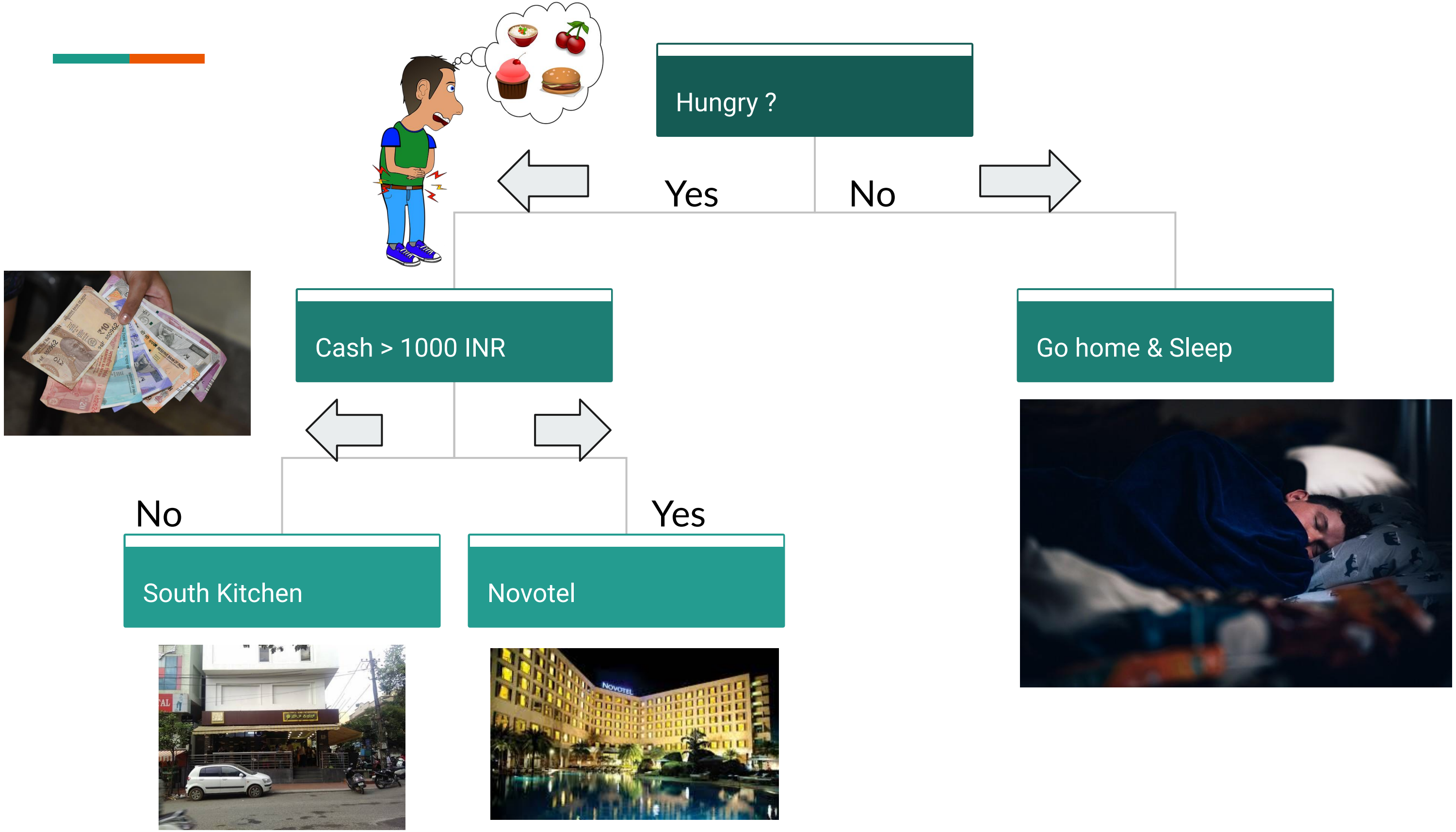
Decision Tree Model

Nachiketh

AWS Machine Learning Certification Course
by Manifold AI Learning

Deciding whether to go to a Restaurant or home ?

