29/04/22
1:00pm

SVM (Support Vector Machines)

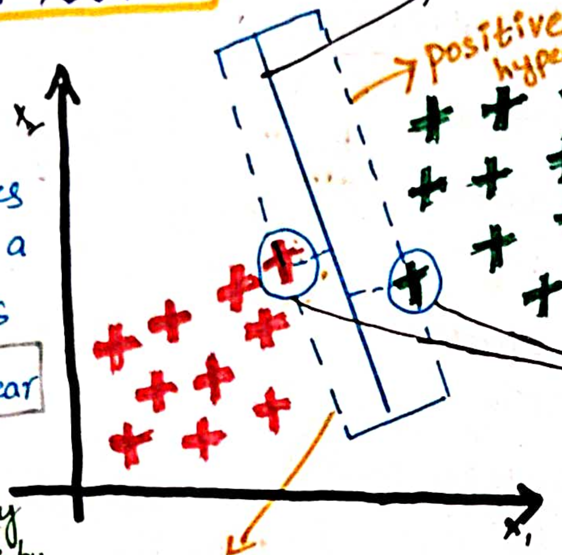
* **Linear Seperable** :-

Seperating The classes by using a Straight is called "Linear SVM"

it identify Best fit line by "Maximum margin classifier".

Maximum Margin Hyperplane (classifier)

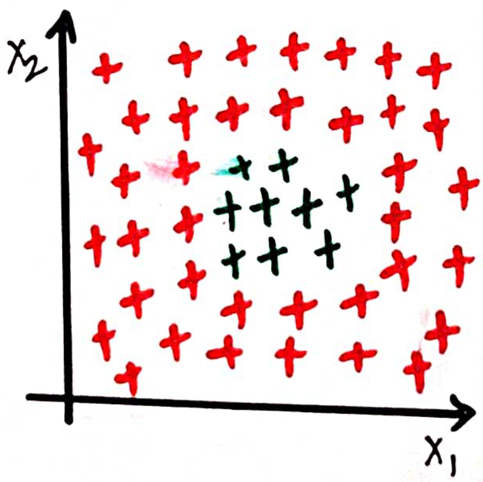
it is used in 3 Dimensional where as, straight is consider in 2D, when it in 3 dimensional view it is called "Hyperplane".



Support Vectors

The classes (or) points which are very close to "difference".

* **Non-linear seperable (or) Kernal SVM**



* In This situation, we use "Non-linear SVM" (or) Kernal SVM

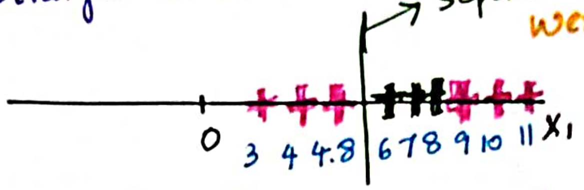
Non-linear

Ex:- Mapping to a higher Dimension

X	y
3	Pink
4	Pink
4.8	Pink
6	green
7	green
8	green
9	Pink
10	Pink
11	Pink

* Earlier, it is degree of '1'. Now, we convert it into "quadratic Equation" and we seperate the data, By writing single condition

These Points are not linearly seperable. we Can't draw a Straight to divide Them.



were we have different class, Before to that is called "Seperator".

So, we Convert Every Data Point to

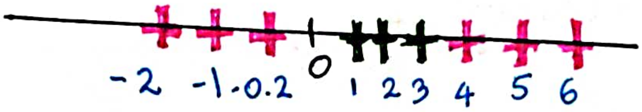
$$\boxed{f = x-5}$$

Why $f = (x-5)$?

Because, we identify "Seperator."

Ex: 9 → seperator if it is 9, $\boxed{f = x-9}$

$f = 3-5 = -2$	$f = 6-5 = 1$	$f = 9-5 = 4$
$f = 4-5 = -1$	$f = 7-5 = 2$	$f = 10-5 = 5$
$f = 4.8-5 = -0.2$	$f = 8-5 = 3$	$f = 11-5 = 6$

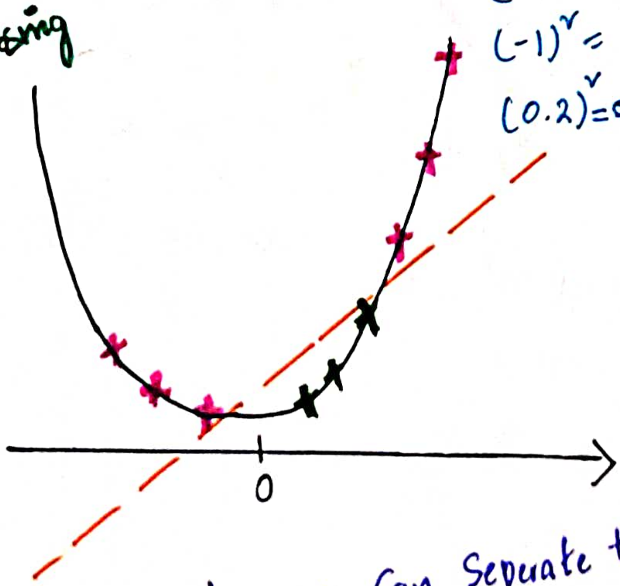


Next, we square the data points.

$$\boxed{f = (x-5)^2}$$

$(-2)^2 = 4$	$1^2 = 1$	$4^2 = 16$
$(-1)^2 = 1$	$2^2 = 4$	$5^2 = 25$
$(0.2)^2 = 0.04$	$3^2 = 9$	$6^2 = 36$

Here, we are increasing "Higher" dimension.

