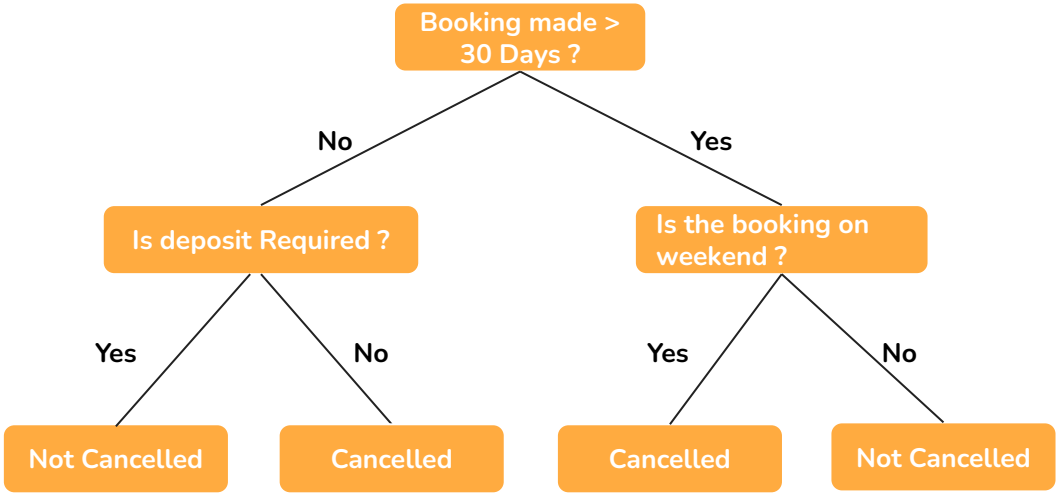
Decision Trees

Discussion questions

- What is a Decision Tree and how does it work?
- What are the common terminologies used in a Decision Tree?
- What is Pruning? How does it avoid overfitting?
- What are the different ways to prune a tree?
- How to read python's visual output of a Decision Tree?
- Classification Metrics

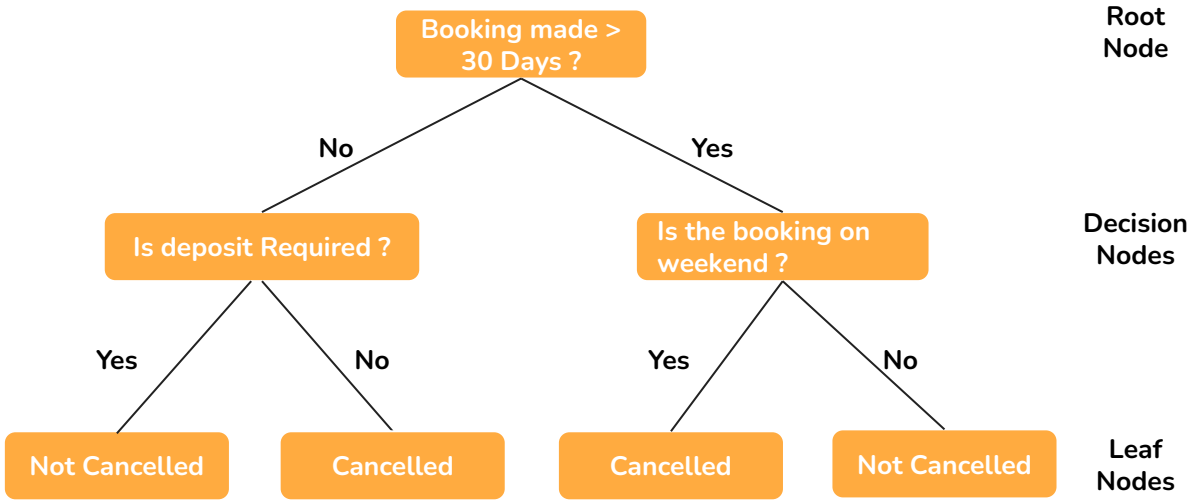
What is a decision tree and how does it work?

- One of the most popular and effective supervised learning techniques
- Works well with both categorical and continuous variables.
- Graphical representation of all the possible solutions to a decision that is based on a certain condition.
- Training sample points are split into two or more sets based on the split condition over input variables.



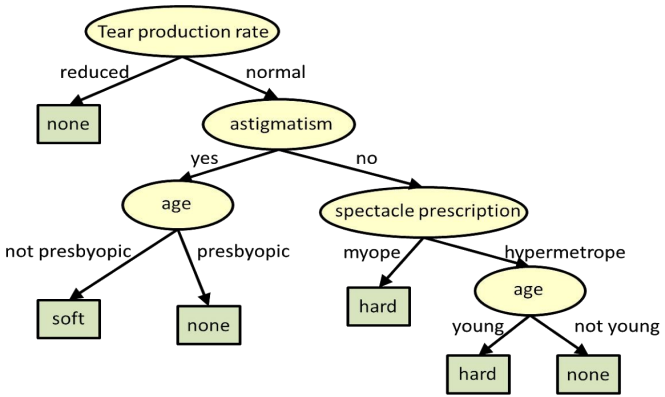
What are the common terminologies used in decision trees?

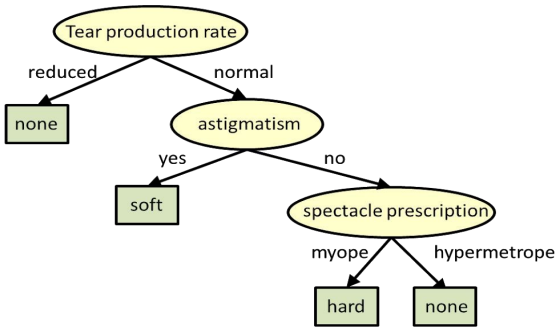
- Root node - Represent the entire set of the population which gets further divided into sets based on splitting decisions.
- Decision node - These are the internal nodes of the tree, These nodes are expressed through conditional expression for input attributes.
- Leaf node/Terminal node - Nodes that do not split further are known as leaf nodes or terminal nodes.
- Splitting - The process of dividing a node into one or more sub-nodes.



