Decision Tree and Support Vector Machine HandWritten Notes

- Darshan R M

"Learn, Share, and Collaborate"

Week - 16

Decision Trees and Support Vector Machines

* Decision Tree classifier (classification problem) ← values of categorical

* Consider, Problem: Predict person play Tennis (or) not.

Dataset →

dependent

Day	Outlook	Temperature	Humidity	Wind	
1	Sunny	Hot		Weak	No
2	Overcast	Mild		Strong	Yes
3	Rain	(		(	)

Independent

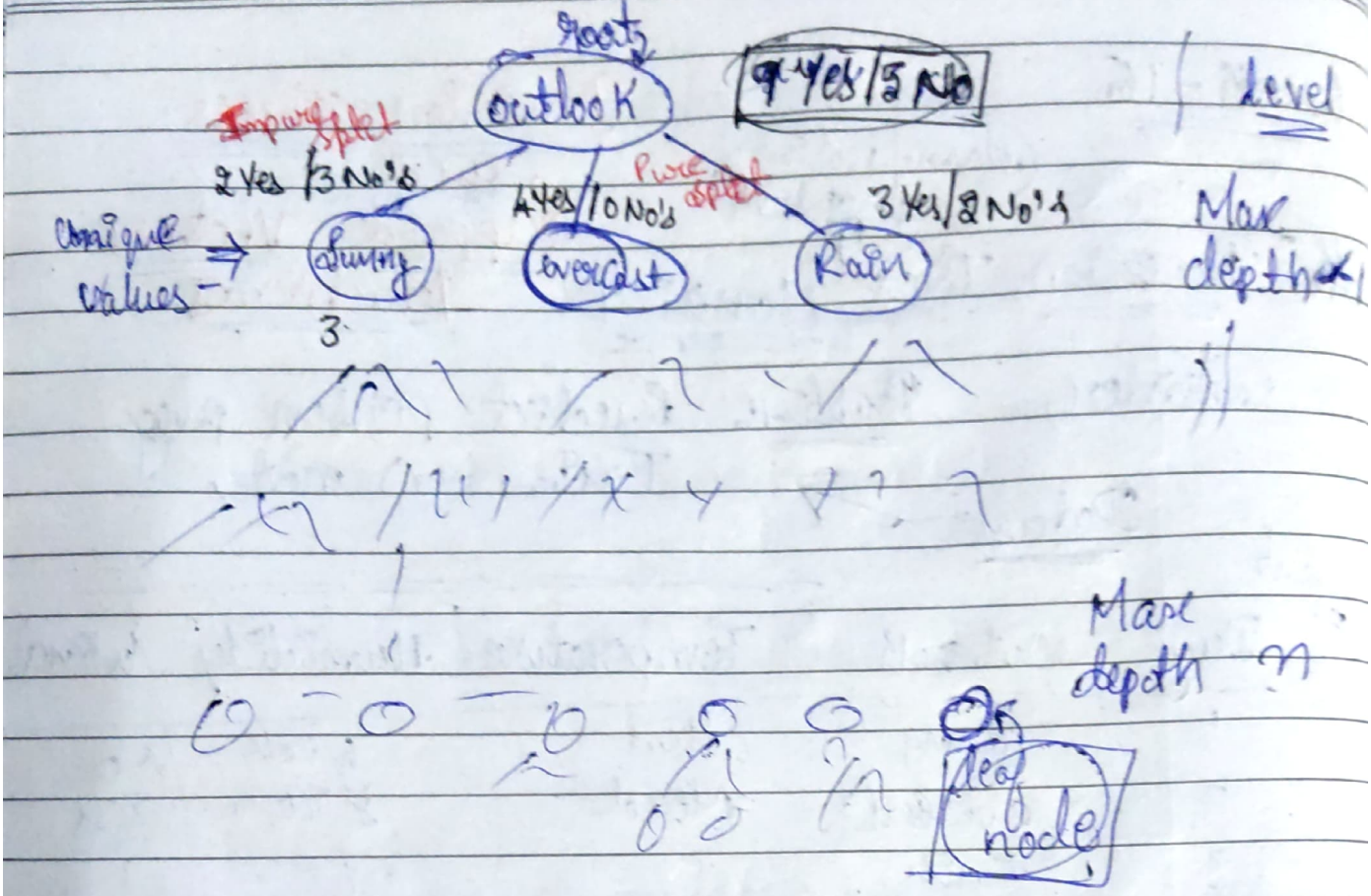
Play Tennis

No
Yes

→ Categorical output

Decision Tree

① Selects root node → from available independent features.



After first Max depth level

(1) Purity split check - we check
Pure Split (or)
Impure Split

No need to shift further

i.e Number of Yes/No = 0

then it will become leaf node.

we have to split further

i.e Both Yes & No ≠ 0

* But in maths we use some techniques

Pure split / Impure

checks → (i) Entropy
→ (ii) Gini Impurity
} Measure of purity

(2) what feature we need to select to start the split?

→ Information gain

* split happens till we reach leaf node.

---

(1) Purity check

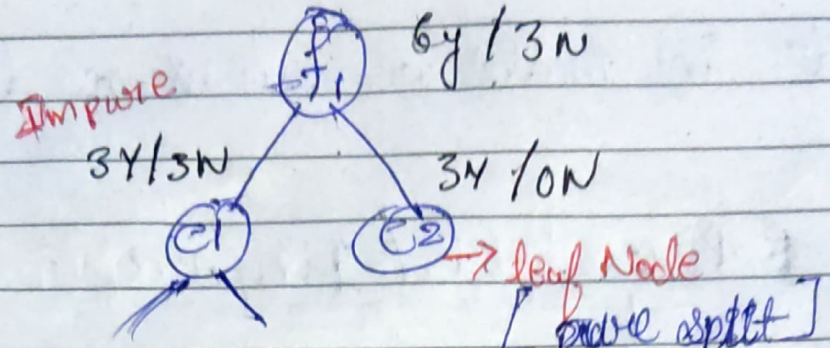
(i) Entropy: → In chemistry, entropy means measure of randomness of system.

For Binary classification

$$H(S) = -(P_+)(\log_2 P_+) - (P_-)(\log_2 P_-)$$

P_+ → Probability of positive category
P_- → —11— of negative —11—

Consider,



* $H(C_1) = -(P_+)(\log_2 P_+) - (P_-)(\log_2 P_-)$

+ve values

* $P_+ = \frac{3}{6} = \frac{1}{2}$ (Total)

$P_- = \frac{3}{6} = \frac{1}{2}$

Interview

Value range b/w 0 to 1

$H(C_1) = -(3/6)(\log_2(3/2)) - (3/6)(\log_2(3/6))$

$H(C_1) = 1$ ↔ entropy

Value 1 indicates impure split.

Completely impure split if $(P_+) = (P_-)$

* $H(C_2) = -(P_+)(\log_2 P_+) - (P_-)(\log_2 P_-)$

* $P_+ = \frac{3}{3} = 1$

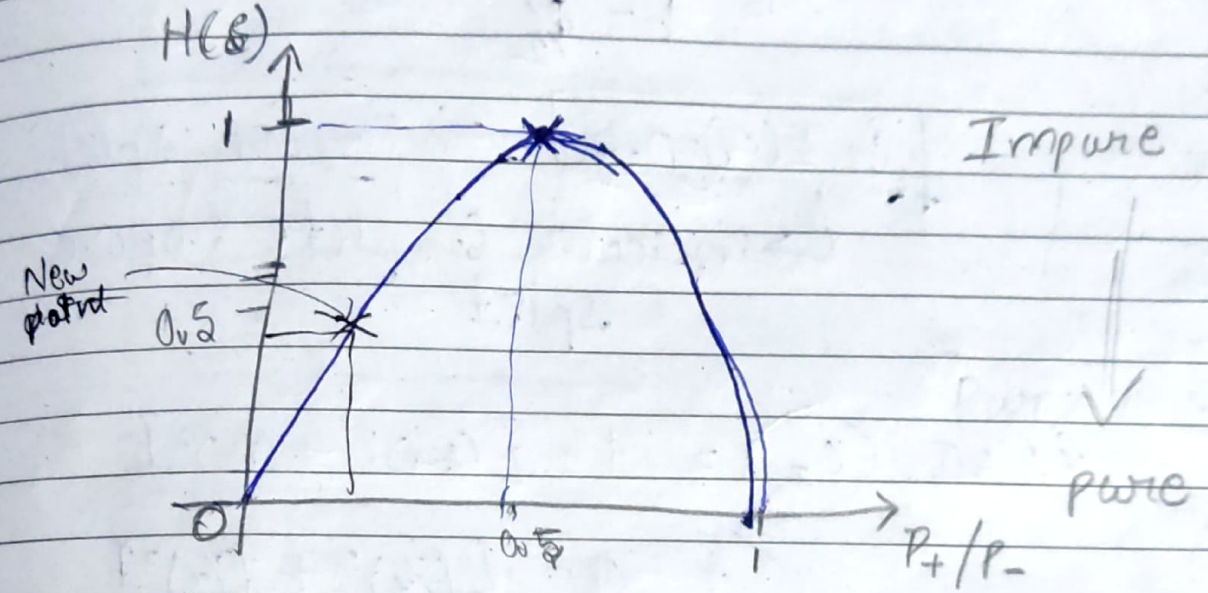
$P_- = \frac{0}{3} = 0$

$\Rightarrow H(C_2) = -(1)(\log_2 1) - (0)(\log_2 0)$

$\boxed{H(C_2) = 0}$

↳ indicates pure split.

