**Decision Tree Tutorial**

# 1. Introduction

A popular supervised machine learning approach called a decision tree divides data recursively according to feature values into a hierarchical, tree-like structure in order to anticipate results. It is praised for its simplicity and interpretability and is adaptable, working well for both classification and regression problems. Decision trees are used in a wide range of fields, including predictive maintenance, customer segmentation, financial analysis, and healthcare.

## Applications

* **Fraud Detection**: By identifying patterns in labeled data, decision trees assist in detecting fraudulent transactions.
* **Customer segmentation**: They help with focused marketing campaigns by grouping clients according to their purchasing patterns.
* **Predictive maintenance** uses past data to forecast when equipment will break down.
* **Healthcare:** Diagnosing diseases based on patient symptoms.

## Advantages

* Their simple hierarchical structure makes them easy to understand and depict.
* Handles both numerical and categorical data effectively.
* Characteristics that don't need to be scaled or normalized, and require very little data preprocessing.

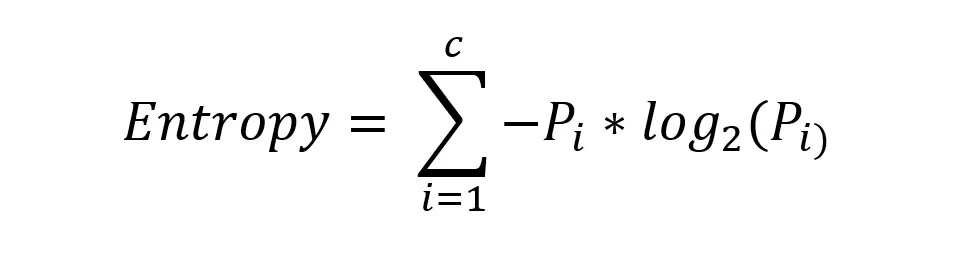
## Disadvantages

* Prone to overfitting, especially with deep trees.
* sensitive to outliers and noisy data, which can impair the ability to generalize.
* does not automatically facilitate online education (model updates are made incrementally).

# 2. Core Concepts

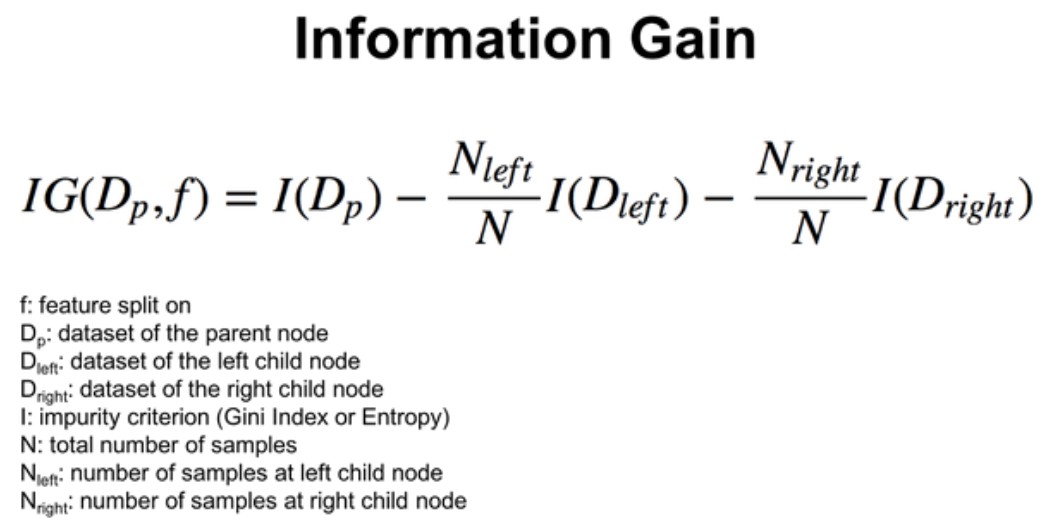
## Entropy

Entropy quantifies the degree of uncertainty or impurity in a dataset. It is employed to assess the split quality at every Decision Tree node. A purer dataset and less unpredictability are indicated by lower entropy.



## Information Gain

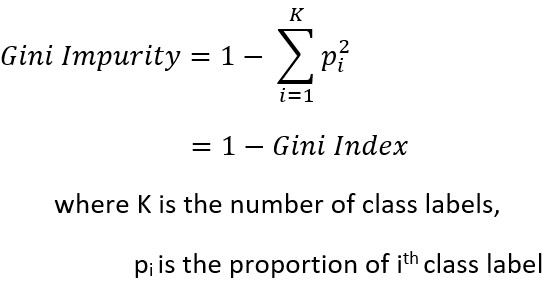
Information Gain After a split, information gain quantifies the decrease in entropy. A better split is indicated by a bigger Information Gain.



