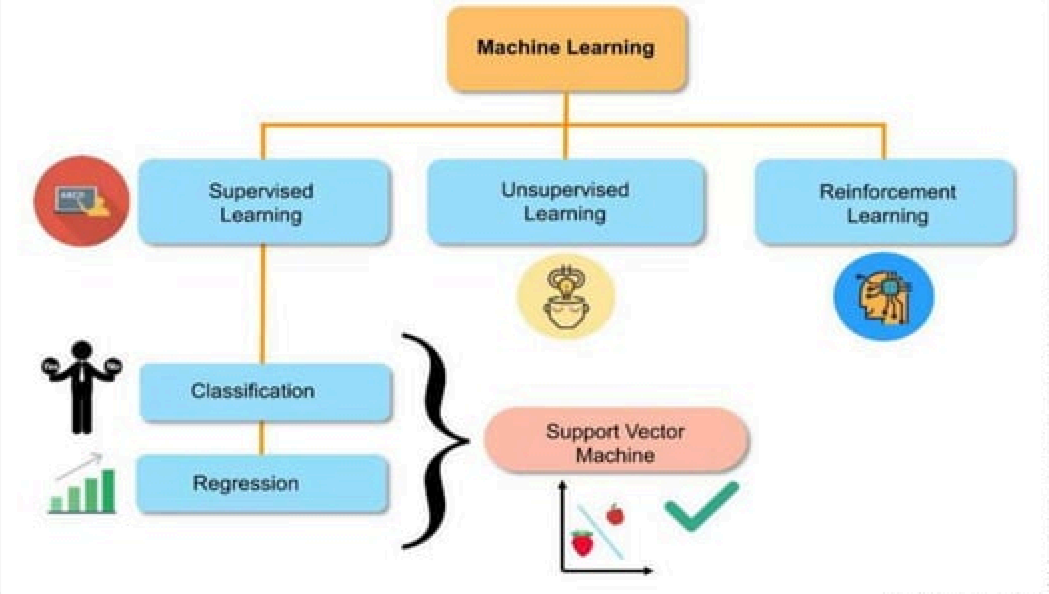
Support Vector Machine (SVM)

Machine Learning



What is SVM ?

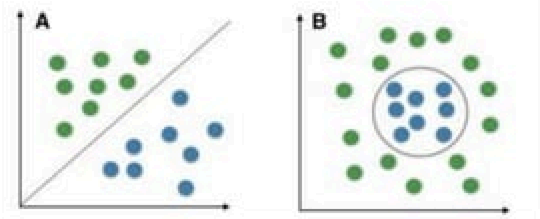
Introduction:

- Support Vector Machine is a **supervised machine learning problem** where to try to find a hyperplane that best separates two classes.
- It is considered as one of the best algorithm as it can handle high-dimensional data, is effective in cases with limited training samples, and can handle nonlinear classification using kernel functions.
- SVM can be used for both regression and classification tasks, but generally work best in classification problem.

Types of SVM Algorithm:

1. **Simple or Linear SVM:** When the data is perfectly linearly separable only then we can use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).

1. **Kernel or Non-Linear SVM:** When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a straight line (if 2D) then we use some advanced techniques like kernel tricks to classify them. In most real-world applications we do not find linearly separable data points hence we use kernel trick to solve them.

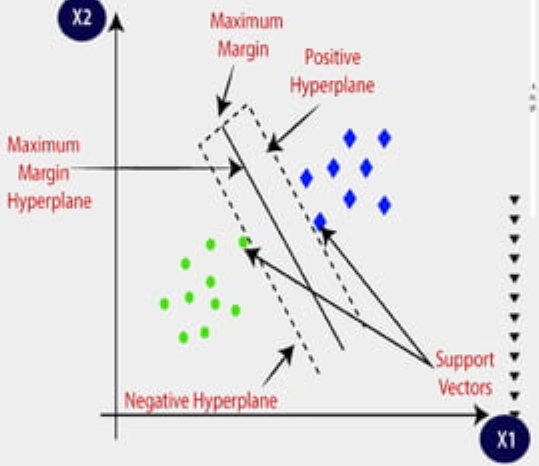


Important Terms:

1. **Support Vectors:** These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.

