Ensemble Learning

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model

1. Basic Ensemble Techniques
2. 2.1 Max Voting
3. 2.2 Averaging
4. 2.3 Weighted Average
5. Advanced Ensemble Techniques
6. 3.1 Stacking
7. 3.2 Blending
8. 3.3 Bagging
9. 3.4 Boosting

Algorithms based on Bagging and Boosting

4.1 Bagging meta-estimator
4.2 Random Forest
4.3 AdaBoost
4.4 GBM
4.5 XGB
4.6 Light GBM
4.7 CatBoost

2. Simple Ensemble Techniques

2.1 Max Voting

The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

2.2 **Averaging**

In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

2.3 **Weighted Average**

This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction

2. Simple Ensemble Techniques

```
from sklearn.ensemble import VotingClassifier
model1 = LogisticRegression(random_state=1)
model2 = tree.DecisionTreeClassifier(random_state=1)
model = VotingClassifier(estimators=[('lr', model1), ('dt', model2)],
voting='hard')
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

```
model1 = tree.DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= LogisticRegression()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)

pred1=model1.predict_proba(x_test)
pred2=model2.predict_proba(x_test)
pred3=model3.predict_proba(x_test)

finalpred=(pred1+pred2+pred3)/3
```

3. Advanced Ensemble techniques

3.1 **Stacking**

Stacking is an ensemble learning technique that uses predictions from multiple models (for example decision tree, knn or svm) to build a new model. This model is used for making predictions on the test set. Below is a step-wise explanation for a simple stacked ensemble:

1. The train set is split into 10 parts.
2. A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.
3. The base model (in this case, decision tree) is then fitted on the whole train dataset.
4. Using this model, predictions are made on the test set.
5. Steps 2 to 4 are repeated for another base model (say knn) resulting in another set of predictions for the train set and test set.
6. The predictions from the train set are used as features to build a new model.
7. This model is used to make final predictions on the test prediction set.

3. Advanced Ensemble techniques

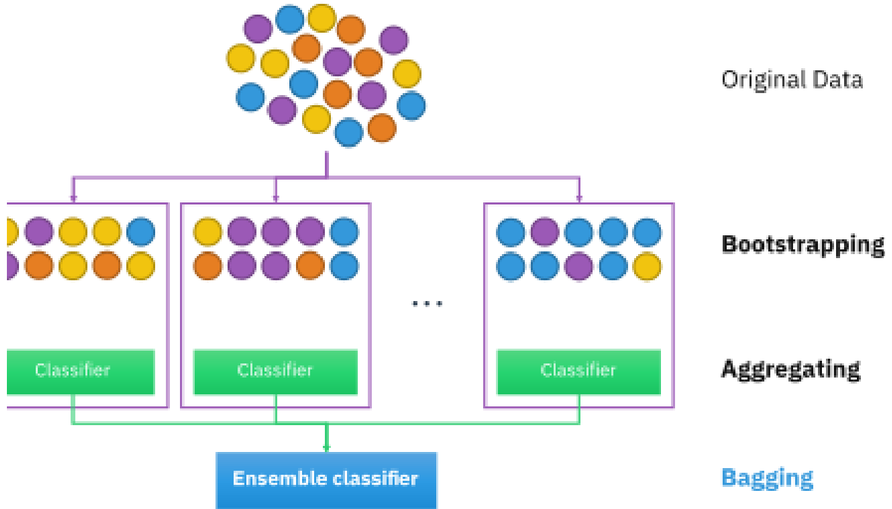
3.2 Blending

Blending follows the same approach as stacking but uses only a holdout (validation) set from the train set to make predictions. In other words, unlike stacking, the predictions are made on the holdout set only. The holdout set and the predictions are used to build a model which is run on the test set. Here is a detailed explanation of the blending process:

1. The train set is split into training and validation sets.
2. Model(s) are fitted on the training set.
3. The predictions are made on the validation set and the test set.
4. The validation set and its predictions are used as features to build a new model.
5. This model is used to make final predictions on the test and meta-features.

3. Advanced Ensemble techniques

Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.



3. Advanced Ensemble techniques

