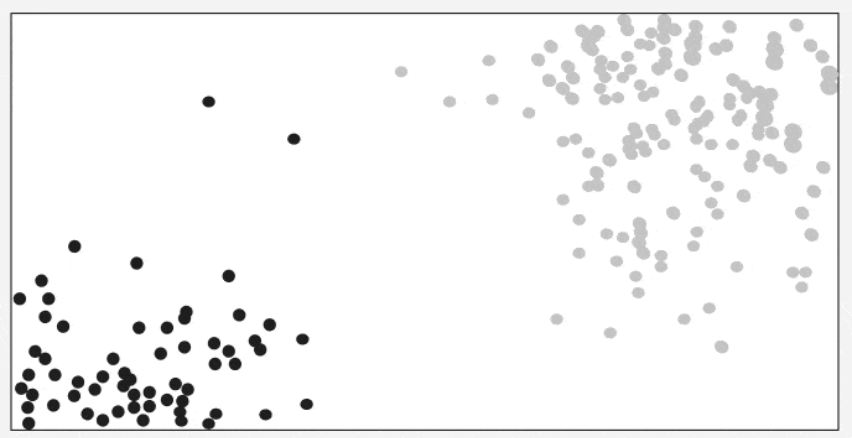
i) If k is too small, the model is too sensitive to noise (overfitting).
ii) If k is too large, the model may oversimplify patterns (underfitting).
iii) A common approach is to use cross-validation to select the best k.

C. How KNN algorithm Works:

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



i) Choose the value of k (number of neighbours).
ii) Measure the distance between the new data point and all training points (commonly using Euclidean distance).
iii) Select the k closest neighbours based on the distance metric.
iv) Classification: Assign the most common class label among the k neighbours.
v) Regression: Calculate the average (or weighted average) of the target values of the k neighbours.



D. Distance Metrics Used in KNN

i) Euclidean Distance (default)

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

ii) Manhattan Distance

$$d(p, q) = \sum_{i=1}^{n} |(q_i - P_i)|$$

iii) Minkowski Distance (generalized form)

$$d(P, q) = \left(\sum_{i=1}^{n} |q_i - P_i|^p \right)^{\frac{1}{p}}$$

iv) Hamming Distance (for categorical variables)

E. Applications of KNN

i) **Classification tasks** (e.g., spam detection, image recognition)
ii) **Regression tasks** (e.g., predicting house prices)
iii) **Recommendation systems** (e.g., suggesting movies based on user preferences)
iv) **Anomaly detection** (e.g., fraud detection in financial transactions)

F. Advantages & Disadvantages

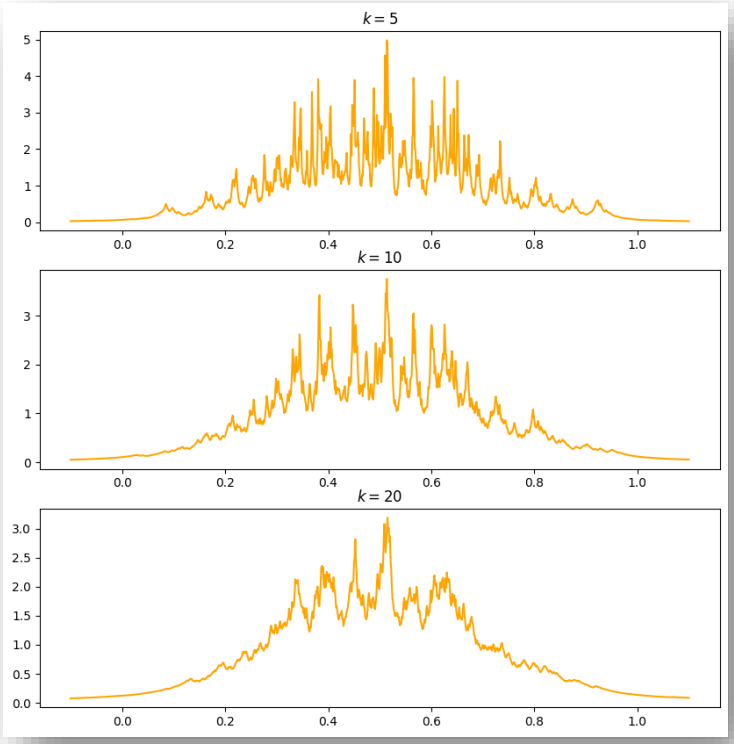
Advantages	Disadvantages
Simple and easy to implement	Computationally expensive for large datasets
No training phase; works on raw data	Sensitive to irrelevant or redundant features
Works well with multi-class classification	Requires choosing an optimal k value
Handles non-linearity in data	Storage-intensive, as it requires storing all training data