1. **Margin:** it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). There are two types of margins hard margin and soft margin.

1. **Hyperplane:** The best-line or decision boundary that can segregate N-dimensional space into classes so that we can easily put the new data point in the correct category.

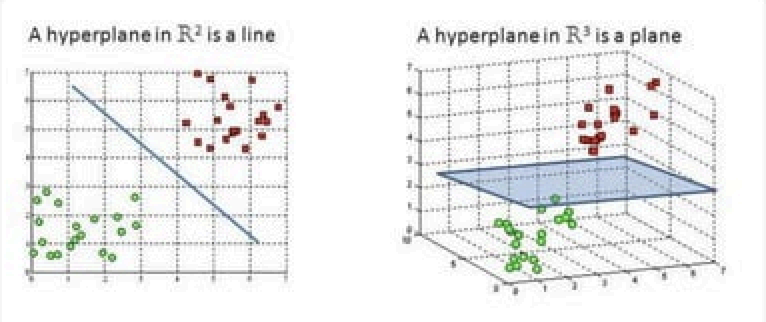


Hyper Plane:

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.

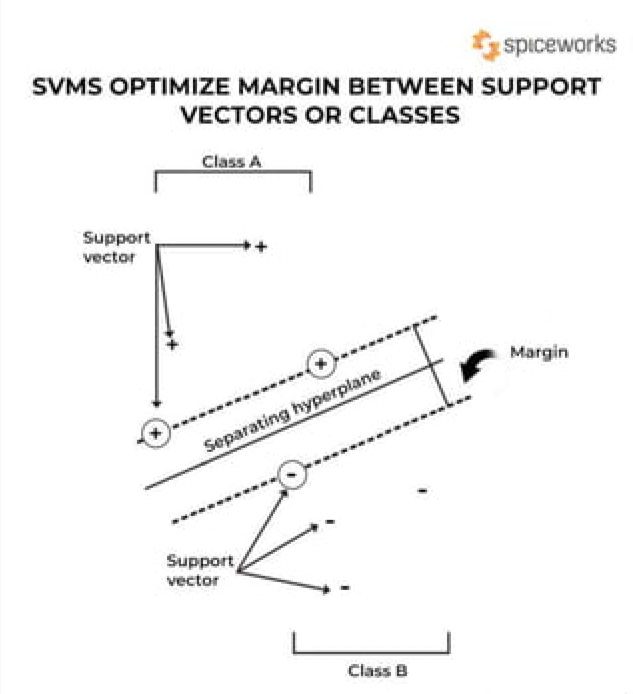
dimension of the hyperplane = the number of features

If, number of input features is 2, hyperplane is just a line.
number of input features is 3, hyperplane becomes a two-dimensional plane.



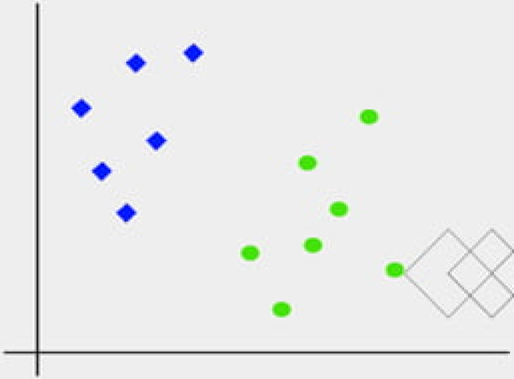
Objective:

Objective of the SVM algorithm is to find a hyperplane that, to the best degree possible, separates data points of one class from those of another class.



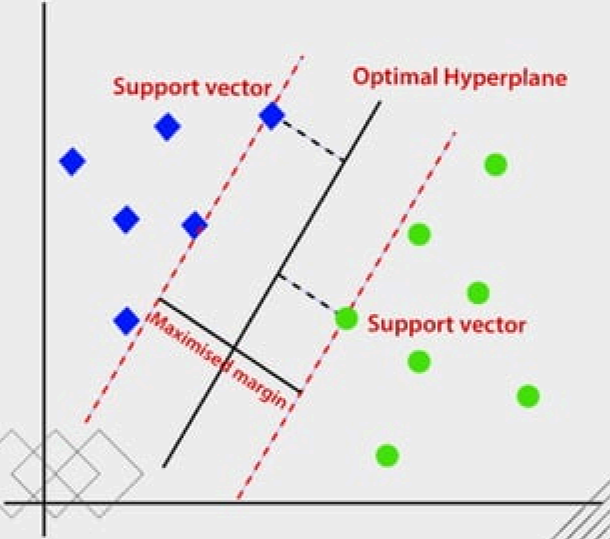
How does SVM work ?