3.4 Boosting

Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model. Let's understand the way boosting works in the below steps.

1. A subset is created from the original dataset.
2. Initially, all data points are given equal weights.
3. A base model is created on this subset.
4. This model is used to make predictions on the whole dataset.
5. Errors are calculated using the actual values and predicted values.
6. The observations which are incorrectly predicted, are given higher weights.
7. Another model is created and predictions are made on the dataset.
8. (This model tries to correct the errors from the previous model)
9. Similarly, multiple models are created, each correcting the errors of the previous model.
10. The final model (strong learner) is the weighted mean of all the models (weak learners).Thus, the boosting algorithm combines a number of weak learners to form a strong learner. The individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Notes

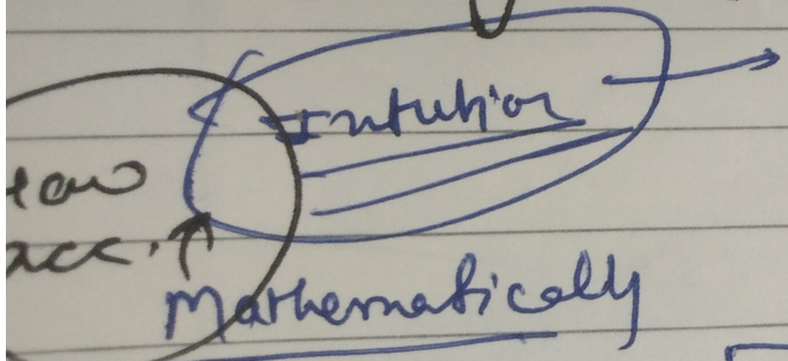
Ensemble Learning :-

1. Voting Ensemble :- Intuition ? why voting works ?

Assumption :

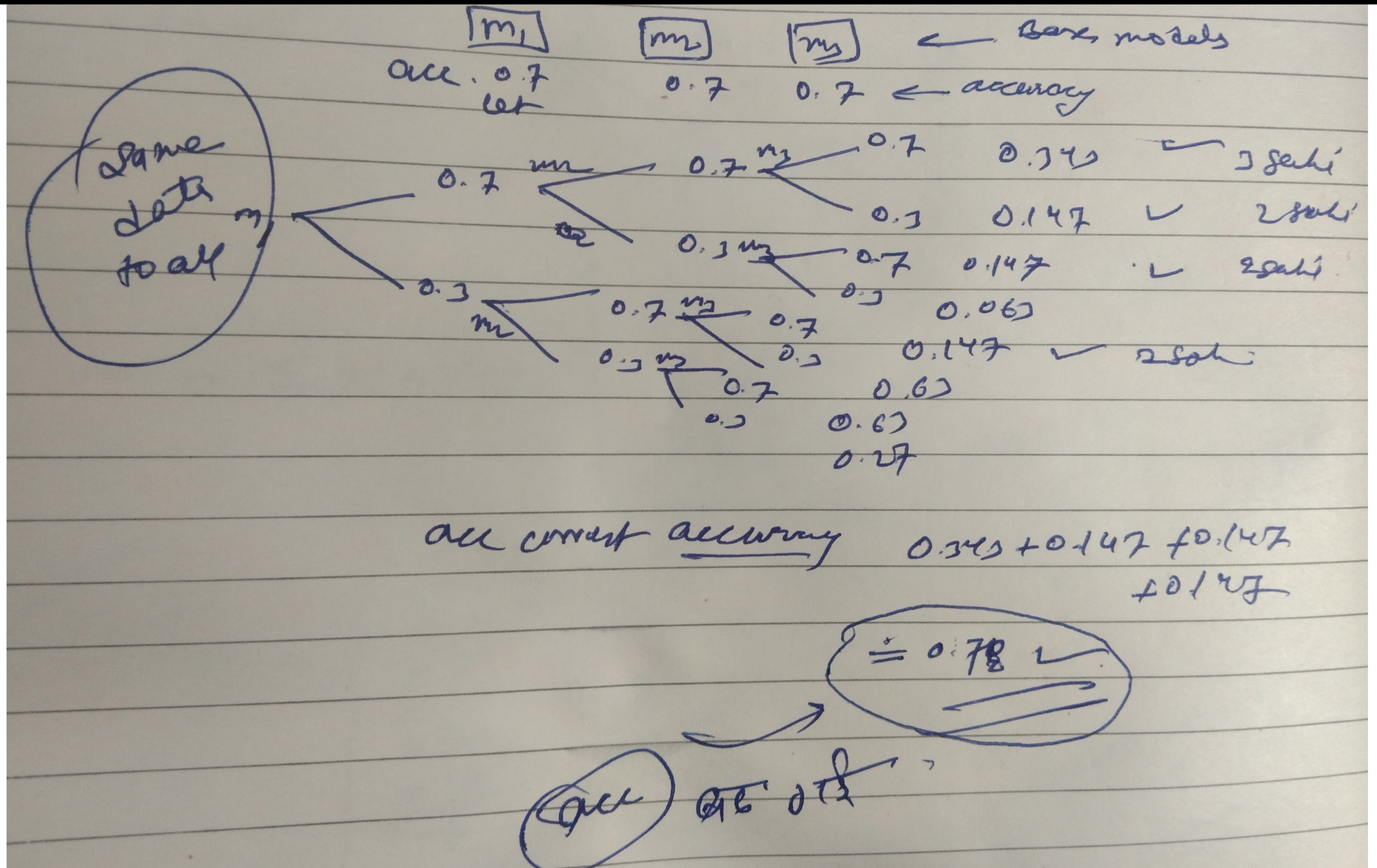
1) All models should be indep. & (Not dissimilar)
   Not should be similar
2) Minimum acc. of all models 50%. (50 >)

why voting works?



here all models are diff eg (abc all people (audience are from diff profession so able to give ans for all)

Notes


m_1 m_2 m_3 ← Base models
acc. 0.7 0.7 0.7 ← accuracy
let
Same data to all
0.7 m_1 0.7 m_2 0.7 0.343 ← 3 Sahi
0.3 0.1 0.147 ← 2 Sahi
m_2 0.3 m_3 0.7 0.147 ← 2 Sahi
0.3 0.1 0.063
m_1 0.7 m_3 0.7 0.147 ← 2 Sahi
0.3 0.1
0.3 m_3 0.7 0.063
0.1 0.063
0.027
all correct accuracy 0.343 + 0.147 + 0.147 + 0.147
≒ 0.78
acc

Notes

Voting classifier :-

In classification, two types of voting : hard & soft voting

Hard voting :- entails picking the prediction with the highest number of votes

Soft voting :- entails combining the probabilities of each prediction in each model & picking the prediction with the highest total prob.

In Regression, Instead of finding the prediction with the highest freq, regression models built with voting take the predictions of each model & compute their average value to derive a final prediction.

Notes

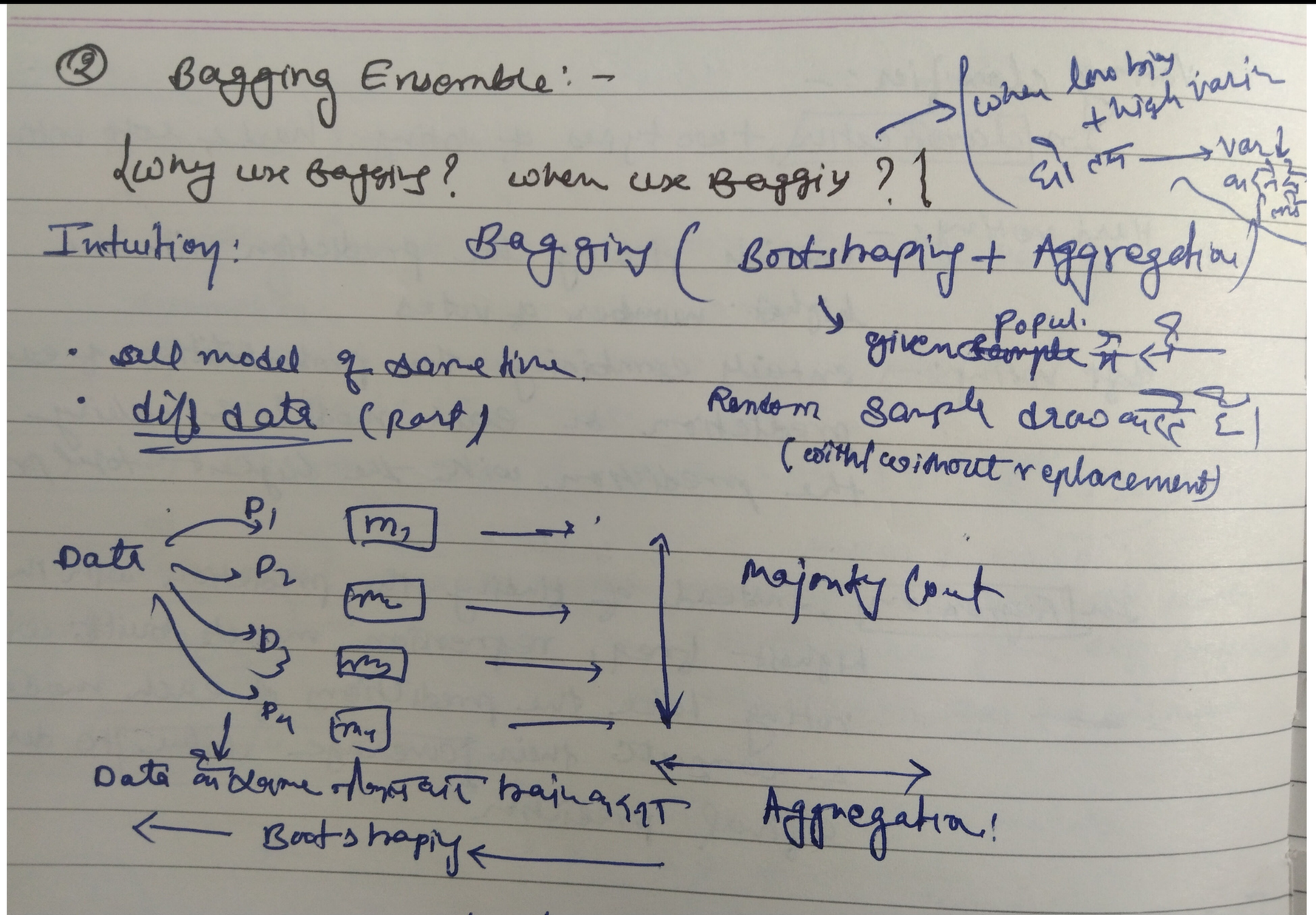
Benefits of Ensemble Learning ⇒

1) Improvement in performance
2) Bias variance ⟶ Both low
3) Robustness ⟶ Performance में ज्यादा change नहीं थोड़ा data change करने से

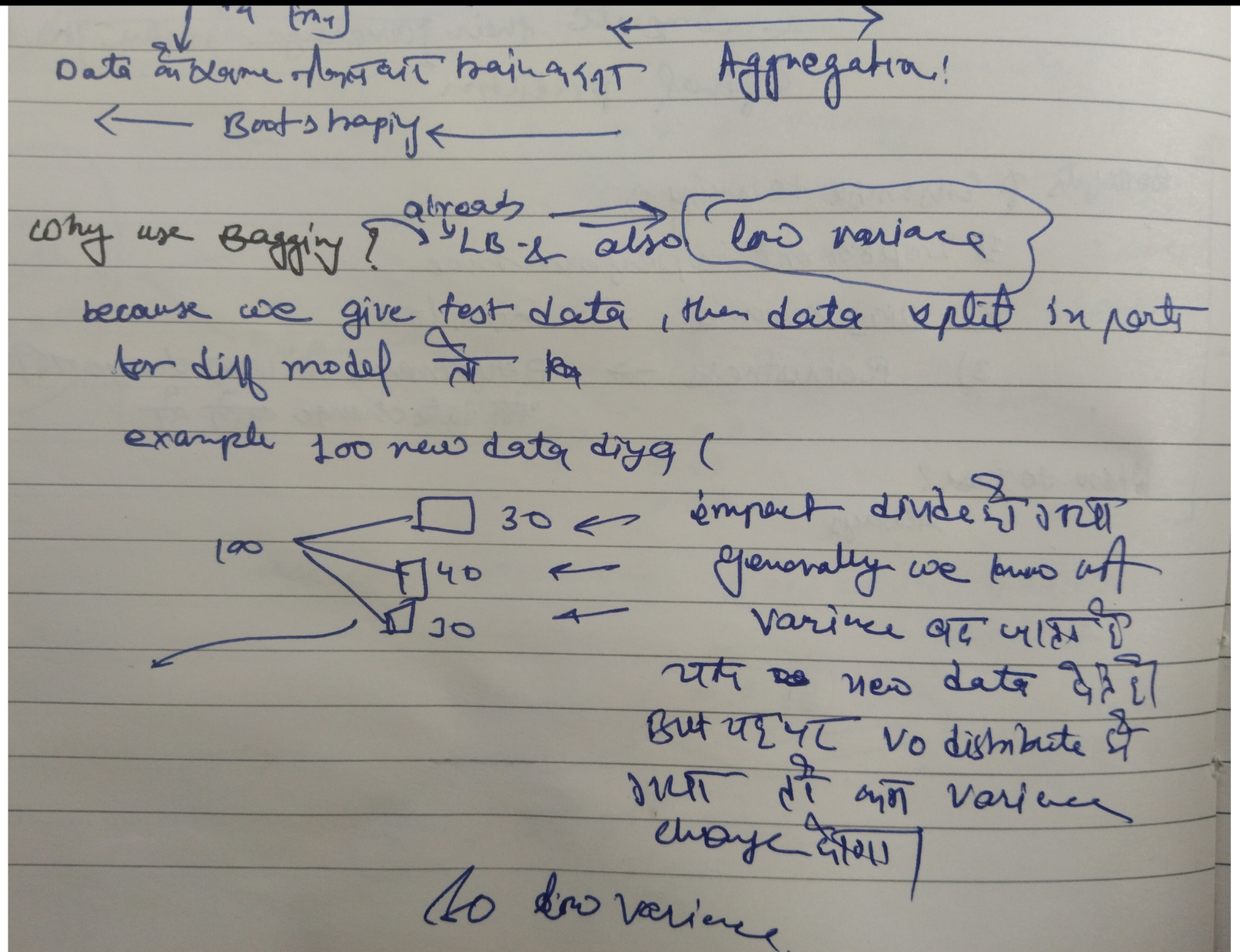
When to use?

Always.

Notes


② Bagging Ensemble :-
Why use Bagging? When use Bagging ?
when low bias + high variance
Intuition:
Bagging (Bootstraping + Aggregation)
Popul.
given sample
Random sample draw
(with/without replacement)
• all model same time
• diff data (part)
Data
P1
P2
D
P4
m1
m2
m3
m4
Majority Count
Data on same Algorithm training
Bootstraping
Aggregation

Notes


Data
Aggregation
Bootstrapping
Why use Bagging ?
already
LB-L also
low variance
because we give test data, then data split in parts
for diff model
example 100 new data diya (
100
30
40
30