↳ Completely pure split



(ii) Gini impurity [G.I]

$$G.I = 1 - \sum_{i=1}^{n} (P)^2$$

~~$-[(P_+)\log P_+] -$~~

For Binary class

$$G.I = 1 - [(P_+)^2 + (P_-)^2]$$

** Consider previous example,

* $G_i.I(E)_2 = 1 - [(P_+)^2 + (P_-)^2]$

$= 1 - [(3/6)^2 + (3/6)^2]$

$= 1 - 1/2$

$G_1.I(C_1) = 0.5$ ⇒ impure split

↳ indicates complete impure split

Interview

Value range b/w p to 0.5

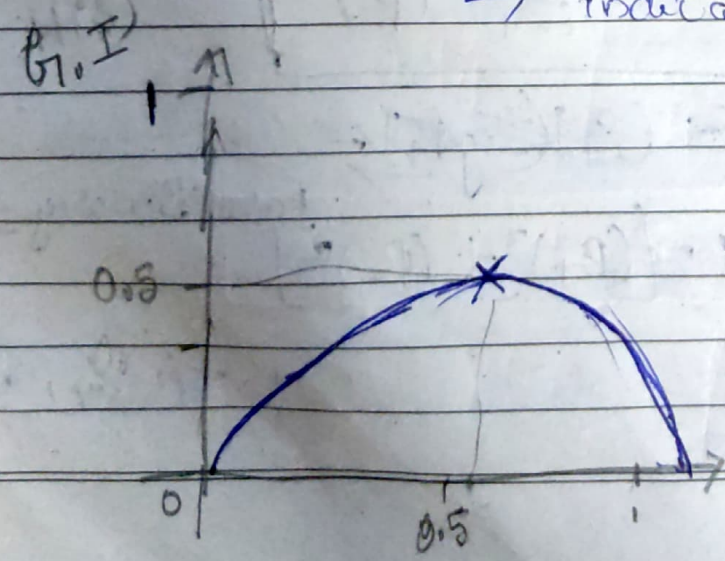
* $G_1.I.(C_2) = 1 - [(P_+)^2 + (P_-)^2]$

$= 1 - [(3/3)^2 + (0/3)^2]$

$= 1 - 1$

$G_1.I(C_2) = 0$ ⇒ pure split

↳ indicates complete pure split



* Multiclass classification problem

consider, 3 categories in o/p

$$H(S) = -(P_{C1})(\log_2 P_{C1}) - (P_{C2})(\log_2 P_{C2}) - (P_{C3})(\log_2 P_{C3})$$

$$Gr.I = 1 - [(P_{C1})^2 + (P_{C2})^2 + (P_{C3})^2]$$

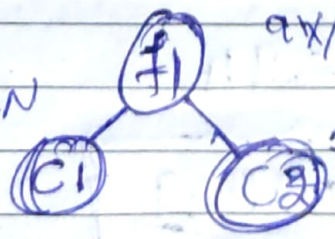
② Information gain :

↳ helps which feature to select to start the split.

$$Gain(S, f_1) = H(S) - \sum_{V \in Val} \frac{|S_V|}{|S|} H(S_V)$$

entropy parent node ← $H(S)$; $H(S_V)$ → entropy of child node

f_1	f_2	f_3	o/p
			9Y/5N
			6Y/2N
			3Y/3N



$H(S) = 0.94$

$H(S_V) =$ 0.81

* for root Node

* Root Node, $H(S) = -(P_+)(\log_2 P_+) - (P_-)(\log_2 P_-)$

$= \frac{-9}{14}\left(\log_2 \frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$

≈ 0.94

* $H(C_1) = -\frac{6}{8}\left(\log_2 \frac{6}{8}\right) - \frac{2}{8}\left(\log_2 \frac{2}{8}\right)$

≈ 0.81

* $H(C_2) = -\frac{3}{6}\left(\log_2 \frac{3}{6}\right) - \frac{3}{6}\left(\log_2 \frac{3}{6}\right)$

$H(C_2) \approx 1$

for first feature

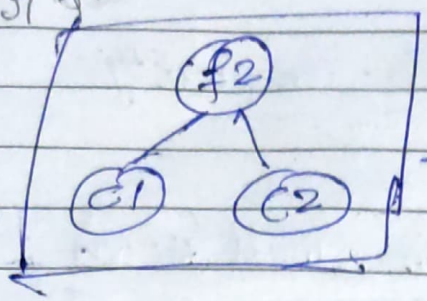
For categorical $\Rightarrow$ $Gain(S, f_1) = H(S) - \sum_{v \in values} \frac{|S_v|}{|S|} H(S_v)$ (entropy of each value)

$= 0.94 - \left[\frac{8}{14}(0.81) + \frac{6}{14}(1)\right]$

$\approx 0.94 -$

For First split $\Rightarrow$ $\boxed{Gain(S, f_1) = 0.049}$

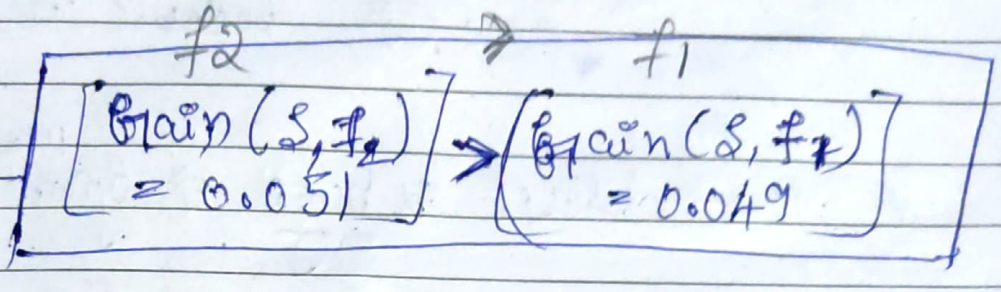
for first split



Consider $G_r(S, f_2)$.

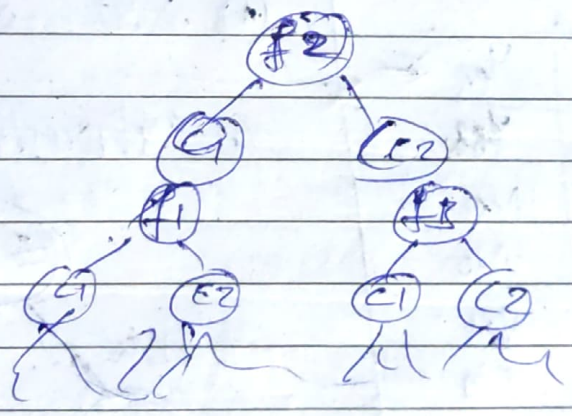
⇒ Information gain = 0.051

* Here,



Conclusion:
We need to start splitting by using f_2 feature.

we can do



d. classification techniques

① Entropy
② Gini impurity
③ Information gain

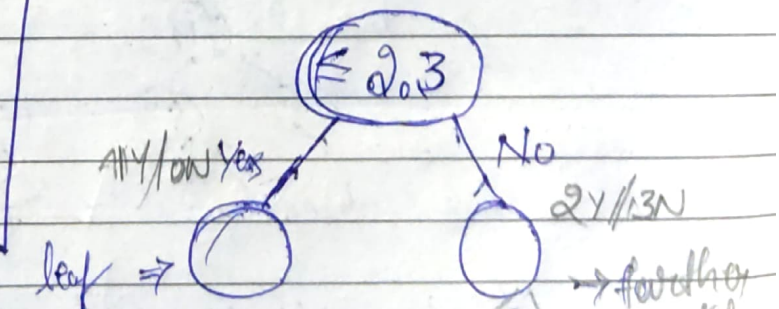
Q) Entropy v/s gini impurity

* when data set is small (10,000) ⇒ use Entropy
  as we use "log" because Time ↓↓.
* when dataset is huge ⇒ use gini impurity
  as we use "simple maths"