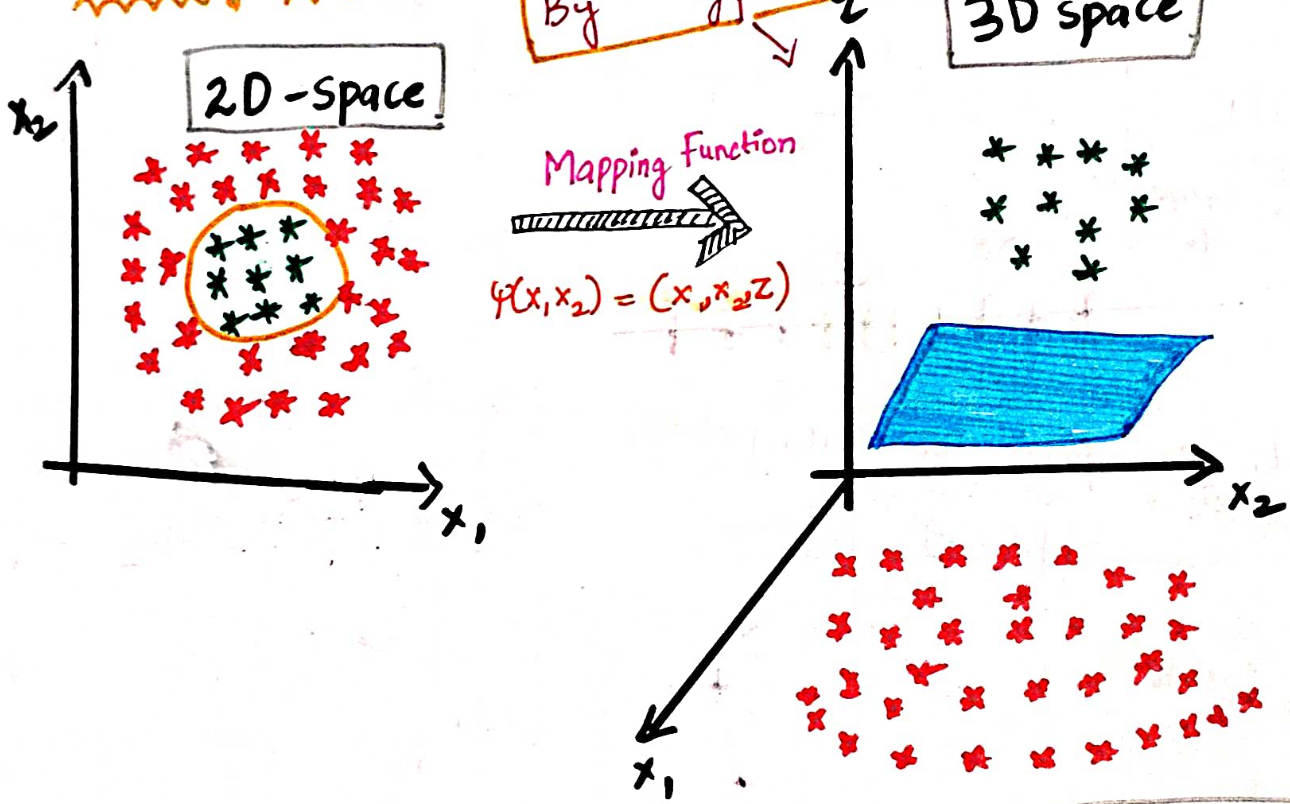


Now, we Can Seperate them by "Linear" line

By Table

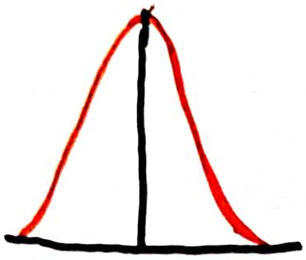
x	y	x-5	$(x-5)^2$
3	Pink	-2	4
4	Pink	-1	1
4.8	Pink	-0.2	0.04
6	green	1	1
7	green	2	4
8	green	3	9
9	pink	4	16
10	pink	5	25
11	pink	6	36

⇒ SVM-Kernal:-



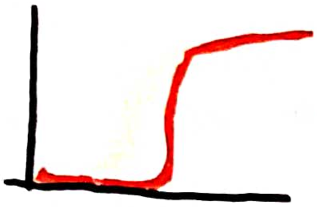
* Mapping to higher dimensional space can be Highly Compute intensive. More RAM needed.

