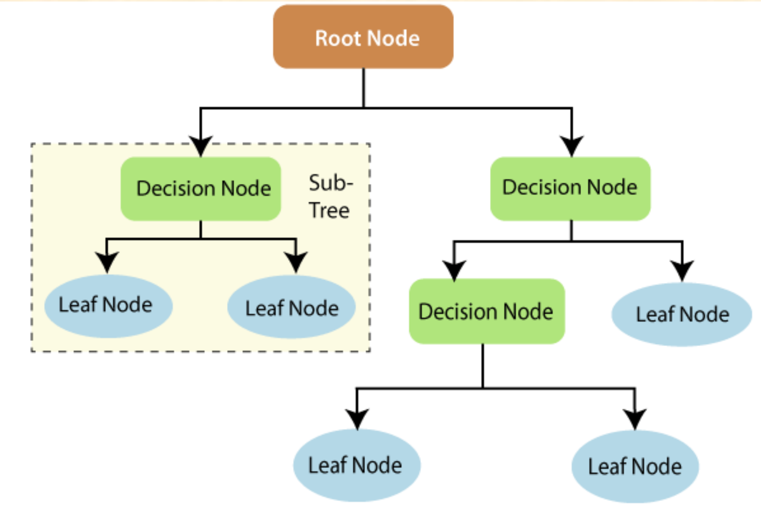
Decision Tree

- The decision tree from the name itself signifies that it is used for making decisions from the given dataset.

- The concept behind the decision tree is that it helps to select appropriate features for splitting the tree into subparts.

- the algorithm used behind the splitting is ID3.



For checking the purity, we have 2 features:

1. **Entropy**: It builds an appropriate decision tree for selecting the best splitter.

$$H(s) = -P_{(} + \log_2 P_{(}+) - P_{(}-) \log_2 P_{(-)}$$

Here $P_{(}+) / P_{(-)} = \%$ of + ve class 1% of - ve class

Entropy can be defined as a measure of the purity of the sub split. The value lies between 0 and 1.

The algorithm calculates the entropy of each feature using the above formula, after every split and as the splitting continues, it selects the best feature and starts splitting according to it.

2. **Gini Impurity:** Gini impurity is a metric to measure how often a randomly chosen element would be incorrectly identified.

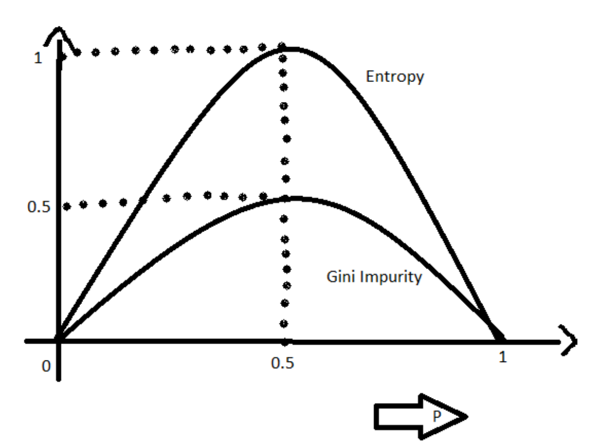
$$GiniIndex = 1 - \sum_j p_j^2$$

It means an attribute with lower Gini index should be preferred.

In the words of Wikipedia, its main goal is to measure how often a randomly chosen element from the set would be incorrectly labeled.