## Gini Impurity

The probability of erroneously identifying a randomly selected sample is measured by Gini Impurity. It is Scikit-learn's default splitting criterion.

Entropy



**Overfitting and Underfitting:**

* **Underfitting** : When a tree is too shallow, it underfits and misses patterns in the data.
* **Overfitting :** This occurs with excessively deep trees that reduce generalization to unknown data by memorizing noise in the data.

## Key Hyperparameters

1. **Max\_depth** : Limits the depth of the tree to prevent overfitting.
2. **Min\_samples\_split** : The bare minimum of samples needed to split a node.
3. **Min\_samples\_leaf :** Minimum samples required to form a leaf node.

# Explanation of How a Decision Tree Works

**The tree-building process consists of the following steps:**

1. **Start at the root node:**
   * The training data is all stored in the root node.
   * Determine the splitting criterion for each feature and threshold, such as entropy or Gini impurity.
   * Select the threshold and characteristic that produce the best split.
2. **Split the data:**
   * Using the chosen feature and threshold, divide the data into subsets. ● Every subset turns into a child node.
3. **Repeat Recursively:**
   * To choose the best feature and threshold for additional splitting, repeat the procedure for every child node.
4. **Stop Splitting :**
   * + Stop splitting when a stopping criterion is met, such as:
     + A node's samples are all members of the same class.
     + It reaches its maximum depth.
     + A node's sample count is less than the minimal threshold (min\_samples\_split).
5. **Assign Class Labels:**
   * Assign a class label to the leaf nodes according to the samples' predominant class.

# Example of How a Decision Tree Works

Let’s take sample data and break down how the split works.

**Scenario**: Using a Titanic passenger's characteristics, determine if they survived.

Sample Dataset :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Passenger ID | Pclass | Sex | Age | Fare | Survived |
| 1 | 1 | Female | 29 | 211.0 | 1 |
| 2 | 3 | Male | 22 | 7.25 | 0 |
| 3 | 3 | Male | 24 | 8.05 | 0 |
| 4 | 2 | Female | 35 | 26.0 | 1 |

**Step-by-Step Tree Building:**

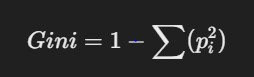
**1. Understand the dataset**

We have the following dataset features:

* Sex : Categorical(Male or Female)
* Pclass : Categorical (Passenger class : 1,2,3)
* Survived : Target variable(0=Did not survive, 1=Survived)

Our goal is to use a passenger's qualities to forecast whether or not they survived. At the root node, possible splits are assessed to start the tree-building process. **2. Step 2: Calculate the Impurity of the Root Node**

* **Gini Impurity**:



where the percentage of samples that survived or did not survive in each class is denoted by Pi

**Example** :

* If 40% of passengers survived (P1 = 0.4) and 60% of passengers perished (P0 = 0.6), then:

Gini =1-((0.6x0.6) + (0.4x0.4)) = 0.48

* **Entropy** :



Using the same proportions :

Entropy = -(0.6log2(0.6) + 0.4log2(0.4)) ≈ 0.97

The root node's impurity is measured by these metrics. Higher homogeneity is indicated by a lower value.

**Step 3: Evaluate Splits for Each Feature :**

The tree determines which feature to split on by calculating the amount of impurity that is decreased when the dataset is split according to each feature.

**1. Splitting by Sex:**

* Divide the dataset into 2 groups :

Group 1 : Passengers with Sex = Female

Group 2 : Passengers with Sex = Male

* For each group, calculate the Gini Impurity or Entropy Example :

Group 1(Females) : 70% survived, 30% did not.

Gini=1−(0.7X0.7+0.3X0.3)=0.42

Group 2(Males) : 20% survived, 30% did not.

Gini=1−(0.2x0.2+0.8x0.8)=0.32

* Calculate the **weighted average impurity** for the split:



* To calculate the Information Gain, compare this number with the root node's impurity.

**2. Splitting the Pclass :**

* Divide the dataset into 3 groups:

Group 1 : Passengers in Pclass = 1.

Group 2 : Passengers in Pclass = 2.

Group 3 : Passengers in Pclass = 3.

* For each group, calculate the Gini Impurity or Entropy.