G. KNN Implementation in Python (Using Scikit-Learn)

i) Code

```
# KNN Density esitmator
gaussian = norm(loc=0.5, scale=0.2)
X = gaussian.rvs(500)
grid = np.linspace(-0.1, 1.1, 1000)
k_set = [5, 10, 20]
fig, axes = plt.subplots(3, 1, figsize=(10, 10))
for i, ax in enumerate(axes.flat):
    K = k_set[i]
    p = np.zeros_like(grid)
    n = X.shape[0]
    for i, x in enumerate(grid):
        dists = np.abs(X-x)
        neighbours = dists.argsort()
        neighbour_K = neighbours[K]
        p[i] = (K/n) * 1/(2 * dists[neighbour_K])
    ax.plot(grid, p, color='orange')
    ax.set_title(f'$k={K}$')
plt.show()
```

ii) Output



8. Support Vector Machine (SVM) Algorithm

A. Key Concepts of SVM

i) **Hyperplane**: SVM aims to find the optimal decision boundary (hyperplane) that best separates different classes in a dataset.

ii) **Support Vectors:** These are the data points that are closest to the hyperplane and influence its position and orientation.

iii) **Margin**: The distance between the hyperplane and the nearest support vectors. SVM maximizes this margin to improve classification performance.

iv) **Kernel Trick**: When data is not linearly separable, SVM uses kernel functions (e.g., polynomial, radial basis function (RBF)) to transform the data into a higher-dimensional space where it becomes separable.

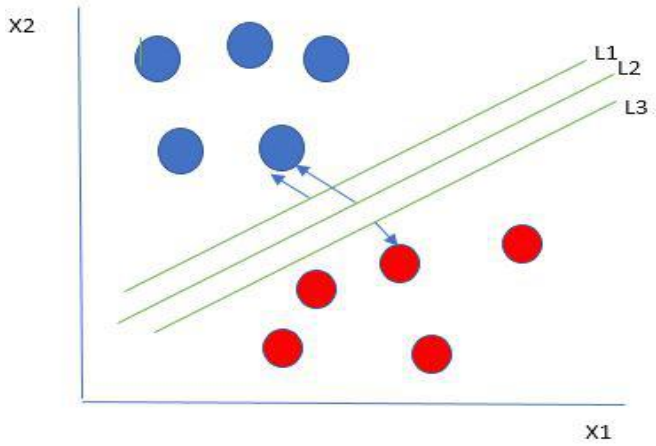
B. Types of SVM

i) **Linear SVM:** Used when the data is linearly separable.

ii) **Non-Linear SVM:** Uses kernel functions to handle non-linearly separable data.

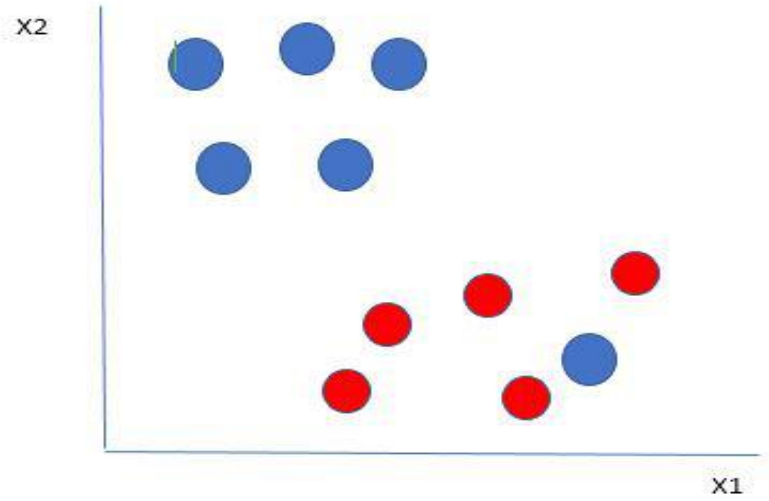
C. How does Support Vector Machine Algorithm Work?

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



The best hyperplane, also known as the **"hard margin,"** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So, from the above figure, we choose L2 as hard margin.