* Types of Kernel functions — Radial Basis function :- Shape of Kernal : R.B.F



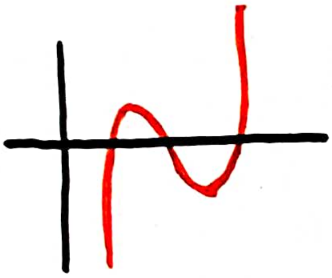
Gaussian Kernal → RBF

$$K[\vec{x}, \vec{l^i}] = e^{-\frac{\|\vec{x} - \vec{l_i}\|^2}{2\sigma^2}}$$



sigmoid kernal

$$K(x,y) = \tanh(\gamma \cdot x^T y + r)$$

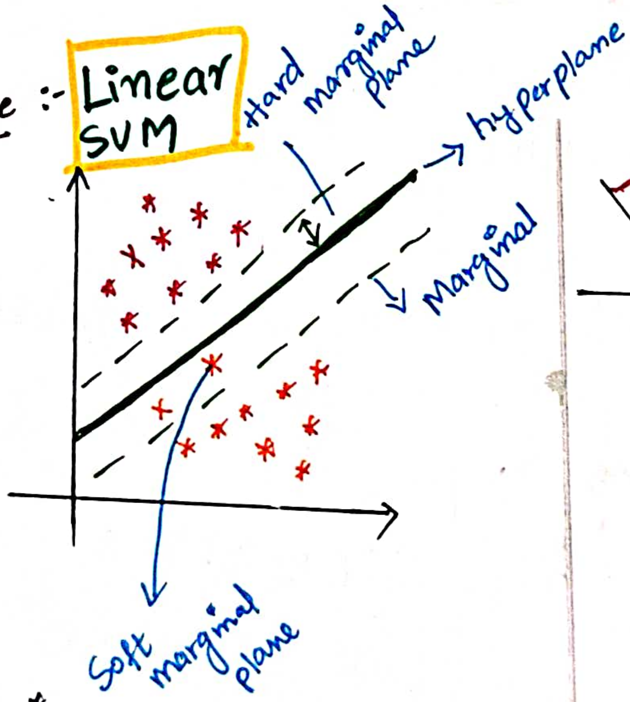


Polynomial Kernal

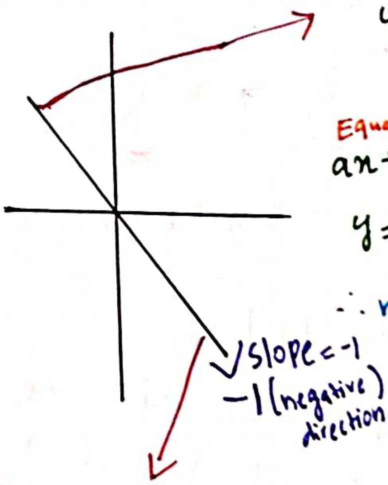
$$K(x,y) = (\gamma \cdot x^T y + r)^d, \ \gamma > 0$$

↓ apply Higher The "degree"

Example :- **Linear SVM**



***



$y = mx + c$ (slope: m, intercept: c)

⇓

Equation For Straight line

$ax + by \pm c = 0$] same

$y = \frac{-c}{b} - \frac{ax}{b}$

$\therefore m = \frac{-a}{b}, \ c = \frac{-c}{b}$

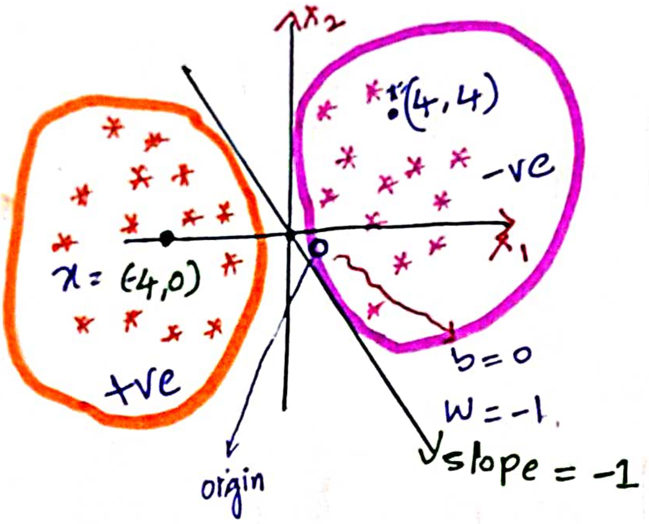
$y = mx + c$

written as

$y = w_1 x + w_2 x + w_3 x + \dots + c$

$$\boxed{y = W^T x + b}$$

→ Transpose



Equation :- $\boxed{y = W^T x + b}$

$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [-4, 0]$$

By multiplication

$$= 4 \Rightarrow (+ve) \text{ value.}$$

Any Point under the line is Positive Value.

New Data point (4,4)