impact divide
generally we have
variance
new data
But
Vo distribute
variance
change
Vo low variance

DATE

Types of Bagging :- { row's sample (मतलब diff rows) diff model को देंगे }

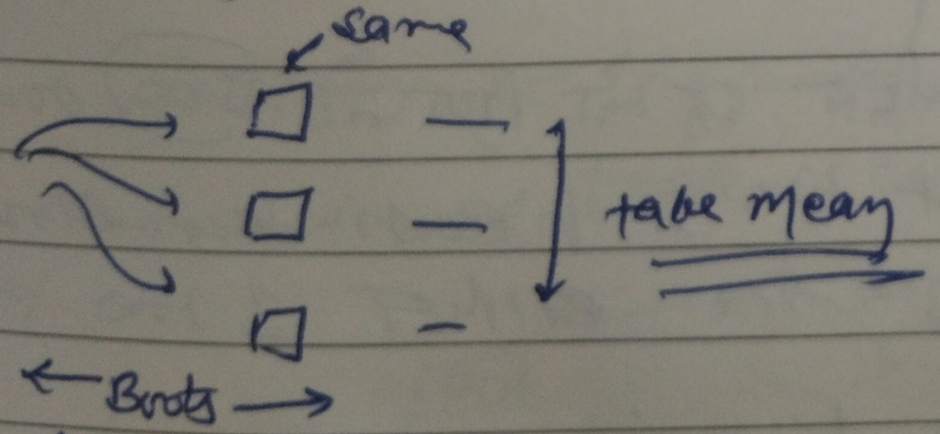
But columns को भी break कर सकते हैं as Row.

I. Pasting row sampling without Replacement

II. Random subspaces : → row sampling नहीं करते, column sampling करते हैं → with/without both ✓

III. Random Patches : Row & column both → with/without both ✓

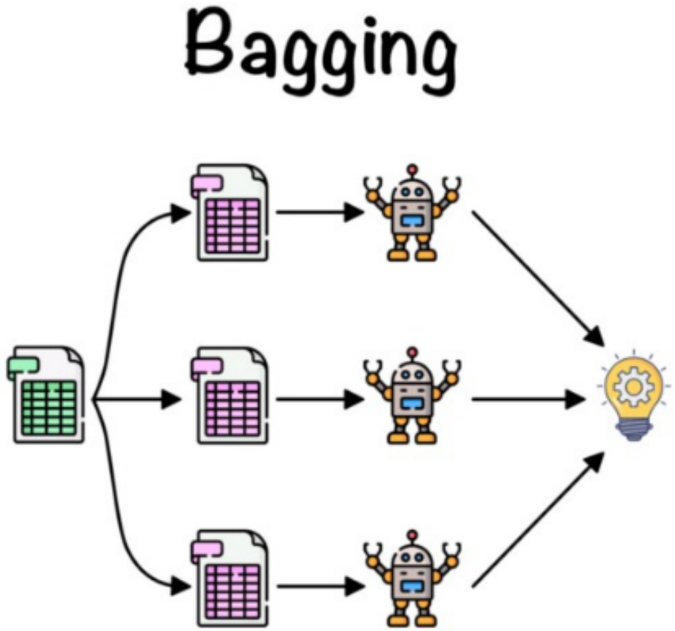
Bagging Regressor :-



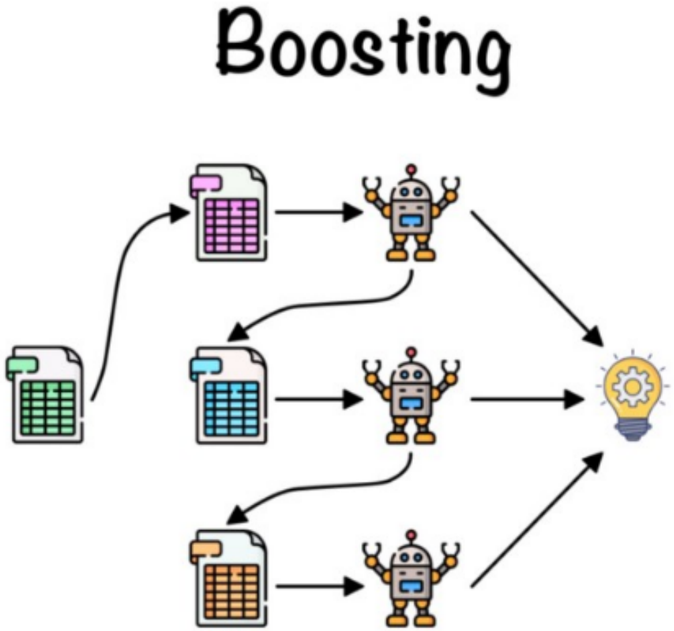
(Pasting, Random subspaces, Random Patches)

Working of Random Forest Algorithm

Random forest works on the Bagging principle.



Parallel



Sequential

Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Wisdom of the crowd

Steps involved in random forest algorithm:

Step 1: In Random forest n number of random records are taken from the data set having k number of records.
Step 2: Individual decision trees are constructed for each sample.
Step 3: Each decision tree will generate an output.
Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

In the original paper on random forests, it was shown that the forest error rate depends on two things:

The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.
The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Features of Random Forests

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

How is Variable Importance Calculated for a Random Forest?

There are two measures of importance given for each variable in the random forest. The first measure is based on how much the accuracy decreases when the variable is excluded. This is further broken down by outcome class. The second measure is based on the decrease of Gini impurity when a variable is chosen to split a node.

Importance for numeric outcomes

Percentage increase in mean square error is analogous to accuracy-based importance, and is calculated by shuffling the values of the out-of-bag samples.

Increase in node purity is analogous to Gini-based importance, and is calculated based on the reduction in sum of squared errors whenever a variable is chosen to split.

https://www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest/

Feature Importance

- **Gini importance (or mean decrease impurity)**, which is computed from the Random Forest structure. Let’s look how the Random Forest is constructed. It is a set of Decision Trees. Each Decision Tree is a set of internal nodes and leaves. In the internal node, the selected feature is used to make decision how to divide the data set into two separate sets with similars responses within. The features for internal nodes are selected with some criterion, which for classification tasks can be gini impurity or infomation gain, and for regression is variance reduction. We can measure how each feature decrease the impurity of the split (the feature with highest decrease is selected for internal node). For each feature we can collect how on average it decreases the impurity. The average over all