KURUNJI VENKATRAMANA GOWDA POLYTECHNIC SULLIA-574327

5TH SEMESTER

AI/ML WEEK-7

Week 7

Decision Trees

LIST OF REGRESSION ALGORITHMS IN MACHINE LEARNING: -

1. Linear Regression
2. Ridge Regression
3. Neural Network Regression
4. Lasso Regression
5. Decision Tree Regression
6. Random Forest
7. KNN Model
8. Support Vector Machines (SVM)
9. Gausian Regression
10. Polynomial Regression

- **LINEAR REGRESSION**

It is one of the most-used regression algorithms in Machine Learning. A significant variable from the data set is chosen to predict the output variables (future values).Linear regression algorithm is used if the labels are continuous, like the number of flights daily from an airport, etc. The representation of linear regression is y = b*x + c. ➢ **2) RIDGE REGRESSION**

Ridge Regression is another popularly used linear regression algorithm in Machine Learning. If only one independent variable is being used to predict the output, it will be termed as a linear regression ML algorithm. ML experts prefer Ridge regression as it minimizes the loss encountered in linear regression (discussed above). In place of OLS (Ordinary Least Squares), the output values are predicted by a ridge estimator in ridge regression. The above-discussed linear regression uses OLS to predict the output values.

- **NEURAL NETWORK REGRESSION**

You all must be aware of the power of neural networks in making predictions/assumptions. Each node in a neural network has a respective activation function that defines the output of the node based on a set of inputs. The last activation function can be manipulated to change a neural network into a regression model. One can use 'Keras' that is the appropriate python library for building neural networks in ML. ➢ **LASSO REGRESSION**

Lasso (Least Absolute Shrinkage and Selection Operator) regression is another widely used linear ML regression (one input variable). The sum of coefficient values is penalized in lasso regression to avoid prediction errors. The determination coefficients in lasso regression are reduced towards zero by using the technique 'shrinkage'. The regression coefficients are reduced by lasso regression to make them fit perfectly with various datasets. Besides ML, the lasso algorithm is also used for regression in Data Mining. ➢ **DECISION TREE REGRESSION**

Non-linear regression in Machine Learning can be done with the help of decision tree regression. The main function of the decision tree regression algorithm is to split the dataset into smaller sets. The subsets of the dataset are created to plot the value of any data point that connects to the problem statement. The splitting of the data set by this algorithm results in a decision tree that has decision and leaf nodes. ML experts prefer this model in cases where there is not enough change in the data set.

- **RANDOM FOREST**

Random forest is also a widely-used algorithm for non-linear regression in Machine Learning. Unlike decision tree regression (single tree), a random forest uses multiple decision trees for predicting the output. Random data points are selected from the given dataset (say k data points are selected), and a decision tree is built with them via this algorithm. Several decision trees are then modeled that predict the value of any new data point. ➢ **7) KNN MODEL**

KNN model is popularly used for non-linear regression in Machine Learning. KNN (K Nearest Neighbours) follows an easy implementation approach for non-linear regression in Machine Learning. KNN assumes that the new data point is similar to the existing data points. The new data point is compared to the existing categories and is placed under a relatable category. The average value of the k nearest neighbors is taken as the input in this algorithm. The neighbors in KNN models are given a particular weight that defines their contribution to the average value.

- **SUPPORT VECTOR MACHINES (SVM)**

SVM can be placed under both linear and non-linear types of regression in ML. The use cases of SVM can range from image processing and segmentation, predicting stock market patterns, text categorization, etc. When you have to identify the output in a multidimensional space, the SVM algorithm is used. In a multidimensional space, the data points are not represented as a point in a 2D plot. The data points are represented as a vector in a multidimensional space. ➢ **GAUSSIAN REGRESSION**

Gaussian regression algorithms are commonly used in machine learning applications due to their representation flexibility and inherent uncertainty measures over predictions. A Gaussian process is built on fundamental concepts such as multivariate normal distribution, nonparametric models, kernels, joint and conditional probability.

