# 8-Decision_Tree

October 20, 2024

## 1 Decision Tree

A Decision Tree is a supervised machine learning algorithm used for classification and regression tasks. It works by breaking down a dataset into smaller and smaller subsets while incrementally developing a tree structure. Each internal node of the tree represents a test or decision based on an attribute (feature), each branch represents the outcome of the decision, and each leaf node represents a class label (in classification) or a continuous value (in regression).

- **Root node**: The top node, which represents the entire dataset and is split based on the most important feature.
- **Internal nodes**: Represent the attributes or features that are used to split the data.
- **Leaf nodes**: Represent the final class or output value.

The main goal of a Decision Tree is to create a model that predicts the target value by learning simple decision rules inferred from the data features.

**When to Use Decision Trees?** Decision Trees are useful when you want a model that:

- **Can handle both categorical and numerical data**: Decision Trees are flexible and can work well with different types of data.
- **Is easy to interpret**: Since Decision Trees mimic human decision-making processes, they are intuitive and easy to understand, even for non-experts.
- **Requires minimal data preprocessing**: Unlike many other algorithms, Decision Trees do not require normalization or scaling of data.
- **Can model non-linear relationships**: Decision Trees can capture complex patterns in data, including interactions between features.

However, Decision Trees are especially effective when:

- The dataset is not too large, as deep trees can become computationally expensive.
- There is a need for a simple, interpretable model.
- You are working with a problem that involves a sequence of decision

**How Does a Decision Tree Work?**

1. **Feature Selection and Splitting**: The tree starts with all the data at the root node. It then selects a feature and splits the data based on this feature to form child nodes. The feature is chosen by evaluating different splitting criteria (more on this later).
2. **Recursive Partitioning**: This process of splitting the data continues recursively at each child node, selecting the best feature at each step until one of the following conditions is met:
   - All samples at a node belong to the same class (for classification).
   - The node reaches a pre-defined depth (maximum depth of the tree).
   - The number of samples at a node is less than the minimum split size.
3. **Prediction**: Once the tree has been constructed, it can be used to classify new samples by passing them down the tree, following the decisions at each node until a leaf node is reached. The class label (or value for regression) at the leaf node is the model's prediction.

**Who Should Use Decision Trees?** Decision Trees are ideal for:

- **Business analysts and decision-makers**: The model is easy to interpret and can provide insights into the data and important decision points.
- **Data scientists and machine learning engineers** who need to solve classification or regression problems.
- **Researchers** in fields like biology, medicine, or finance who deal with complex datasets involving interactions between multiple features.

**Advantages of Decision Trees:**

- **Easy to interpret**: Even non-technical people can understand the decision process.
- **Handles both numerical and categorical data**: Can work with different types of features without requiring feature transformation.
- **No need for feature scaling**: Unlike algorithms like SVM or KNN, Decision Trees don't require scaling.
- **Can capture non-linear relationships**: Decision Trees can split data at any point, allowing them to model complex relationships.

**Disadvantages of Decision Trees:**

- **Prone to overfitting**: Decision Trees can create very complex models that overfit the training data, especially when the tree grows too deep.
- **Unstable**: Small changes in the data can result in significantly different trees.
- **Bias towards dominant classes**: In unbalanced datasets, the tree might be biased toward the dominant class.

To avoid these issues, Decision Trees are often pruned (cutting off branches that do not provide additional information) or ensemble methods like Random Forest or Gradient Boosting are used to create a more robust model.

**Real-World Applications:**

1. **Loan Default Prediction**: Financial institutions use Decision Trees to classify loan applicants based on their risk profile.
2. **Medical Diagnosis**: Decision Trees help in diagnosing diseases based on patient symptoms and medical records.
3. **Customer Churn Prediction**: Companies use Decision Trees to predict if a customer will leave based on historical data.
4. **Marketing**: Decision Trees are used to segment customers based on purchasing behavior and demographics.
5. **Fraud Detection**: Decision Trees help identify fraudulent transactions by learning decision rules from historical data.