trees in the forest is the measure of the feature importance. This method is available in scikit-learn implementation of the Random Forest (for both classifier and regressor). It is worth to mention, that in this method we should look at relative values of the computed importances. This biggest advantage of this method is a speed of computation - all needed values are computed during the Radom Forest training. **The drawbacks of the method is to tendency to prefer (select as important) numerical features and categorical features with high cardinality. What is more, in the case of correlated features it can select one of the feature and neglect the importance of the second one (which can lead to wrong conclusions).**

Feature Importance

Mean Decrease Accuracy - is a method of computing the feature importance on permuted out-of-bag (OOB) samples based on mean decrease in the accuracy. This method is not implemented in the scikit-learn package. The very similar to this method is permutation based importance

Permutation Based Feature Importance (with scikit-learn)
The permutation based importance can be used to overcome **drawbacks of default feature importance computed with mean impurity decrease.** It is implemented in scikit-learn as permutation_importance method. As arguments it requires trained model (can be any model compatible with scikit-learn API) and validation (test data). This method will randomly shuffle each feature and compute the change in the model's performance. The features which impact the performance the most are the most important one.

Out of bag OOB

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:
Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree.
Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710

Important Features of Random Forest

1. Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
2. Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced.
3. Parallelization-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
4. Train-Test split- In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
5. Stability- Stability arises because the result is based on majority voting/ averaging.

Difference Between Decision Tree & Random Forest

Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

Following hyperparameters increases the predictive power:

1. n_estimators- number of trees the algorithm builds before averaging the predictions.
2. max_features- maximum number of features random forest considers splitting a node.
3. mini_sample_leaf- determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. n_jobs- it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
2. random_state- controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
3. oob_score - OOB means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

Advantages and Disadvantages of Random Forest Algorithm

Advantages

1. It can be used in classification and regression problems.
2. It solves the problem of overfitting as output is based on majority voting or averaging.
3. It performs well even if the data contains null/missing values.