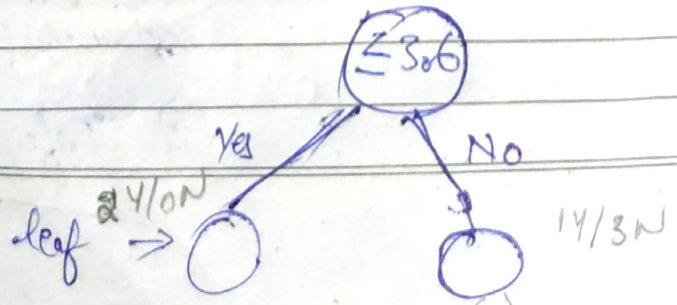
Q) what if my features are in continuous?

f1	o/p
2.3	Yes
3.6	Yes
4	No
5.2	No
6.7	Yes
7.8	No

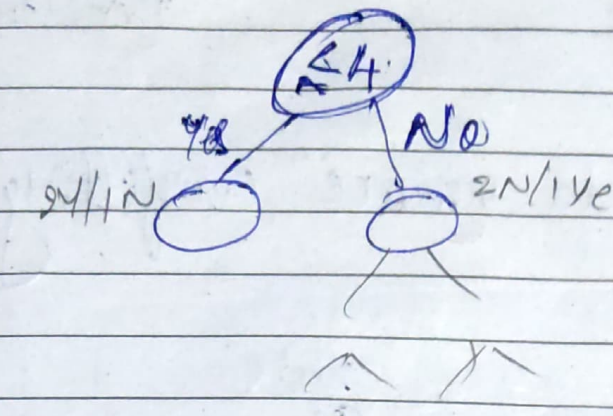
(1) sort the feature f1 (internally done by)
(2) threshold = 2.3 first value



(3) threshold = 3.6



⑤ threshold = 4



↓

and Information gain is Calculated which ever value ~~feature~~ has the highest gain that will be selected then further splitting takes place 11/4

Disadvantage:

① Time. Complexity ↑↑ when dataset is huge.

↓ solution

we have some parameters in Sklearn we can use them.

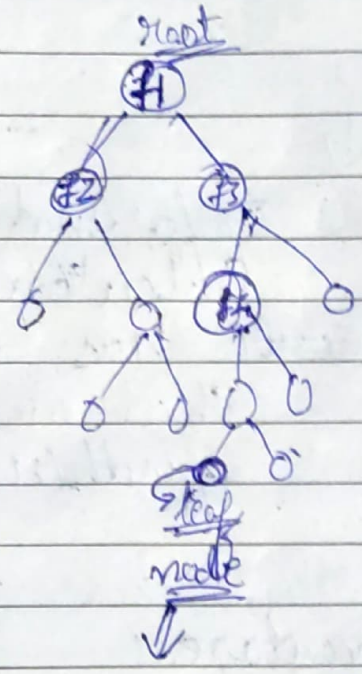
* Post pruning and pre pruning in Decision Tree

⇓

Used to "reduce overfitting".

Training Data

f_1	f_2	f_3	o/p



split till we get leaf node

represents results like Yes/No

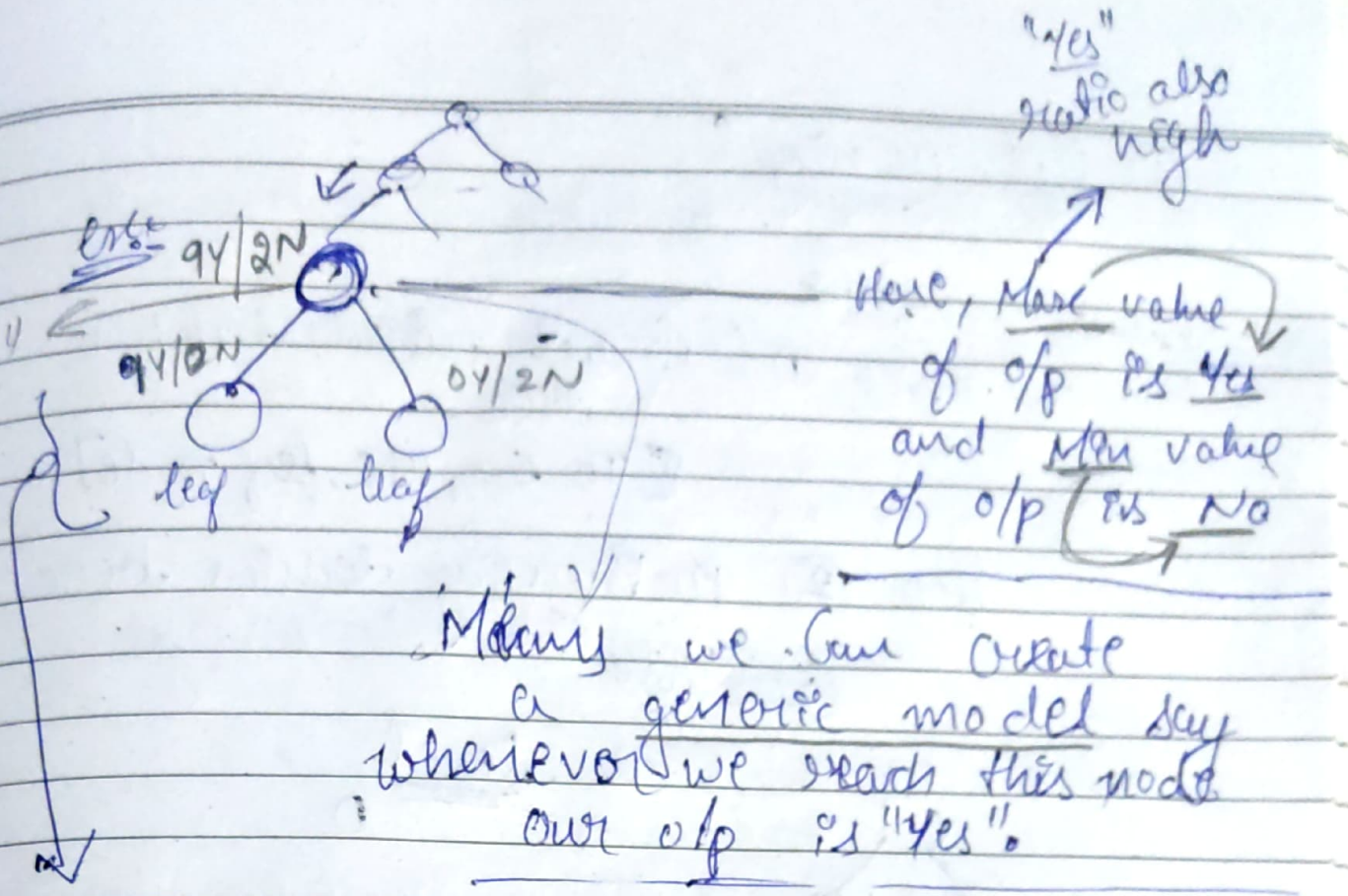
Q.) whenever, we split decision tree till depth / leaf node it leads to "overfitting"

↳ low bais & High variance

i.e Training data → Acc ↑↑

Test data → Acc ↓↓

In case of splitting till depth.



Means we can create a generic model say whenever we reach this node our o/p is "Yes".

Here we are further split training data

& Model is "by" "learning" training data

Aim: To build "generic model"

↓

low bais & low variance

[Train Acc ↑↑] [Test Acc ↑↑]

Solution ↓

now to reduce overfitting

① Post pruning.

② Pre pruning..

→ Cutting

① Post pruning :

steps

step ① Construct entire decision tree.

[To complete leaf node]

step ② Pruning decision tree.

level

0

1

2

3

4

Can cut, prune

(or)

prune

*Max depth = 4

*tell max depth = 3

[one of the parameter for cutting]

* Suitable for smaller dataset.

Parameters in sklearn

① Criterion : {"gini", "entropy", "log_loss"}

↓

Cost function of logistic regression

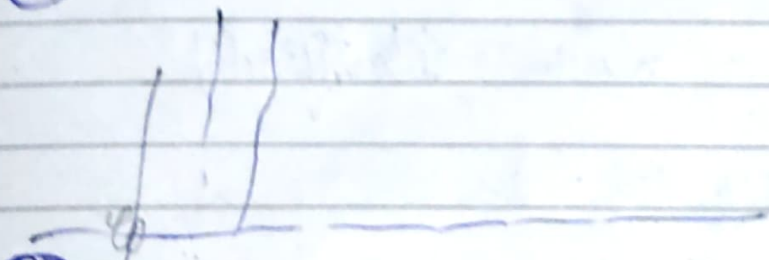
② Splitter : {'best', 'random'}

↓

Take most gain and split

→ Randomly split

③ max_depth



④ Hyperparameter :

Using GridSearchCV and RandomizedSearchCV.

↑

Step ① Hyperparameter Tuning to select best parameters.

② Train model using best parameters.

* Implementation using python

(1) Segregate independent and dependent features.

