L01: ML libraries and platforms

Machine Learning libraries

NumPy:

- NumPy stands for Numerical Python, and SciPy stands for Scientific Python; both are essential Python libraries.
- NumPy is used for arrays that store values of the same datatype.
- In NumPy, dimensions are called axes and the number of axes is called rank. The array object in NumPy is called *ndarray*.
- Basic array operations: add, multiply, slice, flatten, reshape, index arrays & Advanced array operations: stack arrays, split into sections, broadcast arrays.

Pandas:

- Pandas stand for Python Data Analysis Library. Who ever knew that?
- Pandas take data in a CSV or TSV file or a SQL database and create a Python object with rows and columns called a data frame. The data frame is very similar to a table in statistical software, say Excel or SPSS.
- Indexing, manipulating, renaming, sorting, merging data frame
- Update, Add, Delete columns from a data frame
- Impute missing files, handle missing data or NANs
- Plot data with histogram or box plot

Matplotlib:

- Data visualization.
- Matplotlib enables the creation of a wide range of visualizations, from simple plots to complex charts. Visualizations are essential for understanding data patterns and trends.

Scikit-learn:

- General-purpose machine learning library.
- Scikit-learn provides a consistent interface for various machine learning algorithms, making it easy to experiment with models for classification, regression, clustering, and more. Its tools for model selection, hyperparameter tuning, and evaluation simplify the machine learning workflow.

Deep Learning libraries

TensorFlow:

- Deep learning framework.
- TensorFlow is widely adopted for building and training neural networks. Its flexibility and scalability make it suitable for a range of applications, from small

projects to large-scale production systems. TensorFlow also supports deployment on various platforms, including mobile devices.

Keras:

- High-level neural networks API.
- Keras acts as a user-friendly interface for building neural networks, making it accessible to both beginners and experts. Its seamless integration with TensorFlow allows users to take advantage of TensorFlow's capabilities while working with a simplified, intuitive syntax.

PyTorch:

- Deep learning framework with dynamic computation graph.
- PyTorch's dynamic computation graph makes it particularly suitable for research and experimentation. It facilitates easier debugging and model customization. PyTorch is embraced by the research community and is known for its strong support for dynamic architectures.

OpenCV:

- Computer vision library (often used in deep learning applications).
- OpenCV provides a wealth of tools for image and video processing. In deep learning projects, it complements neural networks by offering functionalities like image preprocessing, feature extraction, and augmentation, enhancing the quality of input data for models.

L02: Scikit-learn Documentation

Scikit-learn is the heart or core of the machine learning model.

1. Import Libraries: import pandas as pd, import numpy as np, **from sklearn.model_selection import train_test_split.**
2. Data Loading: Load the dataset using **pd.read_csv()** or similar methods.
3. Data Exploration: Understand the dataset using methods like **head(), describe()**, and visualization tools.
4. Data Preprocessing: Handle missing values, scale features, encode categorical variables.
5. Train-Test Split: Divide the dataset into training and testing sets.

Resources:

1. https://scikit-learn.org/stable/
2. Official source code repo: https://github.com/scikit-learn/scikit-learn
3. https://www.tutorialspoint.com/scikit_learn/scikit_learn_support_vector_machines.htm

L3+L4+L5+L6: Data Pre-processing

Google Colab Link:
https://colab.research.google.com/drive/1EF9YqEPaI86FysSIUE0kcsdAE83HY0QK?usp=sharing

In this notebook we will learn about:
1. Handling missing values
2.Imbalance dataset issue
3.Outliers
4.converting / Encoding data (categorical to numerical)
5.Duplicate data
6.Splitting dataset
7.Scaling data
8.Feature Engineering

L07: Linear Regression code

Google Colab Link:
https://colab.research.google.com/drive/1NPTAcwnkspYFBrIpJ0Pee7Z30BJFK_78?usp=sharing

L14+L15: Support Vector Machine (SVM) Theory Part (1+2)

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. By SVMs, we can do a variety of tasks, like text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection.
- Support vectors are the data points that lie closest to the decision surface (or hyperplane). SVMs maximize the margin.
- They are the data points most difficult to classify.
- They have direct bearing on the optimum location of the decision surface.