4. Each decision tree created is independent of the other thus it shows the property of parallelization.
5. It is highly stable as the average answers given by a large number of trees are taken.
6. It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
7. It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.
8. We don’t have to segregate data into train and test as there will always be 30% of the data which is not seen by the decision tree made out of bootstrap.

Disadvantages

1. Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.
2. Training time is more compared to other models due to its complexity. Whenever it has to make a prediction each decision tree has to generate output for the given input data.

Why use Random Forest Algorithm and How it work

How it works

In the random forest, we grow multiple trees in a model. To classify a new object based on new attributes each tree gives a classification and we say that tree votes for that class. The forest chooses the classifications having the most votes of all the other trees in the forest based on the importance score and takes the average difference from the output of different trees. In general, RF built multiple trees and combines them together to get a more accurate result.

Uses of Random Forest

Banking Sector: The banking sector consists of most users. There are many loyal customers and also fraud customers. To determine whether the customer is a loyal or fraud, Random forest analysis comes in.

Medicines: Medicines needs a complex combination of specific chemicals. Thus, to identify the great combination in the medicines, Random forest can be used.

E-Commerce: When you will find it difficult to recommend or suggest what type of products your customer should see.

Stock Market: Machine learning also plays role in the stock market analysis. When you want to know the behavior of the stock market, with the help of Random forest algorithm, the behavior of the stock market can be analyzed. Also, it can show the expected loss or profit which can be produced while purchasing a particular stock.

Random Forest Regression: When Does It Fail and Why?

Random Forest Regression vs Linear Regression

Random Forest Regression is quite a robust algorithm, however, the question is should you use it for regression?

Generally, Random Forests produce better results, work well on large datasets, and are able to work with missing data by creating estimates for them. However, they pose a major challenge that is that they can't extrapolate outside unseen data.

A Linear Regression model, just like the name suggests, created a linear model on the data. A simple way to think about it is in the form of y = mx+C. Therefore, since it fits a linear model, it is able to obtain values outside the training set during prediction. It is able to extrapolate based on the data.

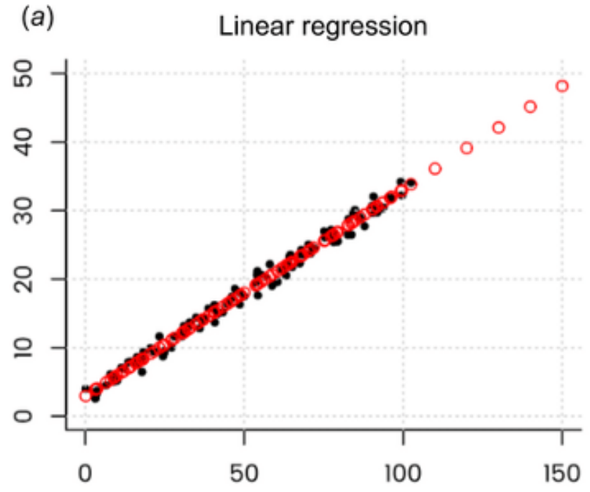
Random Forest Regression: When Does It Fail and Why?

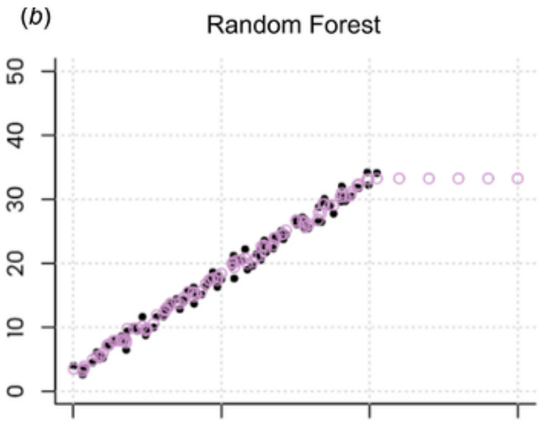
Extrapolation problem