- **POLYNOMIAL REGRESSION**

Polynomial Regression is a regression algorithm that models the relationship between an independent variable (x) and a dependent variable (y) as an nth degree polynomial. The equation for Polynomial Regression is as follows:y= b0 b1x1 b2x12 b2x13 …… bnx1nIt is also known as the special scenario of Multiple Linear Regression in machine learning. Because to make it polynomial regression, some polynomial terms are added to the Multiple Linear Regression equation. It is a linear model that has been modified to improve accuracy. The

dataset used for training in polynomial regression is nonlinear. To fit the non-linear and complicated functions and datasets. The original features are changed into Polynomial features of the required degree (2,3,…,n) and then modelled using a linear model.

Building a model with some other Regression algorithm:

• Random Forest Regression

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.We need to approach the Random Forest regression technique like any other machine learning technique

Design a specific question or data and get the source to determine the required data.

- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model
- Set the baseline model that you want to achieve
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data, or using another data modeling technique

1. Supervised learning classification :-

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a "sub-populations." With the help of these precategorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories.

In machine learning, classification is a supervised learning concept which basically categorizes a set of data into classes. The most common classification problems are – speech recognition, face detection, handwriting recognition, document classification, etc.

2. Types of classifications:

- **Binary classification**

In machine learning, binary classification is a supervised learning algorithm that categorizes new observations into one of two classes. The following are a few binary classification applications, where the 0 and 1 columns are two possible classes for each observation: Application.

- **Multi-label classifications**

Multi-label classification involves predicting zero or more class labels. Unlike normal classification tasks where class labels are mutually exclusive ,multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually nonexclusive classes or "labels."

- **Multi-class classifications**

In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes (classifying instances into one of two classes is called binary classification).

- **Imbalanced classifications**

A classification data set with skewed class proportions is called imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes.

3. Applications of Classification Algorithm

Okay, so now we understand a bit of the mathematics behind classification, but what can these machine learning algorithms do with real-world data?

- Sentiment Analysis
- Email Spam Classification
- Document Classification
- Image Classification

- **Sentiment Analysis :-**

Sentiment analysis is a machine learning text analysis technique that assigns sentiment (opinion, feeling, or emotion) to words within a text, or an entire text, on a polarity scale of *Positive*, *Negative*, or *Neutral*.

- **Email Spam Classification**

One of the most common uses of classification, working non-stop and with little need for human interaction, email spam classification saves us from tedious deletion tasks and sometimes even costly phishing scams.

- **Document Classification**

Document classification is the ordering of documents into categories according to their content. This was previously done manually, as in the library sciences or hand-ordered legal files. Machine learning classification algorithms, however, allow this to be performed automatically.

- **Image Classification**

Image classification assigns previously trained categories to a given image. These could be the subject of the image, a numerical value, a theme, etc. Image classification can even use multi-label image classifiers, that work similarly to multi-label text classifiers, to tag an image of a stream, for example, into different labels, like “stream,” “water,” “outdoors,” etc.

Decisions trees

1. Introduction to Decision tress :-

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

2. What is Decision tress

It is also defined as **Decision trees are an approach used in supervised machine learning, a technique which uses labelled input and output datasets to train models.** The approach is used mainly to solve classification problems, which is the use of a model to categorise or classify an object.

An example of a decision tree can be explained using above binary tree. Let's say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc.

The decision nodes here are questions like 'What's the age?', 'Does he exercise?', 'Does he eat a lot of pizzas'? And the leaves, which are outcomes like either 'fit', or 'unfit'.

3. Advantages of Decision trees are:

- Simple to understand and to interpret.
- Requires little data preparation.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data.
- Able to handle multi-output problems.

4. Types of Decision Trees:

1. Classification trees (Yes/No types)
2. Regression trees (Continuous data types)

➢ Classification trees (Yes/No types):-

What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is Categorical.

