Avinash Navlani
December 29th, 2018

PYTHON

Decision Tree Classification in Python

In this tutorial, learn Decision Tree Classification, attribute selection measures, and how to build and optimize Decision Tree Classifier using Python Scikit-learn package.

As a marketing manager, you want a set of customers who are most likely to purchase your product. This is how you can save your marketing budget by finding your audience. As a loan manager, you need to identify risky loan applications to achieve a lower loan default rate. This process of classifying customers into a group of potential and non-potential customers or safe or risky loan applications is known as a classification problem. Classification is a two-step process, learning step and prediction step. In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret. It can be utilized for both classification and regression kind of problem.

In this tutorial, you are going to cover the following topics:

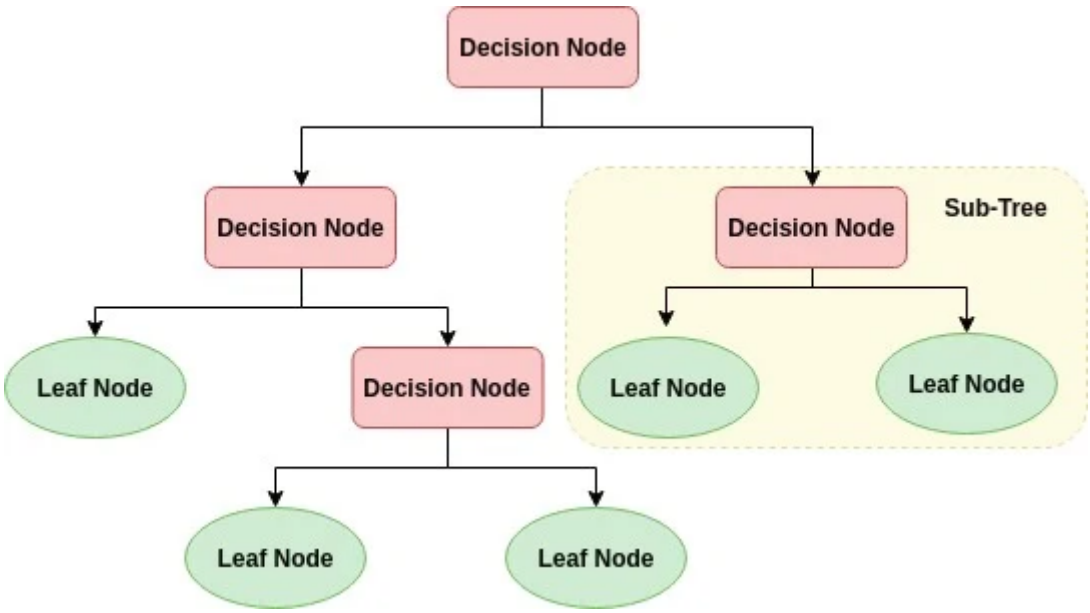
- Decision Tree Algorithm
- How does the Decision Tree algorithm work?
- Attribute Selection Measures

- Gain Ratio
- Gini index

- Optimizing Decision Tree Performance
- Classifier Building in Scikit-learn
- Pros and Cons
- Conclusion