Decision Tree in Real Life



Decision Tree

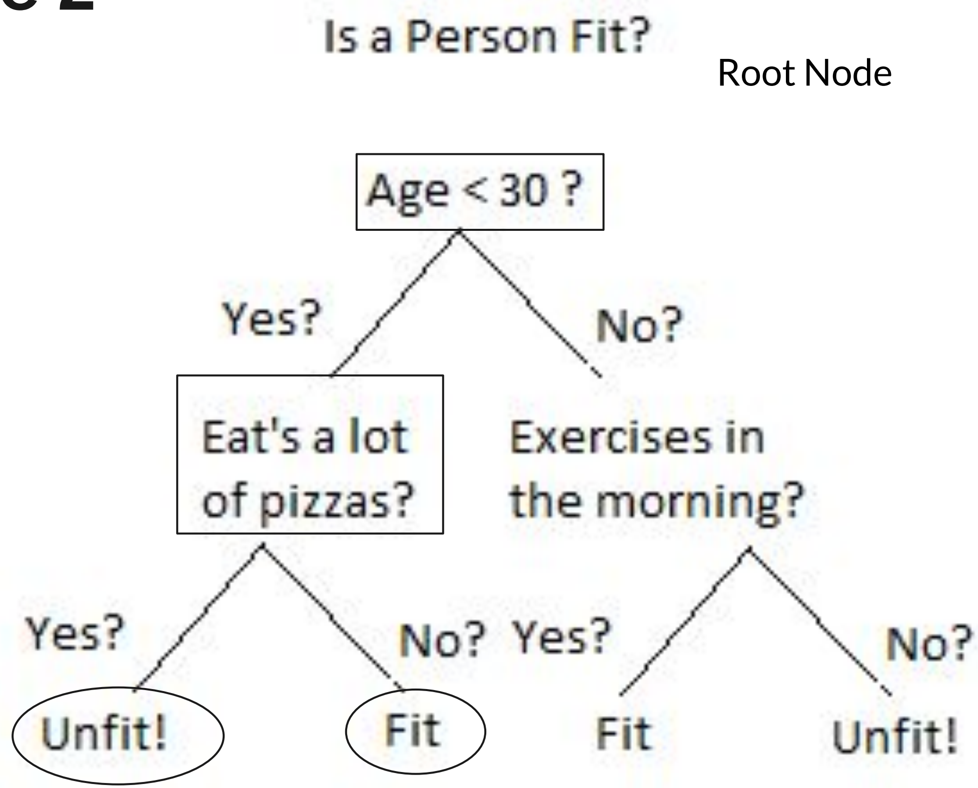
A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter.

Each Decision Tree Contains

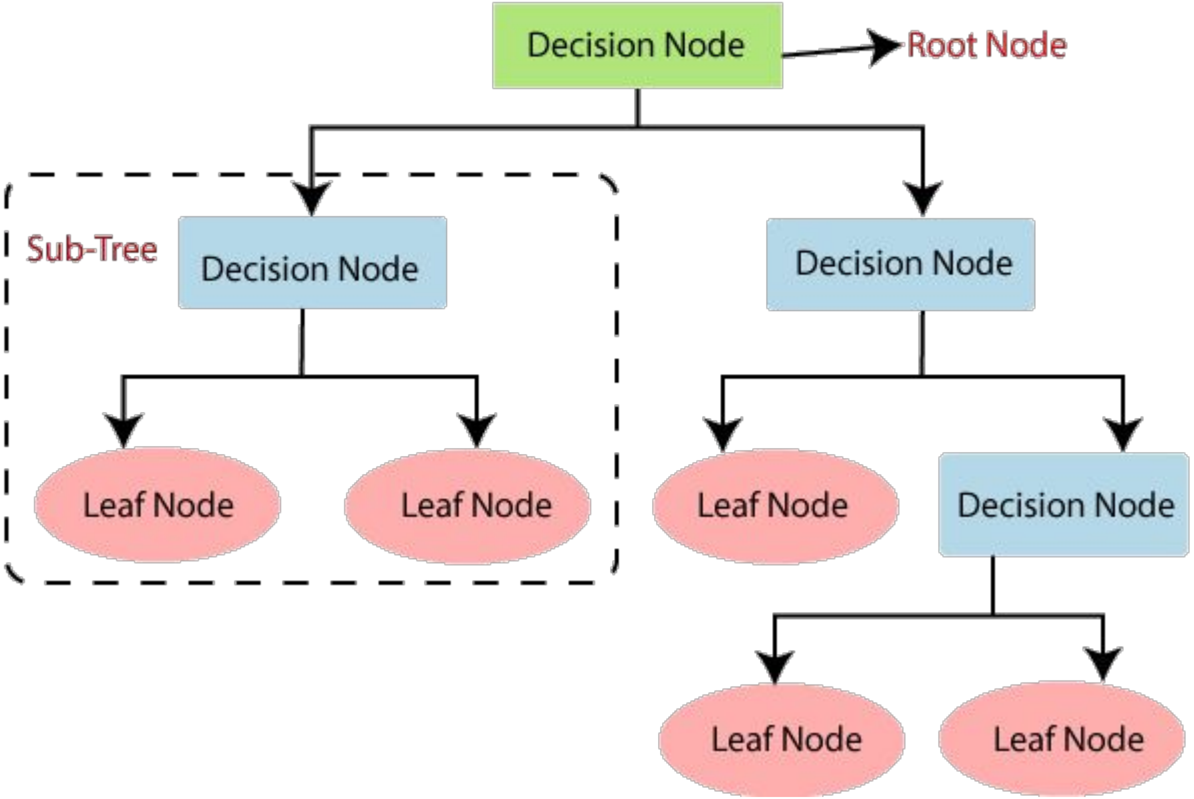
1. **Nodes** : Test for the value of a certain attribute.
2. **Edges/ Branch** : Correspond to the outcome of a test and connect to the next node or leaf.
3. **Leaf nodes** : Terminal nodes that predict the outcome (represent class labels or class distribution).