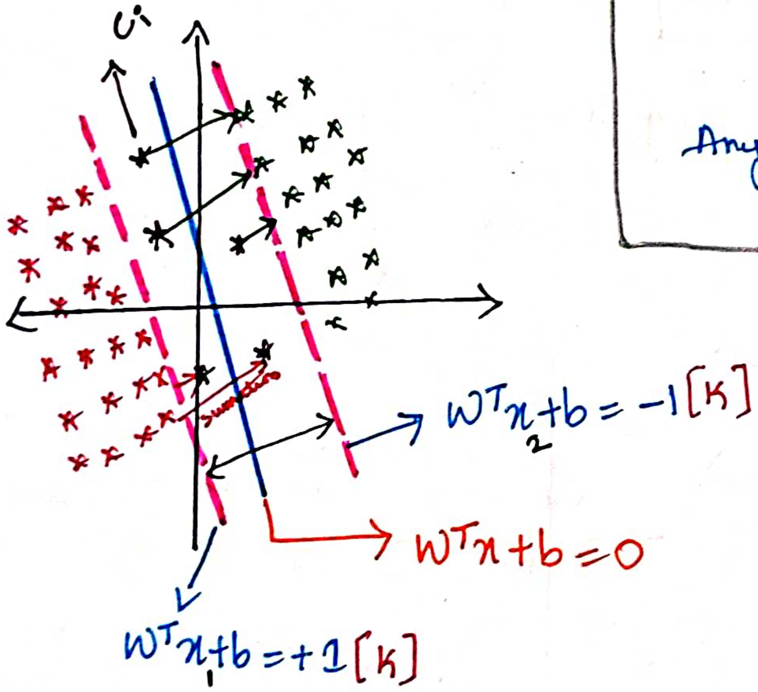
$$\boxed{y = W^T x + b}$$

$$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4\ 4]$$

$$= -4 + 0 = -4$$

Any point above the line is negative value.

⇒ Marginal plane :



Our Aim should increase distance, to $\boxed{\text{Maximize}}$ Perform model well

Vectors + magnitude

$\vec{W}$ $\|W\|$

$$W^T x_1 + b = +1$$

$$W^T x_2 + b = -1$$

$$W^T (x_1 - x_2) = 2 \quad \text{(difference between above \& below plane)}$$

$$\frac{W^T (x_1 - x_2)}{\|W\|} = \frac{2}{\|W\|}$$

Maximize cost function

$$\max_{(w,b)} \frac{2}{\|w\|}$$ (Marginal plane will be bigger)

Such that,

$$\boxed{y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}}$$

Condition

$(w^T x + b)$ greaterthan $= 1 = +1$

$(w^T x + b)$ Lesser than $= -1 = -1$

$y_i * (w^T x_i + b) \geq 1$

$\Downarrow$

For Correct point

$$\boxed{\max_{(w,b)} = \boxed{\frac{2}{\|w\|}} \xRightarrow{\text{Both are Equal}} \min_{(w,b)} = \boxed{\frac{\|w\|}{2}}}$$

For [SVR] → parameter changes.

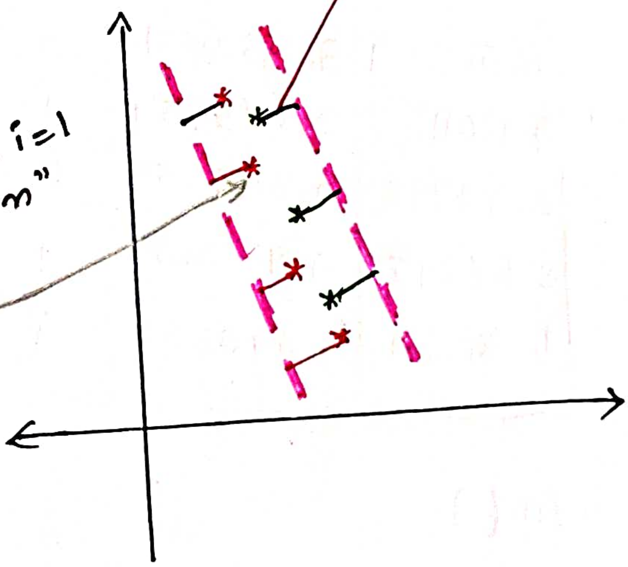
$$\min_{(w,b)} \frac{\|w\|}{2} + \underbrace{c_i \sum_{i=1}^{n} \xi_i}_{\text{"Hypertuning Parameter"}}$$

ξ_i → Eta (summation of the distance of the wrong data points)

c_i → How many wrong prediction (or) Errors, we can Have

$\sum_{i=1}^{n}$ → Sumation $i=1$ to "n"

Errors



29/04/22 4:30pm

29/9/23 6pm
11:00pm

CODE :-

Data :-
The data shown here simulates a medical study in which infected with a virus were given various doses of two medicines and then checked 2 weeks later to see if they were still infected. Given this data, our goal is to create a classification model than predict (given two dosage measurements) if they mouse will still be infected with virus.

load data, with import libraries.

```
# df = pd.read_csv ("mouse_viral_study.csv")
# df.head()
```

Out :

med_1_mL	Med_2_mL	Virus_Present
6.508231	8.582531	0
4.126116	3.073459	1
6.427870	6.369758	0
3.672953	4.905215	1
1.580321	2.440562	1

```
# df.info()
```