(2) Train Test split

Do not do standard scaling

(3) Decision Tree

> from sklearn.tree import DecisionTreeClassifier

> classifier = DecisionTreeClassifier()

> classifier.fit()

can display tree

> tree.plot_tree(classifier, filled=True)

> Post-pruning:
[based of depth]

> class = tree(criterion='gini', max_depth=2)
from observation

④ Prediction

⑤ accuracy_score() classification report

Pre pruning + Hyperparameteric Tuning

① > parameters

> from sklearn.model_selection import GridSearchCV

> clf = GridSearchCV(classifier, param_grid = parameters, scoring = 'accuracy')

> clf.fit(~~x_train, y_train~~)

> clf.best_params_

> clf.best_score_

> ypre = ——

>

regression problem → o/p value are continuous.

* Decision Tree regressor

DT classifier → o/p values are categorical.

① Entropy.
② Gini impurity } Measur of impurity
③ Information gain.

DT Regressor

① Variance reduction.
② Variance.

* Dataset [regressor]

Exp	Career gap	o/p Salary (Continous value)
2	Yes	40K
2.5	Yes	42K
3	No	52K
4	No	60K
4.5	Yes	56K

$\bar{y} = 50K$ (average / mean)

↳ Now, we have to select the feature to split that using feature Can be done using 'Information gain'

* Consider, 'Exp' as float. feature (continuous value)

Threshold = 2 [40K, 42K, 52K, 60K, 56K]

① Exp ≤ 2 — variance = 60.8 [4+2] ← Calculated beyond

Yes → [40K]

No → [42K, 52K, 60K, 56K]

② Threshold = 2.5

≤ 2.5

Yes → [40K, 42K]

No → [52K, 60K, 56K]

③ Threshold = 3

reduction should be more ↑

* Final aim: To Calculate "Variance reduction".

Variance (or) more

$$= \frac{\sum_{i=1}^{n} [y_i - \text{mean}]^2}{n}$$

Now, calculate variance for root, childs

① for threshold 2

$(y_i - \bar{y})^2$ $\left[\bar{y} = 50\right]$

* Variance (root) $= \frac{1}{5}\left[(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\right]$

$= \frac{1}{5}[100 + 64 + 4 + 100 + 36]$

$= 60.8$

* Variance (left) $= \frac{1}{1}[(40-50)^2]$

$= 100$

* Variance (right) $= \frac{1}{4}[(42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2]$

$= 51$

Variance reduction $= \text{Var(root)} - \sum w_i \text{Var(child)}$

$= 60.8 - \left[\frac{1}{5} \times 100 + \frac{4}{5} \times 100\right]$

Variance reduction $= 0$

(a) for threshold 2.5

Variance (root) = 60.8

variance (left) = 82

variance (right) = 46.66

Variance reduction = $60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66\right]$

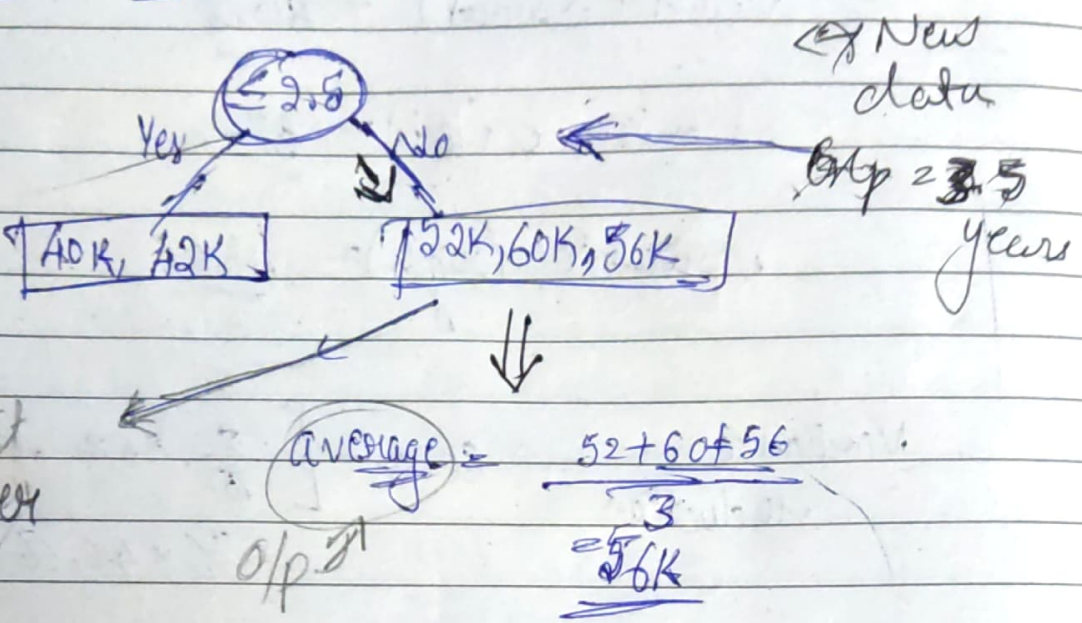
Variance reduction = 0.004

* Q) which split to use ?

$\Rightarrow$ $VR_{T=2.5} > VR_{T=8}$ — More Variance reduction happening

$\therefore$ we will use Threshold = 2.5 split at feature root

→ selected split:



* Implementation with Python

Only 'criterion' parameter will change compared to classifier.

{'squared_error', 'friedman_mse', 'absolute_error', 'poisson'}

> df = df.sample(frac = 0.25)

① Segregate independent and dependent feature.

② train test split

③ Model training

> from sklearn.tree import
DecisionTreeRegressor

(4) Prediction
(5) Accuracy

> from sklearn.metrics import
r2_score

r2 = r2_score(y_test, y_pred)

Hyper parameter tuning

* include ignore warnings

(7) Best parameters selection

> from sklearn.model_selection import
GridSearchCV

* In regression problem we have "scoring parameter different"

> regressorCV = GridSearchCV(regressor, estimator
param_grid = parameters,
scoring = 'neg_mean_squared_error')

> regressorCV.fit(—, —)

> tell.plot_tree(regressor, filled=True)