Decision Tree Algorithm

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



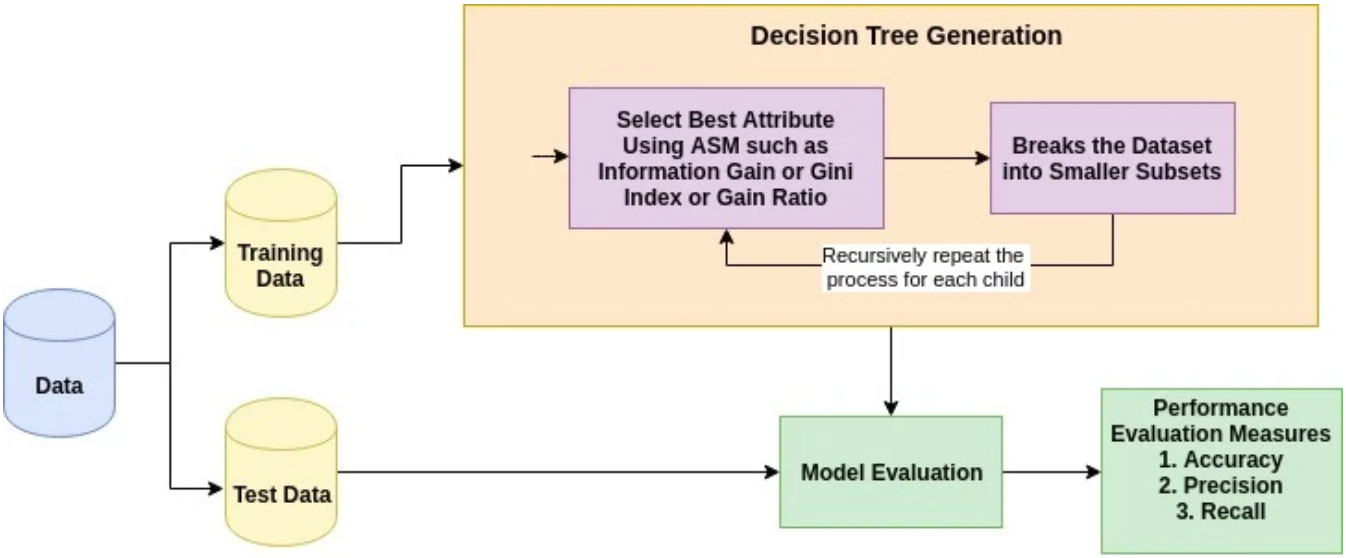
Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the

high dimensional data with good accuracy.

How does the Decision Tree algorithm work?

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
   - All the tuples belong to the same attribute value.
   - There are no more remaining attributes.
   - There are no more instances.



Attribute Selection Measures

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute (Source). In the case of a continuous-valued attribute,

Information Gain

Shannon invented the concept of entropy, which measures the impurity of the input set. In physics and mathematics, entropy referred as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples. Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

$$\text{Info}(D) = -\sum_{i=1}^{m} pi \log_2 pi$$

Where, Pi is the probability that an arbitrary tuple in D belongs to class Ci.

$$\text{Info}_A(D) = \sum_{j=1}^{V} \frac{|Dj|}{|D|} \times \text{Info}(D_j)$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

Where,

- Info(D) is the average amount of information needed to identify the class label of a tuple in D.
- |Dj|/|D| acts as the weight of the jth partition.
- InfoA(D) is the expected informa-tion required to classify a tuple from D based on the partitioning by A.

The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N().

Gain Ratio

unique identifier such as customer_ID has zero info(D) because of pure partition. This maximizes the information gain and creates useless partitioning.

C4.5, an improvement of ID3, uses an extension to information gain known as the gain ratio. Gain ratio handles the issue of bias by normalizing the information gain using Split Info. Java implementation of the C4.5 algorithm is known as J48, which is available in WEKA data mining tool.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Where,

- |Dj|/|D| acts as the weight of the jth partition.
- v is the number of discrete values in attribute A.

The gain ratio can be defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the highest gain ratio is chosen as the splitting attribute (Source).

Gini index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$Gini(D) = 1 - \sum_{i=1}^{m} Pi^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.

The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into

$$\text{Gini}_A(D)=\frac{|D1|}{|D|}\text{Gini}(D_1)+\frac{|D2|}{|D|}\text{Gini}(D_2)$$

In case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split-point and point with smaller gini index chosen as the splitting point.

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

The attribute with minimum Gini index is chosen as the splitting attribute.

Decision Tree Classifier Building in Scikit-learn

Importing Required Libraries

Let's first load the required libraries.

```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

Loading Data

Let's first load the required Pima Indian Diabetes dataset using pandas' read CSV function. You can download the data here.

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'lab
# load dataset
pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)
```

```
pima.head()
```

0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using function train_test_split(). You need to pass 3 parameters features, target, and test_set size.