What is pruning? How does it avoid overfitting?

- One of the problems with the decision tree is it gets easily overfits the training data and becomes too large and complex.
- A complex and large tree poorly generalizes on new data whereas a small tree fails to capture the information of training sample data.
- Pruning shortens the branches of the tree. The process of reducing the size of the tree by turning some branch node into a leaf node and removing the leaf node under the original branch.
- By removing branches we can reduce the complexity of the tree which helps in reducing the overfitting of the tree





What are the different ways to prune a tree?

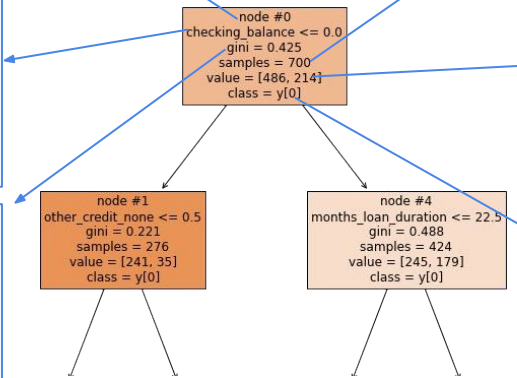
- There are two different ways to prune a tree:
  - Pre-Pruning: Early stopping of a tree before it has fully grown.
    - To perform pre-pruning - Hyper-parameter tuning is one of the ways to tune different hyperparameters of a tree and restrict the growth of the tree by limiting the hyperparameters such as max_depth,min_samples_split, etc.
  - Post-Pruning: Building a full tree and removing the sub-tree nodes.
    - To perform post-pruning - It can be done using the cost complexity pruning technique by choosing the appropriate value of ccp_alpha and removing the less significant sub-tree nodes.

How to read Python's visual output of a decision tree?

Node number : Represents the number of node

Decision Rule : checking _balance <= 0 means that every observation with a checking balance of 0 and less will follow the True arrow i.e. go to the left node in the next level, and the rest will follow the False arrow that is going to the right node in the next level.

GINI: Refers to the quality of the split. Always a number between 0.0 and 0.5,
0.0 -> High purity in the node
0.5 -> High impurity in the node.



Samples: 424 means that there are 424 observations in this node.

Value: List that tells how many samples at the given node fall into each category. value = [486,214] means that of these 700 observations at this level 486 belong to class 0 and 214 to class 1 as the
Class = y[0] represents the majority class 0 in that node.

Color intensity represents the concentration of the majority class in a node. Darker the color higher the concentration of the majority class.

Classification Metrics

Understanding Model Predictions

- Consider a problem of classifying emails as spam or not spam
- An ML model is built to do the classification
- Four possible outcomes

Case 1	Case 2	Case 3	Case 4
The email was not spam and the model also predicted that it would not be spam	The email was not spam but the model predicted that it would be spam	The email was spam and the model also predicted that it would be spam	The email was spam but the model predicted that it would not be spam

Confusion Matrix

- A table that helps you visualize the performance of a classification model
- Especially useful when you have a binary classification problem
    - Email: spam vs not spam
    - Loan Repayment Default: yes vs no
- Can be extended for classification task with more than two classes too

	Predicted Spam	Predicted Not Spam
Actual Spam	40	5
Actual Not Spam	5	50

Confusion Matrix

- **True Positives (TP):** The model predicted the class as positive, and it is actually positive.
- **True Negatives (TN):** The model predicted the class as negative, and it is actually negative.
- **False Positives (FP):** The model predicted the class as positive, but it is actually negative
  - Also known as a **Type I error**
- **False Negatives (FN):** The model predicted the class as negative, but it is actually positive
  - Also known as a **Type II error**

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Accuracy

- A basic measure used to evaluate the overall performance of a classification model
- Represents the ratio of correctly predicted instances to the total instances
- Example:
  - Out of 100 emails, an ML model correctly classified 90 emails as either spam or not spam
  - Accuracy = (# Correct Predictions) / (# Total Predictions) = 90/100 = 0.9 (or 90%)

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

Is Accuracy always a good metric?

- Consider a problem of identifying fraudulent transactions
- An ML model classifies all transactions as ‘Not Fraud’
- Accuracy of the model would be very high as the number of fraudulent transactions would be very low
  - If 10 out of 1000 transactions are fraudulent, the above model would have a 99% accuracy!
- But the model **missed the main point** in this case
- **We need better metrics in such cases**

Precision

- A metric used to measure the accuracy of positive predictions made by a model
- **Represents the ability of the model to reduce the number of false positives**

$$\text{Precision} = \frac{TP}{TP + FP}$$

- For example, marketing campaigns require a high precision value to ensure that a large number of potential customers will interact with their survey or be interested to learn more

Recall

- A metric that measures the proportion of actual positives that are correctly identified by the model
- **Represents the ability of the model to reduce the number of false negatives**

$$\text{Recall} = \frac{TP}{TP + FN}$$

- For example, algorithms used to detect credit card fraud need to have a very high recall to ensure that the maximum number of fraudulent transactions are flagged, as missing even a single fraudulent transaction would have repercussions

F1 Score

- In some business scenarios, we might need a model that has both low false positives as well as low false negatives
- In such cases, F1 score can be used

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

- For example, when detecting spam emails, we want a model to have low false positives as well as low false negatives

Great Learning

Happy Learning !