Hyperplane

In SVM, a hyperplane is a decision boundary that separates the data into different classes. The goal of SVM is to find the best hyperplane that maximally separates the different classes. In a two-dimensional space, a hyperplane is a straight line, and in a three-dimensional space, it is a plane, and so on. A good rule of thumb is that for an n-dimensional space, the hyperplane will generally have an n-1 dimension.

Margin

The margin is the distance between the hyperplane and the closest data points from each class. The goal of SVM is to find the hyperplane that maximizes the margin, which helps to ensure that the decision boundary is as far away as possible from the data points. The larger the margin, the better the separation of the classes.

Support Vectors:

Support vectors are the data points that are closest to the decision boundary (or hyperplane). These points have a special role in determining the location of the decision boundary because the goal of SVM is to find the hyperplane that maximally separates the different classes by maximizing the margin, which is the distance between the hyperplane and the closest data points from each class.

Soft Margin

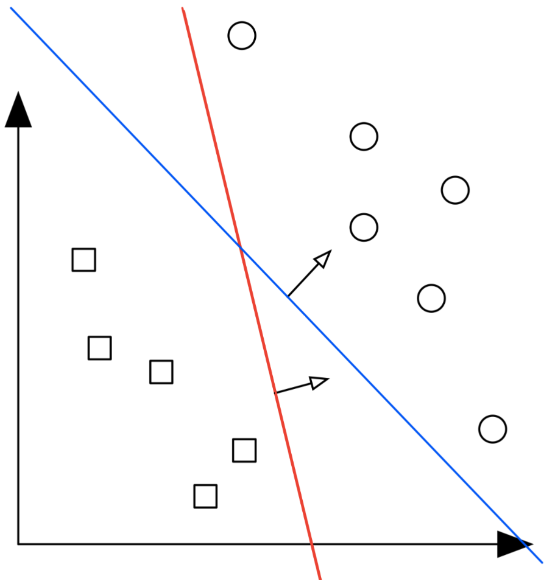
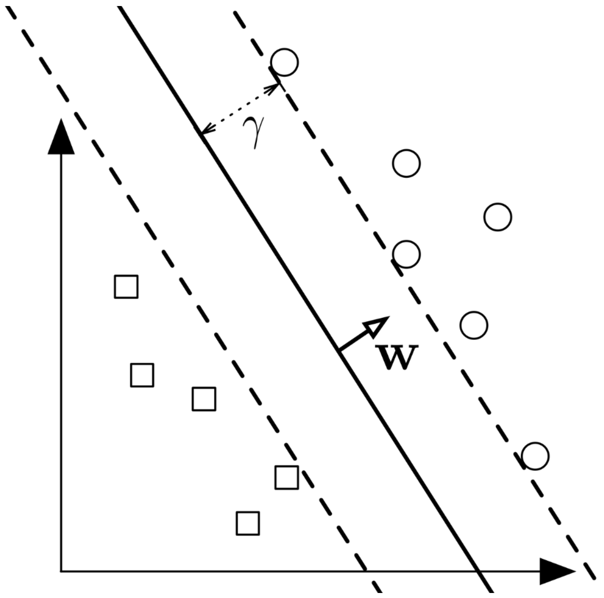
In a hard-margin SVM, the goal is to find a decision boundary that separates the two classes with the largest possible margin. However, in some cases, the data may not be perfectly separable and there may be some overlap between the two classes. In these cases, a hard-margin SVM may not be able to find a decision boundary that separates the two classes perfectly. A soft-margin SVM, on the other hand, refers to the ability of SVM to tolerate misclassifications by allowing some data points to be on the wrong side of the decision boundary. This is achieved by introducing a regularization parameter.