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) #
```

Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

```
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating Model

Let's estimate, how accurately the classifier or model can predict the type of cultivars.

Accuracy can be computed by comparing actual test set values and predicted values.

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6753246753246753
```

Well, you got a classification rate of 67.53%, considered as good accuracy. You can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.

Visualizing Decision Trees

You can use Scikit-learn's *export_graphviz* function for display the tree within a Jupyter notebook. For plotting tree, you also need to install graphviz and pydotplus.

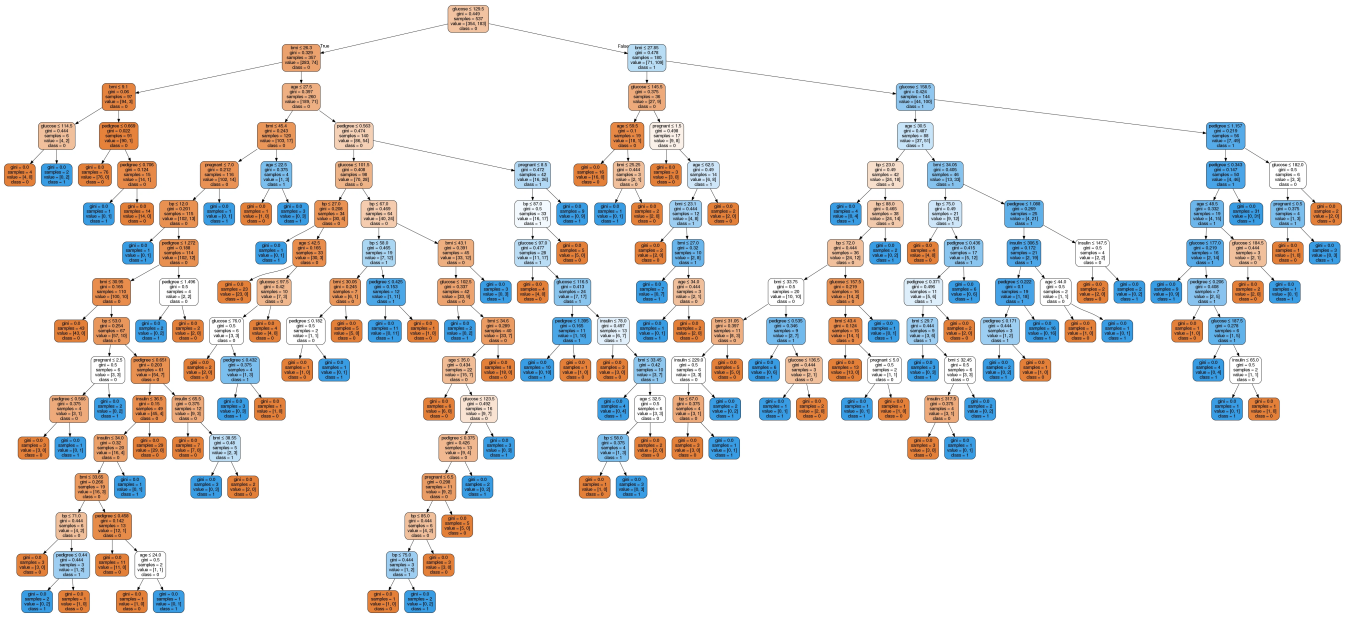
```
pip install graphviz
```