Suppose we have a dataset that has two classes (green and blue). We want to classify that the new data point as either blue or green.



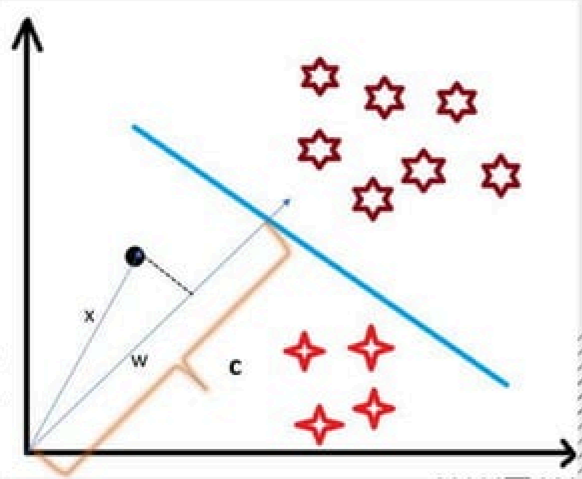
How does SVM work ?

Best Hyperplane = Finding different hyperplanes which classify the labels in the best way then it will choose the one which is farthest from the data points or the one which has a maximum margin.



Hard Margin SVM (Dot Product):

- Consider a random point X and we want to know whether it lies on the right side of the plane or the left side of the plane (positive or negative).
- First we assume this point is a vector (X) and then we make a vector (w) which is perpendicular to the hyperplane.



Decision Rule:

$\vec{X}.\vec{w} = c$ *(the point lies on the decision boundary)*

$\vec{X}.\vec{w} > c$ *(positive samples)*

$\vec{X}.\vec{w} < c$ *(negative samples)*

$\vec{X}.\vec{w} - \mathrm{c} \geq 0$

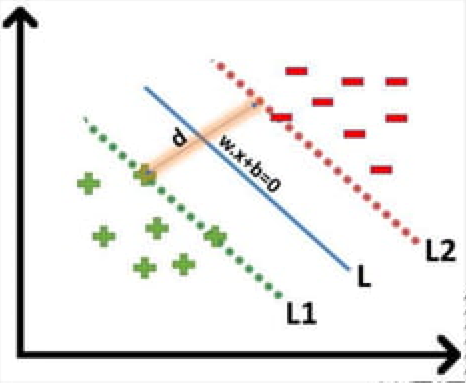
putting $-\mathrm{c}$ as b, we get

$\vec{X}.\vec{w} + \mathrm{b} \geq 0$

hence

$$\mathrm{y} = \begin{cases} +1 & \text{if } \vec{X}.\vec{w} + \mathrm{b} \geq 0 \\ -1 & \text{if } \vec{X}.\vec{w} + \mathrm{b} < 0 \end{cases}$$

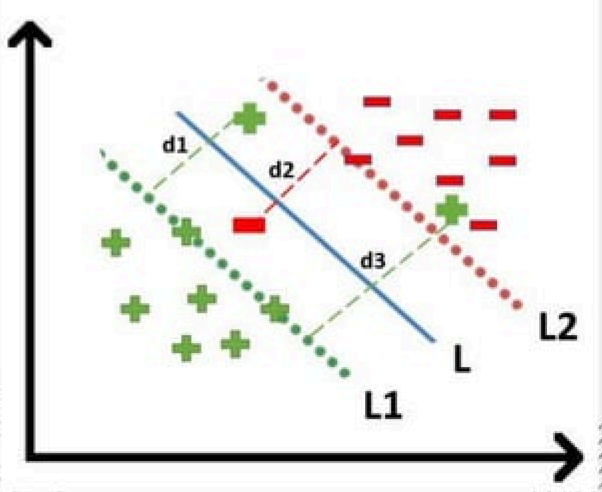
We all know the equation of a hyperplane is w.x+b=0 where w is a vector normal to hyperplane and b is an offset.