Regularization parameter (C)

The regularization parameter is a tunable parameter that controls the trade-off between maximizing the margin and minimizing the number of misclassifications. A smaller value of C will result in a larger margin but more misclassification, while a larger value of C will result in a smaller margin but less misclassification.
By default, the value of C is set to **1**, which corresponds to a hard-margin SVM. To use a soft-margin SVM, you can set a smaller value of C, such as **0.1**.

Note:

- Margin of Separation (d): the separation between the hyperplane and the closest data point for a given weight vector w and bias b.
- Optimal Hyperplane (maximal margin): the particular hyperplane for which the margin of separation d is maximized.

Types of SVM:

There are two main types of SVM:

1. linear SVM.
2. non-linear SVM (Kernel).

1. Linear SVM

The Linear Support Vector Machine algorithm is used when we have linearly separable data. In simple language, if we have a dataset that can be classified into two groups using a simple straight line, we call it linearly separable data, and the classifier used for this is known as Linear SVM Classifier.

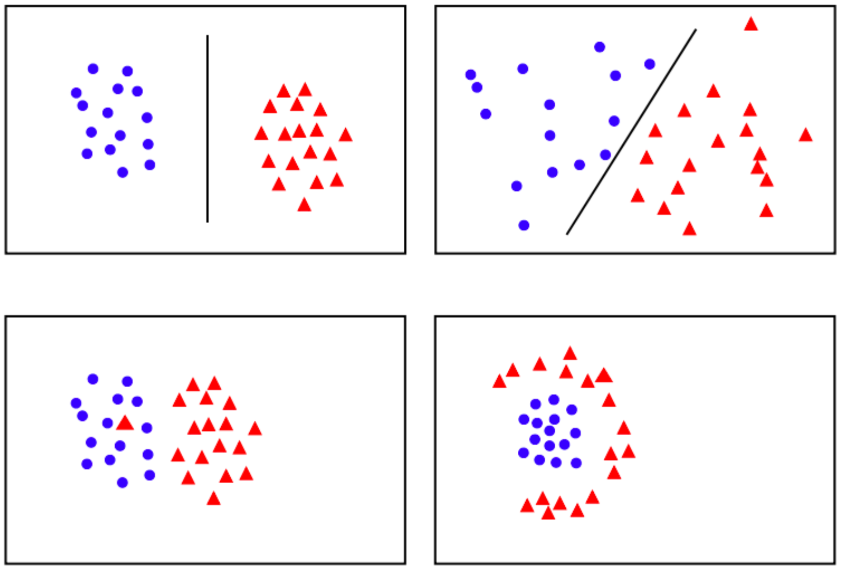
2. Kernel or Non-Linear SVM

The non-linear support vector machine algorithm is used when we have non-linearly separable data. In simple language, if we have a dataset that cannot be classified into two groups using a simple straight line, we call it non-linear separable data, and the classifier used for this is known as a Non-Linear SVM classifier.