```
pip install pydotplus
```

export_graphviz function converts decision tree classifier into dot file and pydotplus convert this dot file to png or displayable form on Jupyter.

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
```

```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```



In the decision tree chart, each internal node has a decision rule that splits the data. Gini referred as Gini ratio, which measures the impurity of the node. You can say a node is pure when all of its records belong to the same class, such nodes known as the leaf node.

Here, the resultant tree is unpruned. This unpruned tree is unexplainable and not easy to understand. In the next section, let's optimize it by pruning.

Optimizing Decision Tree Performance

- **criterion : optional (default="gini") or Choose attribute selection measure**: This parameter allows us to use the different-different attribute selection measure. Supported criteria are "gini" for the Gini index and "entropy" for the information gain.
- **splitter : string, optional (default="best") or Split Strategy**: This parameter allows us to choose the split strategy. Supported strategies are "best" to choose the best split and

- **max_depth : int or None, optional (default=None) or Maximum Depth of a Tree**: The maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting (Source).

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. In the following the example, you can plot a decision tree on the same data with max_depth=3. Other than pre-pruning parameters, You can also try other attribute selection measure such as entropy.

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

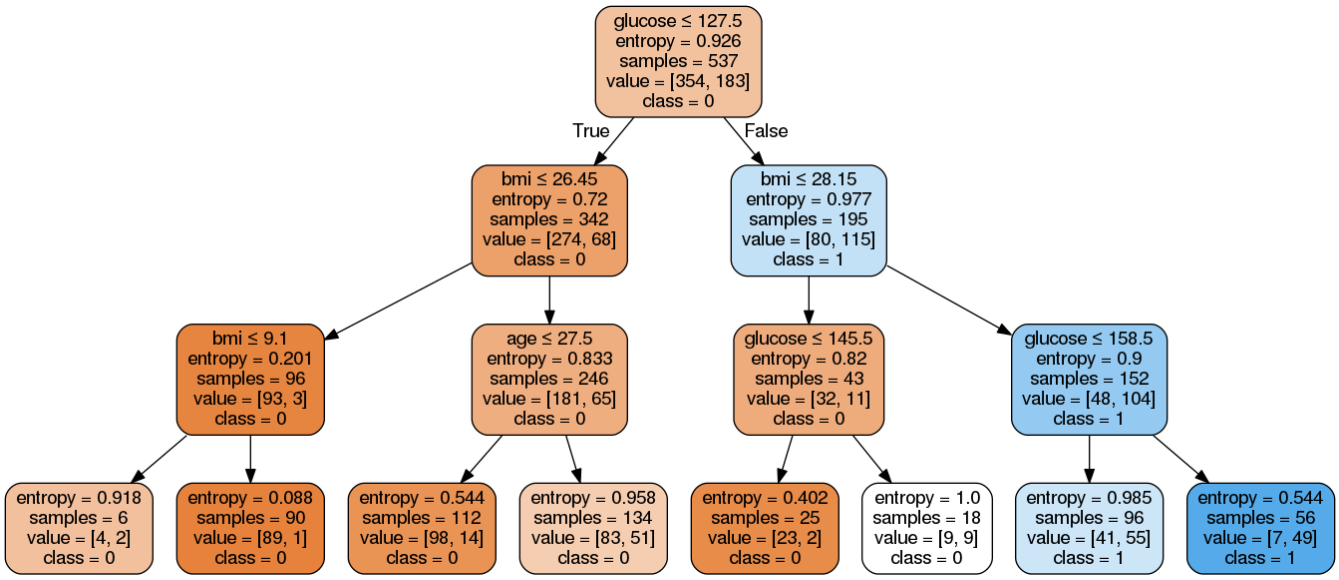
```
Accuracy: 0.7705627705627706
```

Well, the classification rate increased to 77.05%, which is better accuracy than the previous model.

Visualizing Decision Trees

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
```

```
graph.write_png('diabetes.png')
Image(graph.create_png())
```



This pruned model is less complex, explainable, and easy to understand than the previous decision tree model plot.

Pros

- Decision trees are easy to interpret and visualize.
- It can easily capture Non-linear patterns.
- It requires fewer data preprocessing from the user, for example, there is no need to normalize columns.
- It can be used for feature engineering such as predicting missing values, suitable for variable selection.
- The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm. (Source)

Cons

- Sensitive to noisy data. It can overfit noisy data.

- Decision trees are biased with imbalance dataset, so it is recommended that balance out the dataset before creating the decision tree.