Out : Range Index 400 entries 0 to 399

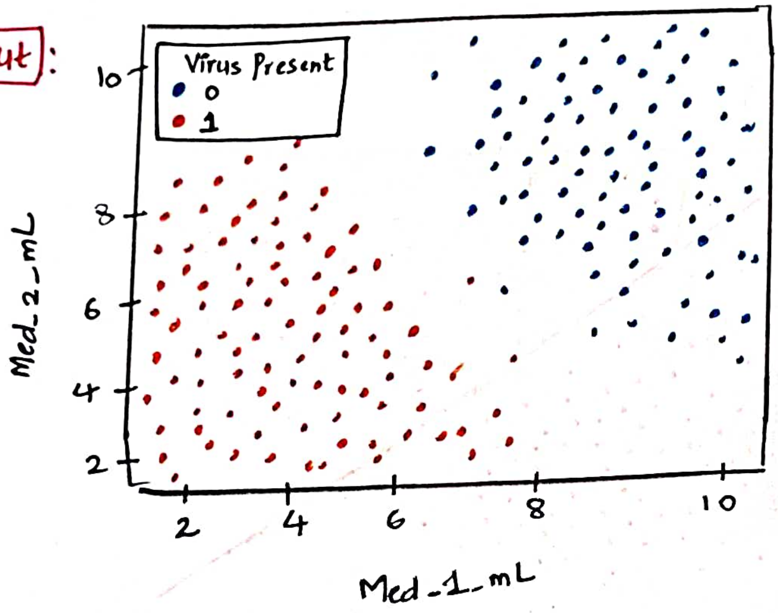
0	Med_1_mL	400 non null	Float 64
1	Med_2_mL	400 non null	Float 64
2	Virus_Present	400 non null	int 64

EDA

```
# sns.scatterplot(x="Med_1_mL", y="Med_2_mL", hue="Virus present",
                  data=df)

# plt.show()
```

Out:



X & Y

```
# x = df.drop("Virus Present", axis=1)
# y = df["Virus present"]
```

MODELLING

```
from sklearn.svm import SVC
# model = SVC(kernel = "linear")
# model.fit(x,y)
```

Out: SVC[kernal = "linear")

30/4/22
12:30 pm

30/4/22
11:00Am

Seperating Hyper plane

#Sns. scatterplot(x = "Med_1_ML", y = "Med_2_mL", hue = "Virus Pres..
Palette = "seismic", data = df)

#we want to some how automatically Create a Seperating hyperplane (a line in 2D)

x = np.linspace (0,10,100)

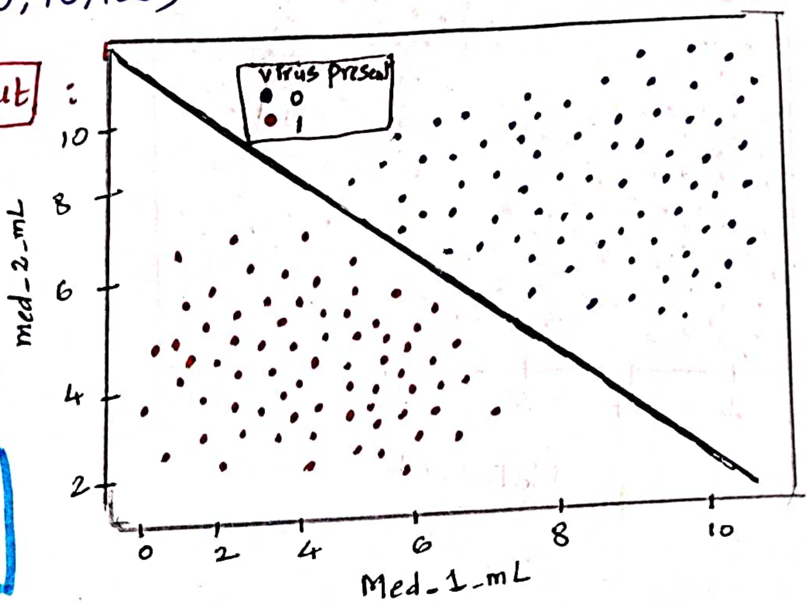
m = -1

b = 11

y = m*x +b

plt.plot (x,y,"k")

Out :



 # user defined function

def plot_svm_boundary (model, x,y):

x = x. Values

y = y. Values

scatter plot

plt. scatter (x[:,0] , x[:,1], c=y, s=30,
Cmap = "seismic")

plot the decision function.

ax = plt . gca ()
xlim = ax . get_xlim()
ylim = ax. get_ylim()

Creating grid to Evaluate model.

```
xx = np.linspace (xlim[0], xlim[1], 30)
yy = np.linspace (ylim[0], ylim[1], 30)
YY, XX = np.meshgrid (yy, xx)
xy = np.vstack ([XX.ravel(), YY.ravel()]).T
Z = model.decision_function (xy).reshape (XX.shape
```

plot decision boundary and margins.

```
ax.contour (XX, YY, Z, colors= "k", Levels =[-1,0,1
            alpha=0.5, line styles = ["--", "-", "--
```