Example :

Group 1: 80% survived, 20% did not

Gini=1−(0.8x0.8+0.2x0.2)=0.32

Group 2 : 50% survived, 50% did not

Gini=1−(0.5x0.5+0.5x0.5)=0.5

Group 3 : 10% survived, 90% did not

Gini=1−(0.1x0.1+0.9x0.9)=0.18

**3. Calculate Weighted Average Impurity for Sex**

From previous calculations:

* **Group 1 (Female):**

○ Gini Impurity: GiniGroup1=0.42

○ Size: ∣Group1∣=30 ● **Group 2 (Male):**

○ Gini Impurity: GiniGroup2=0.32

○ Size: ∣Group2∣=70

* **Total Size:** ∣Total∣=100

The **Weighted Impurity** for the Sex split is calculated as:

Weighted ImpuritySex = ∣Group1∣ / ∣Total∣ x GiniGroup1 + ∣Group2∣ / ∣Total∣ x GiniGroup2

Weighted ImpuritySex = 30 / 100 x 0.42 + 70 / 100 x 0.32

Weighted ImpuritySex = 0.3 x 0.42 + 0.7 x 0.32

Weighted ImpuritySex = 0.126 + 0.224 = 0.35

**4. Calculate Information Gain for Sex**

The **Information Gain (IG)** is the reduction in impurity:

IGSex = GiniRoot − Weighted ImpuritySex

From previous steps:

* GiniRoot=0.48
* Weighted ImpuritySex=0.35

IGSex=0.48 − 0.35=0.13

**5. Calculate Weighted Average Impurity for Pclass**

From previous calculations:

* **Group 1 (Pclass = 1):**

○ Gini Impurity: GiniGroup1=0.32

○ Size: ∣Group1∣=20 ● **Group 2 (Pclass = 2):**

○ Gini Impurity: GiniGroup2=0.50

○ Size: ∣Group2∣=30 ● **Group 3 (Pclass = 3):**

○ Gini Impurity: GiniGroup3=0.18

○ Size: ∣Group3∣=50

* **Total Size:** ∣Total∣=100

The **Weighted Impurity** for the Pclass split is calculated as:

Weighted ImpurityPclass = ∣Group1∣ / ∣Total| x GiniGroup1 + ∣Group2| / ∣Total x ⋅ GiniGroup2

+ ∣Group3| / ∣Total∣ x ⋅GiniGroup3

Weighted ImpurityPclass =20 / 100 x 0.32 + 30 / 100 x 0.50 + 50 / 100 x 0.18

= 0.2 x 0.32 + 0.3 x 0.50 + 0.5 x 0.18

0.304

**6. Calculate Information Gain for Pclass** The **Information Gain (IG)** for Pclass is:

IGPclass =GiniRoot −Weighted ImpurityPclass

From previous steps:

* GiniRoot=0.48
* Weighted ImpurityPclass=0.304
* IGPclass=0.48−0.304=0.176

**7. Compare Information Gain**

* **Information Gain for Sex:** IGSex=0.13
* **Information Gain for Pclass:** IGPclass=0.176

**8. Select the Best Feature**

The feature with the highest Information Gain is selected for splitting. In this case:

* **Pclass** has a higher Information Gain (0.176>0.130.176 > 0.130.176>0.13) than Sex.
* **Result:** The Decision Tree will split on Pclass at the root node.

## Next Steps

* After splitting on Pclass, the process repeats for each child node.
* At each subsequent node, impurity and Information Gain are recalculated to determine the best feature for further splitting.

# We will now use a dataset from Kaggle and Python code to complete the same task: 3. Dataset

A great example for demonstrating Decision Trees is the Titanic dataset, which was taken from the Kaggle Titanic competition. It is a binary classification problem where the goal is to predict survival based on passenger attributes.