Soft margin SVM:

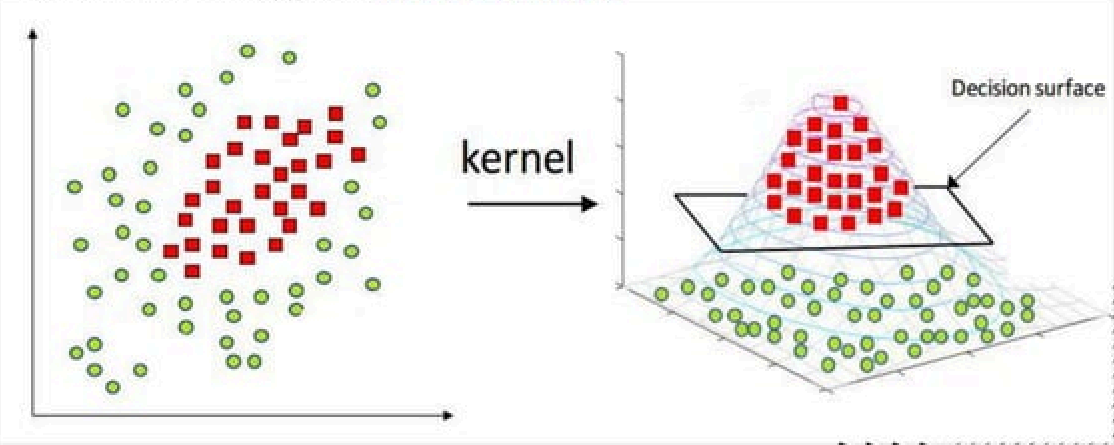
- In real-life applications, we rarely encounter datasets that are perfectly linearly separable. Instead, we often come across datasets that are either nearly linearly separable or entirely non-linearly separable.

- To tackle this problem what we do is modify that equation in such a way that it allows few misclassifications that means it allows few points to be wrongly classified.



Kernel in SVM:

- Kernels = functions that helps to convert lower dimension space to a higher dimension space using some quadratic functions.



Types of Kernel Functions:

- Polynomial Kernel = Separate data of Degree 2.
- RBF Kernel = (Most used) Lift our samples onto a higher-dimensional feature
- Bessel function Kernel: Used for eliminating the cross term in mathematical functions
- Anova Kernel: Multidimensional regression problems

Right Kernel = What type of Dataset we are working on?

Linear Dataset = Linear Kernel Function

Complex Dataset = RBF Kernel Function

Import Necessary Libraries

Necessary imports

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

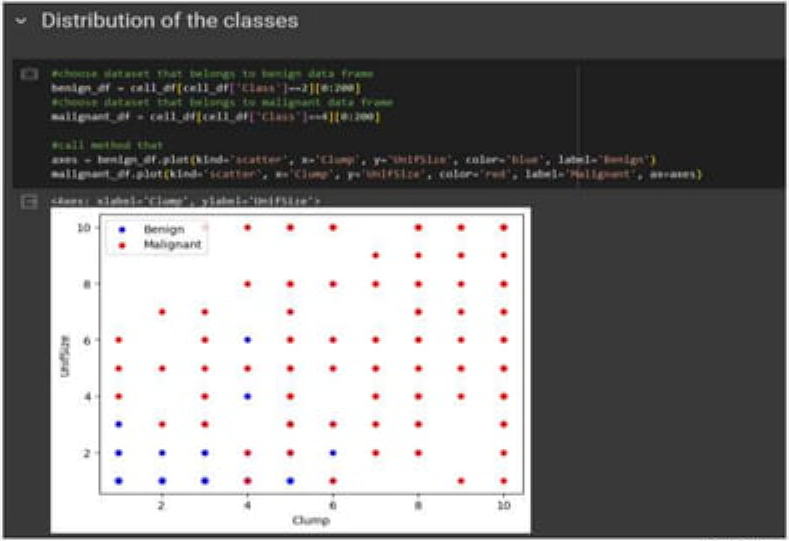
Load Data from CSV file

Load Data from CSV file

```
[ ] cell_df = pd.read_csv('cell_samples.csv')
    cell_df.tail()
    cell_df.shape
    cell_df.size
    cell_df.count()
    cell_df['Class'].value_counts()
```

```
2    458
4    241
Name: Class, dtype: int64
```