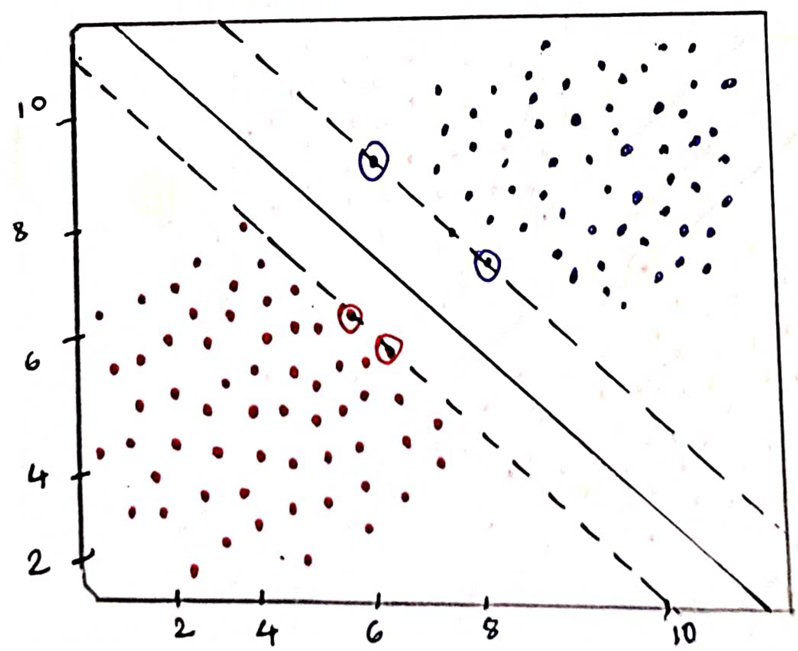
plot support vectors.

```
ax.scatter (model.support_vectors_ [:,0], model.
            support_vectors_ [:,1], s=100, linewidth =1,
            face colours = "none", edge colors = "k")
```

```
Plt.show()
```

plot_svm_boundary (model, x,y)

Out:



Our goal with SVM is to create the best Seperating hyperplane with Maximum margin classifier.

In 2 dimensions, The hyperplane is Simply a line.
straight

In 3 dimensional → plane

Hyper Parameters

C :- Regularization parameter.

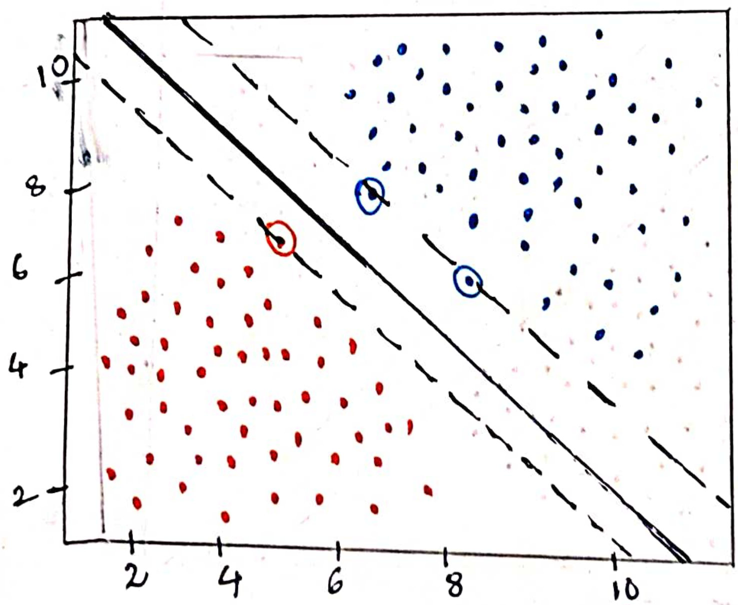
The Strength of the regularization is inversely proportional to "C". Must be strictly positive. The penality is a Squared l2 penalty.

C ↑ (Less Sensitility).
Higher The "c" Value - Lesser The missclassification

* C ↓ (# Lower The "c" value - incorrect answer

```
# model = SVC (Kernel = "Linear", C = 1000)
# model. fit (x,y)
# plot_svm_boundary (model, x,y)
```

out :-

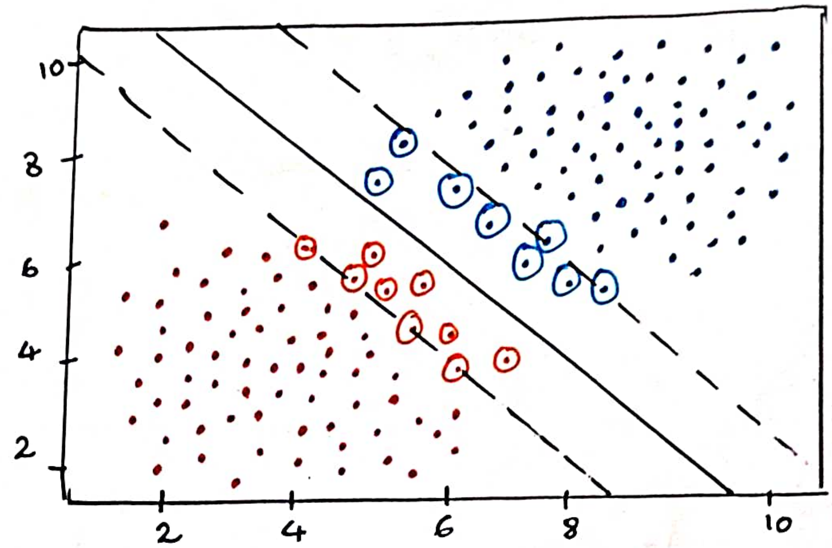


model = svc (Kernel = "linear", C = 0.05)

model. fit (x,y)

plot_svm_boundary (model, x,y)

out :