**Features**

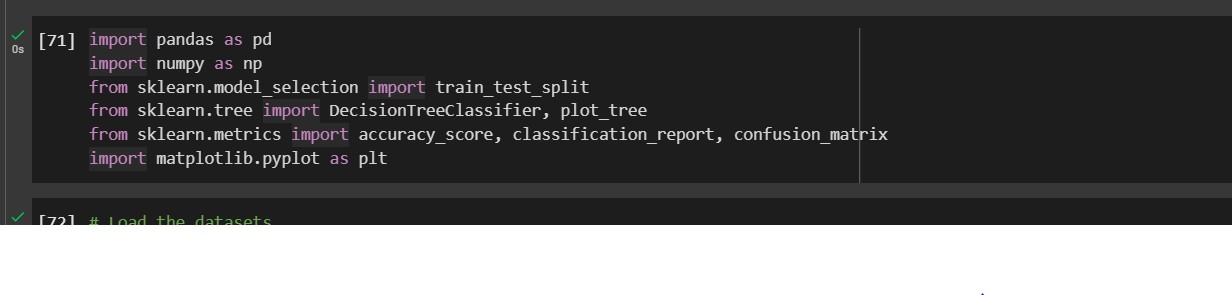
1. **Pclass (Passenger Class) :** Proxy for socioeconomic status.
2. **Sex :** Male or Female.
3. **Age :** Passenger’s Age.
4. **Fare :** Ticket fare price.
5. **Embared :** Port of embarkation(C, Q, S).

**Target Variable**

* **Survived :** Binary outcome (0 = Did not survive, 1 = Survived).

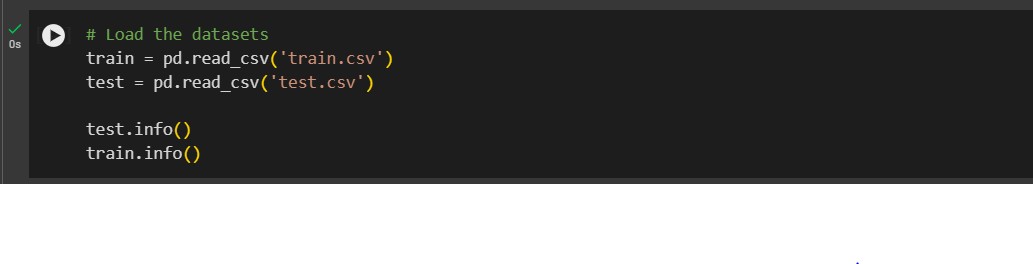
**Preprocessing Steps**

* Initially we need to import libraries to ensure we have tools for data preprocessing, model training, evaluation and visualisation.



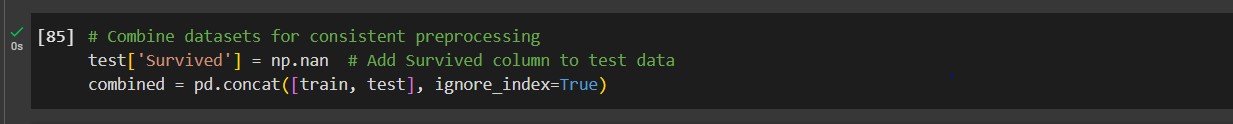
**Loading the datasets**

* Here we are loading the training and testing data and printing the information of the both datasets. info() gives information like how many columns we have and what are the datatypes present and no-null count.



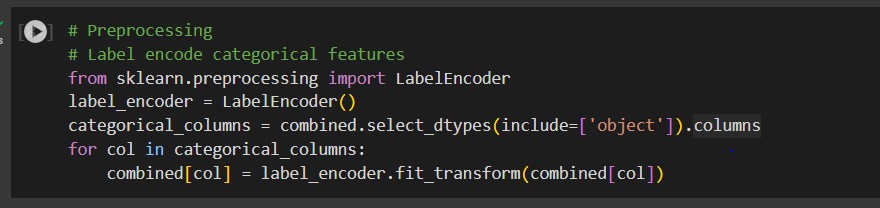
**Combining training and testing dataset.**

* To guarantee consistent structure, include a Survived column in the test dataset containing NaN values.
* To ensure uniform preparation, merge the test and train datasets.
* Applying preprocessing methods consistently across all data is made easier by combining datasets.



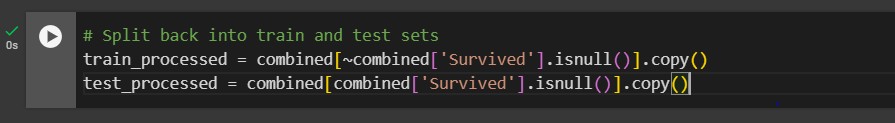
**Label Encoding**

* Use label encoding to translate category variables into numerical values.
* Machine learning models, which often do not directly operate with categorical data, require this.
* All data is guaranteed to be numeric and prepared for model training in this step.