Let's consider a scenario like shown below:

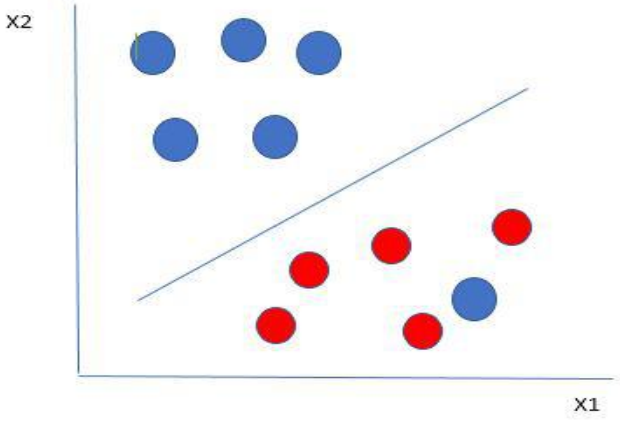


Here, we have one blue ball in the boundary of the red ball.

D. How does SVM classify the data?

It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

A soft margin allows for some misclassifications or violations of the margin to improve generalization. The SVM optimizes the following equation to balance margin maximization and penalty minimization:



Objective Function=(1/margin)+λ∑penalty

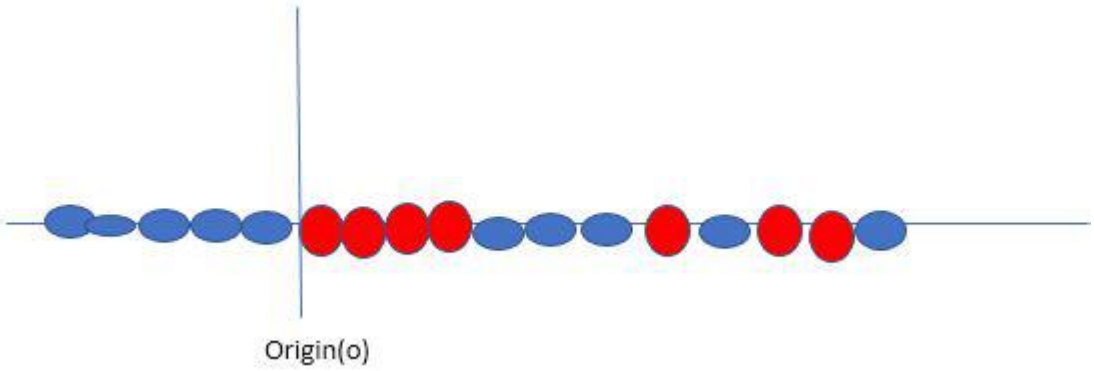
The penalty used for violations is often **hinge loss**, which has the following behaviour:

- If a data point is correctly classified and within the margin, there is no penalty (loss = 0).
- If a point is incorrectly classified or violates the margin, the hinge loss increases proportionally to the distance of the violation.

Till now, we were talking about linearly separable data (the group of blue balls and red balls are separable by a straight line/linear line).

E. What to do if data are not linearly separable?

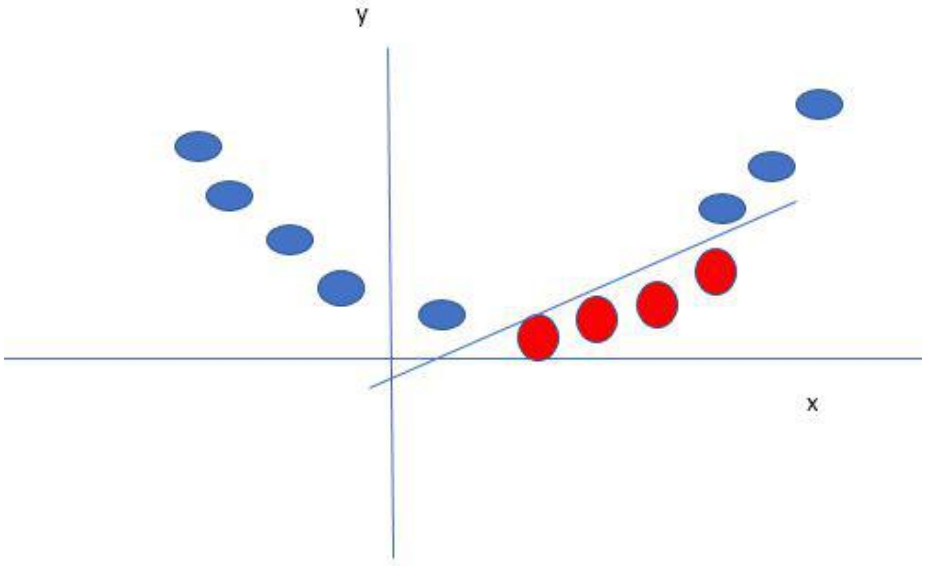
When data is not linearly separable (i.e., it can't be divided by a straight line), SVM uses a technique called kernels to map the data into a higher-dimensional space where it becomes separable. This transformation helps SVM find a decision boundary even for non-linear data.



A **kernel** is a function that maps data points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping.

