Decision Tree classifier

Today your task is to get familiar with decision tree classifier - simple, but powerful case of discrete math usage.

General idea

You are expected to write a quite simple, yet good core logic of decision tree classifier class. Additionaly, you need to test your results and write down a report on what you've done, which principles used and explain the general process.

Hopefully, you have already learned what is decision tree classifier and how it work. For better understanding, and in case if something is still unclear for you, here are some useful links on basics of DTC:

- https://towardsdatascience.com/decision-tree-from-scratch-in-python-46e99dfea775
- https://towardsdatascience.com/decision-tree-algorithm-in-python-from-scratch-8c43f0e40173
- https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/
- https://anderfernandez.com/en/blog/code-decision-tree-python-from-scratch/

Also, for those interested to learn more about machine learning and particulary Desicion Trees - here is a great course on Coursera (you may be interested in the whole course or just this particular week):

- https://www.coursera.org/learn/advanced-learning-algorithms/home/week/4

Dataset

You can use Iris dataset for this task. It is a very popular dataset for machine learning and data science. It contains 150 samples of 3 different species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

Read more on this:
https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
https://en.wikipedia.org/wiki/Iris_flower_data_set

However, using more interesting and intricate datasets is much appreciated. You can use any dataset you want, but it should be a classification one. For example you can use breast cancer or wine datasets, which are also available in sklearn.datasets. Or you can use any other dataset you find interesting.

P.S. In case you are not sure if your dataset is suitable, feel free to ask assistants :).

Sofiia Popeniuk, Victoriia Lushpak

```
# install the required packages

!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install graphviz
!pip install scikit-learn
```

```
Requirement already satisfied: pandas in ./venv/lib/python3.12/site-
packages (2.2.0)
Requirement already satisfied: numpy<2,>=1.26.0 in
./venv/lib/python3.12/site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
./venv/lib/python3.12/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
./venv/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
./venv/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in ./venv/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: numpy in ./venv/lib/python3.12/site-
packages (1.26.4)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: matplotlib in
./venv/lib/python3.12/site-packages (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
./venv/lib/python3.12/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
./venv/lib/python3.12/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
./venv/lib/python3.12/site-packages (from matplotlib) (4.48.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
./venv/lib/python3.12/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in
```

```
./venv/lib/python3.12/site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
./venv/lib/python3.12/site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in
./venv/lib/python3.12/site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
./venv/lib/python3.12/site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
./venv/lib/python3.12/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in ./venv/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: graphviz in ./venv/lib/python3.12/site-
packages (0.20.1)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: scikit-learn in
./venv/lib/python3.12/site-packages (1.4.0)
Requirement already satisfied: numpy<2.0,>=1.19.5 in
./venv/lib/python3.12/site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in
./venv/lib/python3.12/site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in
./venv/lib/python3.12/site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
./venv/lib/python3.12/site-packages (from scikit-learn) (3.2.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# scikit-learn package
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split

iris = load_iris()
dir(iris)
```

```
['DESCR',
 'data',
 'data_module',
```

```
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']

iris.data.shape

(150, 4)
```

This means that we have 150 entries (samples, infos about a flower). The columns being: Sepal Length, Sepal Width, Petal Length and Petal Width(features). Let's look at first two entries:

```
iris.data[0:2]

array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2]])
```

