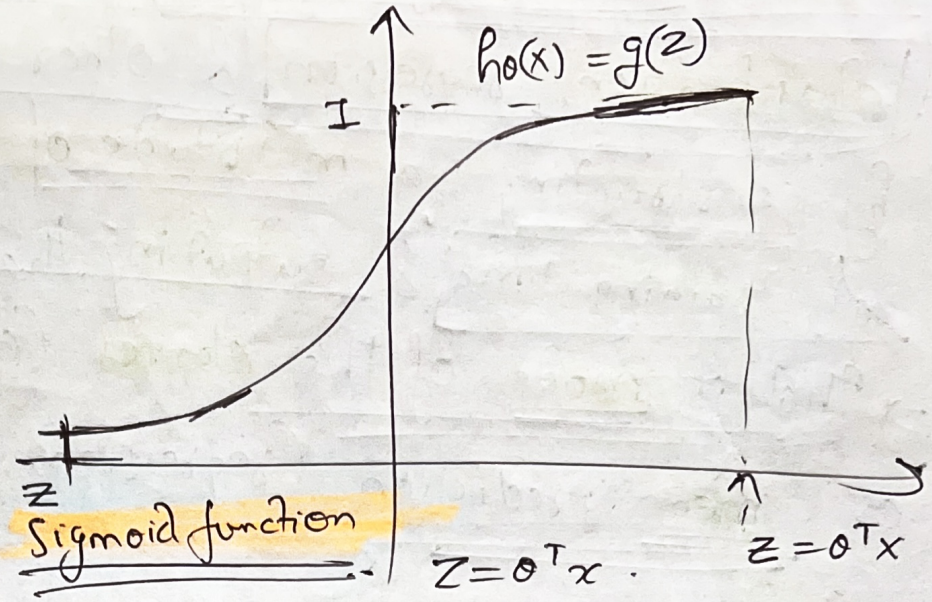
Logistic Regression

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

$e^{\left(\frac{1}{\theta^T x}\right)} \to 0$

$\theta^T x \to \infty$



Sigmoid function

$z = \theta^T x$

$\theta^T x \gg 0$

$\theta^T x \ll 0$

if $y = 1$, we want $h_\theta(x) = 1$, $\theta^T x \gg 0$

if $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

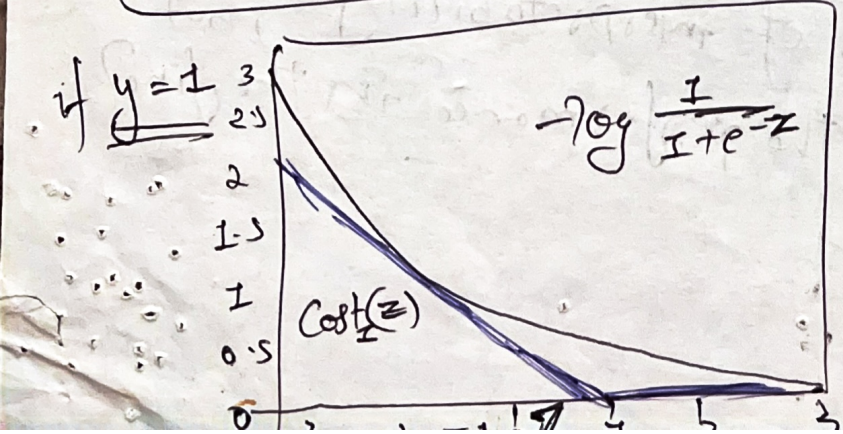
Cost function of logistic regression

$$-h_\theta(x)\left(\log(h_\theta(x))\right) + (1-h_\theta(x))\left(\log(1-h_\theta(x))\right)$$

$$= -\left(y \log h_\theta(x)\right) + (1-y)\left(\log(1-h_\theta(x))\right)$$

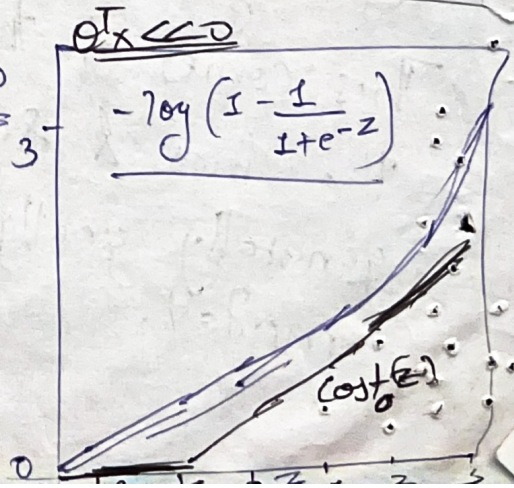
$$= \left[-y \log \frac{1}{1+e^{-\theta^T(x)}} + (1-y) \log\left(1 - \frac{1}{1+e^{-\theta^T(x)}}\right)\right]$$