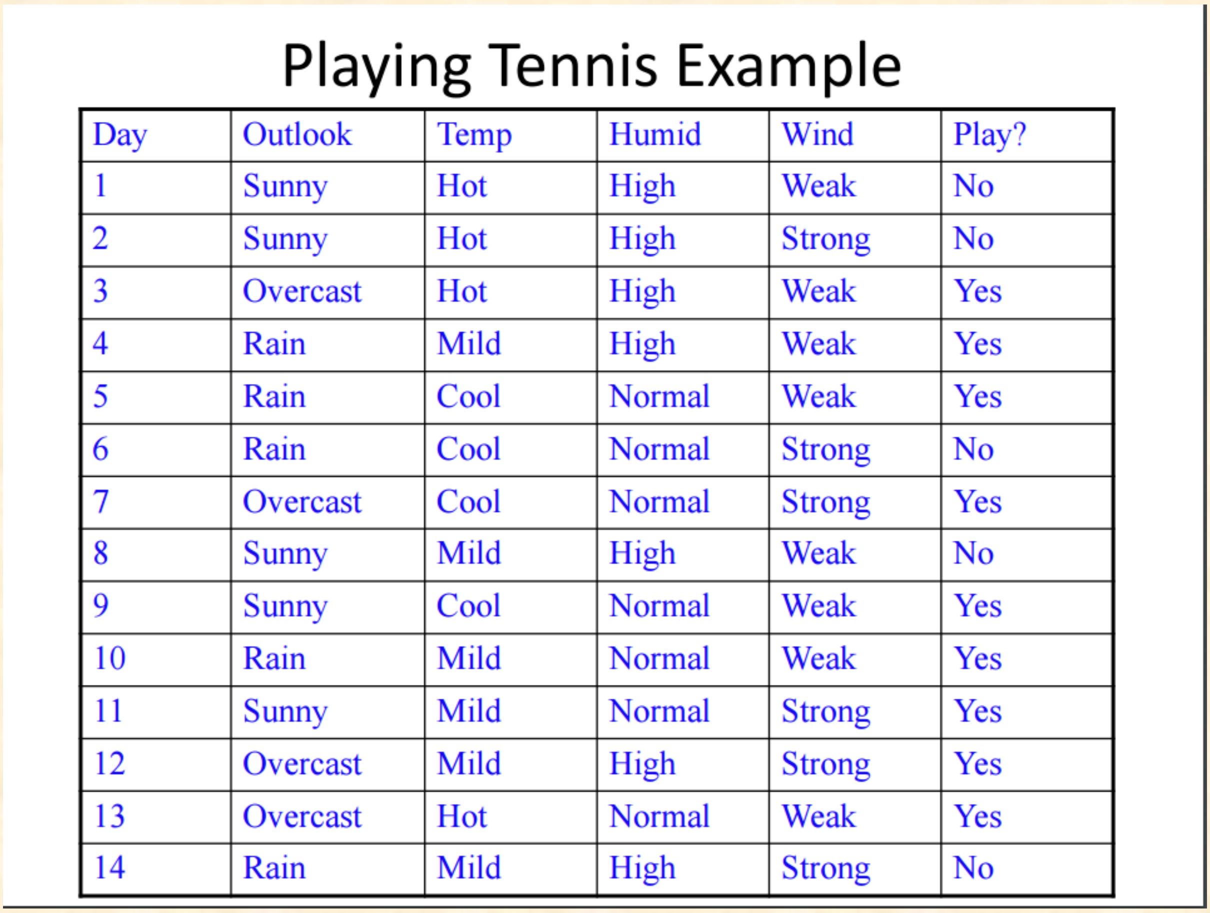
Difference between Entropy and Gini Indexing:

Entropy	Gini Indexing
1. Entropy can be defined as a measure of the purity of the sub split	**1.** It shows that the data point that we have randomly chosen for the splitting from the dataset how it is incorrectly labeled.
2. It is used for C4.5 decision tree.	2. It is used for CART
3. Higher values give inappropriate result,	3. It gives more accurate value and less than the entropy index which is its best quality, the small values show less impurity.
4.Takes more time	4.Takes less time
5. Entropy has a maximum impurity of 1 and maximum purity is 0.	5.The Gini index has a maximum impurity is 0.5 and maximum purity is 0



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

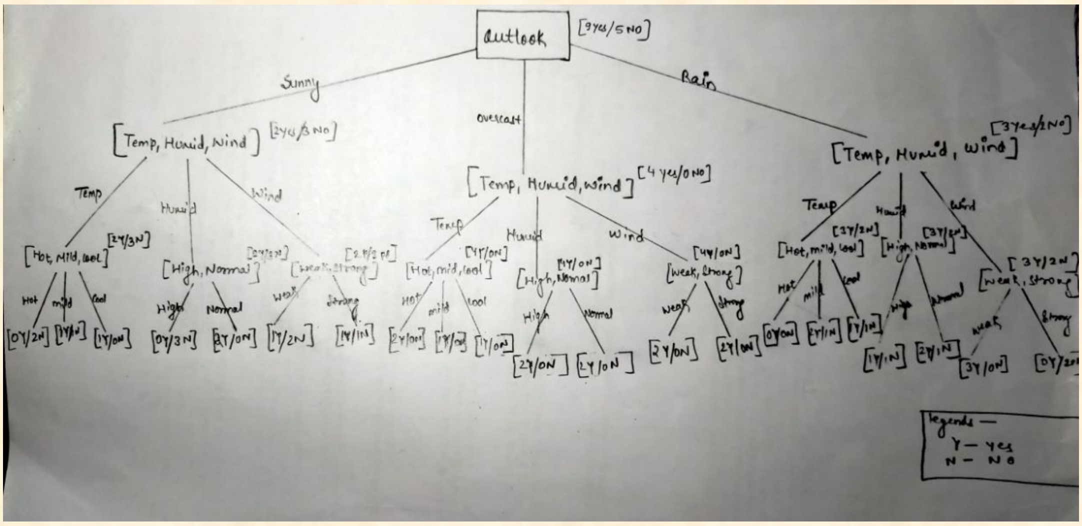
$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$



Playing Tennis Example

Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision tree w.r.t above Tennis dataset