➢ **Regression trees (Continuous data types):-**

A regression tree is basically a decision tree that is used for the task of regression which can be used to predict continuous valued outputs instead of discrete outputs

5. Understanding Entropy, Information gain

Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations. It determines how a decision tree chooses to split data. Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Intuitively, it tells us about the predictability of a certain event. Example, consider a coin toss whose probability of heads is 0.5 and probability of tails is 0.5. Here the entropy is the highest possible, since there's no way of determining what the outcome might be. Alternatively, consider a coin which has heads on both the sides, the entropy of such an event can be predicted perfectly since we know beforehand that it'll always be heads. In other words, this event has no randomness hence it's entropy is zero. In particular, lower values imply less uncertainty while higher values imply high uncertainty

information gain is also called as Kullback-Leibler divergence denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables. Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

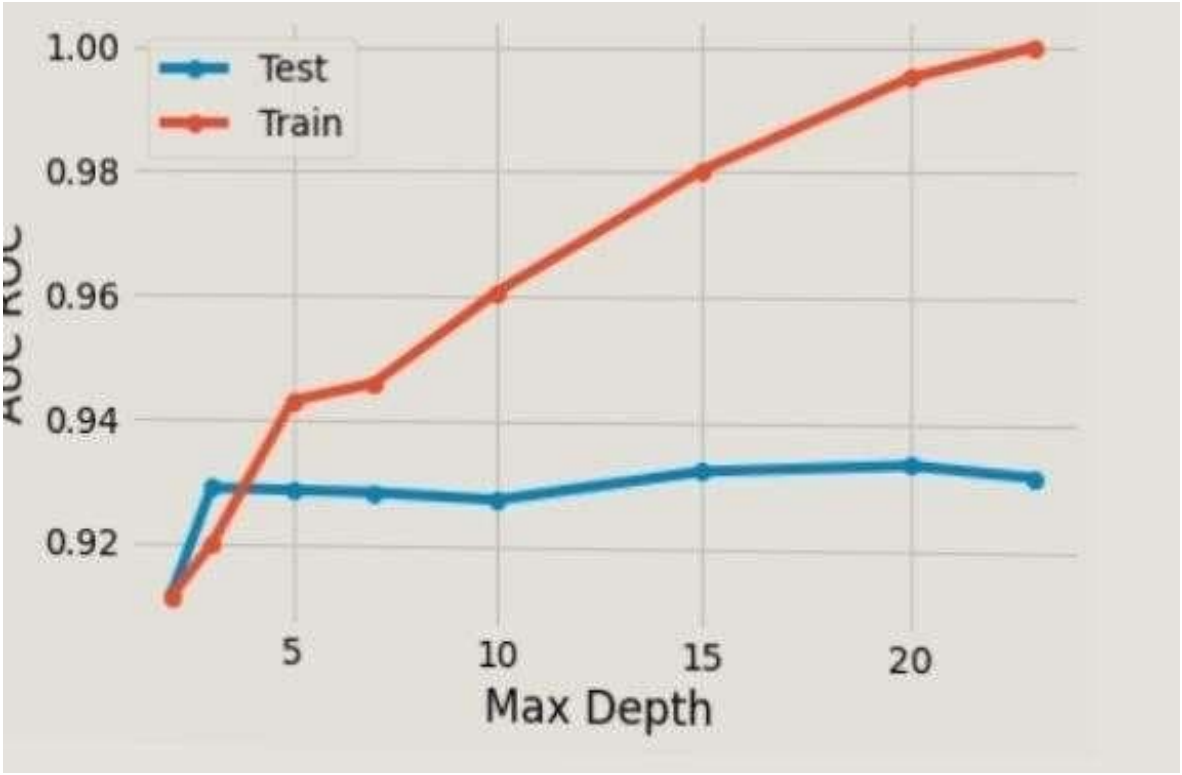
where IG(S, A) is the information gain by applying feature A. H(S) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature A, where P(x) is the probability of event x.