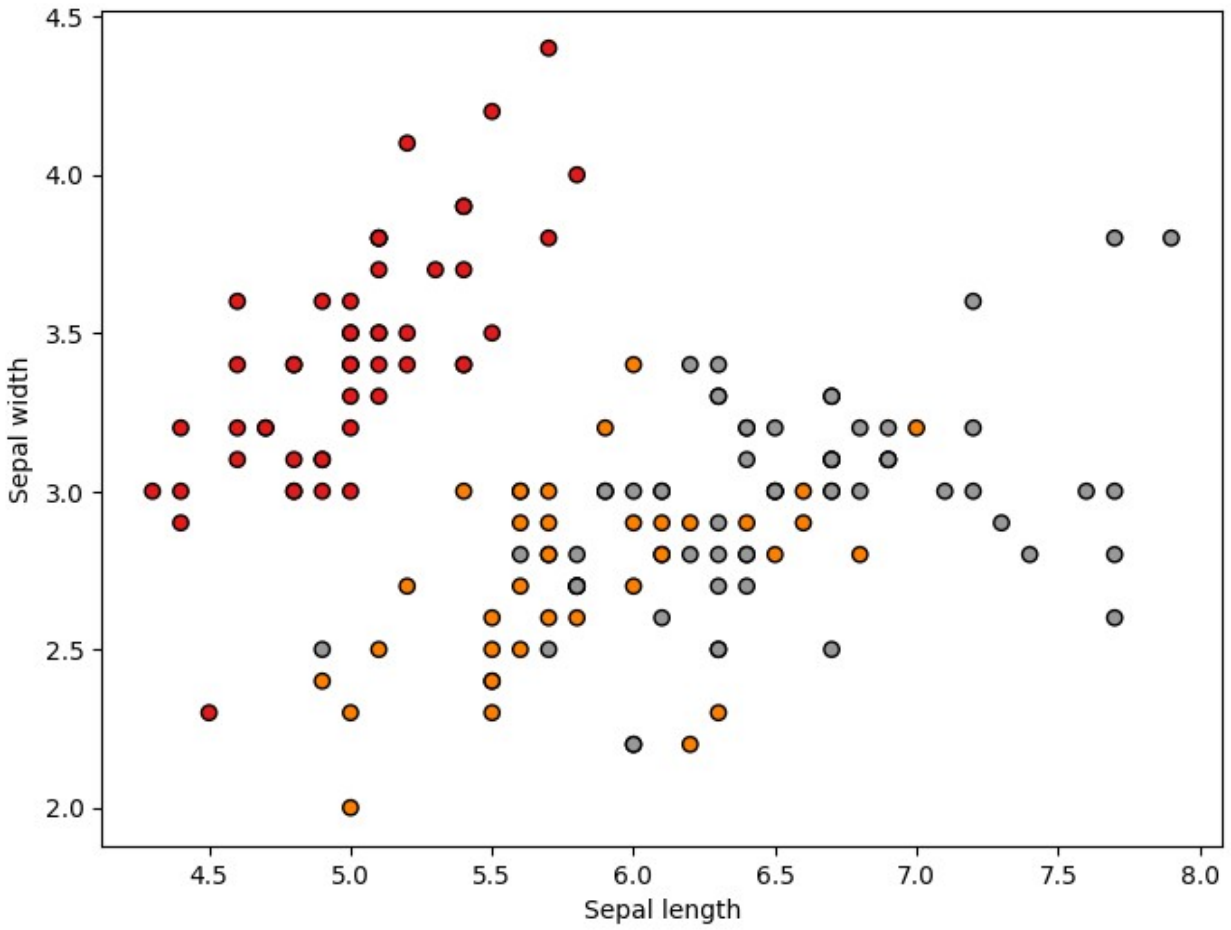
To undestand data little bit better, let's plot some features

```
X = iris.data[:, :2]  # we only take the first two features.
y = iris.target

plt.figure(2, figsize=(8, 6))
plt.clf()

# Plot the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")

Text(0, 0.5, 'Sepal width')
```



From this we can clearly see, that even basing on those two parameters, we can clearly divide (classify) out data into several groups. For this, we will use decision tree classifier: https://scikit-learn.org/stable/modules/tree.html#tree

Example of usage

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

```
clf = DecisionTreeClassifier()

X, y = iris.data, iris.target
X.shape, y.shape
```

```
((150, 4), (150,))
```

Train / test split

We train our model using training dataset and evaluate its performance basing on the test dataset. Reason to use two separate datasets is that our model learns its parameters from data, thus test set allows us to check its possibilities on completely new data.

```
X, X_test, y, y_test = train_test_split(X, y, test_size= 0.20)
```

Model learning

It learns its parameters (where it should split data and for what threshold value) basing on the training dataset. It is done by minimizing some cost function (e.g. Gini impurity or entropy).

```
clf = clf.fit(X, y)
```

Visualization of produced tree

You do not need to understand this piece of code :)

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/execute.py:81, in run_check(cmd,
input_lines, encoding, quiet, **kwargs)
     80     else:
---> 81         proc = subprocess.run(cmd, **kwargs)
     82 except OSError as e:

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
subprocess.py:548, in run(input, capture_output, timeout, check,
*popenargs, **kwargs)
    546     kwargs['stderr'] = PIPE
--> 548 with Popen(*popenargs, **kwargs) as process:
    549     try:

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
subprocess.py:1026, in Popen.__init__(self, args, bufsize, executable,
stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,
universal_newlines, startupinfo, creationflags, restore_signals,
start_new_session, pass_fds, user, group, extra_groups, encoding,
```