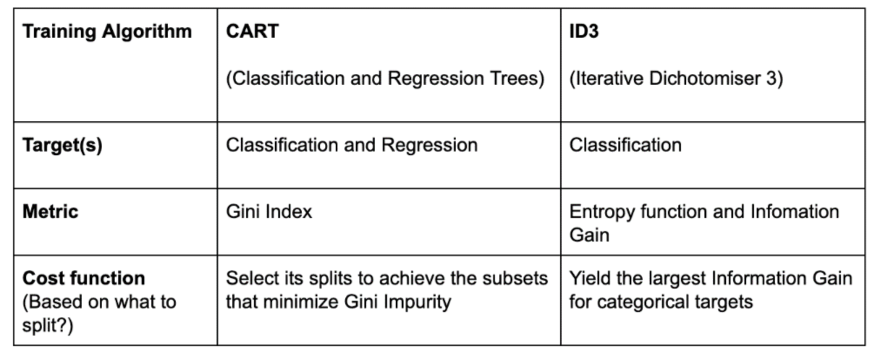
Link for clear pic image

Information Gain: Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way.

It is commonly used in the construction of decision trees from a training dataset, by evaluating the information gain for each variable, and selecting the variable that maximizes the information gain, which in turn minimizes the entropy and best splits the dataset into groups for effective classification.

Information Gain = Entropy before splitting - Entropy after splitting

Some differences



Training Algorithm	**CART** (Classification and Regression Trees)	**ID3** (Iterative Dichotomiser 3)
Target(s)	Classification and Regression	Classification
Metric	Gini Index	Entropy function and Infomation Gain
Cost function (Based on what to split?)	Select its splits to achieve the subsets that minimize Gini Impurity	Yield the largest Information Gain for categorical targets

Practical Implementation

In [3]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
df=pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Wine-Quality-Dataset
/master/winequality-red.csv",sep=';')
```

In [5]:

```
df
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
240
```

In [9]:

```
df.shape
```

Out[9]:

```
(1599, 12)
```

In [10]:

```
df=df.drop_duplicates()
```

In [11]:

```
1599-240
```

Out[11]:

```
1359
```

In [12]:

```
df.shape
```

Out[12]:

```
(1359, 12)
```

In [13]:

```
df['quality'].unique()
```

Out[13]:

```
array([5, 6, 7, 4, 8, 3])
```

In [14]:

```
df['quality'].value_counts()
```

Out[14]:

```
5    577
6    535
7    167
4     53
8     17
3     10
Name: quality, dtype: int64
```

In [15]:

```
from sklearn.model_selection import  train_test_split, GridSearchCV
```

In [16]:

```
X=df.drop('quality',axis=1)
```

In [17]:

```
y=df['quality']
```

In [18]:

```
X
```

Out[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1593	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1359 rows × 11 columns

In [19]:

```
y
```

Out[19]:

```
0       5
1       5
2       5
3       6
5       5
       ..
1593    6
1594    5
1595    6
1597    5
1598    6
Name: quality, Length: 1359, dtype: int64
```

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1
0)
```

In [21]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [22]:

```
model=DecisionTreeClassifier()
```

In [23]:

```
model.fit(X_train,y_train)
```

Out[23]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [24]:

```
model.score(X_train,y_train)
```

Out[24]:

```
1.0
```

In [25]:

```
y_pred=model.predict(X_test)
```

In [26]:

```
from sklearn.metrics import accuracy_score
```

In [27]:

```
accuracy_score(y_test,y_pred)
```

Out[27]:

```
0.5033407572383074
```

In [28]:

```
# Logistic Regression, SVM,( RF, XGBoost, GB, AB)
```

In [35]:

```
grid_param={
          'criterion':['gini','entropy'],
          'max_depth':range(2,32,1),
          'min_samples_leaf':range(1,10,1),
          'min_samples_split':range(2,10,1),
          'splitter':['best',"random"]}
```

In [36]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=model,param_grid=grid_param,cv=5)
```

In [43]:

```
grid_search.fit(X_train,y_train)
```

Out[43]:

```
▸          GridSearchCV
▸ estimator: DecisionTreeClassifier
 ▸       DecisionTreeClassifier
```

In [39]:

```
grid_search.best_params_
# best parameter out of all
```

Out[39]:

```
{'criterion': 'gini',
 'max_depth': 4,
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'splitter': 'random'}
```

In [41]:

```
model_with_best_params=DecisionTreeClassifier(criterion= 'gini',
 max_depth= 4,
 min_samples_leaf= 1,
 min_samples_split= 4,
 splitter= 'random')
```

In [44]:

```
model_with_best_params.fit(X_train,y_train)
```

Out[44]:

```
▾                        DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, min_samples_split=4, splitter='random')
```

In [45]:

```
y_pred_2=model_with_best_params.predict(X_test)
```

In [46]:

```
accuracy_score(y_test,y_pred_2)
```

Out[46]:

```
0.5278396436525612
```

In [47]:

```
# Accuracy Increased
```

In [ ]:

```

```