The predicted values are never outside the training set values for the target variable.

when the Random Forest Regressor is tasked with the problem of predicting for values not previously seen, it will always predict an average of the values seen previously. Obviously the average of a sample can not fall outside the highest and lowest values in the sample.

The Random Forest Regressor is unable to discover trends that would enable it in extrapolating values that fall outside the training set. When faced with such a scenario, the regressor assumes that the prediction will fall close to the maximum value in the training set. Figure 1 above illustrates that.





Potential solutions

There are a couple of options:

- Use a linear model such as SVM regression, Linear Regression, etc
- Build a deep learning model because neural nets are able to extrapolate (they are basically stacked linear regression models on steroids)
- Combine predictors using stacking. For example, you can create a stacking regressor using a Linear model and a Random Forest Regressor.
- Use modified versions of random forest

One of such extensions is Regression-Enhanced Random Forests (RERFs)

Specifically, there are two steps to the process:

- run Lasso before Random Forest,
- train a Random Forest on the residuals from Lasso.

Feature Selection Techniques in Machine Learning

The goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena.

The techniques for feature selection in machine learning can be broadly classified into the following categories:

Supervised Techniques: These techniques can be used for labeled data, and are used to identify the relevant features for increasing the efficiency of supervised models like classification and regression.

Unsupervised Techniques: These techniques can be used for unlabeled data.

From a taxonomic point of view, these techniques are classified as under:

A. Filter methods B. Wrapper methods C. Embedded methods D. Hybrid methods

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

Let’s, discuss some of these techniques:

- Information Gain: already discuss
- Chi-square Test: The Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with the best Chi-square scores. In order to correctly apply the chi-squared in order to test the relation between various features in the dataset and the target variable, the following conditions have to be met: the variables have to be categorical, sampled independently and values should have an expected frequency greater than 5.

From a taxonomic point of view, these techniques are classified as under:

Fisher’s Score

Fisher score is one of the most widely used supervised feature selection methods. The algorithm which we will use returns the ranks of the variables based on the fisher’s score in descending order. We can then select the variables as per the case.

Correlation Coefficient

Correlation is a measure of the linear relationship of 2 or more variables. Through correlation, we can predict one variable from the other. The logic behind using correlation for feature selection is that the good variables are highly correlated with the target. Furthermore, variables should be correlated with the target but should be uncorrelated among themselves.