6. 3 Techniques to Stop Overfitting of Decision Trees:-

Decision Trees are a non-parametric supervised machine learning approach for classification and regression tasks. Overfitting is a common problem, a data scientist needs to handle while training decision tree models. Comparing to other machine learning algorithms, decision trees can easily overfit.

Pre-Pruning:-

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including max_depth, min_samples_leaf, min_samples_split can be tuned to early stop the growth of the tree and prevent the model from overfitting.



Post-Pruning:

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting. Cost complexity pruning (ccp) is one type of post-pruning technique. In case of cost complexity pruning, the ccp_alpha can be tuned to get the best fit model.

Scikit-learn package comes with the implementation to compute the ccp_alpha values of the decision tree using function cost_complexity_pruning_path(). With the increase in ccp_apha values, more nodes of the tree are pruned.

Steps for cost complexity pruning (post-pruning) are :-

- Train a decision tree classifier to its full depth (default hyperparameters).
- Compute the ccp_alphas value using function cost_complexity_pruning_path()

Train decision tree classifiers with different values of ccp_alphas and compute train and test performance scores.

- Plot train and test scores for each value of ccp_alphas values.

From the above plot, ccp_alpha=0.000179 can be considered as the best parameter as AUC-ROC scores for train and test are 0.94 and 0.92 respectively**.**

Random Forest:

Random Forest is an ensemble technique for classification and regression by bootstrapping multiple decision trees. Random Forest follows bootstrap sampling and aggregation techniques to prevent overfitting.Random Forest can be implemented using the Scikit-Learn library. You can further tune the hyperparameters of the Random Forest algorithm to improve the performance of the model. n_estimator parameter can be tuned to reduce the overfitting of the model.

7. What is Pruning?

By default, the decision tree model is allowed to grow to its full depth. Pruning refers to a technique to remove the parts of the decision tree to prevent growing to its full depth. By tuning the hyperparameters of the decision tree model one can prune the trees and prevent them from overfitting.

• **There are two types of pruning**

- Pre-pruning
- Post-pruning.

Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances. Decision trees are the most susceptible out of all the machine learning algorithms to overfitting and effective pruning can reduce this likelihood.

8. What is a decision tree classifier?

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

The main advantage of the decision tree classifier is its ability to using different feature subsets and decision rules at different stages of classification.

9. How does decision tree works?

Decision trees use multiple algorithms to decide to split a node into two or more subnodes. The creation of sub-nodes increases the homogeneity of resultant subnodes. In other words, we can say that the purity of the node increases with respect to the target variable.

Parameters:

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_features : int, float, string or None, optional (default=None)

The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=sqrt(n_features).
- If "sqrt", then max_features=sqrt(n_features).

- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

max_depth : int or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. Ignored if `max_samples_leaf` is not None.

min_samples_split : int, optional (default=2)

The minimum number of samples required to split an internal node.

min_samples_leaf : int, optional (default=1)

The minimum number of samples required to be at a leaf node.

max_leaf_nodes : int or None, optional (default=None)

Grow a tree with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes. If not None then `max_depth` will be ignored.

random_state : int, RandomState instance or None, optional (default=None)

