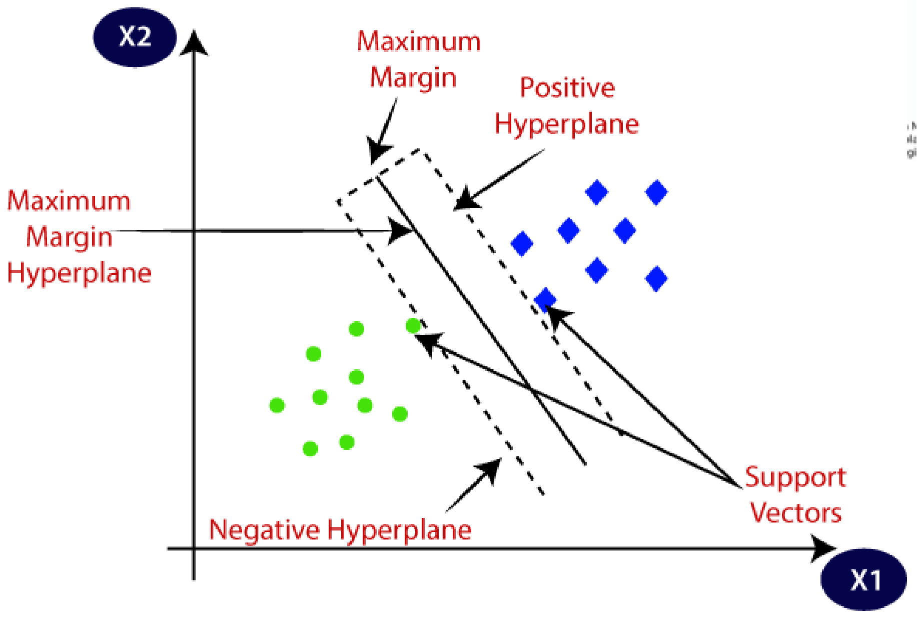
Support Vector Machine (SVM)

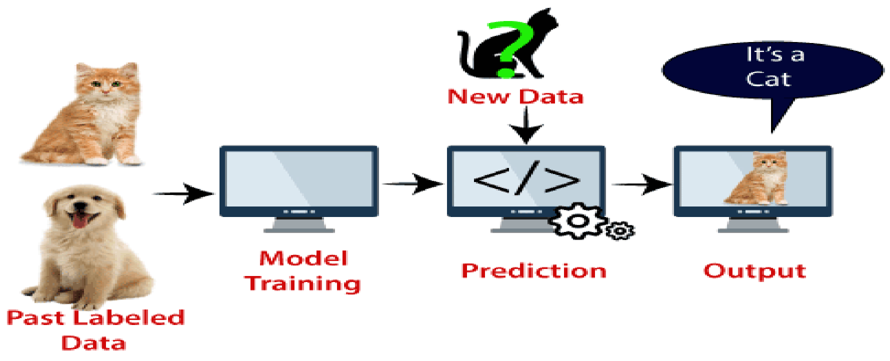
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create 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 in the future. This best decision boundary is called a hyperplane.

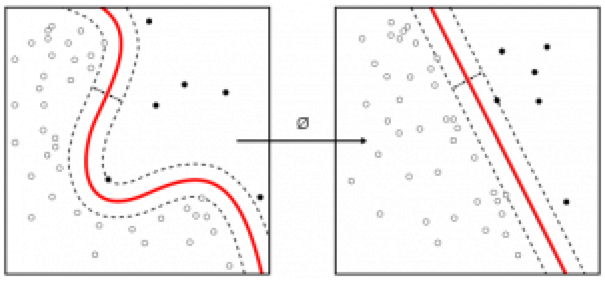
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

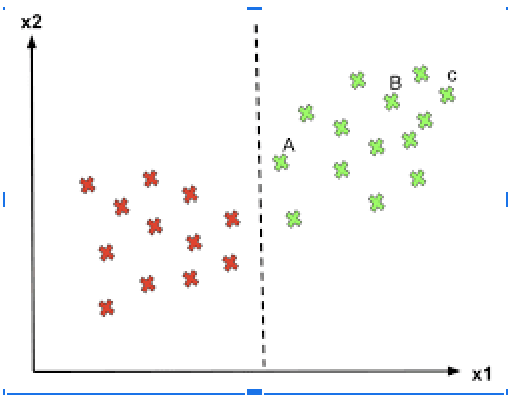


- There is also a subset of SVM called SVR which stands for Support Vector Regression which uses the same principles to solve regression problems.
- SVM is most commonly used and effective because of the use of the Kernel method which basically helps in solving the non-linearity of the equation in a very easy manner.



