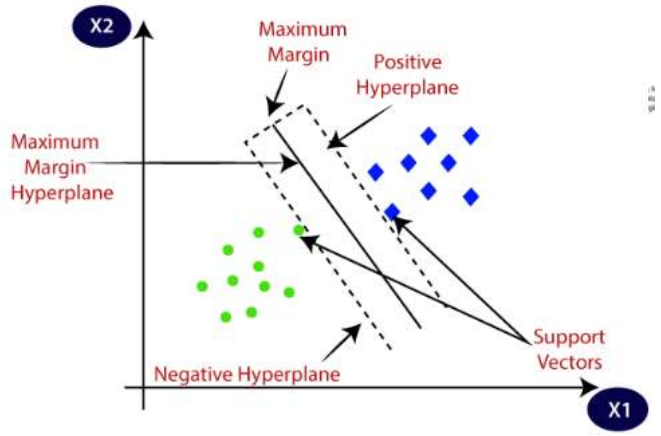
Machine Learning 💻

Support Vector Machine

By ⟹ PRINCE👑❤️

Support Vector Machine Algorithm

- **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.**
- **The plane that devides two different classes/predictor variables is a plane that becomes our *Support Vector Classifier*.**
- **Any arbitrary geometrical figure inside a vector field can be termed as a plane.**



SVM algorithm can be used for Face Detection, image classification, text categorization.

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**.*

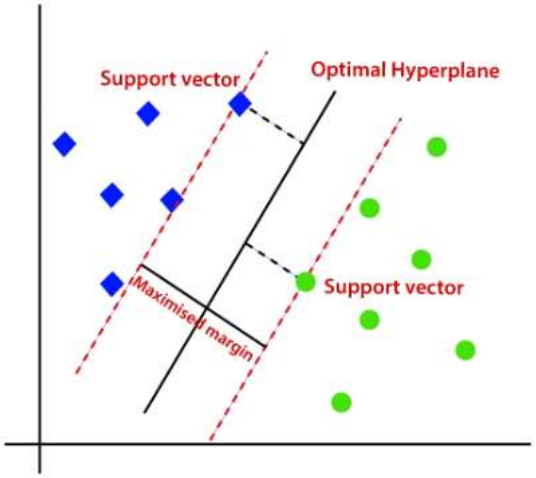
Hyperplane and Support Vectors in the SVM algorithm :

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.

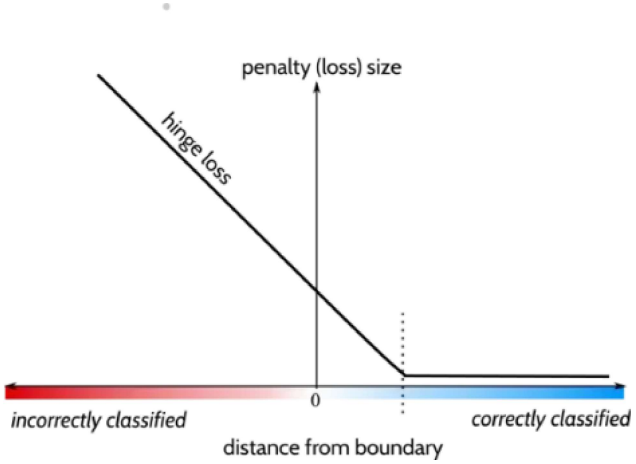
Support Vectors ⟶

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.



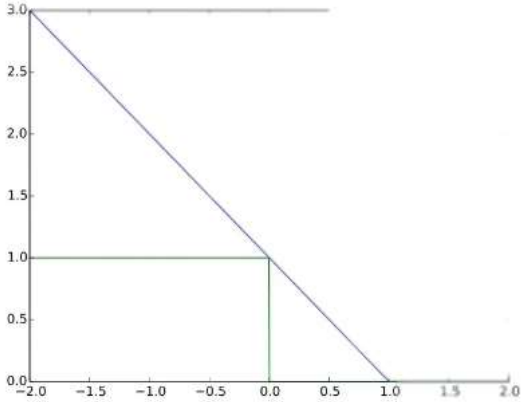
Hinge Loss :

Hing Loss is a type of cost function that is specifically tailored to *Support Vector Machine*.



1. *That dotted line on the x-axis represents the number 1. This means that when an instance's distance from the boundary is greater than or at 1, our loss size is 0.*
2. *If the distance from the boundary is 0 (meaning that the instance is literally on the boundary), then we incur a loss size of 1.*
3. *We see that correctly classified points will have a small(or none) loss size, while incorrectly classified instances will have a high loss size.*
4. *A positive distance from the boundary incurs a low hinge loss, or no hinge loss at all, and the further we are away from the boundary (and on the right side of it), the lower our hinge loss will be.*
5. A negative distance from the boundary incurs a high hinge loss. This essentially means that we are on the wrong side of the boundary, and that the instances will be classified incorrectly.

Numerical Visualization of Hinge Loss ⟶



When the point is at the boundary, the hinge loss is one(denoted by the green box), and when the distance from the boundary is negative(meaning it's on the wrong side of the boundary) we get an incrementally larger hinge loss.

