Decision Trees In Python

Shashwati Shradha

Contents

1 Introduction

Decision trees are widely used data structures for data compression and prediction. They are a set of rules used to predict the behavior of a data set, assuming that the similar behavior continues.

The purpose of this document is to summarize decision trees. We will focus on different types of decision trees, ways of measuring the accuracy and improving the model. We will also be using scikit-learn package in Python to code decision trees.

1.1 Representation of data

The goal of classification is to find a better representation of the data. Decision trees, unlike linear models are able to worked with non-linear data. They partition the data into rectangular regions in such a way that the concentration of a particular type of class is maximized in that region, along with minimizing impurity.

1.2 Supervised Learning

Decision trees is a form of supervised learning. This means that the the model learns through examples. When we are given a data set, we divide it into two groups: Train data and Test data. We train the model using the train data set and hope to get the least predictive error in the test data set.

2 Creating a Decision Tree

2.1 Split the Data Set

The first step in creating a decision tree is to split the given data set into a train data and a test data. We will use the train data to make a our and the accuracy of our model will be tested on the test data.

2.2 Recursive Partitioning

The algorithm used to make the decision trees is called the Top-Down Induction of Decision Trees (TDIDT). The principles guiding this algorithms are:

- The tree is generated using recursive partitioning.
- The same attribute cannot be chosen twice.
- Adequacy Condition. No two instances with the same attributes belong to different classes

According the TDIDT algorithm, for each attribute, we place the instances of the training data into one of the values of the attributes. We repeat this until we have reached a pure class.

3 Types of Decision Trees

3.1 Classification Trees

Classification trees are used to separate data sets into two or more discrete classes. The response variables are categorical in nature. The attributes can be categorical or numeric in nature. To decide the output at a leaf node, we use majority vote.

3.2 Regression Trees

Regression trees are used when the response variables are numeric or continuous. The attributes can be categorical or numeric in nature. To decide the output at a leaf node, we use the average.

4 Classification Tree for Titanic Data set

4.1 Decription

- **Pclass**: Passenger Class (1- 1st; 2- 2nd; 3- 3rd)
- **survival**: Survival (0- No; 1- Yes)
- **name**: Name
- **sex**: Sex
- **age**: Age
- **sibsp**: Number of Siblings/Spouses Aboard
- **parch**: Number of Parents/Children Aboard
- **ticket**: Ticket Number
- **fare**: Passenger Fare (British pound)
- **cabin**: Cabin
- **embarked** Port of Embarkation (C- Cherbourg; Q- Queenstown; S- Southampton)
- **boat**: Lifeboat
- **body**: Body Identification Number
- **home.dest**: Home/Destination

4.2 Code

```
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import pandas as pd
import numpy as np

dataset = './titanic.csv'
use = ["pclass", "sex", "age", "sibsp", "parch", "embarked", "survived"]

## 1- Getting data
titanic = pd.read_csv(dataset, usecols = use)

## 2- Preprocessing
clean = preprocessing.LabelEncoder()
titanic["sex"] = clean.fit_transform(titanic["sex"])
titanic["embarked"] = clean.fit_transform(titanic["embarked"].astype(str))
titanic[use] = titanic[use].replace(np.NaN, 0)

## 3- Making features and target attributes"
features = titanic[["pclass", "sex", "age", "sibsp", "parch", "embarked"]]
target = titanic["survived"]

## 4- Making the train and test set (80-20)
feat_train, feat_test, tar_train, tar_test =
train_test_split(features, target, test_size=0.20, random_state=2)

## 5- Building the decision tree
clf = DecisionTreeClassifier(max_depth = 6, random_state=1)

clf = clf.fit(feat_train, tar_train)

## 6- Making predictions
y_pred = clf.predict(feat_test)

## 7- Getting accuracy
accuracy = accuracy_score(y_pred, tar_test)

print("Accuracy: ", accuracy)
```

4.3 Summary

The titanic data set contains all the information people who were on board of titanic. The data collected will be used to predict whether a person would have survived on the ship or not. We do this using python's machine learning package scikit learn. We import the necessary packages, import the data and focus on the features variable to construct the model. We train our model using 80% of the data and use the 20% to test our accuracy. Some points to remember:

- **max_depth** can be used to avoid overfitting
- **min_samples_leaf** can be used to avoid overfitting too
- **random_state** is used to make sure our random set is reproducible
- **test_size** should be changed to test the model

5 Splitting criterion for Classification Trees

5.1 Entropy

Entropy is used to get the uncertainity in the dataset.