Conclusion

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you covered a lot of details about Decision Tree; It's working, attribute selection measures such as Information Gain, Gain Ratio, and Gini Index, decision tree model building, visualization and evaluation on diabetes dataset using Python Scikit-learn package. Also, discussed its pros, cons, and optimizing Decision Tree performance using parameter tuning.

Hopefully, you can now utilize the Decision tree algorithm to analyze your own datasets.

If you want to learn more about Machine Learning in Python, take DataCamp's Machine Learning with Tree-Based Models in Python course.

COMMENTS

Dendani Bilal
16/02/2019 02:49 PM

Thank you Avinash.

It was very helpful however I got the following error after I run the code until **y_pred = clf.predict(X_test).** The following is the error :

ValueError: could not convert string to float: 'DiabetesPedigreeFunction'

▲ 20

Avinash Navlani
18/03/2019 09:08 PM

▲ 0

Zahid Parvez
19/03/2019 07:35 AM

You need to factorize your categorical data, the model cant deal with non-numeric data.

Have a look at:

- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
- https://stackoverflow.com/questions/38108832/passing-categorical-data-to-sklearn-decision-tree

▲ 6

Emmanuel Davidson
28/07/2019 11:05 PM

delete the first row of the dataframe

pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

pima = pima.iloc[1:]

▲ 7

Virendra Mishra
11/04/2019 05:02 PM

i got the same error and i used the below script before train the model.

lE = preprocessing.LabelEncoder()

data = yourdatasetname.apply(lE .fit_transform)

▲ 1

Colin Contreary
20/04/2019 09:06 AM

If you downloaded the dataset from Kaggle, it comes with a header row. Try replacing it as follows:

<code>

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

pima = pd.read_csv("diabetes.csv")

pima.columns = col_names

That worked for me.

▲ 9

Ami Joy
24/04/2019 11:35 PM
Thanks it worked for me also :)

▲ 1

naksha v
24/11/2019 01:43 AM
thanks a lot it worked :)

▲ 1

Luis Alfonso Gómez Zúñiga
23/12/2019 01:16 PM
This work for me!! Thanks

▲ 1

Hamid K Hamedani
04/01/2020 04:21 AM
Great! worked for me as well. Thanks, Could you let us know what was the problem and what your code does?

▲ 1

GANESH PANDIT
05/01/2020 03:52 PM
i'm getting error so please can you send me the complete code

so that i can check where i did mistake

▲ 3

Hikmah Nasir
30/01/2020 12:29 AM
Thanks a lot, this is worked.

▲ 1

Avinash Navlani
21/04/2019 11:03 AM
Colin Contreary is right. May be your dataset has columns in it. if you see the data none of the column is String. So need to apply label encoding.

sahaia salim
14/06/2019 03:50 PM

```
ValueError: could not convert string to float: 'Insulin'
```

▲ 5

Garvit Jain
06/08/2019 12:28 PM

Replace the statement

pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)

with

pima = pd.read_csv("diabetes.csv", header=0, names=col_names)

▲ 3

---

kiran kumari
09/04/2019 04:02 AM

Thanks a lot Sir! But I got an error :

```
InvocationException: GraphViz's executables not found
```

Can you please help me out?

▲ 4

Virendra Mishra
11/04/2019 05:06 PM

you need to install Graphviz, instal pydotplus and setup environment variable. after doing these three step you will see the tree.let me know if you still see error

▲ 5

Colin Contreary
20/04/2019 08:48 AM

I had the same problem of "GraphViz's executables not found." I'm using Windows 10. What worked for me was going into the Environment Variables and adding the graphviz

You can search "Environment Variables" in the search bar of your desktop, or find them manually under Control Panel > System > Advanced System Settings.

Click into Environment Variables and then click Path under the User Variables box.

On my computer, the graphviz folder is stored in
C:\Users\Colin\Anaconda3\Library\bin\graphviz

Find where the graphviz folder is stored on yours, copy that location, then make a new entry in your Path settings with that location.

Then make sure you close down your Jupyter notebook and restart it. Hopefully, the graphviz package will now work!