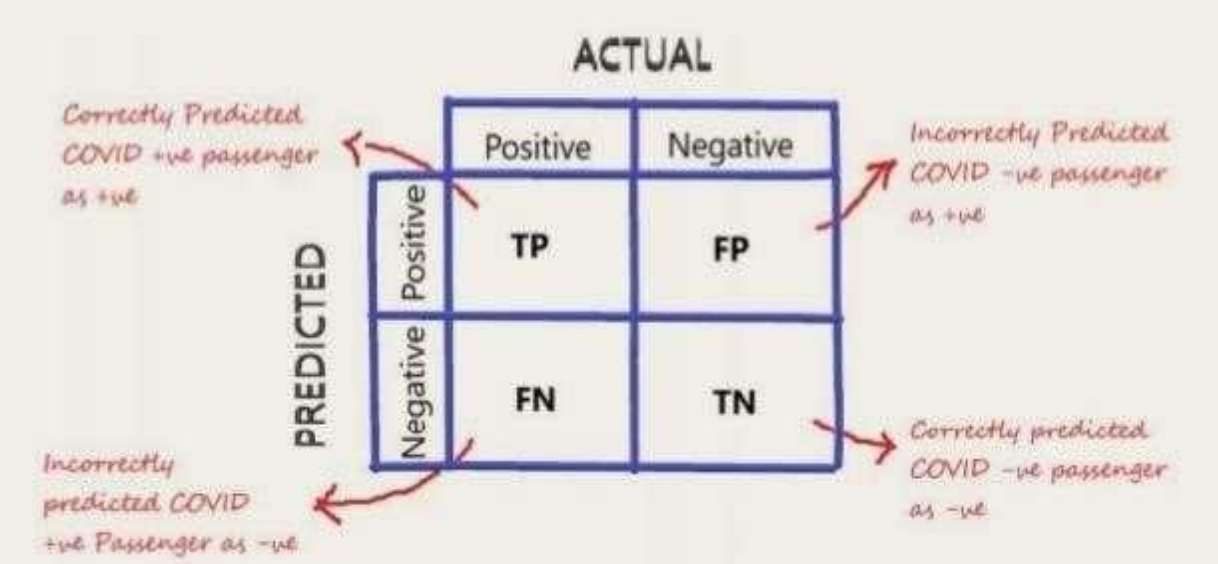
If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

1. Evolutions matrics for Classification:

A. Confusion Matrix

Confusion Matrix is a useful machine learning method which allows you to measure Recall, Precision, Accuracy, and AUC-ROC curve. Below given is an example to know the terms True Positive, True Negative, False Negative, and True Negative. True Positive: You projected positive and its turn out to be true.

For better visualization of the performance of a model, these four outcomes are plotted on a confusion matrix.



B. Accuracy

Yes! You got that right, we want our model to focus on True positive and True Negative. Accuracy is one metric which gives the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy = Number of correct predictions / Total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TP}$$

C. SPECIFICITY:

Specificity is the metric that evaluates a model's ability to predict true negatives of each available category. These metrics apply to any categorical model.

D. F1- score

It is defined as the harmonic mean of the model's precision and recall.

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

We use Harmonic mean because it is not sensitive to extremely large values, unlike simple averages. Say, we have a model with a precision of 1, and recall of 0 gives a simple average as 0.5 and an F1 score of 0. If one of the parameters is low, the second one no longer matters in the F1 score. The F1 score favors classifiers that have similar precision and recall. Thus, the F1 score is a better measure to use if you are seeking a balance between Precision and Recall.