For example, consider data points that are not linearly separable. By applying a kernel function, SVM transforms the data points into a higher-dimensional space where they become linearly separable.

- **Linear Kernel**: For linear separability.
- **Polynomial Kernel**: Maps data into a polynomial space.
- **Radial Basis Function (RBF) Kernel**: Transforms data into a space based on distances between data points.



In this case, the new variable y is created as a function of distance from the origin.

F. Advantages of SVM

i) Effective in high-dimensional spaces.
ii) Works well when there is a clear margin of separation.
iii) Uses kernels to handle complex decision boundaries.

G. Disadvantages of SVM

i) Computationally expensive for large datasets.
ii) Sensitive to the choice of kernel and hyperparameters.
iii) May not perform well with overlapping classes.

H. Implementing SVM Algorithm in Python

i) Algorithm

(a) Predict if cancer is Benign or malignant. Using historical data about patients diagnosed with cancer enables doctors to differentiate malignant cases and benign ones are given independent attributes.
(b) Load the breast cancer dataset from sklearn.datasets.
(c) Separate input features and target variables.

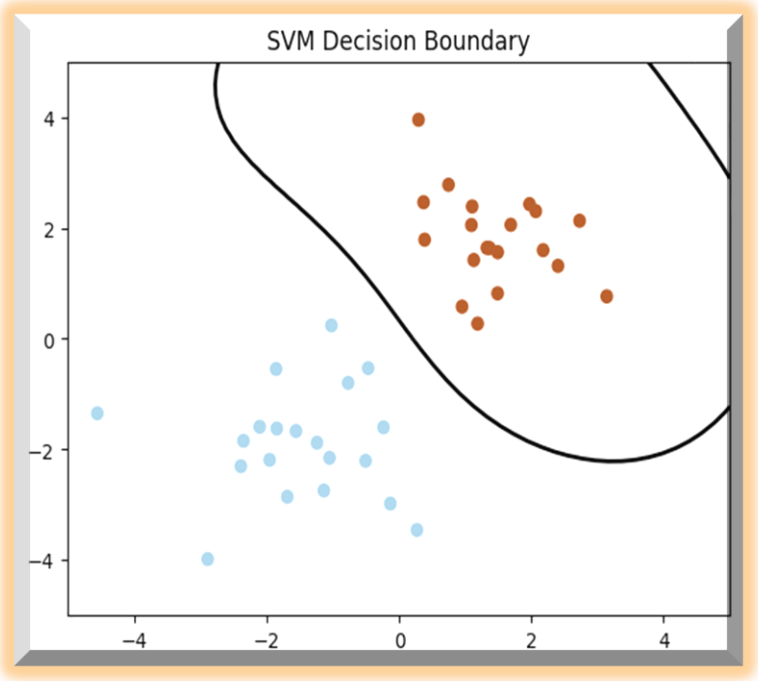
(d) Build and train the SVM classifiers using RBF kernel.

(e) Plot the scatter plot of the input features**.**

ii) Code

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

# Generate sample data
np.random.seed(0)
X = np.r_[np.random.randn(20, 2) - [2, 2], np.random.randn(20, 2) + [2, 2]]
Y = [0] * 20 + [1] * 20

# Train SVM model with RBF kernel
clf = svm.SVC(kernel='rbf', C=1.0, gamma='scale')
clf.fit(X, Y)

# Plot decision boundary
xx, yy = np.meshgrid(np.linspace(-5, 5, 50), np.linspace(-5, 5, 50))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)
plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='k')
plt.title("SVM Decision Boundary")
plt.show()
```

iii) Output



9.Conclusion

Non-parametric methods play a crucial role in both statistics and machine learning by providing flexible and data-driven approaches to analysis. They are especially useful when parametric assumptions do not hold. While they offer significant advantages in terms of adaptability, they often require more computational resources and larger datasets for optimal performance. As data science continues to evolve, non-parametric techniques remain vital tools for analyzing real-world data without restrictive assumptions.

10. References

- **"All of Nonparametric Statistics"** – Larry Wasserman, Springer (2006).
- **"The Elements of Statistical Learning: Data Mining, Inference, and Prediction"** – Trevor Hastie, Robert Tibshirani, and Jerome Friedman, Springer (2009).
- **"Density Estimation for Statistics and Data Analysis"** – Bernard W. Silverman, Chapman & Hall/CRC (1986).
- **"Pattern Recognition and Machine Learning"** – Christopher M. Bishop, Springer (2006).
- **"Nonparametric Statistical Inference"** – Jean Dickinson Gibbons and Subhabrata Chakraborti, Chapman & Hall/CRC (2011).