[142]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report

df = pd.read_csv('500hits.csv', encoding="latin-1")
df.head()
```

[142]:
```
        PLAYER  YRS     G     AB     R     H   2B   3B   HR   RBI    BB  \
0      Ty Cobb   24  3035  11434  2246  4189  724  295  117   726  1249
1   Stan Musial   22  3026  10972  1949  3630  725  177  475  1951  1599
2  Tris Speaker   22  2789  10195  1882  3514  792  222  117   724  1381
3   Derek Jeter   20  2747  11195  1923  3465  544   66  260  1311  1082
4  Honus Wagner   21  2792  10430  1736  3430  640  252  101     0   963

     SO   SB   CS     BA  HOF
0   357  892  178  0.366    1
1   696   78   31  0.331    1
2   220  432  129  0.345    1
3  1840  358   97  0.310    1
4   327  722   15  0.329    1
```

[143]:
```
df = df.drop(columns=['PLAYER', 'CS'])
df.head()
```

[143]:
```
   YRS     G     AB     R     H   2B   3B   HR   RBI    BB    SO   SB     BA  \
0   24  3035  11434  2246  4189  724  295  117   726  1249   357  892  0.366
1   22  3026  10972  1949  3630  725  177  475  1951  1599   696   78  0.331
2   22  2789  10195  1882  3514  792  222  117   724  1381   220  432  0.345
3   20  2747  11195  1923  3465  544   66  260  1311  1082  1840  358  0.310
4   21  2792  10430  1736  3430  640  252  101     0   963   327  722  0.329

   HOF
```

```
0    1
1    1
2    1
3    1
4    1
```

[144]:
```
X = df.iloc[:, 0:13]
y = df.iloc[:, 13]

X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=17,␣
 ↪test_size=0.2)

dtc = DecisionTreeClassifier()

dtc.get_params()
```

[144]:
```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'monotonic_cst': None,
 'random_state': None,
 'splitter': 'best'}
```

[145]:
```
dtc.fit(X_train, y_train)
```

[145]:
```
DecisionTreeClassifier()
```

[146]:
```
y_prediction = dtc.predict(X_test)

print(confusion_matrix(y_test, y_prediction))
```

```
[[52  9]
 [11 21]]
```

[147]:
```
print(classification_report(y_test, y_prediction))
```

```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84        61
           1       0.70      0.66      0.68        32

    accuracy                           0.78        93
```

```
   macro avg       0.76      0.75      0.76        93
weighted avg       0.78      0.78      0.78        93
```

[148]:
```python
dtc.feature_importances_
```

[148]:
```
array([0.        , 0.02598978, 0.03173403, 0.03633493, 0.39759474,
       0.06589131, 0.01565832, 0.05810452, 0.04515849, 0.12784069,
       0.04098071, 0.05207056, 0.10264192])
```

[151]:
```python
features = pd.DataFrame(dtc.feature_importances_, index=X.columns)
features
```

[151]:

| | 0 |
|---|---|
| YRS | 0.000000 |
| G | 0.025990 |
| AB | 0.031734 |
| R | 0.036335 |
| H | 0.397595 |
| 2B | 0.065891 |
| 3B | 0.015658 |
| HR | 0.058105 |
| RBI | 0.045158 |
| BB | 0.127841 |
| SO | 0.040981 |
| SB | 0.052071 |
| BA | 0.102642 |

[156]:
```python
dtc2 = DecisionTreeClassifier(criterion='entropy', ccp_alpha=0.04)

dtc2.fit(X_train, y_train)

y_pred2 = dtc2.predict(X_test)

print(confusion_matrix(y_test, y_pred2))

print(classification_report(y_test, y_pred2))
```

```
[[50 11]
 [ 9 23]]
              precision    recall  f1-score   support

           0       0.85      0.82      0.83        61
           1       0.68      0.72      0.70        32

    accuracy                           0.78        93
   macro avg       0.76      0.77      0.77        93
weighted avg       0.79      0.78      0.79        93
```

[157]:
```
features2 = pd.DataFrame(dtc2.feature_importances_, index=X.columns)
features2
```

[157]:

| | 0 |
|---|---|
| YRS | 0.000000 |
| G | 0.000000 |
| AB | 0.000000 |
| R | 0.000000 |
| H | 0.837977 |
| 2B | 0.000000 |
| 3B | 0.000000 |
| HR | 0.000000 |
| RBI | 0.000000 |
| BB | 0.000000 |
| SO | 0.000000 |
| SB | 0.000000 |
| BA | 0.162023 |