Kernal

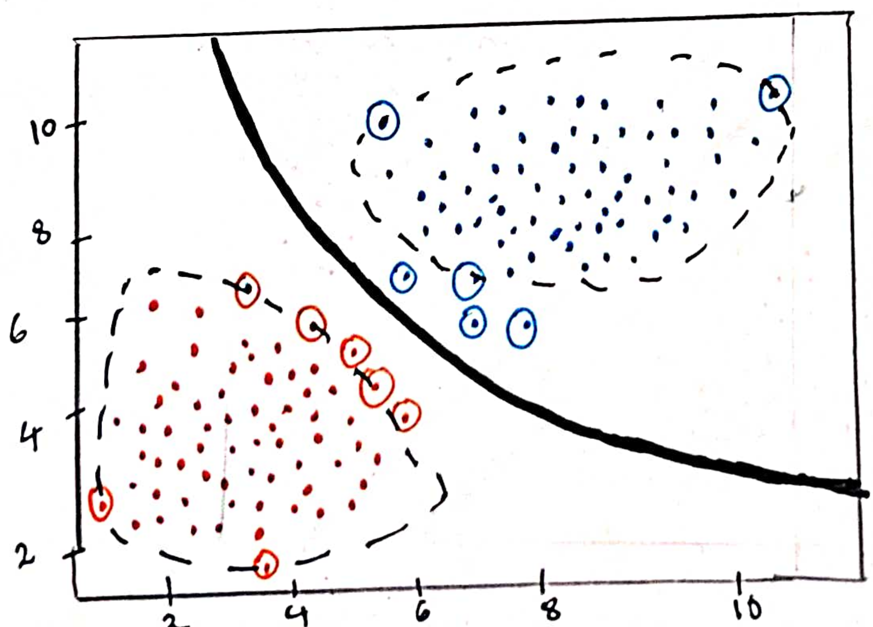
1. RBF

model = SVM (kernel = "rbf", c = 1)

model. fit (x,y)

plot_svm_boundary (model, x,y)

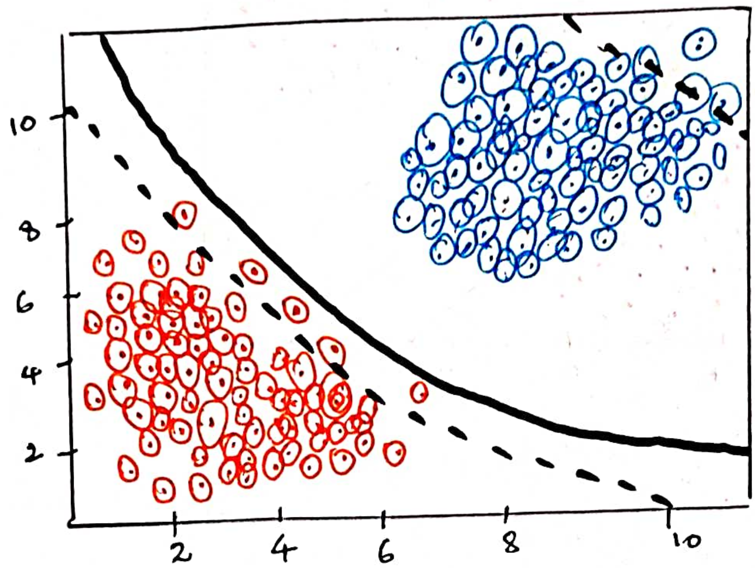
out :



2. Sigmoid

```
# model = SVC(kernel = "sigmoid")
# model.fit(x,y)
# plot_svm_boundary(model, x,y)
```

Out :

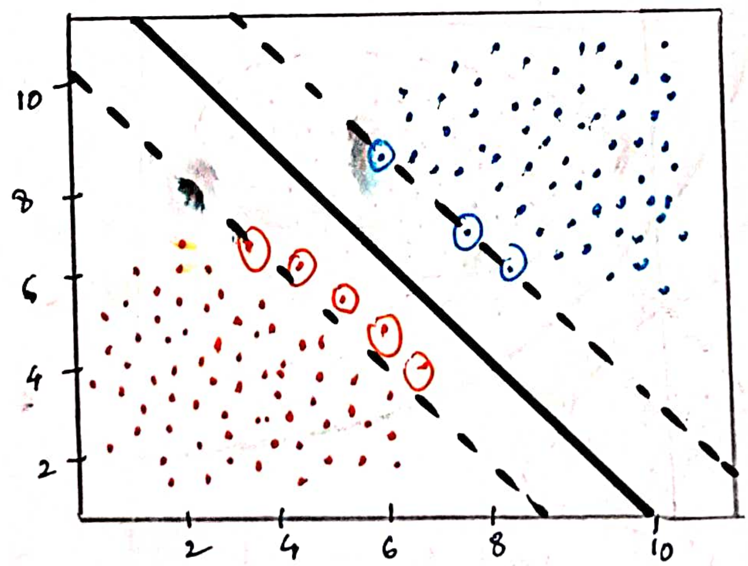


3. Polynomial

"Linear" Both are same.

```
# model = SVC(kernel= "poly", C=1, degree=1)
# model . fit(x,y)
# plot_svm_boundary(model, x, y)
```

Out :



