Transcript of Tutorial Video

Hi, everyone.

00:01

So today I'll be walking you through an end-to-end implementation of a decision tree to predict heart disease.

00:07

I'll be focusing on how pruning affects accuracy and model performance.

00:13

So, we'll start with the data pre-processing, after which we will build an unpruned decision tree.

00:19

And then we'll perform some pre-pruning or apply some pre-pruning and then post-pruning techniques.

00:24

So, you see why, how to effectively train a decision tree model and also why pruning is essential to overfit, to dealing with overfitting.

00:37

The data set we're using is the heart disease data set.

00:40

And it contains 303 rows and about 14 columns.

00:45

Contains information like age, cholesterol levels, and so on and so forth.

00:49

target column indicates if a person has heart disease or not.

00:55

And now we've loaded the dataset using pandas.

00:57

And just to find out if we have any null values in our dataset, I'll just run this code.

01:06

So that's missing values just to find is null the sum of is null values.

01:12

We'll run that and then we can look through the data set and we can see that um for this bit we have about four is null values and here we've got two is null values.

01:24

And so, because it's quite negligible when we consider the fact that there's about 303 entries, had we values can drop those values.

01:36

We can handle it by dropping those values.

01:38

And so what we just do is we'll run this and say clevelanddata.dropna values and run that again.

01:47

And then to see if we've successfully handled that we'll just run that code

again to see if we still have any is null values or not.

01:55

And when we take a look at it again, we can see that the is null values are all gone now and they're all handled.

02:03

The next thing we do is we prepare the target column for binary classification.

02:08

So originally the data sets the target column had values ranging from zero to four, and each successive number represented how severe the heart disease was.

02:22

So, a value of one would be less severe than, less severe level of heart disease than a person with a value of two, for example.

02:33

But because we just want to find out if heart disease is present or not, we're just going to convert that to a binary form.

02:39

So that's either 0 for no heart disease and 1 for heart disease.

02:44

And so to do that, we just run this line of code using .loc.

02:48

And what's going on here is it selects all the rows and just the target column.

02:54

And over here, we have a Boolean mask created where it tries to find out essentially if the value is greater than zero.

03:03

And if it is greater than zero, it assigns a true value.

03:06

And if it is zero, it gives it a false value.

03:09

And then we just convert that to an integer where the true value then becomes 1 and the false value becomes 0.

03:17

So, the next thing we'll do is the train-test split.

03:20

And we just do that by importing train-test split from scikit-learn.

03:24

And what we're going to do is, obviously, we're going to choose x aseverything else, which is our training.

03:34

Data accepts the target and then Y is going to be our target.

03:38

So that's why we've dropped the target in this line of code.

03:43

And in this one we've got just the target.

03:45

And so, we perform the train test split.

03:47

And so, to perform the train test split, essentially, we need to have X train, X test, Y train, Y test equal to train test split.

03:55

For the test size we can either do.

03:59

80% to 20% or 70% to 30%.

04:02

For this particular situation, I'm going to do a 70/30 split.

04:06

So, I'm just going to put 0.3 for the test size and random state 42 to ensure reproducibility.

04:13

Let me just, okay, and then we can run that.

04:20

matplotlib.pyplot, as plt And so we begin by , training a basic decision tree without any constraints.

04:24

So, what that means is that the tree is allowed to grow to its full depth um which could potentially lead to overfitting.

04:29

So, what we're going to do is we're just going to import some basic libraries that we need from scikit-learn.

04:34

So that's a decision tree classifier, accuracy score and and so this is the code to perform the training over here and like I explained the random state of 42 that just ensures that each time the code is run the same split occurs.

04:50

So just for reproducibility and then we make the predictions on the test set which is what we've got going on here.

04:57

And then let's see what the accuracy looks like on this on pruned basic decision tree.

05:03

So, we just run that code.

05:04

Um We already imported accuracy score and then we've got just the Y test and the prediction.

05:09

And then let's print to see what the accuracy looks like.

05:15

Got an issue here, let's see.

05:17

Right, so I've just got that there.

05:19

Okay, and then we run it again.

05:23

Okay.

05:24

So, we've got an unpruned decision tree accuracy of 0.6889.

05:31

Okay, fair enough.

05:32

Let's see.