Distribution of the Classes



Identification of the unwanted rows

Identifying the unwanted rows

```
cell_df.dtypes
#remove the non numerical datas / discard non numeric data
#convert the values to numeric
cell_df = cell_df[pd.to_numeric(cell_df['BareNuc'], errors='coerce').notnull()]
cell_df['BareNuc'] = cell_df['BareNuc'].astype('int')
cell_df.dtypes
```

```
ID            int64
Clump         int64
UnifSize      int64
UnifShape     int64
MargAdh       int64
SingEpiSize   int64
BareNuc       int64
BlandChrom    int64
NormNucl      int64
Mit           int64
Class         int64
dtype: object
```

Remove unwanted Columns

Remove of unwanted columns

```
[ ] cell_df.columns

    feature_df = cell_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize',
           'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]
    #convert to numpy n-D array

    #Independent variable X
    X = np.asarray(feature_df)

    #Dependent variable -> y
    y = np.asarray(cell_df['Class'])

    y[0:5]
```

```
array([2, 2, 2, 2, 2])
```

Divide data as Train/Test

Divide the data as Train/Test dataset

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
    #546 X 9
    X_train.shape
    #546 X 1
    y_train.shape
    #137 X 9
    X_test.shape
    #137 X 1
    y_test.shape
```

```
(137,)
```

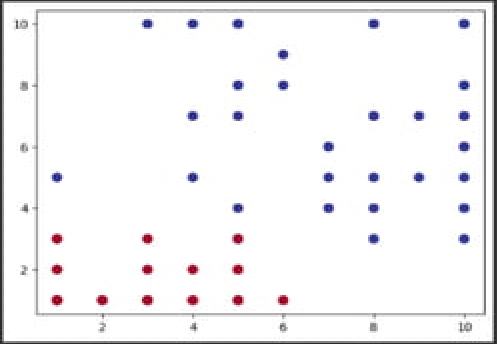
Modeling

Modeling (SVM with Scikit-learn)

```
[ ]  from sklearn import svm
     #Support Vector Classifier
     classifier = svm.SVC(kernel='linear', gamma='auto', C=2)

     classifier.fit(X_train, y_train)

     y_predict = classifier.predict(X_test)
```

Model Output :

```
# Plot the results
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, s=50, cmap='RdYlBu')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_predict, s=50, cmap='RdYlBu')

plt.show()
```



Evaluation of Results

Evaluation (Results)

```
[ ] from sklearn.metrics import classification_report
    print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           2       1.00      0.94      0.97        90
           4       0.90      1.00      0.95        47

    accuracy                           0.96       137
   macro avg       0.95      0.97      0.96       137
weighted avg       0.97      0.96      0.96       137
```

Metrics:

- **Accuracy:** The portion of correctly classified instances among the total instances.
- **Precision and recall:** Precision measures the portion of true positive predictions among all positive predictions while recall measures the portion of true positive predictions among all actual positives.
- **F1 Score:** The harmonic mean of precision and recall, which provides a balance between the two metrics.
- **Support:** Support refers to the number of occurrences of each class in the actual data. It represents the number of samples belonging to each class.

Advantages of SVM

- SVM works better when the data is linear
- It is more effective in high dimensions
- With the help of the kernel trick, we can solve any complex problem
- SVM is not sensitive to outliers
- Can help us with Image Classification

Disadvantage of SVM

- Choosing a good kernel is not easy
- It does not show good result on a big dataset
- The SVM hyperparameters are Cost -C and gamma.
- It is **not easy** to fine-tune these hyper-parameters.
- It is hard to visualize their impact

Conclusion:

- We have **demonstrated** SVM by segregation two different classes.
- We discussed its concept of working, math intuition behind SVM, implementation in python, the tricks to classify non-linear datasets, Pros and cons, and finally, we solved a problem with the help of SVM.

Support vector machine is an elegant and powerful algorithm.

THANK YOU!

Any Questions?