if $y = 1$



if $y = 0$

$\theta^T x \ll 0$



SVM It is a powerful yet flexible supervised ML algorithms, which are used for classification & ~~regression~~.

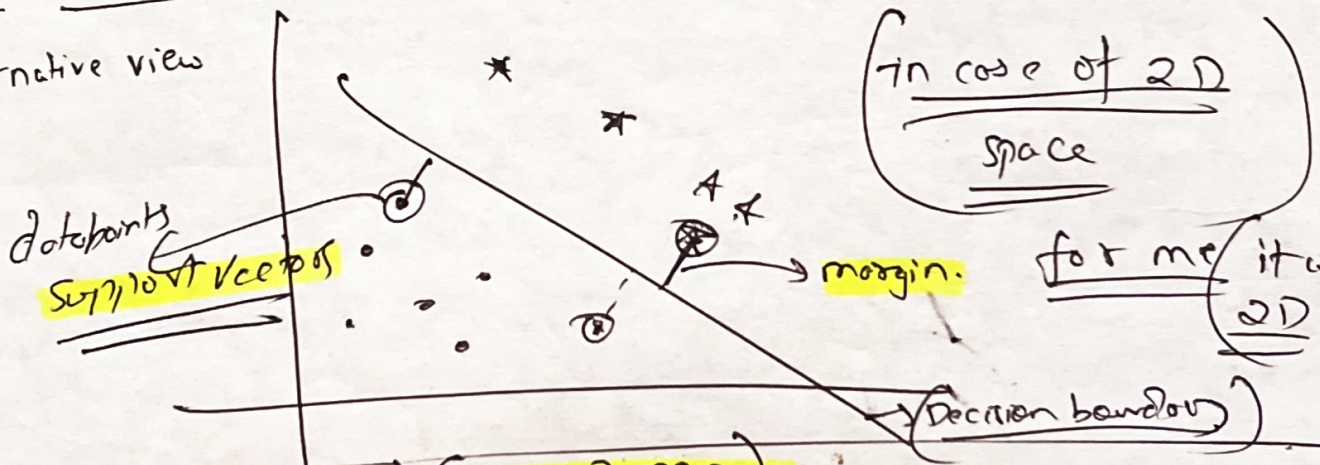
Working of SVM (basically a representation of different classes in a hyperplane)

Hyperplane it is generated in ~~an iterative~~ manner by SVM so that the error can be minimized.

Goal of SVM is to divide the datasets into classes to find a Maximum Marginal Hyperplane (MMH).

Support vectors: points close to hyperplane. Separting will be define using these.

Alternative view



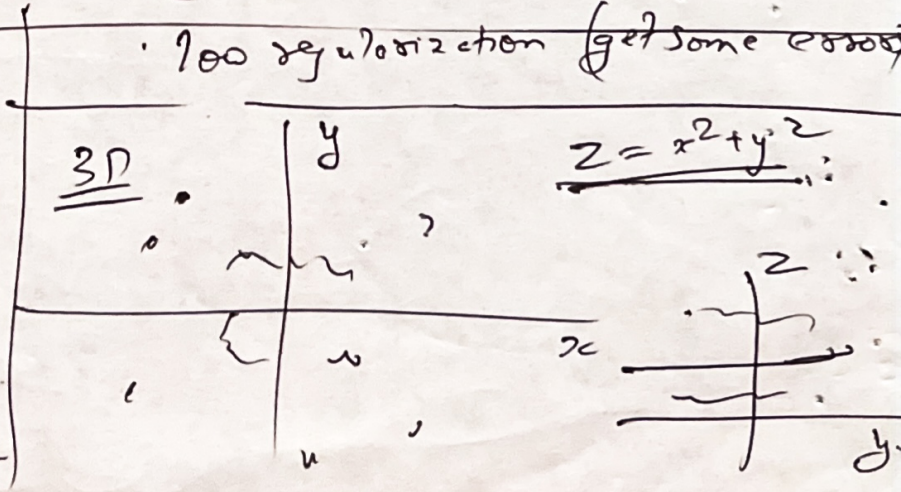
in case of 2D space

for me it was 2D

3D case (Hyperplane) (for N Dimen)