05:33

So, let's just visualize and see what that unpruned tree looks like.

05:38

So, we just import matplotlib.pyplot, as plt, and then we visualize the decision tree.

05:43

Let's just quickly run this code and see what that looks like.

05:48

Okay.

05:52

Let's see what our tree looks like.

05:54

Perfect.

05:54

Okay, so we can see how large between the decision tree's is and how most likely it's going to be doing a lot of overfitting, picking up a lot of noise, and that's just why, that's why it's as large as it is.

06:08

see then what we can do about that.

06:11

And so, to control overfitting, what we can do is we can introduce a max depth so we don't let the tree grow to the full, to its full extent, but we introduce like a max depth constraint.

06:25

And so, once we do that, let's see what we have.

06:27

This is just the code to, the only thing I've included here is a max depth of five.

06:32

That's the only thing that's different.

06:34

And then we just make the predictions again on the test set.

06:38

All right.

06:39

And then let's see what, we'll also just look at what the accuracy, the difference in the accuracy it looks like.

06:45

So, let's just run that, take a look.

06:48

at what we have.

06:49

Okay, so we can see that the accuracy is a lot better.

06:52

So as opposed to 0.6889, we've trees got 0.7667, and that's using just

because it's making a better prediction as opposed to what we had before.

07:03

So, it's generalizing better and not overfitting.

07:07

Another way to simplify a decision tree after it's fully grown is a method called cost complexity pruning.

07:13

And what that does is that it aims to strike a balance between a tree's predictive performance and its complexity by introducing something called CCP alphas.

07:24

So, what we want to do is calculate a sequence of CCP alpha values to find the optimum that, CCP alpha the value.

07:30

Now these values like.

07:34

represent the thresholds for removing have.

07:36

branches off of the decision trees.

07:37

post

07:38

And so, the pruning path is generated using the cost complexity pruning path, which we've visualized here.

07:45

using

07:46

One thing to note here is that CCP alphas contains different pruning thresholds.

07:52

So, at CCP alpha equal to zero, then the tree is at maximum depth and it's unpruned.

07:58

But as the CCP alpha's value increases, then we essentially have that the tree is getting more and more pruned.

08:04

Next, we create a list the unpruned decision trees using different alpha values, following the path that's been created by the cost complexity pruning path.

08:15

So that's what's going on here.

08:17

And we just have that.

08:18

For every new alpha, a new tree is created and trained on the training data.

08:24

And it just follows the path that we've created.

08:26

We then evaluate the performance of each tree on the test data.

08:30

Usually, trees with a larger CCP alpha value are better at doing the predictions than trees with lower CCP values, because those ones then tend to overfit, um which is an issue we would have a lot less with trees with higher CCP alpha values.

08:51

Finally, we identify which tree has the best accuracy, which is the best balance between tree complexity and accuracy on the test data.

09:01

And so, we just run that.

09:05

And so, we just run that, the tree with the max pruned accuracy, just to see what that looks like.

09:13

So, let's see what we have.

09:14

Okay, so after post pruning, we've got a tree with the test accuracy of 0.7889, which is even higher than when we introduced the max depth of five, which was even higher than obviously the unpruned decision tree.

09:31

And now we just visualize using matplotlib again to see what the final tree looks like.

09:37

just to see what the difference between that and the first decision tree was.

09:43

So, we just run that.

09:46

Okay.

09:47

And we can see that this is a much simpler tree, as opposed to what we had, which was the original unpruned decision tree, which included all the noise, all the noise from the data, And there was a lot of overfitting going on.

10:03

And we just have a much simpler, better decision tree, which performs the prediction significantly better at a higher level of accuracy as well.

10:13

And so just to summarize what we have, or what we've discussed, um we've seen how the un-prune tree essentially has high training accuracy, but low-test accuracy because it overfits to the training data.

10:26

And then we've seen how limiting the depth with pruning, introducing a maximum depth will provide a better test accuracy.

10:35

But even more optimized is the post-pruning process, which we went through um with the cost complexity pruning.

10:43

And we saw how that strikes the perfect balance between the tree complexity and the test accuracy

10:49

and so, this just shows how pruning is essential in creating effective decision trees that generalize well to unseen data and that's all I have for you today thank you so much