* Support vector ~~classifier~~ Machines

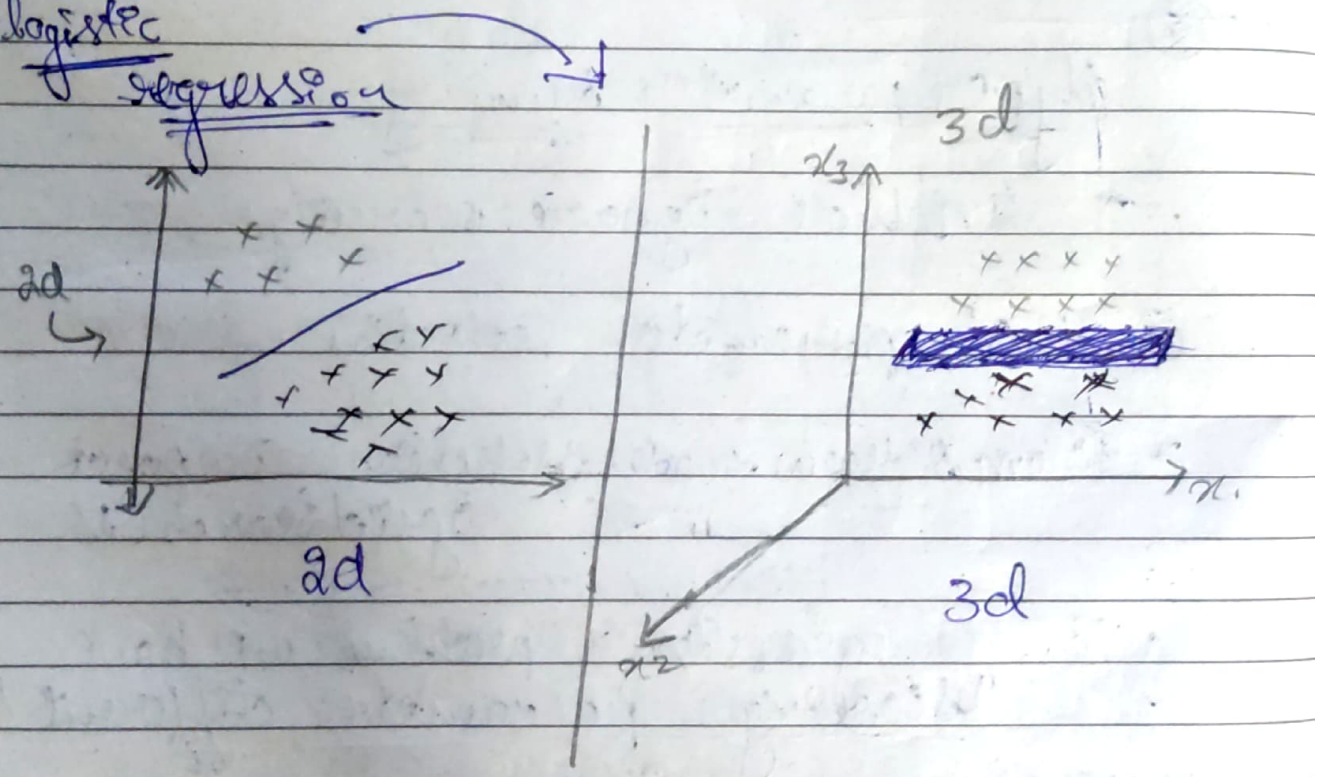
* Support vector Machine

2 Types:

(SVC) ① Support vector classifier → classification

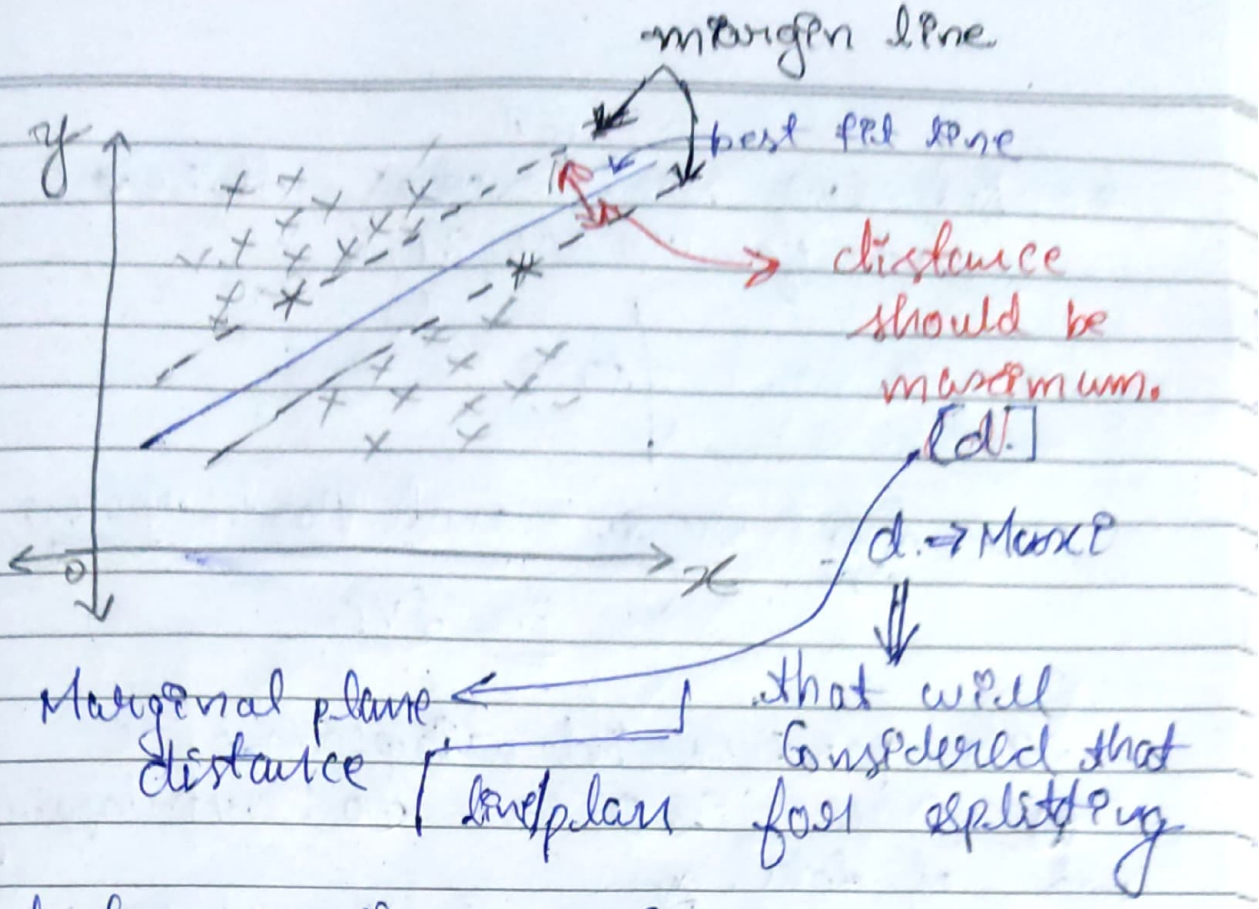
(SVR) ② Support vector regressor → Regression

* logistic regression



① Support vector classifier (SVC):

↳ classification problem.

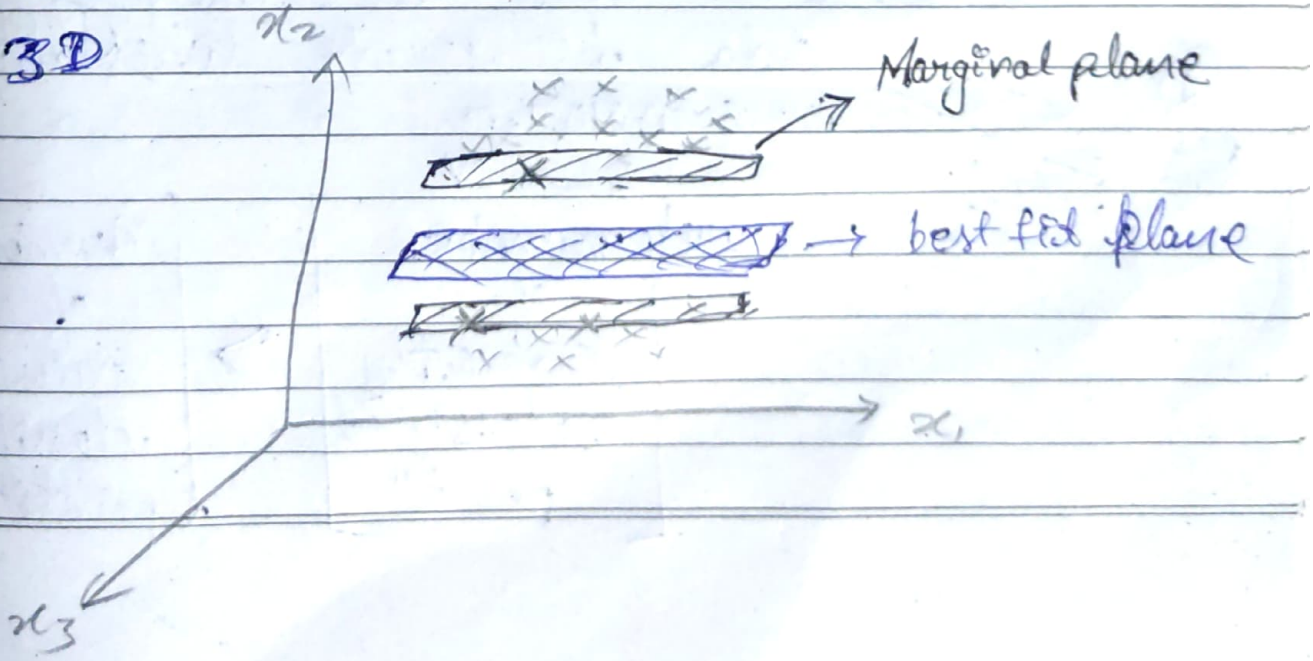


* distance is maximum.

⇒ Aim: To create best fit line among with we create margin line. which passes through nearest points.

Called "support vectors"

* 3D



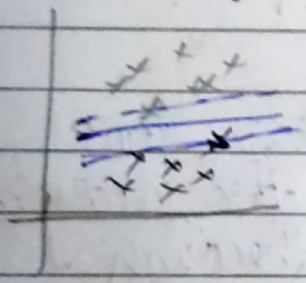
* But in real world data there will be lot of overlapping.

↓ so

* Soft margin and Hard margin in SVC

→ In previous 2d diagram, the error is 0 as no overlapping of data.

(1) Hard



" None of the data points are miss-classified "

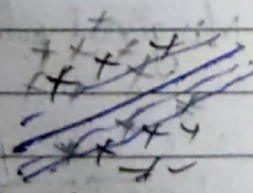
↓

we say as "Hard margin".

↓

almost impossible in our data.

(2) Soft



→ when there is overlapping of data we create lines with some error.