From a taxonomic point of view, these techniques are classified as under:

Variance Threshold

The variance threshold is a simple baseline approach to feature selection. It removes all features which variance doesn’t meet some threshold. By default, it removes all zero-variance features, i.e., features that have the same value in all samples. We assume that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature variables or feature and target variables into account, which is one of the drawbacks of filter methods.

Mean Absolute Difference (MAD)

‘The mean absolute difference (MAD) computes the absolute difference from the mean value. The main difference between the variance and MAD measures is the absence of the square in the latter. The MAD, like the variance, is also a scale variant.’ [1] This means that higher the MAD, higher the discriminatory power.

From a taxonomic point of view, these techniques are classified as under:

B. Wrapper Methods:

Forward Feature Selection, Backward Feature Elimination, Exhaustive Feature Selection, Recursive Feature Elimination

C. Embedded Methods:

LASSO Regularization (L1), Random Forest Importance

Feature Selection Techniques in Machine Learning (analytic vidhya)

Notes

Random Forest :- → is a bagging based technique

→ group of trees.

Bagging → Bootstrapped + Aggregation

Decision Tree ← Sampling ← (Randomly data में से select करना)

eg 1000 ← (data)

select 500 randomly data

500 500 500

DT1 DT2 DT3 . . .

with replacement

with replacement

without replacement

Notes

without replacement

with replacement

eg. let 500 samples (1000 data) है उसे हम decision tree को देना है।

let सबसे पहले DT को दिया तो 500 rows select करते let first time 1, second time 42000 फिर 3rd time वापिस 1 row आ सकती है → that is with replacement

with Replacement → Duplicate row आ सकती है

without Replacement → No Duplicate row

↓

3 types of sampling (3 types से data select

→ row sampling, → column sampling, → Both row/colu

Notes

Q How Random forest work so well? bias-variance trade off

high Var + low bias → Change करदेता है

low bias + low variance

variance कैसे कम करता है? → Same reason as in bagging

Imp # Bagging Vs Random forest ?

- diff algo use करते है (models). only Decision tree

Q= If all models of bagging are Decision Tree then it is Random Forest? NO

diff in feature sampling.

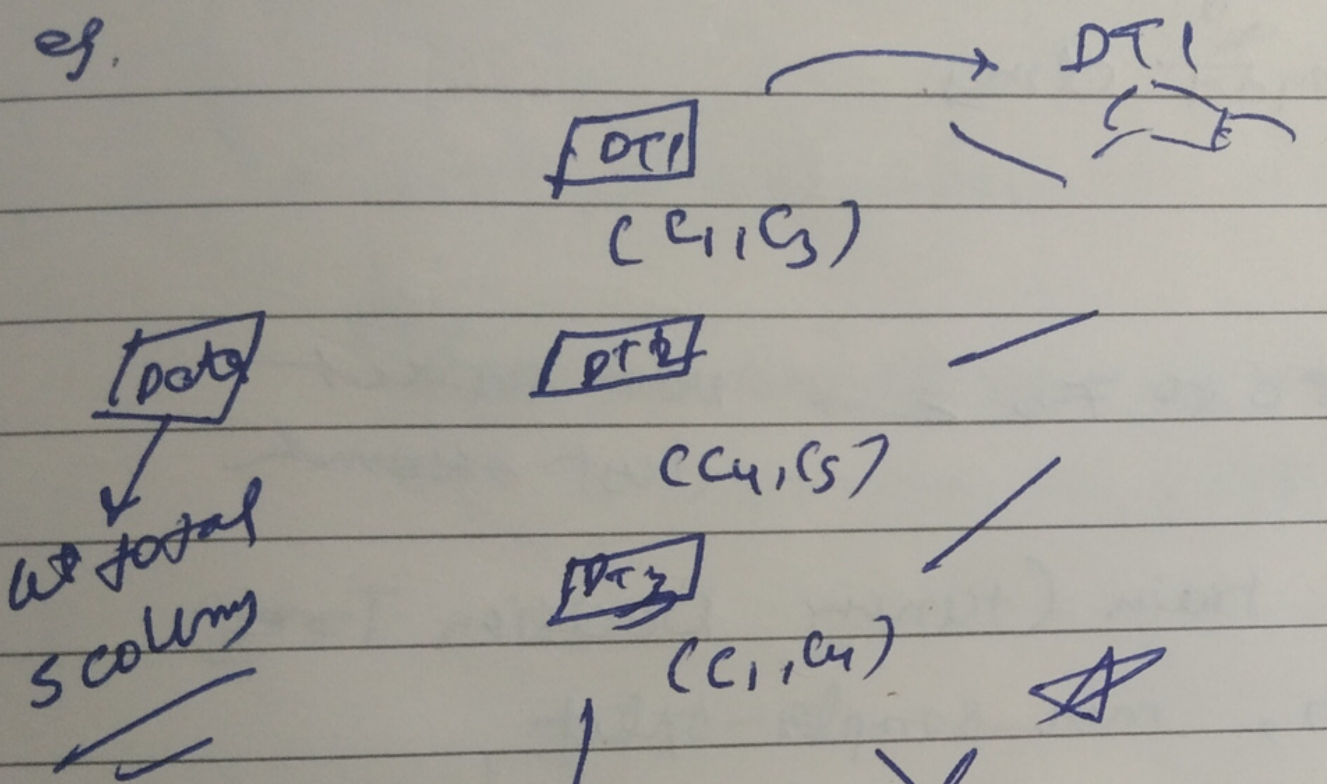
Sampling ← row sampling
feature (column sampling)

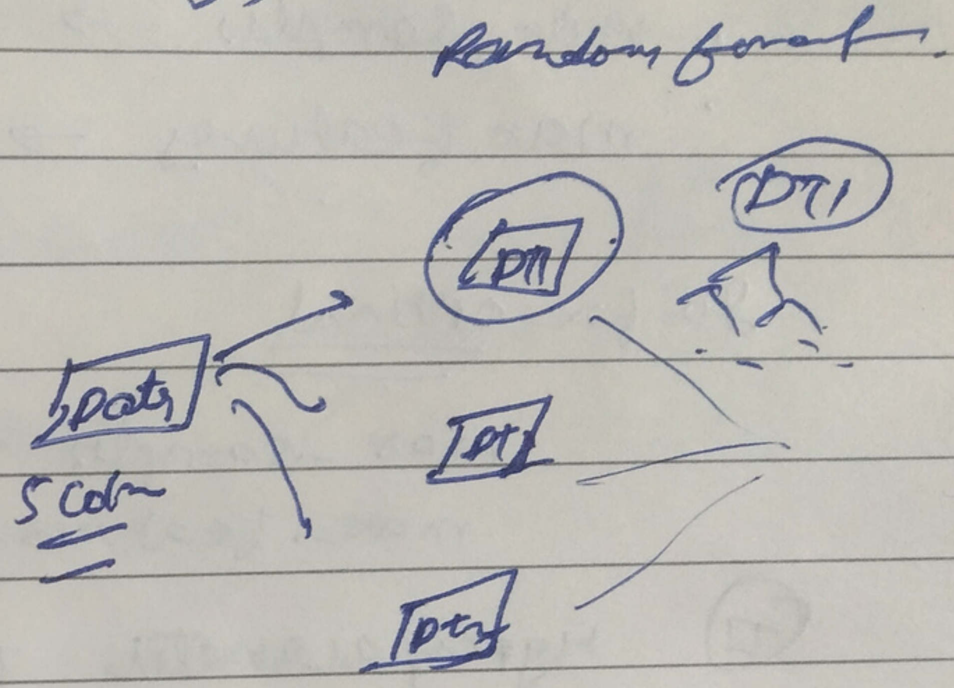
Random forest.

Notes

Sampling
- row sampling
- feature (column sampling)

eg.





↓
we give
2 column randomly
↓
Bagging

यहा पर पहले ही fix हो जाता है कि कौनसे tree को कौन-कौन से clm दो। let DT1 को C4 & C5 दिया तो पूरा tree इस C4 C5 पर ही बनेगा But in random forest. यहा पर fix नही होता जितनी बार भी एक particular tree split होगा उतनी बार उसको randomly new-columns मिलेंगे।

Notes