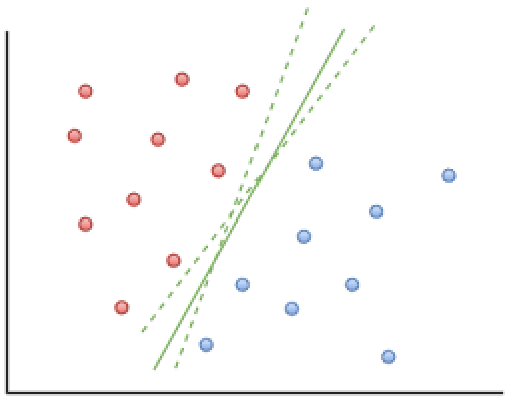
- The equation of the main separator line is called a hyperplane equation.
- A classification problem can have only two (binary) classes for separating or can have more than two too which are known as a multi-class classification problems.
- But not all classification predictive models support multi-class classification, algorithms such as the Logistic Regression and Support Vector Machines (SVM) were designed for binary classification and do not natively support classification tasks with more than two classes.
- But if someone stills want to use the binary classification algorithms for multi-classification problems, one approach which is widely used is to split the multi-class classification datasets into multiple binary classification datasets and then fit a binary classification model on each.
- As already mentioned above, SVM works much better for binary class
- It would be easy to understand the math since our target variable (variable / unseen data targeted to predict, whether the point is a male or a female)
- Note: This will be a One Vs One approach.

Case 1: (Perfect Separation for Binary Classified data)



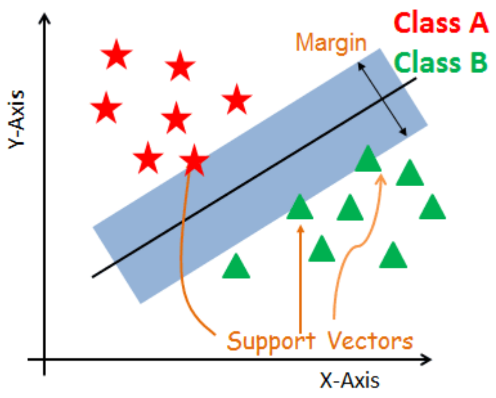
- Continuing with our example, if the hyperplane will be able to differentiate between males and females perfectly without doing any miss-classification, then that case of separation is known as Perfect Separation.
- Here, in the figure, if males are green and females are red and we can see that the hyperplane which is a line here has perfectly differentiated the two classes.

- (Training data — Data through which algorithm/model learns the pattern on how to differentiate by looking at the features
- Testing data — After the model is trained on training data, the model is asked to predict the values for unseen data where only the features are given, and now the model will tell whether its a male or a female)



- Now, there could be many hyperplanes giving 100% accuracy, as seen in the photograph.
- “” So to choose the optimal/best hyperplane, place the hyperplane right at the center where the distance is maximum from the closest points and give the least test errors further. “”
- To notice: We have to aim at the least TEST errors and NOT TRAINING errors.
- So, we have to maximize the distance to give some space to the hyperplane equation which is also the goal / main idea behind SVM.

The goal of the algorithm involved behind SVM:



- Finding a hyperplane with the maximum margin (margin is basically a protected space around hyperplane equation) and algorithm tries to have maximum margin with the closest points (known as support vectors).
- In other words, “The goal is to maximize the minimum distance.”