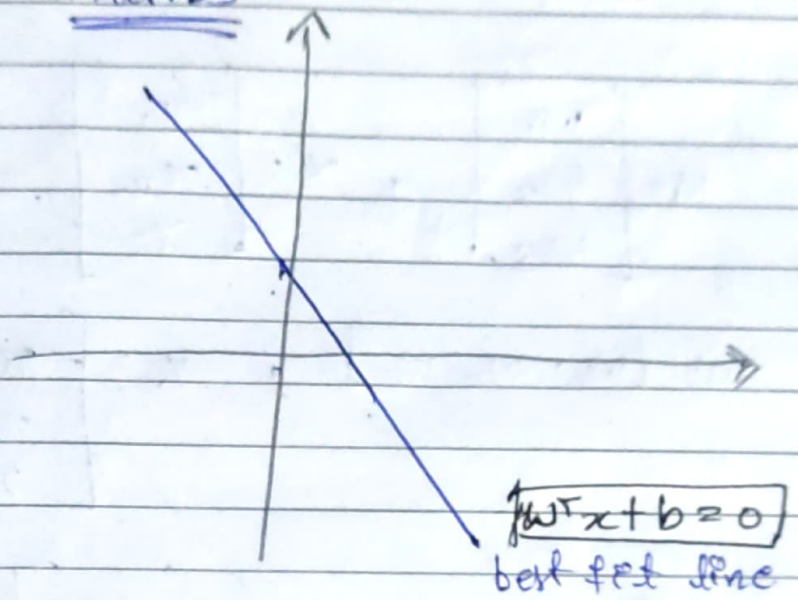
↓

but [number of correctly classified points] > [Number of incorrectly classified points]

Soft margin :- some data point are mass classified

we take [Error] as hyperparameter

SVC maths



equation of a straight line

$y = mx + c$

(or)

$h_\theta(x) = \theta_0 + \theta_1 x$

$\Leftrightarrow$ (we also write) $ax + by + c = 0$

$by = -ax - c$

$y = \left(\frac{-a}{b}\right) x - \left(\frac{c}{b}\right)$

m [slope] c [intercept]

multiple linear regression

↓ 3 i/p feature

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

research papers ↓

$$y = b + [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

⇒ vector form,

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$w^T = [w_1 \; w_2 \; w_3], \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Dot product,

$$w^T \cdot x = [w_1 \; w_2 \; w_3] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boxed{w^T \cdot x = w_1 x_1 + w_2 x_2 + w_3 x_3}$$

$$\boxed{y = w^T x + b} \Leftrightarrow y = mx + c$$

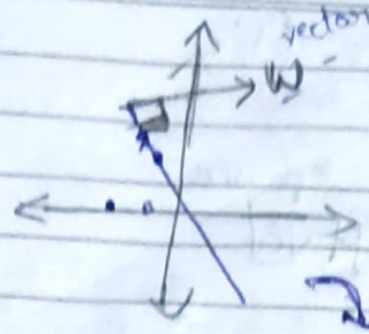
$$ax + by + c = 0$$

$$\boxed{w^T x + b = 0}$$

[d] Scalar → only magnitude

[$\vec{v}$] vector → magnitude and direction ? represents

* Slope

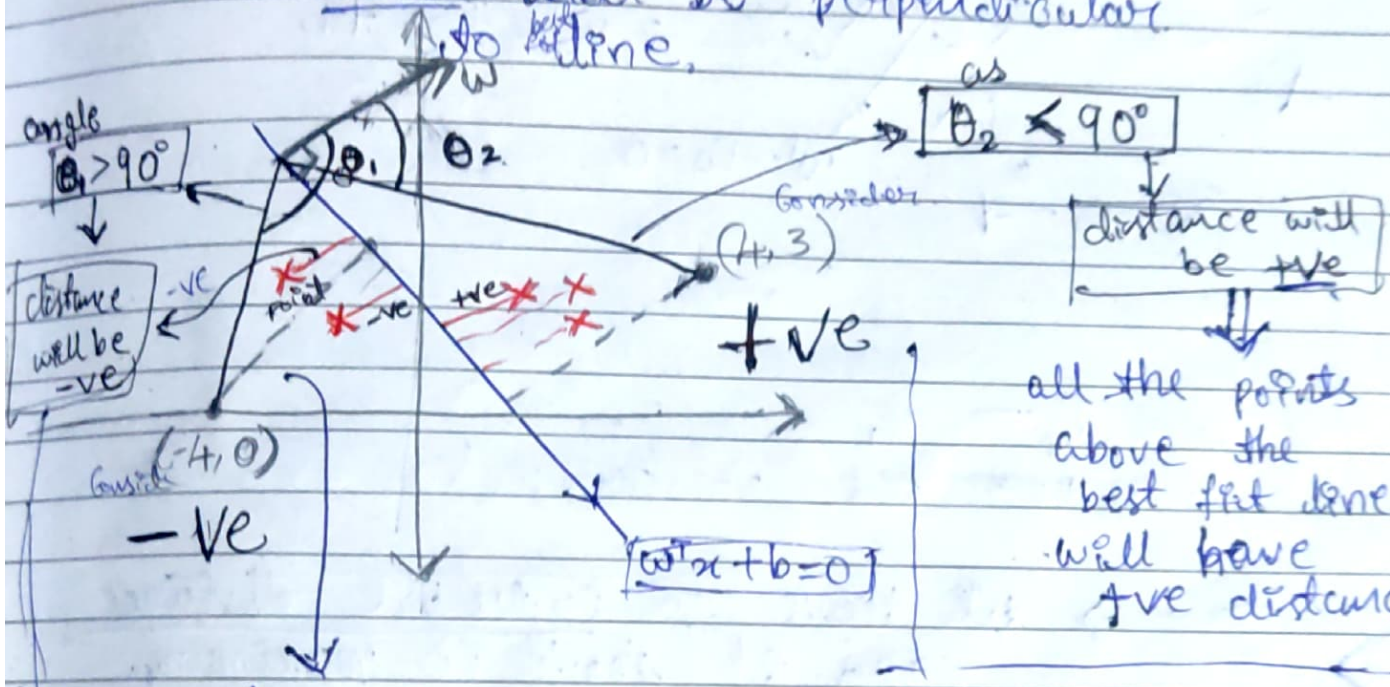


slope = $\frac{\Delta y}{\Delta x}$

-ve slope → line [pointing downwards]

which has both magnitude and direction

* vector will be perpendicular to line.



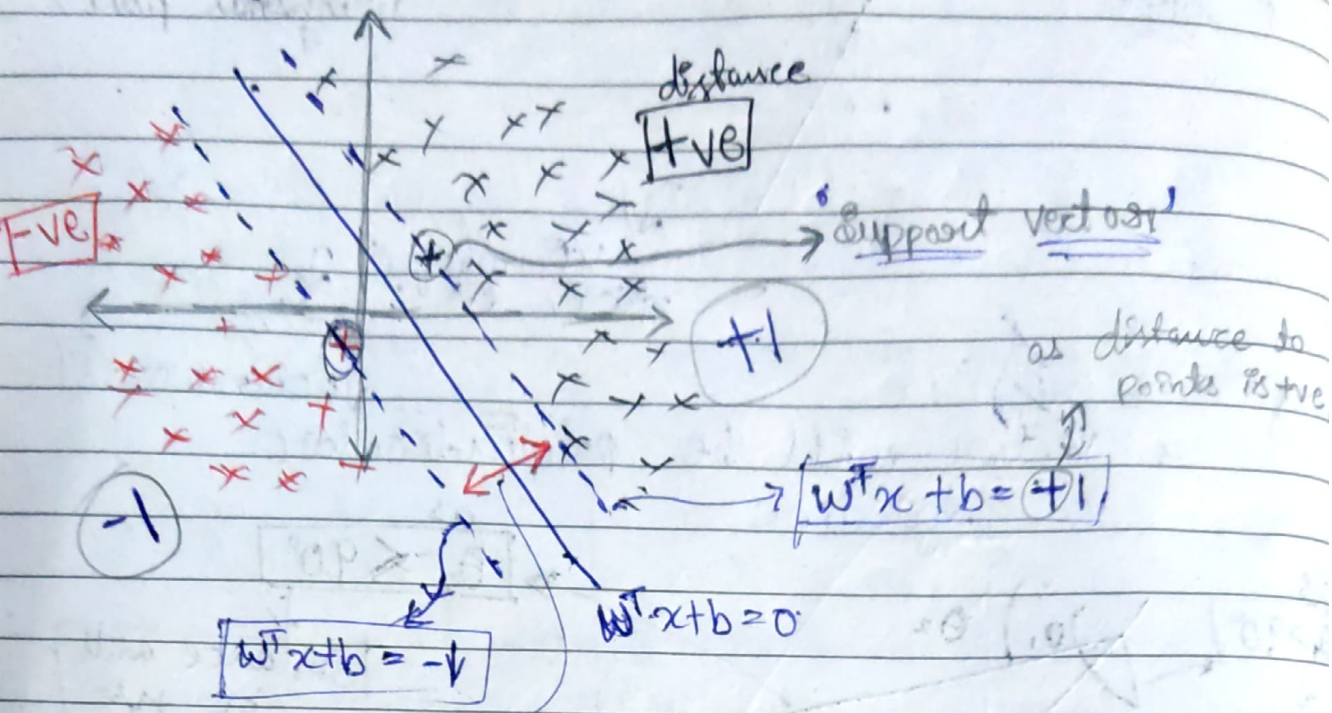
* how to Calculate the distance ?
↳ using vector

magnitude will be negative. ⇒ all the points below best fit line will have -ve distance

-ve Magnitude

represents direction.

