**Decision-Tree for Classification**

In chapter 1, you'll be introduced to a set of supervised learning models known as Classification-And-Regression-Tree or CART. In chapter 2, you'll understand the notions of bias-variance trade-off and model ensembling. Chapter 3 introduces you to Bagging and Random Forests. Chapter 4 deals with boosting, specifically with AdaBoost and Gradient Boosting. Finally in chapter 5, you'll understand how to get the most out of your models through hyperparameter-tuning.

**3. Classification-tree**

Given a labeled dataset, a classification tree learns a sequence of if-else questions about individual features in order to infer the labels. In contrast to linear models, trees are able to capture non-linear relationships between features and labels. In addition, trees don't require the features to be on the same scale through standardization for example.

**4. Breast Cancer Dataset in 2D**

To understand trees more concretely, we'll try to predict whether a tumor is malignant or benign in the Wisconsin Breast Cancer dataset using only 2 features. The figure here shows a scatterplot of two cancerous cell features with malignant-tumors in blue and benign-tumors in red.

**5. Decision-tree Diagram**

When a classification tree is trained on this dataset, the tree learns a sequence of if-else questions with each question involving one feature and one split-point. Take a look at the tree diagram here. At the top, the tree asks whether the concave-points mean of an instance is <= 0-point-051. If it is, the instance traverses the True branch; otherwise, it traverses the False branch. Similarly, the instance keeps traversing the internal branches until it reaches an end. The label of the instance is then predicted to be that of the prevailing class at that end. The maximum number of branches separating the top from an extreme-end is known as the maximum depth which is equal to 2 here.

**6. Classification-tree in scikit-learn**

Now that you know what a classification tree is, let's fit one with scikit-learn. First, import DecisionTreeClassifier from sklearn.tree as shown in line 1. Also, import the functions train\_test\_split() from sklearn.model\_selection and accuracy\_score() from sklearn.metrics. In order to obtain an unbiased estimate of a model's performance, you must evaluate it on an unseen test set. To do so, first split the data into 80% train and 20% test using train\_test\_split(). Set the parameter stratify to y in order for the train and test sets to have the same proportion of class labels as the unsplit dataset. You can now use DecisionTreeClassifier() to instantiate a tree classifier, dt with a maximum depth of 2 by setting the parameter max\_depth to 2. Note that the parameter random\_state is set to 1 for reproducibility.

**7. Classification-tree in scikit-learn**

Then call the fit method on dt and pass X\_train and y\_train. To predict the labels of the test-set, call the predict method on dt. Finally print the accuracy of the test set using accuracy\_score(). To understand the tree's predictions more concretely, let's see how it classifies instances in the feature-space.

**8. Decision Regions**

A classification-model divides the feature-space into regions where all instances in one region are assigned to only one class-label. These regions are known as decision-regions. Decision-regions are separated by surfaces called decision-boundaries. The figure here shows the decision-regions of a linear-classifier. Note how the boundary is a straight-line.

**9. Decision Regions: CART vs. Linear Model**

In contrast, as shown here on the right, a classification-tree produces rectangular decision-regions in the feature-space. This happens because at each split made by the tree, only one feature is involved.

**Classification-Tree Learning**

Welcome back! In this video, you'll examine how a classification-tree learns from data.

**2. Building Blocks of a Decision-Tree**

00:07 - 00:23

Let's first start by defining some terms. A decision-tree is a data-structure consisting of a hierarchy of individual units called nodes. A node is a point that involves either a question or a prediction.

**3. Building Blocks of a Decision-Tree**

00:23 - 01:13

The root is the node at which the decision-tree starts growing. It has no parent node and involves a question that gives rise to 2 children nodes through two branches. An internal node is a node that has a parent. It also involves a question that gives rise to 2 children nodes. Finally, a node that has no children is called a leaf. A leaf has one parent node and involves no questions. It's where a prediction is made. Recall that when a classification tree is trained on a labeled dataset, the tree learns patterns from the features in such a way to produce the purest leafs. In other words the tree is trained in such a way so that, in each leaf, one class-label is predominant.

**4. Prediction**

01:13 - 01:43

In the tree diagram shown here, consider the case where an instance traverses the tree to reach the leaf on the left. In this leaf, there are 257 instances classified as benign and 7 instances classified as malignant. As a result, the tree's prediction for this instance would be: 'benign'. In order to understand how a classification tree produces the purest leafs possible, let's first define the concept of information gain.