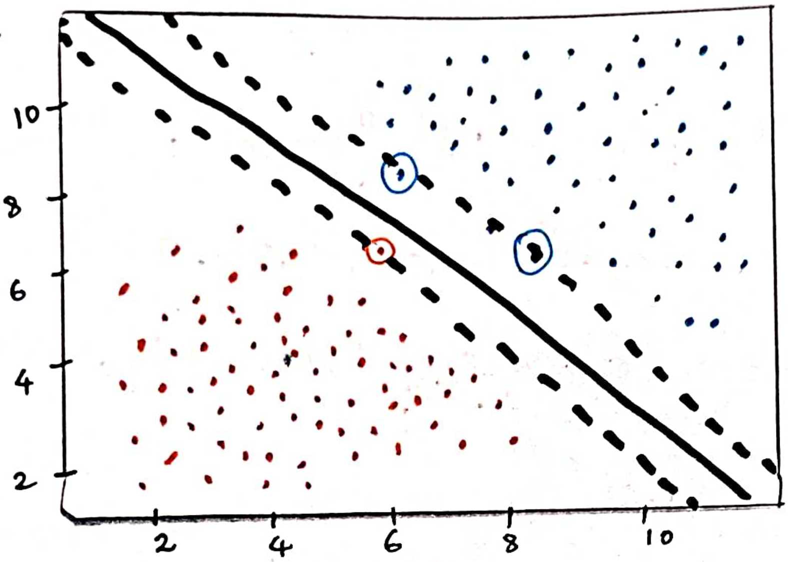
model = SVC (kernel = "Poly", C = 1, degree = 2)

"Quadratic" curve.

model . fit (x, y)

plot_svm_boundary (model, x, y)

out :-



gamma

gamma :- {scale, auto} or float

for "rbf", "Poly" & "sigmoi

Default = "scale" kernel Coefficient

$$\frac{1}{\text{num. of feature}} \times x.\text{Var}()$$

"gamma = scale" $\Rightarrow$ $\frac{1}{\text{num. of feature}} \times x.\text{Var}()$

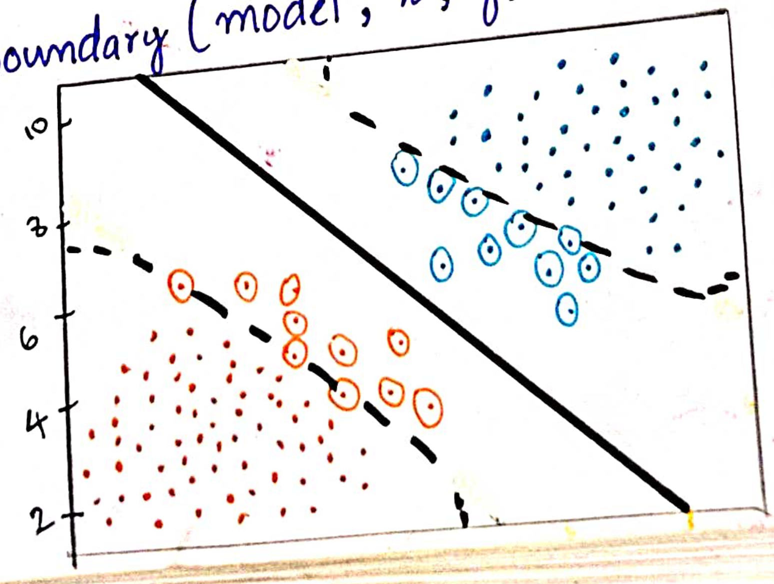
"gamma = auto" $\Rightarrow$ $\frac{1}{\text{n\_features}}$.

model = SVC (kernel = "rbf", C = 1, gamma = 0.01)

model . fit (x, y)

plot_svm_boundary (model, x, y)

out :



Hyper parameter Tuning

from sklearn.model_selection import GridSearchCV

```
# estimator = SVC()
# param_grid = {"c":[0,0.01,0.1], "kernel": ["linear","rbf"]
                , "gamma":[0.01,0.02,0.1]}
# grid = GridSearchCV(estimator, param_grid, cv=5)
# grid.fit(x,y)
```

out: GridSearchCV(cv=5, estimator=SVC(), param_grid={"c": [0,0.01,0.1], "kernel": ("linear","rbf"), gamma [0.01, 0.02, 0.1]}

```
# grid.best_params_
```

out: {"c": 0.01, "gamma": 0.01, "kernel": "rbf"}

If we don't write gamma, kernel = "linear" gives best prediction and accuracy.
:Only for this problem

```
# grid.best_score_
```

out: 1.0

train test split

```
from Sklearn.model_selection import train_test_split
# x_train, x_test , y_train, y_test = train_test_split (x,y,
                        test_size = 0.25 , random_state = 29)
```

Scaling

```
from Sklearn . preprocessing import StandardScaler
# sc = Standard Scaler ()
# x_train = sc. fit_transform (x_train)
# x_test = sc. fit_transform (x_test)
```

model

```
# model = SVC ( Kernel = "rbf", C = 0.01, gamma = 0.01)
                        "Linear"
# model. fit (x_train, y_train)
```

out SVC (C = 0.01, gamma = 0.01, Kernel = "Linear")

Prediction

```
                       Predict
# ypred_train = model.        (x_train)
# ypred_test = model. Predict (x_test)
```

accuracy

```
from sklearn.metrics import accuracy_score

# accuracy_score (y_test, ypred_test)
```

Out: 1.0

30/4/22

5:00pm.

Que:- What is "model Selection"?

Ans:- Identify The best Algorithm with best Hyperparameter

By applying hyperparameter "tunning"