C-Parameter :

► C parameter in SVM is Penalty parameter of the error term. You can consider it as the degree of correct classification that the algorithm has to meet or the degree of optimization the the SVM has to meet.

► For greater values of C, there is no way that SVM optimizer can misclassify any single point.

C ∝ (1 / Margin) ∝ (1 / Accuracy, Precison, Recall, F1 Score)

OR

Margin ∝ (Accuracy, Precison, Recall, F1 Score) ∝ 1 / C

- For greater value of C Parameter, Accuracy, Precison, Recall, F1 Score will be lower.
- C should be minimal, C = 1.0
- Updating of these parameters is known as parameter tuning.

Gamma Parameter :

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

Gamma ∝ C ∝ (1 / Accuracy, Precison, Recall, F1 Score)

OR

(Accuracy, Precison, Recall, F1 Score) ∝ C ∝ Gamma

- In order to research/experiment and bring out the best result through parameter tuning, SVM requires large Dataset.
- Gamma ranges from zero (0) to infinity (∞).
- 'gamma' in Python.

Note :

1. If we haven't decleared 'C' or 'gamma' inside the SVM then there is a possibility that the accuracy will be lower.
2. If you have decleared after the training there is a possibility that the accuracy or other performance metrics might increase.

Kernel Parameter :

Kernel parameters are used as a tuning parameter to improve the classification accuracy. There are mainly four different types of kernels (Linear, Polynomial, RBF, and Sigmoid) that are popular in SVM classifier.

Types of Kernel Parameter ⟶

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Function
4. SIgmoid

Linear Kernel ⟶

- Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line.
- It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set.

Polynomial Kernel ⟶

- A polynomial kernel is a kind of SVM kernel that uses a polynomial function to map the data into a higher dimensional space. It does into this by taking the dot product of the data points in the original space and the polynomial function on the new space.

- For degree d polynomials, the polynomial kernel is defined as :

$$K(x_1, x_2) = (x_1^T x_2 + C)^d$$

where, *C is a constant and x1 and x2 are vectors in the original space.*

Radical Basis Kernal (RBF) ⟶

- Redical Basis Kernel function that is used in machine learning to find a non-linear classifier or regression line.
- RBF Kernel is popular because of its similirity t KNN. It has the advantage of KNN and overcomes the space complexity problem.
- Kernel function is used to transform n-dimensional input, where m is much higher than n then find the dot product in higher dimensional efficiently.
- The main idea to use kernel is: A linear classifier or regression curve in higher dimensions becomes a Non-linear classifier or regression curve in lower dimensions.

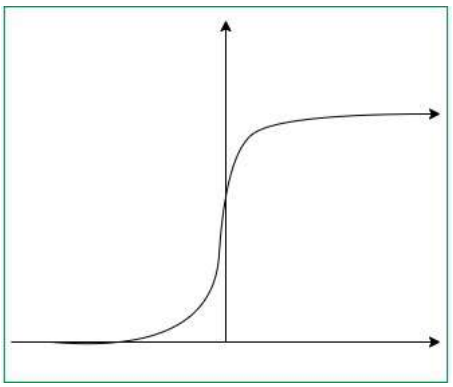
Mathematical Definition of Radial Basis Kernel :

$$K(x_1, x_2) = \exp[-(|| x_1 - x_2 ||)^2 / 2\sigma^2]$$

- *Where x_1 and x_2 are vector point in any fixed dimensional space.*

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.



Python Impelementation For Support Vector Machine (SVM) ⟶

Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Working on the Bank Data

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Uploading and Reading Bank Data
df = pd.read_csv('/content/drive/MyDrive/Dataset/Bank.csv')
```

```
df.head()
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account
0	1	25	1	49	91107	4	1.6	1	0	0	1	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0

```
df.shape
```

```
(5000, 14)
```

```
df.isna().sum()
```

```
ID                  0
Age                 0
Experience          0
Income              0
ZIP Code            0
Family              0
CCAvg               0
Education           0
Mortgage            0
Personal Loan       0
Securities Account  0
CD Account          0
Online              0
CreditCard          0
dtype: int64
```

Applying Machine Learning

```
x = df.drop(['ID','ZIP Code','Personal Loan'], axis = 1)
y = df[['Personal Loan']]
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 1)
print(xtrain.shape, xtest.shape)
print(ytrain.shape, ytest.shape)
```

```
(3500, 11) (1500, 11)
(3500, 1) (1500, 1)
```

```
# Importing SVM
from sklearn.svm import SVC
model = SVC()
```

```
model.fit(xtrain, ytrain)
```

```
SVC()
```

```
ypred = model.predict(xtest)
print(ypred)
```

```
[0 0 0 ... 0 0 0]
```

Accuracy Score, Confusion Matrix, Classification Report ⟶

Without Kernel (C-Parameter and Gamma)

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
accuracy_score(ytest, ypred)
```