If not, you can try reading this StackOverflow post in case something else works:
https://stackoverflow.com/questions/18438997/why-is-pydot-unable-to-find-graphvizs-executables-in-windows-8

▲ 4

Avinash Navlani
21/04/2019 10:56 AM
Graphviz installation is tricky thing. Initially I also got problem in that. Follow the **Colin Contreary's** Solution.

▲ 1

LIA FAROKHAH
29/08/2019 02:08 PM
Change path and it work

```
InvocationException: GraphViz's executables not found
```

▲ 2

Michael Bhekumuzi Maima
02/10/2019 01:59 PM
Hi Colin,

You can visualize the decision without creating a PATH, check the code below:

```
decisionGraph = Source(tree.export_graphviz(clf,
                                            out_file     = None,
                                            feature_names = features,
```

filled = True,

rounded = True))

display(SVG(decisionGraph.pipe(format = "svg")))

▲ 1

Bob Fried
03/01/2020 11:00 PM

I tried the update Environment variables fixed.. I now get a new error:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-e2816f0728af> in <module>
      5
      6 dot_data = StringIO()
----> 7 export_graphviz(clf, out_file=dot_data,
      8                 filled=True, rounded=True,
      9                 special_characters=True,feature_names = feature_cols,

NameError: name 'clf' is not defined
```

[ ]:

▲ 1

Jennifer Jenkins
25/08/2019 06:09 AM

I solved this problem by going to Terminal and typing

conda install graphviz. I am using Anaconda on a Mac.

▲ 1

ANURAG NAGAR
06/05/2019 11:27 PM

could not convert string to float.

My data is of string type(catagorical). How could I move ahead?

▲ 2

David Galli
07/05/2019 05:49 PM

State machines are the basics of computer science. It's kind of funny to suggest this as an extra reading material... on the other hand, it's also funny to think that they are a good way to deal with decision trees. Since, if you ever took an uk essay writing automata class, you'd know that state machines are limited in that they don't have any "memory", which means that you can only describe regular languages using this formalism.

▲ 1

Kevin Ochieng
31/05/2019 01:45 PM

DataCamp kindly try to be clear and procedural like other platforms like towardsdatascience and medium.com ,every time i follow your tutorials errors become my friends and you get other steps skipped without informing the learnings

▲ 1

Suraj Meena
29/06/2019 01:05 AM

Do we need to encode categorical variables or not for decision tree to work ?

▲ 1

Suraj Meena
29/06/2019 01:06 AM

Also, if we do not need to encode categorical variables then how will decision tree know that particular columns in our dataset are categorical ?

▲ 1

javid iqbal
30/07/2019 06:48 PM

Dear Avinash

```
----> 9                 special_characters=True,feature_names = feature_cols,class_
```

▲ 1

AJ Ferrara
01/08/2019 07:45 AM

great tutorial. was able to apply on my own data. can you please reccomend some tips on balancing a dataset with only 1% = 1? I'd like to keep all 1%, about 800 tecords. Thanks

▲ 2

Avinash Navlani
23/08/2019 10:12 AM

You can try oversampling and under-sampling.

▲ 1

Smruti Vyavahare
14/08/2019 03:03 PM

in this decision tree red faint red grey dark nodes are there. how color changed in code where mentioned.please tell in detail

▲ 1

Prateek Ramsinghani
16/08/2019 04:15 AM

Thanks for the tutorial, I was able to use the same code and develop the tree but the problem is, I had all categorical variables and converted them using label encoder while developing the tree it shows values in decimals (can be considered similar to pregnant > 1.5 and < 1.5 in the above example) which does not make any sense, can someone please suggest a way in which I can keep the values as whole numbers.

▲ 1

Ertan Iyidost

```
IndexError: list index out of range
```

- I tried to increase my class_names,

```
class_names=['true','false','something else']
```

- I tried to label my feature_names with numbers,