**5. Information Gain (IG)**

01:43 - 02:32

The nodes of a classification tree are grown recursively; in other words, the obtention of an internal node or a leaf depends on the state of its predecessors. To produce the purest leafs possible, at each node, a tree asks a question involving one feature f and a split-point sp. But how does it know which feature and which split-point to pick? It does so by maximizing Information gain! The tree considers that every node contains information and aims at maximizing the Information Gain obtained after each split. Consider the case where a node with N samples is split into a left-node with Nleft samples and a right-node with Nright samples.

**6. Information Gain (IG)**

02:32 - 02:58

The information gain for such split is given by the formula shown here. A question that you may have in mind here is: 'What criterion is used to measure the impurity of a node?' Well, there are different criteria you can use among which are the gini-index and entropy. Now that you know what is Information gain, let's describe how a classification tree learns.

**7. Classification-Tree Learning**

02:58 - 03:42

When an unconstrained tree is trained, the nodes are grown recursively. In other words, a node exists based on the state of its predecessors. At a non-leaf node, the data is split based on feature f and split-point sp in such a way to maximize information gain. If the information gain obtained by splitting a node is null, the node is declared a leaf. Keep in mind that these rules are for unconstrained trees. If you constrain the maximum depth of a tree to 2 for example, all nodes having a depth of 2 will be declared leafs even if the information gain obtained by splitting such nodes is not null.

**8. Information Criterion in scikit-learn (Breast Cancer dataset)**

03:42 - 03:58

Revisiting the 2D breast-cancer dataset from the previous lesson, you can set the information criterion of dt to the gini-index by setting the criterion parameter to 'gini' as shown on the last line here.

**9. Information Criterion in scikit-learn**

03:58 - 04:13

Now fit dt to the training set and predict the test set labels. Then determine dt's test set accuracy which evaluates to about 92%.

**Decision-Tree for Regression**

Welcome back! In this video, you'll learn how to train a decision tree for a regression problem. Recall that in regression, the target variable is continuous. In other words, the output of your model is a real value.

**2. Auto-mpg Dataset**

00:16 - 00:56

Let's motivate our discussion of regression by introducing the automobile miles-per-gallon dataset from the UCI Machine Learning Repository. This dataset consists of 6 features corresponding to the characteristics of a car and a continuous target variable labeled mpg which stands for miles-per-gallon. Our task is to predict the mpg consumption of a car given these six features. To simplify the problem, here the analysis is restricted to only one feature corresponding to the displacement of a car. This feature is denoted by displ.

**3. Auto-mpg with one feature**

00:56 - 01:17

A 2D scatter plot of mpg versus displ shows that the mpg-consumption decreases nonlinearly with displacement. Note that linear models such as linear regression would not be able to capture such a non-linear trend. Let's see how you can train a decision tree with scikit-learn to solve this regression problem.

**4. Regression-Tree in scikit-learn**

01:17 - 02:09

Note that the features X and the labels y are already loaded in the environment. First, import DecisionTreeRegressor from sklearn-dot-tree and the functions train\_test\_split() from sklearn-dot-model\_selection and mean\_squared\_error as MSE() from sklearn-dot-metrics. Then, split the data into 80%-train and 20%-test using train\_test\_split. You can now instantiate the DecisionTreeRegressor() with a maximum depth of 4 by setting the parameter max\_depth to 4. In addition, set the parameter min\_sample\_leaf to 0-dot-1 to impose a stopping condition in which each leaf has to contain at least 10% of the training data.

**5. Regression-Tree in scikit-learn**

02:09 - 02:35

Now fit dt to the training set and predict the test set labels. To obtain the root-mean-squared-error of your model on the test-set; proceed as follows: - first, evaluate the mean-squared error, - then, raise the obtained value to the power 1/2. Finally, print dt's test set rmse to obtain a value of 5-dot-1.

**6. Information Criterion for Regression-Tree**

02:35 - 03:03

Here, it's important to note that, when a regression tree is trained on a dataset, the impurity of a node is measured using the mean-squared error of the targets in that node. This means that the regression tree tries to find the splits that produce leafs where in each leaf the target values are on average, the closest possible to the mean-value of the labels in that particular leaf.

**7. Prediction**

03:03 - 03:17

As a new instance traverses the tree and reaches a certain leaf, its target-variable 'y' is computed as the average of the target-variables contained in that leaf as shown in this formula.

**8. Linear Regression vs. Regression-Tree**

03:17 - 04:03

To highlight the importance of the flexibility of regression trees, take a look at this figure. On the left we have a scatter plot of the data in blue along with the predictions of a linear regression model shown in black. The linear model fails to capture the non-linear trend exhibited by the data. On the right, we have the same scatter plot along with a red line corresponding to the predictions of the regression tree that you trained earlier. The regression tree shows a greater flexibility and is able to capture the non-linearity, though not fully. In the next chapter, you'll aggregate the predictions of a set of trees that are trained differently to obtain better results.