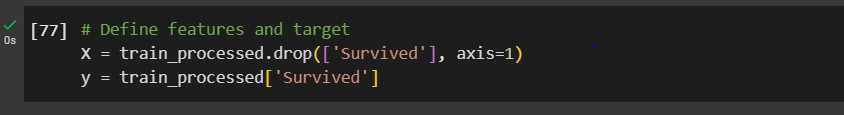


**Splitting target and features**

* In the below step we are splitting back the data into test and train for model building.



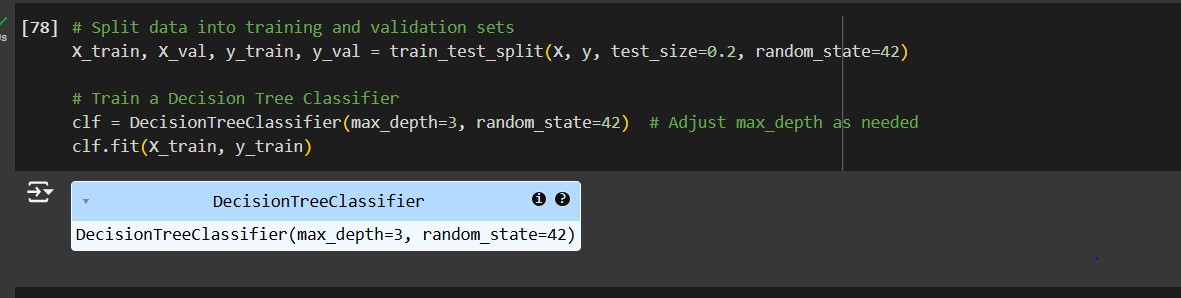
* Defining target and features



* In the below step we are splitting the data into test size in 20% of the data and 80% of the training set. We are using random\_state to ensure reproducibility of results.

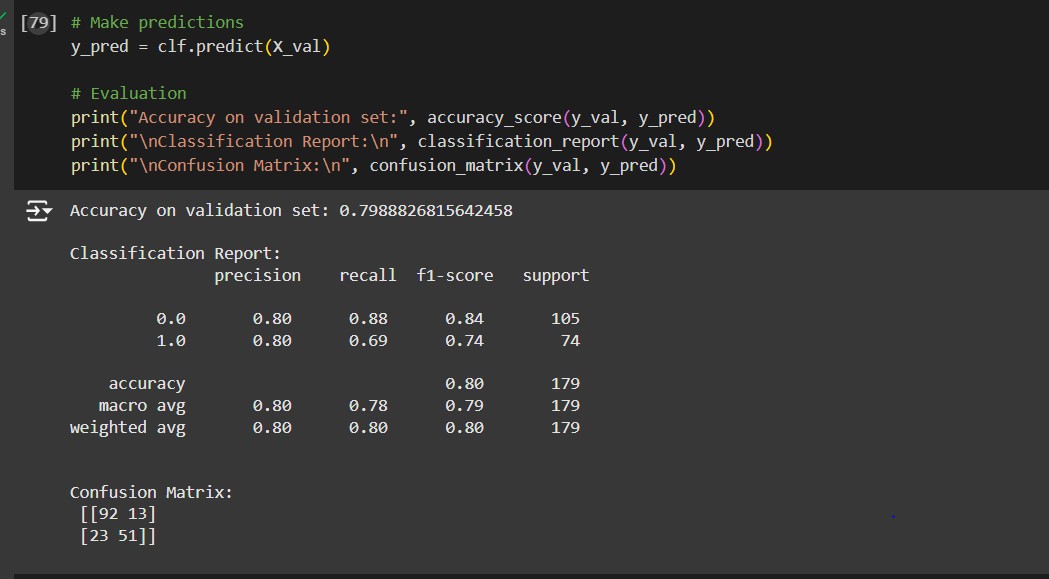
**Training the Decision tree**

* Here we are initializing the decision tree classifier with maximum depth of 3. Training the model data with X\_train and y\_train and predicting the outcomes on the validation set(x\_val)



**Evaluating the model**

* In the below step we are evaluating the model based on classification as this is a classification model. Classification models can be evaluated using a confusion matrix.



* From the above figure we can say that we got around 80% of accuracy from the model building.

**Validation Accuracy : ~ 80**

|  |  |  |
| --- | --- | --- |
| Metric | Class 0(Did not survive) | Class 1 (Survived) |
| Precision | 0.80 | 0.80 |
| Recall | 0.88 | 0.69 |
| F1-Score | 0.84 | 0.74 |

**Confusion matrix**

|  |  |  |
| --- | --- | --- |
|  | Predicted 0 | Predicted 1 |
| Actual 0 | 92 | 13 |
| Actual 1 | 23 | 0.74 |

**Insights:**

* Class 0 (Did Not Survive) is predicted with higher recall compared to Class 1. ● The model struggles slightly with predicting Class 1 due to class imbalance.