```
feature_names=['1', '2', '3','4','5','6','7','8','9','10','11','12','13','14'],
```

but they didn't solve the problem.

I have 14 feature columns, 1 target column and 15 columns in total.

Does anyone knows how can I fix it?

▲ 3

Sai Prasanna
17/08/2019 09:58 PM
way to calculate entropy without using decision tree alogarithm

▲ 2

Avinash Navlani
23/08/2019 10:09 AM
Yes, using formula. you can calculate for small dataset and for large dataset you need to write program.

▲ 1

Veer Pratap Singh
25/08/2019 02:34 PM
very helpfull

▲ 3

29/08/2019 07:34 PM

Great tutorial, thank you for sharing..

▲ 3

pcckee
02/09/2019 01:56 AM

I've been struggling with this concept for few weeks. This article really helps!

▲ 2

Ernesto Ernane
03/09/2019 07:58 PM

obrigado foi útil

▲ 1

87 H
10/09/2019 04:21 AM

It is very helpful to understand the decision tree. Do you have any recommendation how to apply the formula to practical questions, that would be helpful to understand the formula well. Thanks.

▲ 1

aliyu abdul
23/09/2019 07:39 PM

Quick question.

Please how can the decision tree be used for image dataset

▲ 1

MD NAZMUL ISLAM
02/10/2019 02:09 AM

Great writing :)

▲ 1

Mayank Tripathi
05/10/2019 11:22 AM

▲ 1

Brent Niland
07/10/2019 09:20 AM

Hi Avinash, thank you for this article. Where do I place the diabetes.csv file after downloading so that python can read it?

▲ 1

Brent Niland
09/10/2019 03:27 AM

I have made it to this point: graph.write_png('file_name.png') At this point I received this InvocationException: GraphViz's executables not found. It went to the \pydotplus\graphviz.pyc location on my hard drive. It appears it found graphviz, but it said executables not found. What needs to happen for it to find the "executables"?? Is there a graphviz.exe file that needs to be placed somewhere or what? Thanks.

▲ 3

Sri Sreshtan
05/11/2019 05:51 PM

Hello guys,

I am fairly new to the world of programming and coding. I tried to run a decision tree code in python in order to predict the financial distress of companies.

However as I was progressing in running the code, I got the following error: " ValueError: could not convert string to float: 'rev.ebit'. "

▲ 0

Eve M
11/11/2019 11:16 AM

anyone else getting an index error?

▲ 1

Anirban Ghosh
26/11/2019 08:19 AM

How to convert plot decision tree with multiple categorical variables..

Mahesh Babu
28/11/2019 11:49 AM
in this tutorial we are finding the final decision tree, how to figure out the label for unknown sample data?

▲ 1

N S
04/12/2019 02:40 PM
Can you give me a different between decision tree ID3 and C4.5 in python ?

▲ 2

Wallace Wambugu
11/12/2019 06:01 AM
Hi, I am having an issue with the stringIO.externals.SIX from where we import stringIO it gives a deprecation warning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7, do you have any suggestions of how to get around it?

▲ 1

Gaurav Lal
20/12/2019 05:21 PM

i am very new to machine learning , please explain

what is value= [ ], and class represent. say for example for a root node glucose<=12.75, entropy =0.928, samples 537, values[354,183],class 0

i want to know how class o is selected? is it corresponding to value 354 or 183

▲ 1

Luis Alfonso Gómez Zúñiga
23/12/2019 02:08 PM
I have the next error with graphviz:

some idea??

```
"(https://pypi.org/project/six/).", DeprecationWarning)
```

▲ 2

A Dekker
17/01/2020 07:24 PM

It was really helpful, thanks! But in the end I got a syntax error about the 'graph' in the one last sentence. Can somebody maybe help me with solving this error?

▲ 0

Darshan Agrawal
23/02/2020 05:05 PM

Is there any way to do post pruning using sklearn?

▲ 1

Nishi Dutta
03/03/2020 10:52 AM

Can someone please explain What does the "sklearn.externals.six import StringIO" api do? Why is it used?

▲ 6