Linear separability

linearly separable

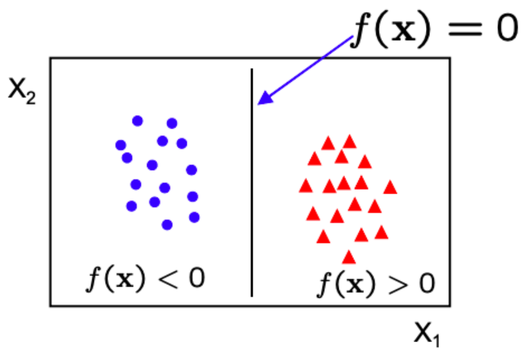
not linearly separable



Linear classifiers

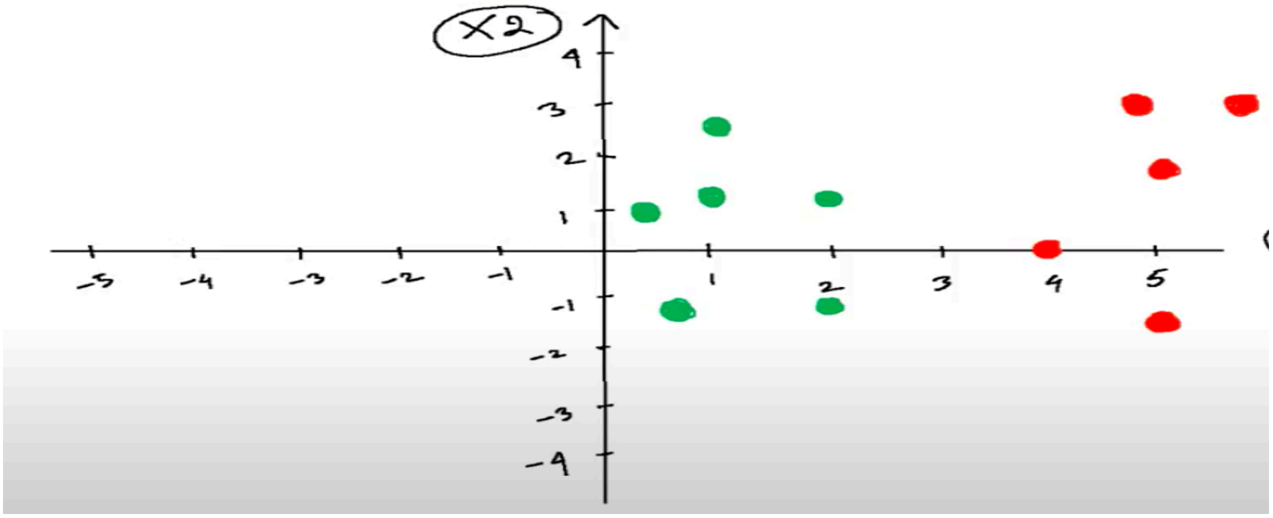
A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear Support Vector Machine (LSVM):



support vector: $\boxed{3}$

$\boxed{x1}$

$s_1 = \begin{pmatrix}2\\1\end{pmatrix}$ $\qquad \tilde{s}_1 = \begin{pmatrix}2\\1\\1\end{pmatrix}$

$s_2 = \begin{pmatrix}2\\-1\end{pmatrix}$ $\qquad \tilde{s}_2 = \begin{pmatrix}2\\-1\\1\end{pmatrix}$

$s_3 = \begin{pmatrix}4\\0\end{pmatrix}$ $\qquad \tilde{s}_3 = \begin{pmatrix}4\\0\\1\end{pmatrix}$

for 3 support vectors,
we will have 3 parameters

$\alpha_1 \quad \alpha_2 \quad \alpha_3$

$$\alpha = \begin{pmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix}$$

$$\underbrace{\underset{(3\times 1)}{\tilde{S}}\ \underset{(1\times 3)}{\tilde{S}^T}}_{3\times 3}\alpha$$

$$\tilde{S} = \begin{pmatrix}\tilde{s}_1\\ \tilde{s}_2\\ \tilde{s}_3\end{pmatrix} \quad (3\times 1)$$

$$\tilde{S}^T = \begin{pmatrix}\tilde{s}_1 & \tilde{s}_2 & \tilde{s}_3\end{pmatrix} \quad (1\times 3)$$

$$\tilde{S}\tilde{S}^T = \begin{bmatrix}\tilde{s}_1\tilde{s}_1 & \tilde{s}_1\tilde{s}_2 & \tilde{s}_1\tilde{s}_3\\ \tilde{s}_2\tilde{s}_1 & \tilde{s}_2\tilde{s}_2 & \tilde{s}_2\tilde{s}_3\\ \tilde{s}_3\tilde{s}_1 & \tilde{s}_3\tilde{s}_2 & \tilde{s}_3\tilde{s}_3\end{bmatrix}^T \quad (3\times 3)$$