* Marginal plane in SVC



points
Point, $P_1 \Leftarrow x_1 \Rightarrow (x_1, x_2)$ coordinates
$P_2 \Leftarrow x_2 \Rightarrow (x_1, x_2)$

⇒ we have to calculate distance and it should be maximum.

$$w^T x_1 + b = +1$$
$$w^T x_2 + b = -1$$
$$(-) \quad (-) \quad (+)$$

$$w^T (x_1 - x_2) = 2$$

* divide by ||w|| magnitude

Maximize

$$\Rightarrow \left(\frac{w^T}{\|w\|}\right)(x_1 - x_2) = \left(\frac{2}{\|w\|}\right)$$ distance b/w marginal planes.

$\hat{w}$

* Work of unit vector, $\hat{x} = \frac{\vec{x}}{\|\vec{x}\|}$

$\Rightarrow$ aim $\Rightarrow$ To get maximize the distance b/w marginal plane.

* Cost function

Maximize $\frac{2}{\|w\|}$ $\Rightarrow$ distance b/w marginal plane.

by changing w, b slope, intercept

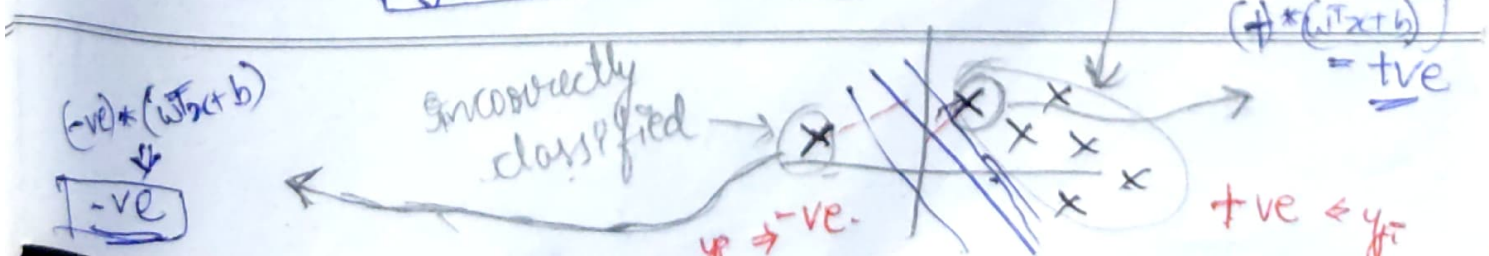
+ Constraint such that

Hard margin $\Rightarrow$ $y_i = \begin{cases} +1, & \text{if } w^T x + b \geq 1 \\ -1, & \text{if } w^T x + b < 1 \end{cases}$

$\Downarrow$

* For all Correctly classified data point

$$y_i * w^T x + b \geq 1$$



Modified Cost function of SVC

Can be written

$$\max_{w,b} \frac{2}{\|w\|} \iff \min_{w,b} \frac{\|w\|}{2}$$

We write in this form because in other algo we minimize cost function.

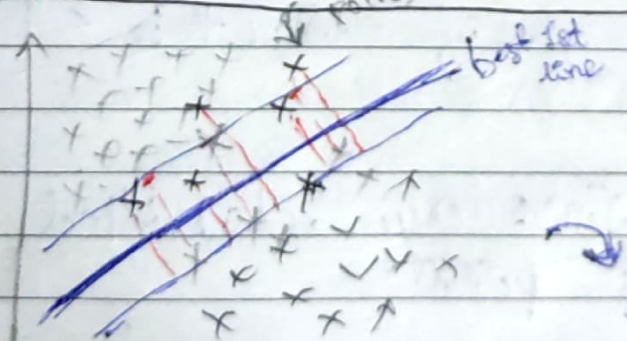
* what happens when we have overlapped datapoint/2 How many mis-classified datapoint

Cost function of Soft margin SVC

$$\text{Cost fun} = \min_{w,b} \frac{\|w\|}{2} + C \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

Misclassified points

Hinge loss



we don't have as we have the correct classification

Some error also there so it can classify remaining points

ξ_i → hyperparameter

if $\xi = 0$ then it says correct data points can remain

if $\xi > 0$ then it says incorrect data points can remain

if cross threshold then we change hyperparameters

{ which say how many points we can consider for classification }

beta $\sum \xi_i$ → summation of the distance of incorrect data points from the marginal plane.

If the threshold cross then we will change the best fit line

$\epsilon = \frac{1}{\lambda}$ → in logistic regression also

*Implementation of SVC with python

① Data ex. / created → import make_classification

② train_test_split

③ Model Training

> from sklearn.svm import SVC

> svc = SVC(kernel = 'linear')

> svc.fit(x_train, y_train)

> svc.coef_

④ predictions

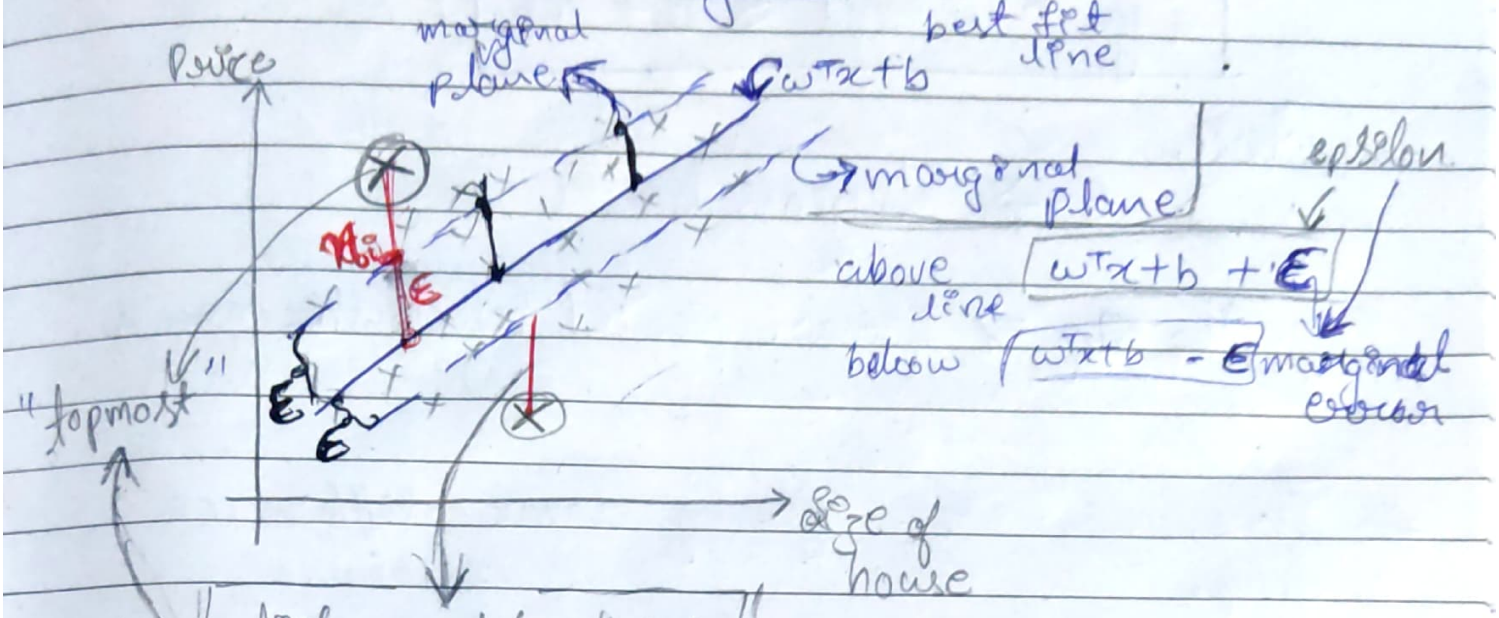
⑤ Performance metrics.