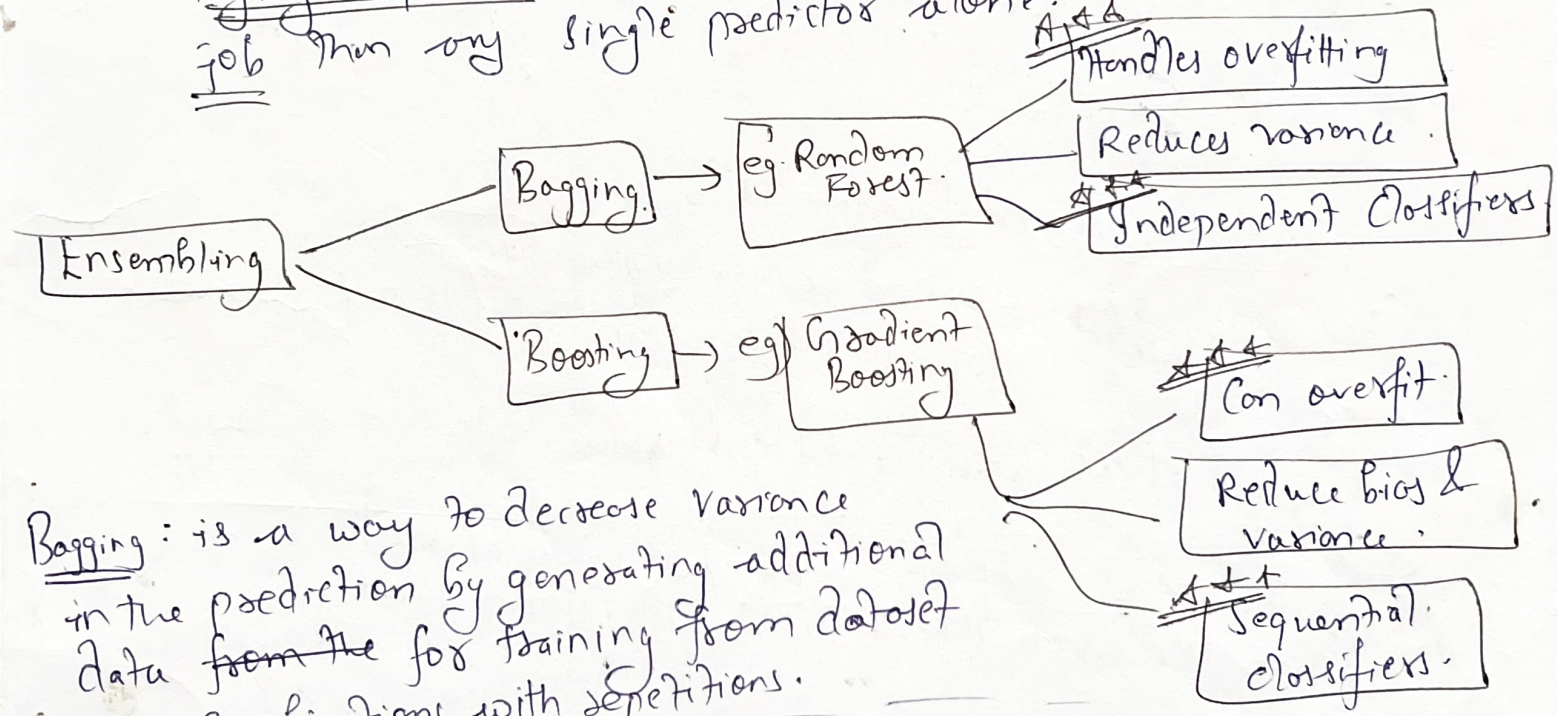
Hyperpm (gamma & regularization) (C)

too regularization get some error



Ensembling When we try to predict the target variable using any ML technique, the main cause of diff b/w actual & predicted values are. noise, variance & bias.
Ensembling helps to reduce these factors (except noise).