```
errors, text, umask, pipesize, process_group)
   1023             self.stderr = io.TextIOWrapper(self.stderr,
   1024                     encoding=encoding, errors=errors)
-> 1026     self._execute_child(args, executable, preexec_fn,
close_fds,
   1027                         pass_fds, cwd, env,
   1028                         startupinfo, creationflags, shell,
   1029                         p2cread, p2cwrite,
   1030                         c2pread, c2pwrite,
   1031                         errread, errwrite,
   1032                         restore_signals,
   1033                         gid, gids, uid, umask,
   1034                         start_new_session, process_group)
   1035 except:
   1036     # Cleanup if the child failed starting.

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
subprocess.py:1950, in Popen._execute_child(self, args, executable,
preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags,
shell, p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite,
restore_signals, gid, gids, uid, umask, start_new_session,
process_group)
   1949         err_msg = os.strerror(errno_num)
-> 1950     raise child_exception_type(errno_num, err_msg,
err_filename)
   1951 raise child_exception_type(err_msg)

FileNotFoundError: [Errno 2] No such file or directory:
PosixPath('dot')

The above exception was the direct cause of the following exception:

ExecutableNotFound                          Traceback (most recent call
last)
Cell In[142], line 4
      2 dot_data = tree.export_graphviz(clf, out_file=None)
      3 graph = graphviz.Source(dot_data)
----> 4 graph.render("iris")

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/_tools.py:171, in
deprecate_positional_args.<locals>.decorator.<locals>.wrapper(*args,
**kwargs)
    162     wanted = ', '.join(f'{name}={value!r}'
    163                        for name, value in deprecated.items())
    164     warnings.warn(f'The signature of {func.__name__} will be
reduced'
    165                   f' to {supported_number} positional args'
    166                   f' {list(supported)}: pass {wanted}'
```

```
    167                   ' as keyword arg(s)',
    168                   stacklevel=stacklevel,
    169                   category=category)
--> 171 return func(*args, **kwargs)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/rendering.py:122, in Render.render(self, filename,
directory, view, cleanup, format, renderer, formatter, neato_no_op,
quiet, quiet_view, outfile, engine, raise_if_result_exists,
overwrite_source)
    118 filepath = self.save(filename, directory=directory,
skip_existing=None)
    120 args.append(filepath)
--> 122 rendered = self._render(*args, **kwargs)
    124 if cleanup:
    125     log.debug('delete %r', filepath)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/_tools.py:171, in
deprecate_positional_args.<locals>.decorator.<locals>.wrapper(*args,
**kwargs)
    162     wanted = ', '.join(f'{name}={value!r}'
    163                        for name, value in deprecated.items())
    164     warnings.warn(f'The signature of {func.__name__} will be
reduced'
    165                   f' to {supported_number} positional args'
    166                   f' {list(supported)}: pass {wanted}'
    167                   ' as keyword arg(s)',
    168                   stacklevel=stacklevel,
    169                   category=category)
--> 171 return func(*args, **kwargs)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/rendering.py:324, in render(engine, format,
filepath, renderer, formatter, neato_no_op, quiet, outfile,
raise_if_result_exists, overwrite_filepath)
    320     raise exceptions.FileExistsError(f'output file exists:
{os.fspath(outfile)!r}')
    322 cmd += args
--> 324 execute.run_check(cmd,
    325                   cwd=filepath.parent if filepath.parent.parts
else None,
    326                   quiet=quiet,
    327                   capture_output=True)
    329 return os.fspath(outfile)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/execute.py:84, in run_check(cmd,
input_lines, encoding, quiet, **kwargs)
     82 except OSError as e:
```

```
     83     if e.errno == errno.ENOENT:
---> 84         raise ExecutableNotFound(cmd) from e
     85     raise
     87 if not quiet and proc.stderr:

ExecutableNotFound: failed to execute PosixPath('dot'), make sure the
Graphviz executables are on your systems' PATH
```

```
dot_data = tree.export_graphviz(clf, out_file=None,
                      feature_names=iris.feature_names,
                      class_names=iris.target_names,
                      filled=True, rounded=True,
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/execute.py:79, in run_check(cmd,
input_lines, encoding, quiet, **kwargs)
     78         kwargs['stdout'] = kwargs['stderr'] = subprocess.PIPE
---> 79     proc = _run_input_lines(cmd, input_lines, kwargs=kwargs)
     80 else:

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/execute.py:99, in _run_input_lines(cmd,
input_lines, kwargs)
     98 def _run_input_lines(cmd, input_lines, *, kwargs):
---> 99     popen = subprocess.Popen(cmd, stdin=subprocess.PIPE,
**kwargs)
    101     stdin_write = popen.stdin.write

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
subprocess.py:1026, in Popen.__init__(self, args, bufsize, executable,
stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,
universal_newlines, startupinfo, creationflags, restore_signals,
start_new_session, pass_fds, user, group, extra_groups, encoding,
errors, text, umask, pipesize, process_group)
   1023             self.stderr = io.TextIOWrapper(self.stderr,
   1024                     encoding=encoding, errors=errors)
-> 1026     self._execute_child(args, executable, preexec_fn,
close_fds,
   1027                         pass_fds, cwd, env,
   1028                         startupinfo, creationflags, shell,
   1029                         p2cread, p2cwrite,
   1030                         c2pread, c2pwrite,
```

```
   1031                            errread, errwrite,
   1032                            restore_signals,
   1033                            gid, gids, uid, umask,
   1034                            start_new_session, process_group)
   1035 except:
   1036     # Cleanup if the child failed starting.

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
subprocess.py:1950, in Popen._execute_child(self, args, executable,
preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags,
shell, p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite,
restore_signals, gid, gids, uid, umask, start_new_session,
process_group)
   1949         err_msg = os.strerror(errno_num)
-> 1950     raise child_exception_type(errno_num, err_msg,
err_filename)
   1951 raise child_exception_type(err_msg)

FileNotFoundError: [Errno 2] No such file or directory:
PosixPath('dot')

The above exception was the direct cause of the following exception:

ExecutableNotFound                          Traceback (most recent call
last)
File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/IPython/core/formatters.py:974, in
MimeBundleFormatter.__call__(self, obj, include, exclude)
    971     method = get_real_method(obj, self.print_method)
    973     if method is not None:
--> 974         return method(include=include, exclude=exclude)
    975     return None
    976 else:

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/jupyter_integration.py:98, in
JupyterIntegration._repr_mimebundle_(self, include, exclude, **_)
     96 include = set(include) if include is not None else
{self._jupyter_mimetype}
     97 include -= set(exclude or [])
---> 98 return {mimetype: getattr(self, method_name)()
     99         for mimetype, method_name in MIME_TYPES.items()
    100         if mimetype in include}

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/jupyter_integration.py:112, in
JupyterIntegration._repr_image_svg_xml(self)
    110 def _repr_image_svg_xml(self) -> str:
    111     """Return the rendered graph as SVG string."""
```

```
--> 112     return self.pipe(format='svg', encoding=SVG_ENCODING)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/piping.py:104, in Pipe.pipe(self, format, renderer,
formatter, neato_no_op, quiet, engine, encoding)
     55 def pipe(self,
     56          format: typing.Optional[str] = None,
     57          renderer: typing.Optional[str] = None,
   (...)
     61          engine: typing.Optional[str] = None,
     62          encoding: typing.Optional[str] = None) ->
typing.Union[bytes, str]:
     63     """Return the source piped through the Graphviz layout
command.
     64
     65     Args:
   (...)
    102         '<?xml version='
    103     """
--> 104     return self._pipe_legacy(format,
    105                              renderer=renderer,
    106                              formatter=formatter,
    107                              neato_no_op=neato_no_op,
    108                              quiet=quiet,
    109                              engine=engine,
    110                              encoding=encoding)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/_tools.py:171, in
deprecate_positional_args.<locals>.decorator.<locals>.wrapper(*args,
**kwargs)
    162     wanted = ', '.join(f'{name}={value!r}'
    163                        for name, value in deprecated.items())
    164     warnings.warn(f'The signature of {func.__name__} will be
reduced'
    165                   f' to {supported_number} positional args'
    166                   f' {list(supported)}: pass {wanted}'
    167                   ' as keyword arg(s)',
    168                   stacklevel=stacklevel,
    169                   category=category)
--> 171 return func(*args, **kwargs)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/piping.py:121, in Pipe._pipe_legacy(self, format,
renderer, formatter, neato_no_op, quiet, engine, encoding)
    112 @_tools.deprecate_positional_args(supported_number=2)
    113 def _pipe_legacy(self,
    114                  format: typing.Optional[str] = None,
   (...)
    119                  engine: typing.Optional[str] = None,
```