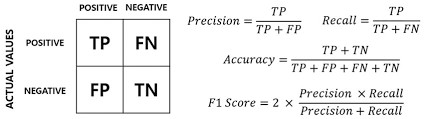
**Metrics in the Classification report**

**Precision :** The percentage of accurate positive predictions (True Positives) among all forecasts for that class is known as precision.

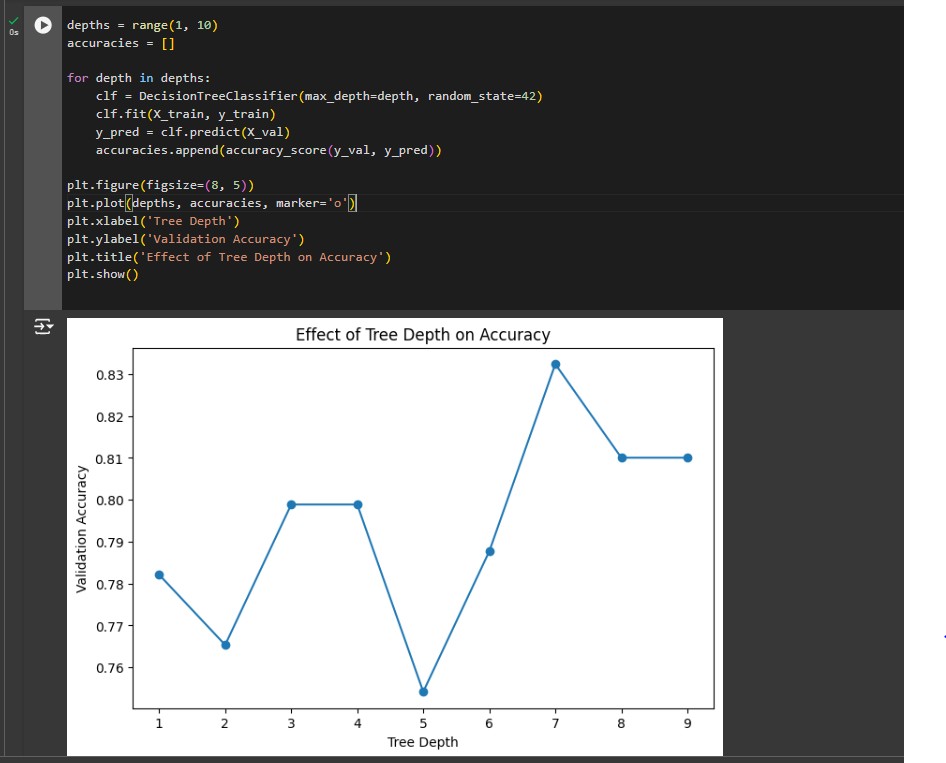
**Recall**: Recall (also called Sensitivity) is the proportion of actual positives that were correctly identified.

**F1-Score :** The F1-score is the harmonic mean of Precision and Recall, providing a balanced metric when there is class imbalance.

**Accuracy :** One metric used to assess a classification model's overall performance is accuracy. It calculates the percentage of accurate predictions the model made out of all the guesses.



**Visualizing the decision tree**



* In the above figure it shows how the accuracy of the decision tree changes based on the depth of the tree. After the decision tree depth 7 the accuracy of the model dropped from 83% to 82% it is where the overfitting of the model started.
* We have to make sure that the model is not overfitted or under fitted.
* Decision tree is very prone to getting overfitted as we need to check each and every depth of the model very carefully.

# 4. Conclusions

## Strengths

* Decision trees are simple to use and interpret.
* They manage numerical and categorical data without requiring a lot of preparation.

## Limitations

* Prone to overfitting in the absence of appropriate tuning (limiting tree depth, for example).
* sensitive to data outliers and noise.

## Recommendations

* To increase performance, employ ensemble techniques like Random Forest or Gradient Boosting.
* Try varying hyperparameters such as min\_samples\_leaf and min\_samples\_split.

**Github Link**

[**Machine Learning Tutorial**](https://github.com/ankarapurenusri/Machine-learning-tutorial)

# References

* Kaggle Titanic Dataset: <https://www.kaggle.com/competitions/titanic>
* Scikit-learn Documentation: <https://scikit-learn.org/stable/modules/tree.html>