Decision Tree - Example 2

Check Wheter the person is fit or not ?



Parts of Decision Tree



Types of Decision Tree

1. Classification Trees (Yes/No)

   Binary recursive partitioning. An iterative process to split the data into partitions

2. Regression Decision Trees (Predicting a Real Value)

   The partition of the data space into cluster (or dense) regions and empty (or Sparse) regions

Method Followed

Basic Working of Divide & Conquer Rule

1. Select a test for root node. Create branch for each possible outcome of the test.
2. Split instances into subsets. One for each branch extending from the node.
3. Repeat recursively for each branch, using only instances that reach the branch.
4. Stop recursion for a branch if all its instances have the same class.

Steps for generating Decision Tree

1. Start with Complete Training data in the root node
2. Decide on measure of impurity (Gini Impurity Index or Entropy). Search for predictor variable that minimizes the impurity.
3. Repeat step 2 for each subset of the data until :
   a. All dependent variables are exhausted
   b. Stopping criteria are met
4. Generate Business Rules for the leaf

Gini Impurity Index & Entropy

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

where $p(c_i)$ is the probability/percentage of class c_i in a node.

Prerequisites - Run the Commands on Anaconda Prompt

- Pydotplus
  - `$ conda install pydotplus`

- GraphViz
  - `$ conda install -c anaconda graphviz`

Dataset

Breast Cancer Wisconsin (Diagnostic) Data Set

<u>Importing the Data:</u>

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
```

About the Dataset

```
**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)
```

Malignant = 0
Benign = 1

```
- class:
        - WDBC-Malignant
        - WDBC-Benign
```

Hands-On Implementation

1. Model Initialization

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