```
    120                   encoding: typing.Optional[str] = None) ->
typing.Union[bytes, str]:
--> 121     return self._pipe_future(format,
    122                              renderer=renderer,
    123                              formatter=formatter,
    124                              neato_no_op=neato_no_op,
    125                              quiet=quiet,
    126                              engine=engine,
    127                              encoding=encoding)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/piping.py:149, in Pipe._pipe_future(self, format,
renderer, formatter, neato_no_op, quiet, engine, encoding)
    146 if encoding is not None:
    147     if codecs.lookup(encoding) is
codecs.lookup(self.encoding):
    148         # common case: both stdin and stdout need the same
encoding
--> 149         return self._pipe_lines_string(*args,
encoding=encoding, **kwargs)
    150     try:
    151         raw = self._pipe_lines(*args,
input_encoding=self.encoding, **kwargs)

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/piping.py:212, in pipe_lines_string(engine,
format, input_lines, encoding, renderer, formatter, neato_no_op,
quiet)
    206 cmd = dot_command.command(engine, format,
    207                             renderer=renderer,
    208                             formatter=formatter,
    209                             neato_no_op=neato_no_op)
    210 kwargs = {'input_lines': input_lines, 'encoding': encoding}
--> 212 proc = execute.run_check(cmd, capture_output=True,
quiet=quiet, **kwargs)
    213 return proc.stdout

File ~/Documents/Дискретна/algorithms_lab/venv/lib/python3.12/site-
packages/graphviz/backend/execute.py:84, in run_check(cmd,
input_lines, encoding, quiet, **kwargs)
     82 except OSError as e:
     83     if e.errno == errno.ENOENT:
---> 84         raise ExecutableNotFound(cmd) from e
     85     raise
     87 if not quiet and proc.stderr:

ExecutableNotFound: failed to execute PosixPath('dot'), make sure the
Graphviz executables are on your systems' PATH

<graphviz.sources.Source at 0x14f626840>
```

```
X_test.shape
```

```
(30, 4)
```

Prediction step

Now we can use our model to predict which type has a flower, basing on its parameters.

This is conducted basically via traversing the tree that you can see above.

```
predictions = clf.predict(X_test)
```

We can also measure the accuracy of our model

```
sum(predictions == y_test) / len(y_test)
```

```
0.9333333333333333
```

To get clearer intuition about predicion, let's look at those X, that should be labeled to some flower

```
y_test
```

```
array([0, 0, 0, 2, 2, 2, 0, 0, 1, 2, 2, 2, 0, 1, 1, 2, 1, 1, 1, 2, 2,
1,
       1, 2, 0, 2, 2, 2, 2, 2])
```

Here you can traverse the tree above by yourself and make sure that prediction works

```
X_test[1]
```

```
array([5. , 3.3, 1.4, 0.2])
```

```
clf.predict([X_test[1]])
```

```
array([0])
```

Finally, it is your turn to write such classifier by yourself!

Gini impurity

Decision trees use the concept of Gini impurity to describe how “pure” a node is. A node is pure (G = 0) if all its samples belong to the same class, while a node with many samples from many different classes will have a Gini closer to 1.

$$G=1-\sum_{k=1}^{n} p_k^2$$

For example, if a node contains five samples, with two belonging to the first class (first flower), two of class 2, one of class 3 and none of class 4, then

$$G = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{2}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0.64$$

```
class Node:

    def __init__(self, X, y, gini):
        self.X = X
        self.y = y
        self.gini = gini
        self.feature_index = 0
        self.threshold = 0
        self.left = None
        self.right = None

class MyDecisionTreeClassifier:

    def __init__(self, max_depth):
        self.max_depth = max_depth

    def gini(self, groups, classes):#
        '''
        A Gini score gives an idea of how good a split is by how mixed
the
        classes are in the two groups created by the split.