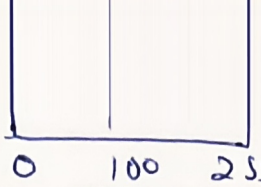
Define: An ensemble is a collection of predictors which come together to give a final prediction.
The reason we use ensembles is that many different predictors trying to predict some target variables will perform a better job than any single predictor alone.



Bagging: is a way to decrease variance in the prediction by generating additional data ~~from the~~ for training from dataset using combinations with repetitions.

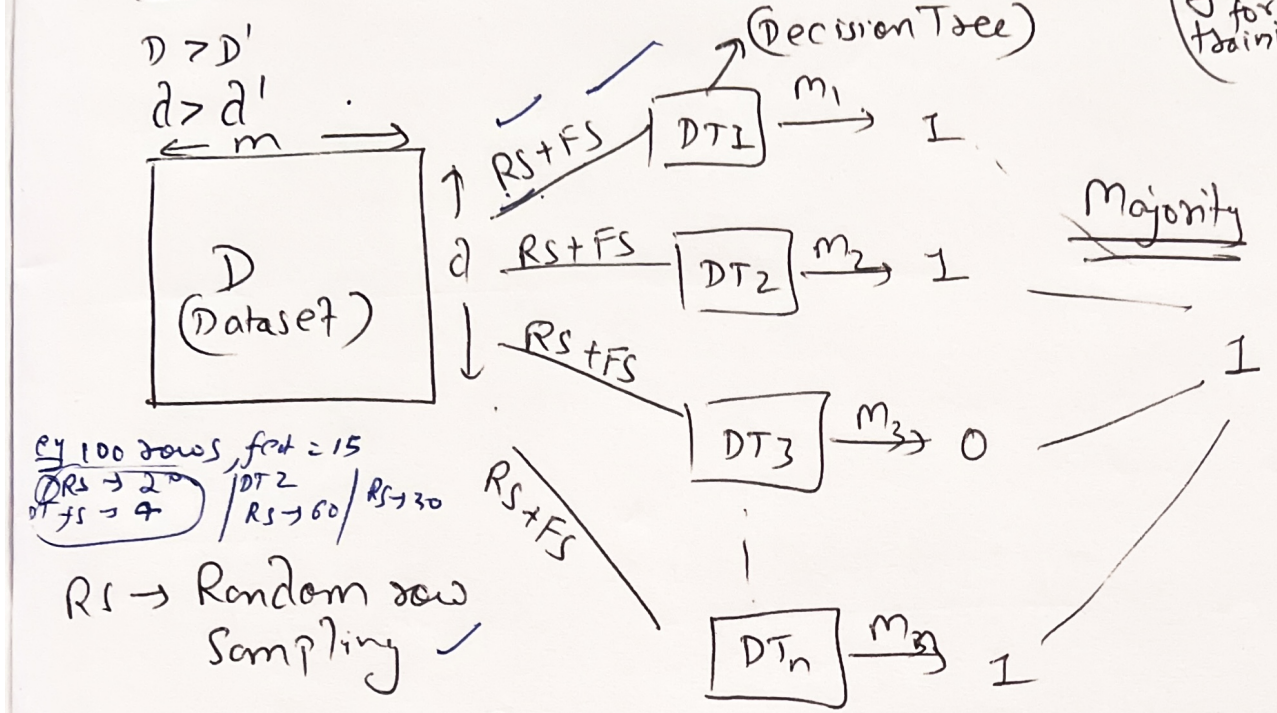
Boosting: It is an ensemble technique in which the predictors are not made independently, but sequentially.
This technique employs the logic in which the subsequent predictors learn from the mistakes of the previous predictors.

Bagging: It is simple ensembling technique in which we build many independent predictors/models & combine them using averaging technique.



MNIST (Handwritten digits)

'Random Forest' (Bagging)



eg 100 rows, feat = 15
RS → 20
FS → 4
DT2
RS → 60
RS → 30

RS → Random row Sampling

FS → Feature Sampling

m, d → no. of cols, no. of rows.

Note: In case of Classification Problem, the output is Majority vote or mode.

whereas;

In Regression Problem, output is mean or median.

Note

In DT
① Low Bias
② High var.

(will be good for training set)

(not good for test data)

whereas;

In RF;
High var → gets converted to low var.

Random Forest or random decision forests are an ensembling learning method for classification, regression & other tasks that operate by constructing a multitude of decision trees at training time & outputting the class.

⇒ That is mode of the classes (classification)

⇒ or mean prediction (regression) of the individual trees.