```
clf_tree = DecisionTreeClassifier(criterion = "gini",
                                  max_depth = 3)
```

2. Fit

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

3. Predict and Evaluate

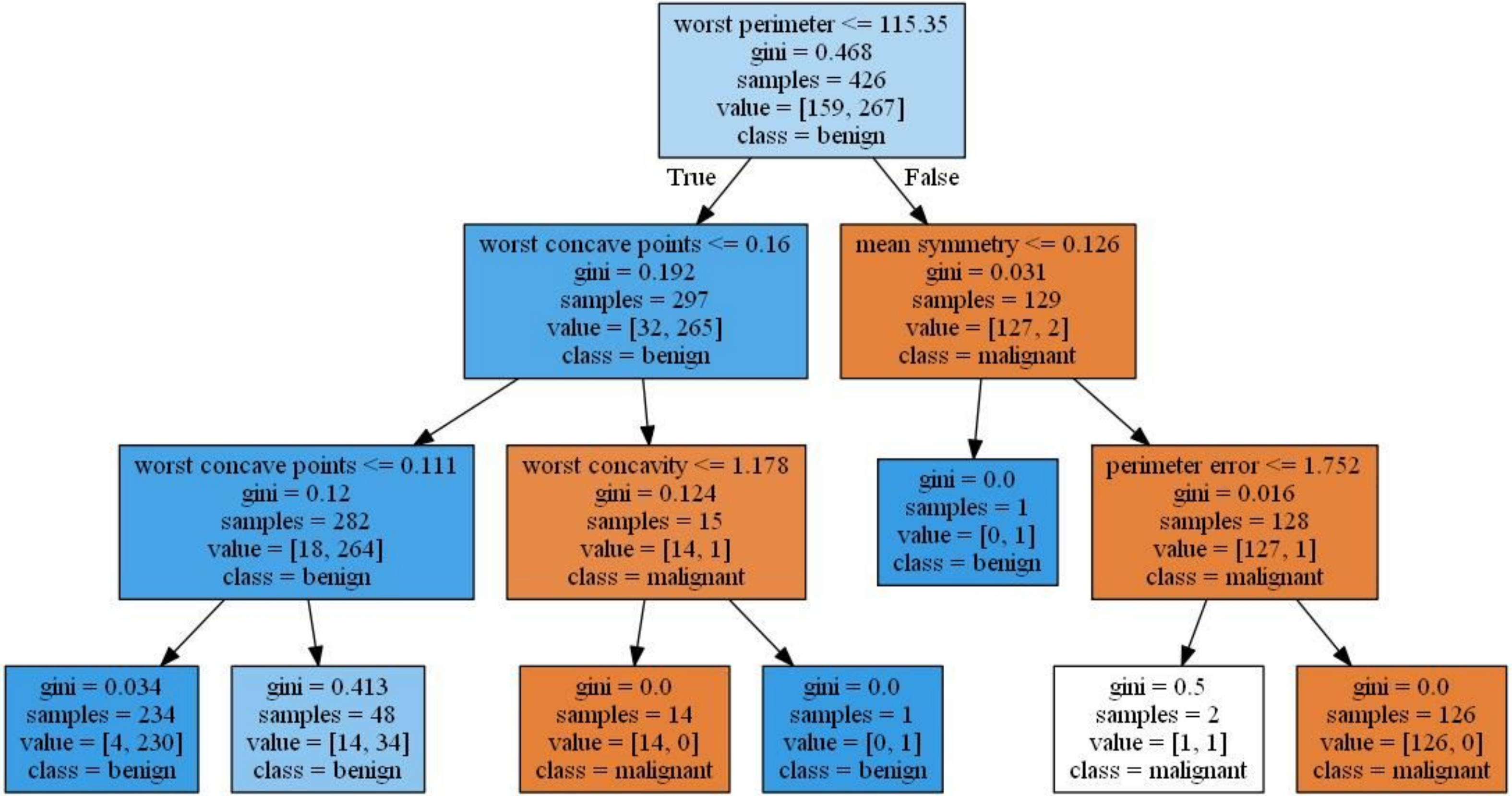
```
In [13]: y_pred = clf_tree.predict(X_test)

In [14]: from sklearn import metrics

In [15]: metrics.roc_auc_score(y_test,y_pred)

Out[15]: 0.9361635220125787
```

Generate Decision Tree



GridSearchCV - Hyperparameter Tuning

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, scoring=None, n_jobs=None, iid='deprecated', refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False)
```
[source]

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified.

Arguments for GridSearchCV

- estimator : A scikit-learn model which implements the estimator interface - ML Model
- param_grid : A dictionary with parameter names (string) as keys and lists of parameter settings to try as values
- scoring : Is a string ; an accuracy measure : i.e, roc_auc
- cv : integer values (Specifies the number of folds for k-fold)

GridSearchCV

```
from sklearn.model_selection import GridSearchCV
tuned_parameters = [{"criterion" : ["gini","entropy"],
                    "max_depth" : range(2,15)}]

clf_tree = DecisionTreeClassifier()

clf = GridSearchCV(clf_tree,
                  tuned_parameters,
                  cv = 10,
                  scoring = "roc_auc")
```

Advantages of Decision Tree

1

Inexpensive to construct.

2

Extremely fast at classifying unknown records.

3

Easy to interpret for small-sized trees

4

Accuracy comparable to other classification techniques for many simple data sets.

5

Excludes unimportant features.

Disadvantages of Classification with Decision Trees:

1

Easy to overfit.

2

Decision Boundary restricted to being parallel to attribute axes.

3

Decision tree models are often biased toward splits on features having a large number of levels.

4

Small changes in the training data can result in large changes to decision logic.

5

Large trees can be difficult to interpret and the decisions they make may seem counter intuitive.