        A perfect separation results in a Gini score of 0,
        whereas the worst case split that results in 50/50
        classes in each group result in a Gini score of 0.5
        (for a 2 class problem).
        '''

        return 1.0 - sum((groups[x]/ sum(groups)) ** 2 for x in
range(classes))

    def split_data(self, X, y):
        """Find the best split for a node.
        "Best" means that the average impurity of the two children,
weighted by their
        population, is the smallest possible. Additionally it must be
less than the
        impurity of the current node.
        To find the best split, we loop through all the features, and
consider all the
        midpoints between adjacent training samples as possible
thresholds. We compute
        the Gini impurity of the split generated by that particular
feature/threshold
```

```
        pair, and return the pair with smallest impurity.
        Returns:
            best_idx: Index of the feature for best split, or None if
no split is found.
            best_thr: Threshold to use for the split, or None if no
split is found.
        """
        m = y.size
        if m <= 1:
            return None, None
        num_parent = [np.sum(y == c) for c in range(self.n_classes_)]
        best_gini = 1.0 - sum((n / m) ** 2 for n in num_parent)
        best_idx, best_thr = None, None
        for idx in range(self.n_features_):
            thresholds, classes = zip(*sorted(zip(X[:, idx], y)))
            num_left = [0] * self.n_classes_
            num_right = num_parent.copy()
            for i in range(1, m):
                c = classes[i - 1]
                num_left[c] += 1
                num_right[c] -= 1
                gini_left = self.gini(num_left, self.n_classes_)
                gini_right = self.gini(num_right, self.n_classes_)
                gini = (i * gini_left + (m - i) * gini_right) / m
                if thresholds[i] == thresholds[i - 1]:
                    continue

                if gini < best_gini:
                    best_gini = gini
                    best_idx = idx
                    best_thr = (thresholds[i] + thresholds[i - 1]) / 2
        return best_idx, best_thr
    def build_tree(self, X, y, depth=0):
        """Build a decision tree by recursively finding the best
split."""
        num_samples_per_class = [np.sum(y == i) for i in
range(self.n_classes_)]
        predicted_class = np.argmax(num_samples_per_class)
        node = Node(
            gini=self.gini(num_samples_per_class, self.n_classes_),
            X=num_samples_per_class,
            y=predicted_class,
        )
        if depth < self.max_depth:
            idx, thr = self.split_data(X, y)
            if idx is not None:
                indices_left = X[:, idx] < thr
                X_left, y_left = X[indices_left], y[indices_left]
                X_right, y_right = X[~indices_left], y[~indices_left]
```

```
                node.feature_index = idx
                node.threshold = thr
                node.left = self.build_tree(X_left, y_left, depth + 1)
                node.right = self.build_tree(X_right, y_right, depth +
1)
        return node

    def fit(self, X, y):
        self.n_classes_ = len(set(y))
        self.n_features_ = X.shape[1]
        self.tree_ = self.build_tree(X, y)

    def predict(self, inputs):
        """Predict class for a single sample."""
        node = self.tree_
        while node.left:
            if inputs[node.feature_index] < node.threshold:
                node = node.left
            else:
                node = node.right
        return node.y

    def evaluate(self, X_test, y_test):
        predictions = [self.predict(x_test) for x_test in X_test]
        correct = sum(y == y_pred for y, y_pred in zip(y_test,
predictions))
        return correct / len(y_test)

clf = MyDecisionTreeClassifier(max_depth=5)
clf.fit(X, y)
clf.evaluate(X_test, y_test)
```

```
0.9166666666666666
```

After executing the Decision Tree Classifier model implemented based on the MyDecisionTreeClassifier class, a test accuracy result of 0.9166666666666666 was obtained.

The evaluate method, which was used to assess the model, is implemented within the class and is responsible for comparing predicted labels with the labels in the test set and calculating the percentage of correctly classified samples.

The programmatic code of the MyDecisionTreeClassifier class includes the following methods:

The gini method, which computes the Gini coefficient to assess the purity of a split.
The split_data method, which finds the optimal data split using the Gini criterion.

```
The build_tree method, which recursively constructs the decision tree,
searching for the optimal split at each step of the construction
process.
The fit method, which trains the model on input data.
The predict method, which utilizes the constructed decision tree to
predict the class for new samples.
The evaluate method, which evaluates the model's accuracy on the test
data set.
The obtained result demonstrates a fairly high accuracy of the model
on the test data, indicating its effectiveness in data classification.
```

For those who want to do it a little bit more complicated ;) (**optional**)

Consider also using some techniques to avoid overfitting, like pruning or setting a maximum depth for the tree. You can also try to implement some other metrics, to measure the quality of a split and overall performance. Also, you can try to implement some other algorithms, like proper CART, ID3 or C4.5. You can find more information about them here:
https://scikit-learn.org/stable/modules/tree.html#tree