So in Bagging → Tree-level complexity (Tree बनने से पहले)
In Random forest → Tree बनने के बाद at every node

So we can also say Random forest beats bagging in some cases (all model are Decision Tree)

Hyperparameters in (both Random Forest classifier & Random Forest Regressor)

cols divide into 2 part

for classifier

(I) Hyperparameters for tuning Random forest

- Num estimators → कितने D.Tree लेने हैं
- bootstrap → True ← replacement
  → False ← Not Replacement
- max_samples → No. of rows.
- max features → कितने cols

So for optimal

max_samples → 50 to 70 ← not exact just observe
max_features →

(II) Hyperparameter for train (tuning Decision Tree)

- Criterion, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_split

(III) Common hyperparameter in all algo.

oob_score, n_jobs, random_state, verbose, warm_state, class_weight ..... so many others

~~max_samples~~

for Random forest Regression

↳ nearly all same as in classifier

check on sklearn — eg criteria यहाँ पर mse, mae होते है
classifier में gini & entropy था
बाकी same

generally Random forest without hyperparameter भी
अच्छा करता है।

Q. How to find best hyperparameter?

so many methods.

① GridSearchCV ⟷ use when आप data & जो algo use कर रहे है उसमें कम hyperparameter है।

② RandomSearchCV ⟷ data large + hyperparameter भी high

→ सभी combination को check करता, → कुछ Randomly select करते है

③ (fast Result But NOT better as compare to gridsearchCV.)

ODB Evaluation

#OOB Evaluation:- ← in all ensemble,

Out of Bag evaluation:-

generally in random forest data को models में देते हैं तो data (rows) repeat हो जाते हैं, तो कुछ data किसी model में नहीं जाता है → इसको out of Bag samples

OOB hidden

mathematically proved → so we can use them in testing

≈ nearly 37% rows को जाता है

↓ so we can use it as test data

eg ⇒ rf = RandomForestClassifier(oob_score = True)

rf.oob_score_ ← इसे देखे गए

↳ check accuracy score in test data

nearly come same

~~but if~~

ODB Evaluation

Feature Importance (using Random forest & D.T)

What is Feature Importance.

Feature selection करने कि एक method

Feature importance से सभी feature (clm) का एक importance निकालते हैं। —

eg in image. 28×28 = 784....

Feature importance not only give कि कितने सेल हैं कितने नहीं. But also calculate भी करते कितना importance है।

eg. loan ← Data analysis → इनकम सही इनकम

यदि किसी को loan नहीं मिलता तो वो जानना चाहेगा क्यों नहीं? किस feature से नहीं।

ODB Evaluation

eg in RandomForest

rf.feature_importances_

obj

sns.heatmap(rf.feature_importances_.reshape(28,28))

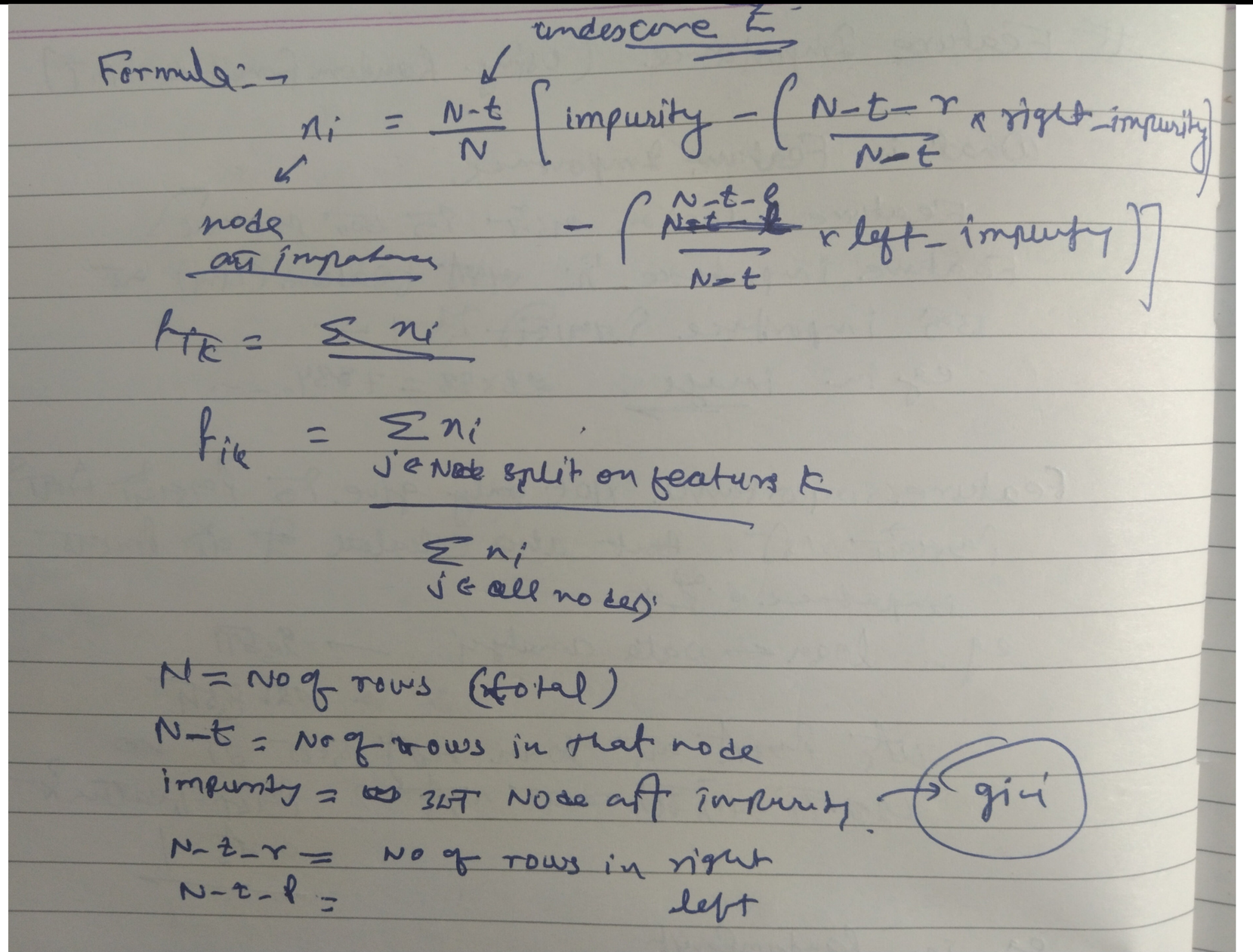
→ check

* How feature importance is calculated?

depends on model

for DecisionTree → using gini (impurity)

ODB Evaluation



ODB Evaluation

In Random forest :- every tree calculate the feature importace of every feature then take average

Warning: impurity-based feature importaces can be misleading for high cardinality features (many unique values),

↑ feature_importaces → and data → for low cardinality

for high cardinality → So we use sklearn.inspection.permutation_importaces.

ODB Evaluation