$$\alpha_1 \tilde{s}_1\tilde{s}_1 + \alpha_2 \tilde{s}_1\tilde{s}_2 + \alpha_3 \tilde{s}_1\tilde{s}_3 = -1$$

$$\alpha_1 \tilde{s}_2\tilde{s}_1 + \alpha_2 \tilde{s}_2\tilde{s}_2 + \alpha_3 \tilde{s}_2\tilde{s}_3 = -1$$

$$\alpha_1 \tilde{s}_3\tilde{s}_1 + \alpha_2 \tilde{s}_3\tilde{s}_2 + \alpha_3 \tilde{s}_3\tilde{s}_3 = +1$$

$$\alpha_1 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}2\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}2\\-1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = -1$$

$$\tilde{s}_2\tilde{s}_1 + \alpha_2 \tilde{s}_2\tilde{s}_2 + \alpha_3 \tilde{s}_2\tilde{s}_3 = -1$$

$$\tilde{s}_3\tilde{s}_1 + \alpha_2 \tilde{s}_3\tilde{s}_2 + \alpha_3 \tilde{s}_3\tilde{s}_3 = +1$$

$$\alpha_1 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}2\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}2\\-1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix}2\\-1\\1\end{pmatrix}\begin{pmatrix}2\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\-1\\1\end{pmatrix}\begin{pmatrix}2\\-1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}2\\-1\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix}4\\0\\1\end{pmatrix}\begin{pmatrix}2\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}4\\0\\1\end{pmatrix}\begin{pmatrix}2\\-1\\1\end{pmatrix} + \alpha_3 \begin{pmatrix}4\\0\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = 1$$

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$
$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$
$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = 1$$

-1

1

$$\begin{pmatrix}2\\1\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = -1$$

$$\begin{pmatrix}2\\-1\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = -1$$

$$\begin{pmatrix}4\\0\\1\end{pmatrix}\begin{pmatrix}4\\0\\1\end{pmatrix} = 1$$

$$\boxed{\begin{aligned}6\alpha_1 + 4\alpha_2 + 9\alpha_3 &= -1\\ 4\alpha_1 + 6\alpha_2 + 9\alpha_3 &= -1\\ 9\alpha_1 + 9\alpha_2 + 17\alpha_3 &= 1\end{aligned}}$$

Var < eq
param

$\alpha_1 = -3.25$

$\alpha_2 = -3.25$

$\alpha_3 = 3.5$

$$\tilde{w} = \alpha_1 \tilde{s}_1 + \alpha_2 \tilde{s}_2 + \alpha_3 \tilde{s}_3$$

$$= -3.25 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + -3.25 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + 3.5 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} -6.50 \\ -3.25 \\ -3.25 \end{pmatrix} + \begin{pmatrix} -6.50 \\ 3.25 \\ -3.25 \end{pmatrix} + \begin{pmatrix} 7.0 \\ 0 \\ 3.5 \end{pmatrix}$$

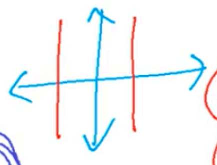
$$\tilde{w} = \begin{pmatrix} 1 \\ 0 \\ \underbrace{\boxed{-3}}_{\text{bias}} \end{pmatrix} \qquad w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$b = -3$$

$$y = wx + b$$

$$y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} x - 3$$

$$\tilde{w} = \begin{pmatrix} 1 \\ 0 \\ \underbrace{\boxed{-3}}_{\text{bias}} \end{pmatrix} \qquad w = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