F. **ROC/AUC Curve**

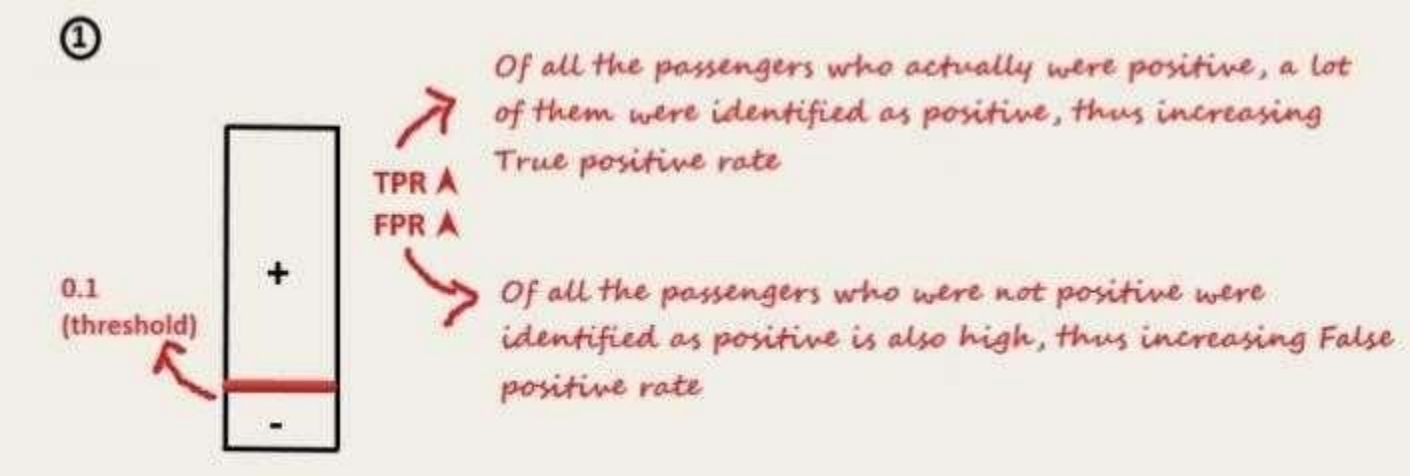
The receiver operator characteristic is another common tool used for evaluation. It plots out the sensitivity and specificity for every possible decision rule cutoff between 0 and 1 for a model. For classification problems with probability outputs, a threshold can convert probability outputs to classifications. we get the ability to control the confusion matrix a little bit. So by changing the threshold, some of the numbers can be changed in the confusion matrix. But the most important question here is, how to find the right. Threshold? Of course, we

don’t want to look at the confusion matrix every time the threshold is changed, therefore here comes the use of the ROC curve.

For each possible threshold, the ROC curve plots the False positive rate versus the true positive rate.

False Positive Rate: Fraction of negative instances that are incorrectly classified as positive.

True Positive Rate: Fraction of positive instances that are correctly predicted as positive



Hyperparameter tuning in Decision TreeClassifier:

Hyperparameter and parameters: Parameters are the model features that the model learns from the data. Whereas, Hyperparameters are arguments accepted by a modelmaking function and can be modified to reduce overfitting, leading to a better generalization of the model.

Hyperparameter tuning in Decision Trees

This process of calibrating our model by finding the right hyperparameters to generalize our model is called Hyperparameter Tuning. We will look at a few of these hyperparameters: **a. Max Depth**

This argument represents the maximum depth of a tree. If not specified, the tree is expanded until the last leaf nodes contain a single value. Hence by reducing this meter, we can preclude the tree from learning all training samples thereby, preventing over-fitting

```
model2.tree_.max_depth
```

```
57
```

```
model2 = DecisionTreeClassifier(max_depth=3, random_state=42)
model2.fit(train_inputs, train_targets)
model2.score(train_inputs, train_targets)
```

```
0.7230238095238095
```

```
model2.score(val_inputs, val_targets)
```

```
0.7301428571428571
```

We can check the current maximum depth of our decision tree with model2.tree_.max_depth. It is evident that even though the training accuracy has reduced, the validation accuracy has improved. Since we are not sure what depth our ideal model would have, we can run a for loop from a range of numbers to find out. You can see below, we've used this basic for loop to print the training and validation accuracy of our model across the range 1–21.

```
for max_d in range(1,21):
  model = DecisionTreeClassifier(max_depth=max_d,
random_state=42)
  model.fit(train_inputs, train_targets)
  print('The Training Accuracy for max_depth {}
is:'.format(max_d), model.score(train_inputs, train_targets))
  print('The Validation Accuracy for max_depth {}
is:'.format(max_d), model.score(val_inputs,val_targets))
  print('')
```

By carefully looking at the results, we can find the max_depth value where the validation accuracy starts decreasing, and the training accuracy starts mounting inordinately.

By carefully looking at the results, we can find the `max_depth` value where the validation accuracy starts decreasing, and the training accuracy starts mounting inordinately.