⑥ Hyperparameter tuning

grid_cv = GridSearchCV(SVC(), param_grid = para, refit = True, verbose = 3)

↑ all info of training (each steps)

* Support vector regression [SVR]



distance b/w point and "lower most" marginal plane

Cost function

$$CF \rightarrow \min_{w,b}\left(\frac{\|w\|}{2}\right) + C\sum_{i=1}^{n} \eta_i$$

$\downarrow$

Hinge loss

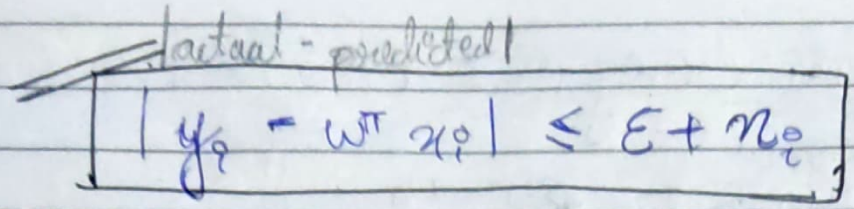
Constraint

actual - predict

$$|y_i - w^Tx| \leq \varepsilon + \eta_i$$

η_i → error parameter

we created a best fit line and marginal line. and make sure that all points in marginal line. so make distance between marginal line more.



where,

$\varepsilon \Rightarrow$ Marginal error

$\eta_i \Rightarrow$ Error above / below the margin.

* Here, we are giving some difference gap.

* Implementation

① Create dataset

↳ from sklearn.datasets import make_regression

x, y = make_regression (n_samples = 1000, n_features = 2, n_targets = 1, noise = 3.0)

② Train test split

③ Model for any reg

> from sklearn.svm import SVR

④ Prediction.

⑤ Performance metrics.

> from sklearn.metrics import r2_score

⑥ Hyperparameter tuning

> from sklearn.model_selection import GridSearchCV

> parameters = {
"epsilon": [0.1, 0.2, 0.3],
}

> grid_cv = GridSearchCV(SVR(), param_grid = ____, refit = True, scoring = 'neg_mean_squared_error', verbose = 3)

> grid_cv.fit(x_train, y_train)

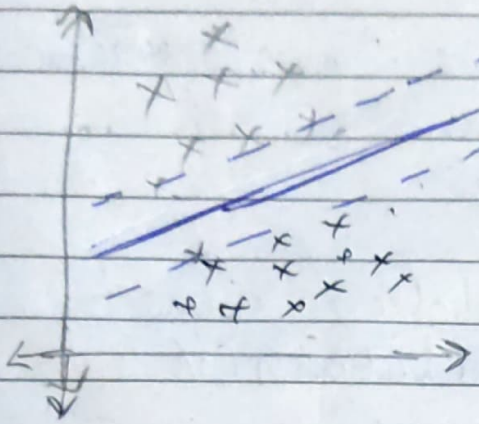
> grid_cv.best_param_

> y_pred = grid_cv.predict(x_test)

> print(r2_score(y_test, y_pred))

(tomorrow)

* SVM Kernels



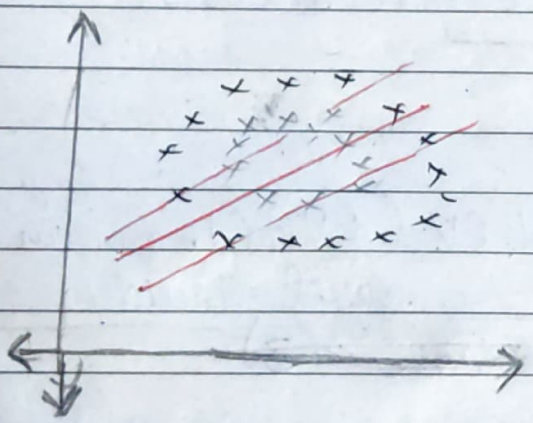
If there were any outliers then we handled those through hyperparameter

Linear SVC

↓

Straight ~~best fit~~ + line

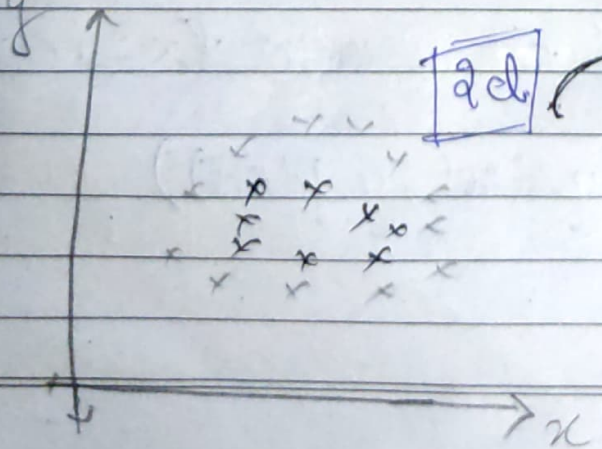
Marginal lines



→ Cannot use linear SVC.

→ More overlapped

Solve use SVM Kernels to solve it



⇒ Transformation ⇒

↓

using Mathematical Formula

Applied formula on x to make 1d as 2d

$Z = x^2$

Consider Dataset

x	y	$I = x^2$
2	Yess	4
3	No	9
4	Yess	16
-	yes	-
-	-	-
-	-	-

linear SVC

$I = x^2$

1d point

SVM Kernel

Data transformation technique

means applying mathematical formula

Cannot create belt for line.

Formula on x and y

$Z = f(x, y)$

Z is obtained by apply Maths formula

Top view 2d

view

Z

3d

Now we can easily classify datapoints after transforming into 3d.

then we apply "linear SVC"

we may get nearly 100% accuracy.

There are 3 types of Transformation given

① Polynomial Kernel.
② R.B.F Kernel. } ⇒ Transformation
③ Sigmoid Kernel.

↓ it because of "maths formula"

* This basically convert lower dimension to higher dimension and easily separate points using a linear SVC.

* ~~Implementation~~

① Polynomial Kernel

$$f(x, y) = (x^T y + c)^d$$

where, c ⇒ Constant
d ⇒ degree of polynomial
depends on no of o/p category.
if 2 category then → d = 2

* Consider

$x = [x_1, x_2]$

$y = [x_1, x_2]$

then

$\Rightarrow x^T \cdot y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot [x_1 \quad x_2]$

$= \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 \end{bmatrix}$

↓

created 3 new features before we having $(x_1 \& x_2)$

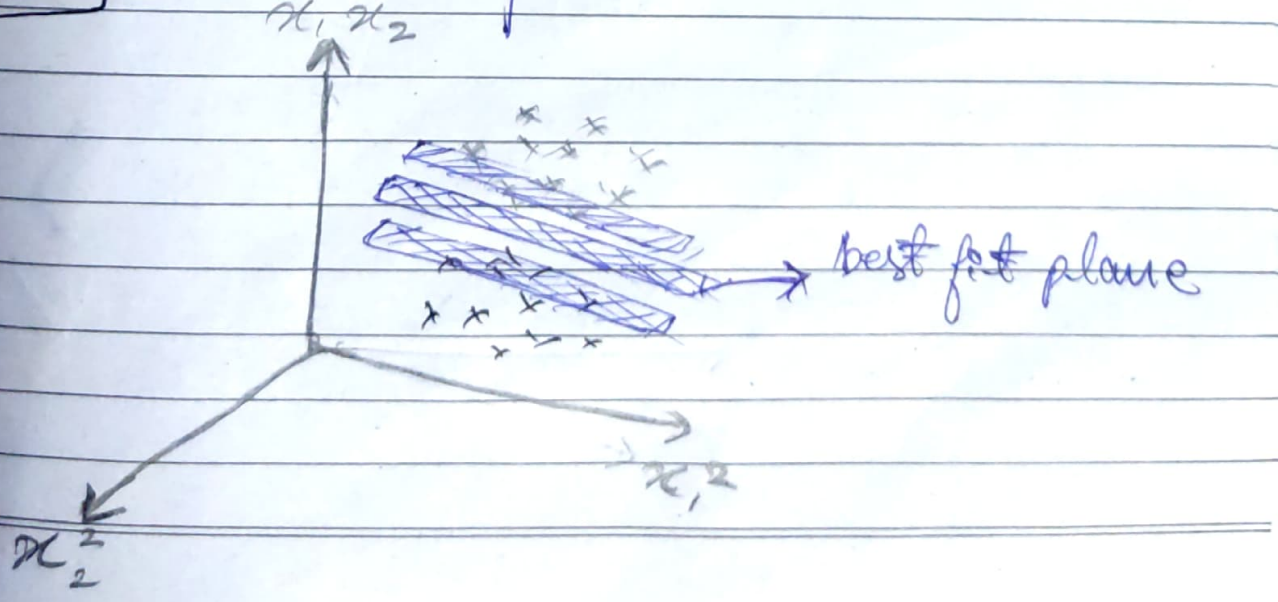
2d

$x_1, x_2 \mid y$

Transformation ⇓ Transforming [polynomial kernel Transformation]

$x_1^2, \quad x_1 x_2, \quad x_2^2 \mid y$

3d



② RBF Kernel :

$$K(x_1, x_2) = \exp\left(\frac{|x_1 - x_2|^2}{2\sigma^2}\right)$$

③ Sigmoid Kernel :

$$\delta(x) = \frac{1}{1 + e^{-x}}$$