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

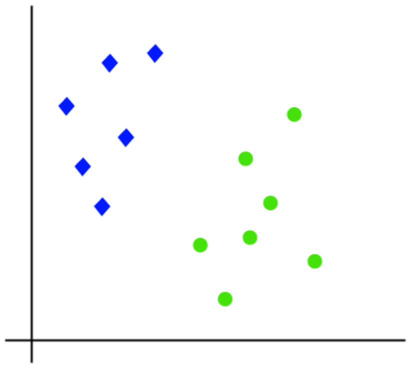
Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

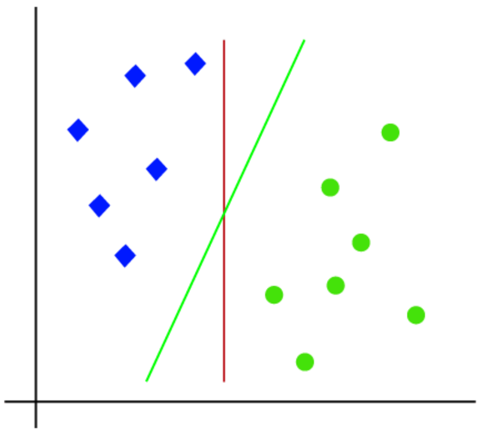
How does SVM works?

Linear SVM:

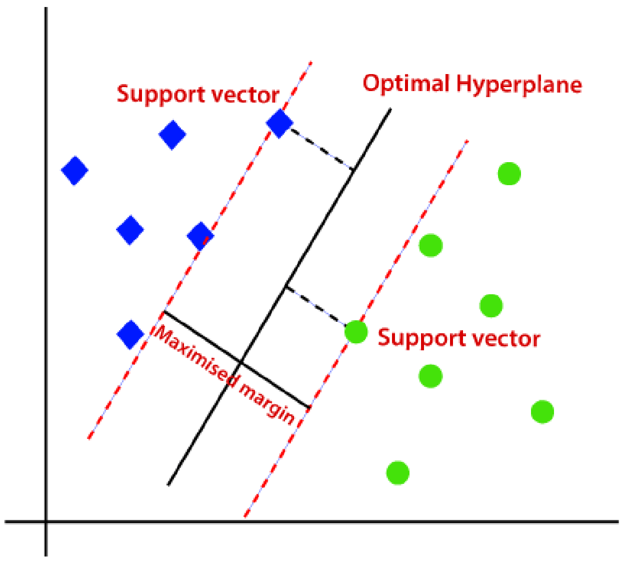
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

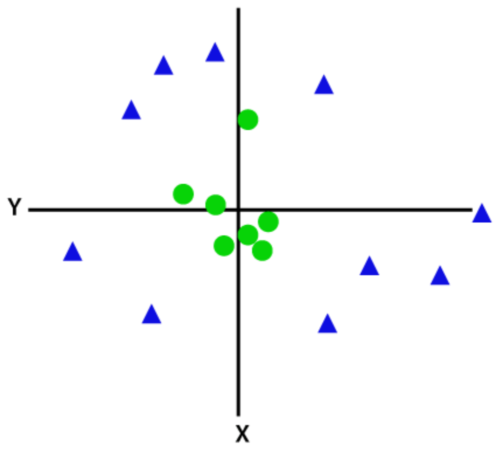


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

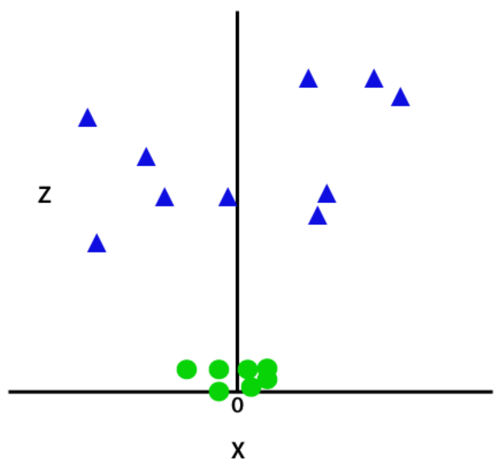
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