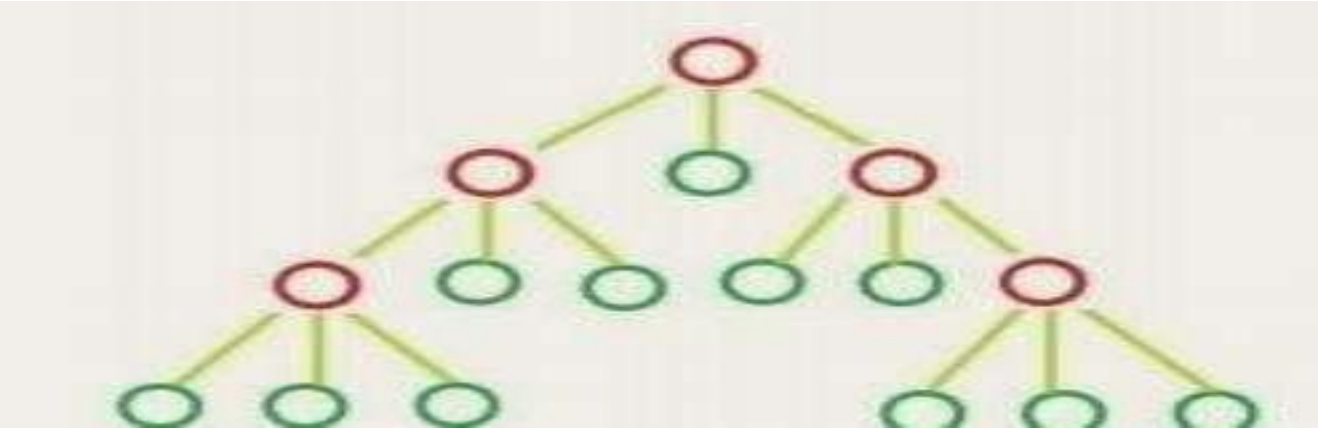
```
Training Accuracy for max_depth 7 is: 0.7359523809523809
Validation Accuracy for max_depth 7 is: 0.7284285714285714

Training Accuracy for max_depth 8 is: 0.7395
Validation Accuracy for max_depth 8 is: 0.7305

Training Accuracy for max_depth 9 is: 0.7458333333333333
Validation Accuracy for max_depth 9 is: 0.7277857142857143
```

b. Max leaf nodes

As the name suggests, this Hyperparameter caps the number of leaf nodes in a decision tree. It will allow the branches of a tree to have varying depths, another way to control the model's complexity



How?: In this case, the model will not find the best split layer by layer. Instead, it will look at all the possible splits (left and right) and only split the node with the lowest Gini Value, irrespective of the level.

To change the number of maximum leaf nodes, we use, max_leaf_nodes.

```
model2 = DecisionTreeClassifier(max_leaf_nodes=30, random_state=42)
model2.fit(train_inputs, train_targets)
```

```
DecisionTreeClassifier(max_leaf_nodes=30, random_state=42)
```

```
model2.score(train_inputs, train_targets)
```

```
0.7327142857142858
```

```
model2.score(val_inputs, val_targets)
```

```
0.7335
```

```
model2.tree_.max_depth
```

```
9
```

Here is the result of our model's training and validation accuracy at different values of max_leaf_node Hyperparameter:

```
Training Accuracy for max_leaf_node 40 is: 0.7327142857142858
Validation Accuracy for max_leaf_node 40 is: 0.7335

Training Accuracy for max_leaf_node 50 is: 0.7334285714285714
Validation Accuracy for max_leaf_node 50 is: 0.732
```

While tuning the hyper-parameters of a single decision tree is giving us Some improvement, a stratagem would be to merge the results of diverse decision trees (like a forest) with moderately different parameters. It is what we will understand in a random forest. So let's practice some other hyper-parameters like max_features, min_samples_split, etc., under random forests.