$\boxed{x = a}$

$\boxed{x = -a}$

$$b = -3 \quad \boxed{x = +3}$$

$$w = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$y = b$$

$$y = wx + b$$

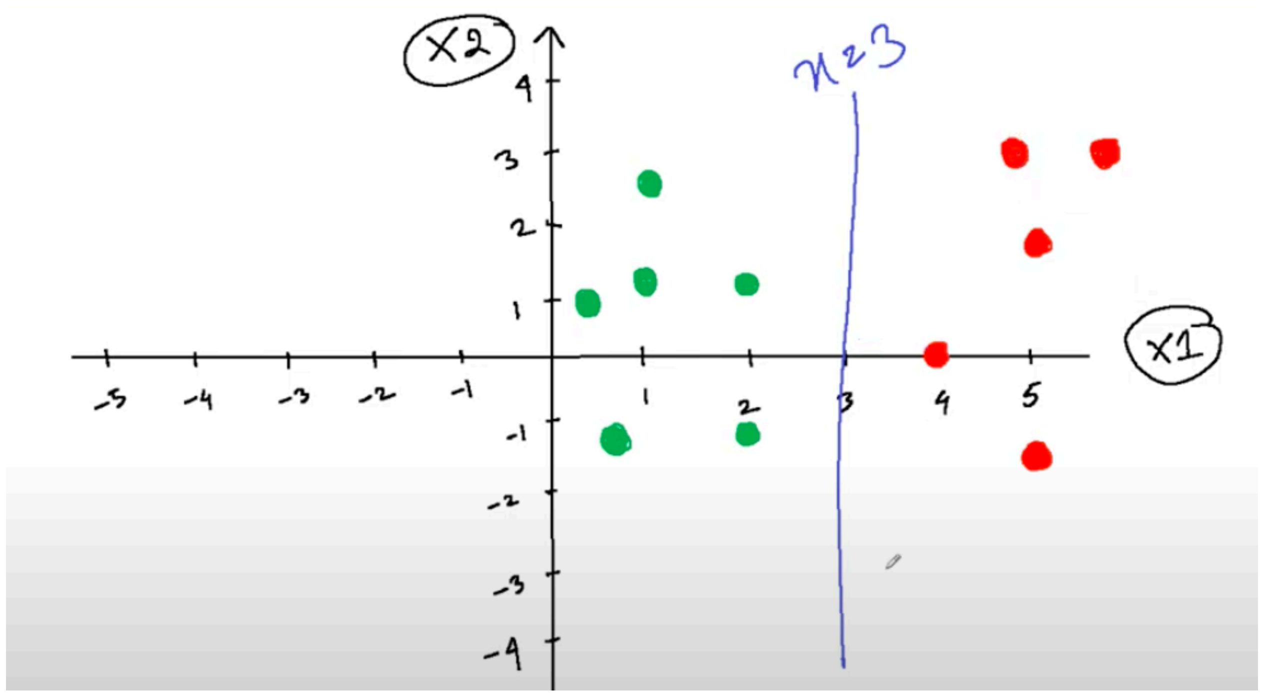
$$y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} x - 3$$

Decision boundary $\rightarrow$ linear SVM

$$y = mx + c \longrightarrow \text{math}$$

$$y = Wx + b \rightarrow \text{ML}$$

$$\tilde{W} = \sum \alpha_i S_i$$



Kernel trick

The kernel trick is used in support vector machines (SVMs), which are a type of machine learning algorithm that can be used for classification and regression tasks. SVMs work by finding a hyperplane that separates the data points into two classes. However, if the data is not linearly separable, the kernel trick can be used to map the data to a higher dimensional space where it becomes linearly separable.

There are many different kernel functions that can be used, each with its own strengths and weaknesses. Some of the most common kernel functions include:

- The linear kernel: This is the simplest kernel function, and it simply computes the dot product of two data points.
- The polynomial kernel: This kernel function is more powerful than the linear kernel, and it can be used to model non-linear relationships between the data points.
- The Gaussian kernel: This kernel function is even more powerful than the polynomial kernel, and it is often used for image classification tasks.

L16: SVM classifier code

Google Colab Link:
https://colab.research.google.com/drive/1Kvm_faeSyZOfKop6WuSrQ_BTpChS2xrw?usp=sharing