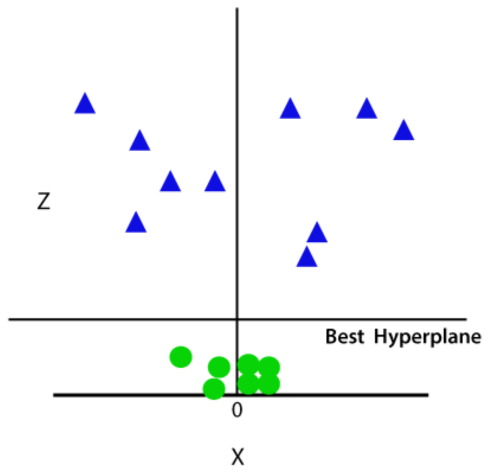
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

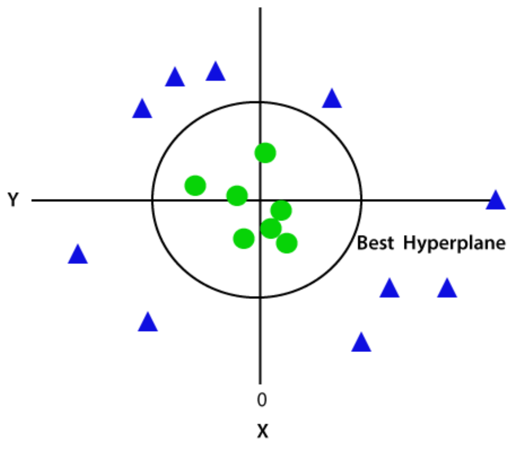
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

Major Kernel Functions in Support Vector Machine (SVM)

Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

Gaussian Kernel: It is used to perform transformation when there is no prior knowledge about data.

Gaussian Kernel Radial Basis Function (RBF): Same as above kernel function, adding radial basis method to improve the transformation.

Sigmoid Kernel: this function is equivalent to a two-layer, perceptron model of the neural network, which is used as an activation function for artificial neurons.

Polynomial Kernel: It represents the similarity of vectors in the training set of data in a feature space over polynomials of the original variables used in the kernel.

Linear Kernel: used when data is linearly separable.

SAMPLES

Name of the Kernel	**Mathematical Formula**
Linear	$k(x,y) = x^T . y$
Polynomial	$k(x,y) = (x^T, y)^P \, or \, k(x,y) = (x^T . y + 1)^P$ where p is the polynomial degree
RBF(Gaussian)	$\phi(x) = \exp(-\frac{x^2}{2\sigma^2}), \sigma > 0$

$$sigmoid\ function = \frac{1}{1 + e^{-(\Theta)}}$$
$$where\ \Theta = (1)x + 0$$

Python Implementation

In [2]:
```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [3]:
```python
data = pd.read_csv('d:bill_authentication.csv')
data.head()
```

Out[3]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

In [4]:
```python
data.shape
```

Out[4]:
```
(1372, 5)
```

In [6]:
```python
data.isnull().sum()
```

Out[6]:
```
Variance    0
Skewness    0
Curtosis    0
Entropy     0
Class       0
dtype: int64
```

In [7]:
```python
x = data.drop('Class',axis=1)
y = data['Class']
```

In [10]:
```python
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)
```

In [12]:
```python
from sklearn.svm import SVC
model = SVC(kernel='linear')
```

In [13]:
```python
model.fit(x_train,y_train)
```

Out[13]:
```
SVC(kernel='linear')
```

In [15]:
```python
ypre = model.predict(x_test)
```

In [18]:
```python
# comparing the result
df = pd.DataFrame({'acutal value':y_test,'predicted value':ypre})
df
```

Out[18]:

	acutal value	predicted value
660	0	0
326	0	0
1051	1	1
32	0	0
803	1	1
...	...	...
1171	1	1
702	0	0
786	1	1
1189	1	1
603	0	0

275 rows × 2 columns

In [21]:
```python
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,ypre))
print(classification_report(y_test,ypre))
```

```
[[138   1]
 [  1 135]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       139
           1       0.99      0.99      0.99       136

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```