Random Forest corrects for decision trees' habits of overfitting to their training sets.

Note:

In general, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training set i.e. have low bias & high variance.

Random Forest are a way of averaging multiple deep decision trees, trained on different parts of the same training data set, with the goal of reducing the variance.

This comes at the expense of a small increase in the bias & some loss of interpretability, but generally greatly boosts the performance in the final model.

Random Forest (Hyperparameter)

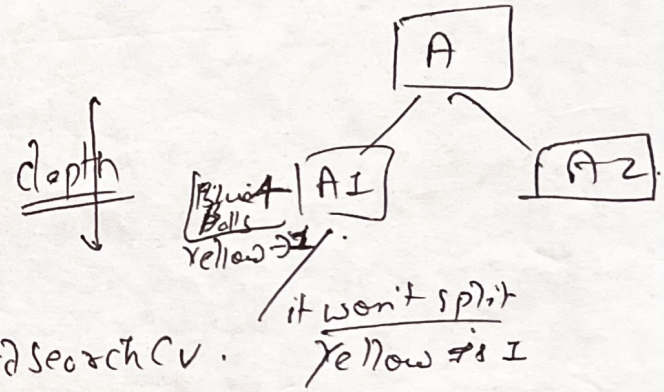
Stratification ??

① n_estimators (no. of decision trees)
default 10

② max_depth (specifies max. depth of each tree).
default None

③ min_samples_leaf (it specifies the min. no. of samples required to be at a leaf node).
default = 1

④ min_samples_split (min. samples required to split an internal leaf node)
default = 2



Code:

```
from sklearn.model_selection import GridSearchCV.
from sklearn.ensemble. import RandomForestClassifier.
param_grid = ['max_depth' : [30, 50, 80],
              'max_features' : ____ ]

# Model
rf = RandomForestClassifier()
grid_search = GridSearchCV (estimator = rf,
    param_grid = param_grid, cv = 3, n_jobs = -1, verbose = 2)

# Fit to model
grid_search.fit (X, y)
grid_search.best_param_
```

X, y → (training dependent & independent variable)

```
max_depth = 80
max_feature = 2.
n_estimator = 100.
min_samples_split = 8.
criterion = None
```

```
rfc = RandomForestClassifier (  )
model_rf = rfc.fit (X, y)
y_pred_rf = model_rf.predict (X_test)
Kfold = StratifiedKFold (n_splits = 10)
results = cross_val_score (rfc, X, y, cv = kfold)
```

Random Forest

ex) of parameter of random forest: (are variable & thresholds used to split each node learned during training)

ex) of hyperparameter of rf:

(1) no. of decision trees in the forest (n-estimator)

(2) no. of features considered while splitting a node (max-features)

(3) max number of levels in each decision tree (max-depth)

(4) min number of data points placed in a node before the node is split (min-samples-split)

(5) min number of data points allowed in a leaf node (min-samples-leaf)

For ex (if min-samples-split = 5 & there are 7 samples at an internal node, then split is allowed - the split result in 2 leaves, one with sample of size 1 & another with 6 size samples. if min-samples-leaf = 2 then won't be allowed)

Parameter	Hyperparameters
Model parameters are learned during training - such as slope, intercept in linear regression	set by Data Scientist before training.

Decision Tree

Splitting measures (Starting from root node & further down by splitting the nodes)
- Gini Index
- Entrophy (Information gain)

Gini Index / Gini Impurity

measure the degree or probability of a particular variable being wrongly classified when it is randomly chosen.

But what is impurity?

If all elements belong to some class than it is pure.

Note: The degree of Gini Index varies from 0 to 1, where 0 denotes that all elements belong to a certain class or if there exits only one class.

Note

& 'I' denotes that the elements are randomly distributed across various class.

Note

A Gini index of 0.5 denotes equally distributed elements into some classes.

Formula of Gini Index

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2$$

where, P_i is the probability of an object classified to a particular class.

Note: We prefer attribute/feature with least Gini Index of root node.

Entropy

Formula for Entropy

$$E(S) = \sum_{i=1}^{c} -P_i \log_2(P_i)$$

→ Not preffered due to 'log' function as it increases computational complexity.

where; P denotes the probability & E(S) denotes entropy.

Note) we prefer attribute with least entropy at root node.

- Note [UDF]

User Defined functions make it possible for user to write it's own transformation.

```
from pyspark.sql.functions import udf
```