```
0.9073333333333333
```

```
cf = confusion_matrix(ytest, ypred)
```

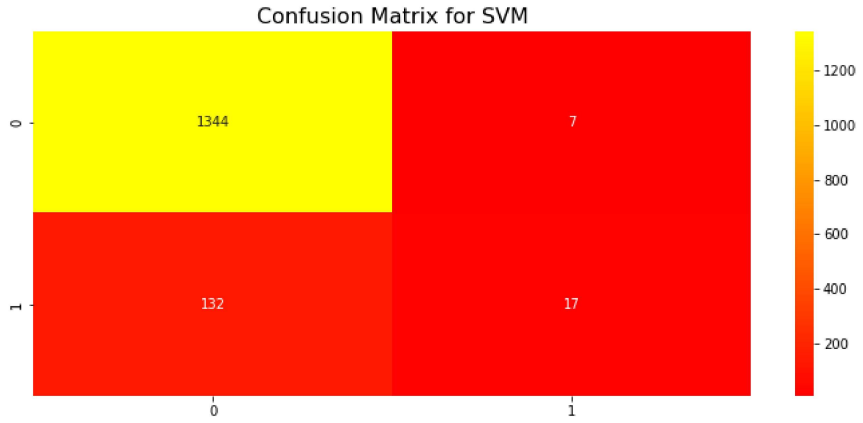
```
print(classification_report(ytest, ypred))
```

```
                  precision    recall  f1-score   support

               0       0.91      0.99      0.95      1351
               1       0.71      0.11      0.20       149

        accuracy                           0.91      1500
       macro avg       0.81      0.55      0.57      1500
    weighted avg       0.89      0.91      0.88      1500
```

```
plt.figure(figsize = (12,5))
plt.title('Confusion Matrix for SVM', fontsize = 16)
sns.heatmap(cf, annot = True, fmt = 'g', cmap = 'autumn')
plt.show()
```



Now Applying Kernel, C-Parameter and Gamma Parameter

After applying SVM to set the C-Parameter or Gamma Parameter is known as Parameter Tuning

```
model = SVC(kernel = 'linear', C = 1.0, gamma = 'auto')
# set C = 1.0 = minimal value of C for good accuracy
# You can take gamma = 'auto', if you set C = 1.0 already no need to take gamma as a parameter.
model.fit(xtrain, ytrain)
ypred = model.predict(xtest)
print(ypred)
```

```
[0 0 0 ... 0 0 0]
```

1. **Set C = 1.0 = minimal value of C for good accuracy.**
2. **You can take gamma = 'auto', if you set C = 1.0 already no need to take gamma as a parameter.**

Comparing Accuracy Score, Confusion Matrix and Classification Report with and without Linear Kernel.

```
# Accuracy Score
accuracy_score(ytest, ypred)
```

```
0.9473333333333334
```

```
# Confusion Matrix
cf2 = confusion_matrix(ytest, ypred)
cf2
```

```
array([[1340,   11],
       [  68,   81]])
```

```
# Classification Report
print(classification_report(ytest, ypred))
```

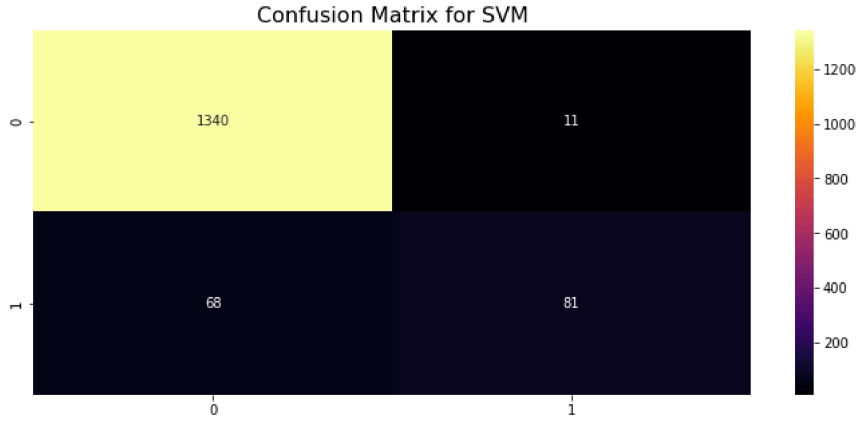
```
                  precision    recall  f1-score   support

               0       0.95      0.99      0.97      1351
               1       0.88      0.54      0.67       149

        accuracy                           0.95      1500
       macro avg       0.92      0.77      0.82      1500
    weighted avg       0.94      0.95      0.94      1500
```

Visualization of Confusion Matrix

```
plt.figure(figsize = (12,5))
plt.title('Confusion Matrix for SVM', fontsize = 16)
sns.heatmap(cf2, annot = True, fmt = 'g', cmap = 'inferno')
plt.show()
```




```
# Accuracy (Direct Calculation)
# Total Truth Value = 1340 + 81
# Total Values = Truth + False  = 1340 + 11 + 68 + 81
acc = (1340 + 81) / (1340 + 11 + 68 + 81)
acc
```

```
0.9473333333333334
```