DigiLocal

MAKING OUR DIGITAL FUTURE

Predicting Predators

Introduction

Which of these two animals is a predator?

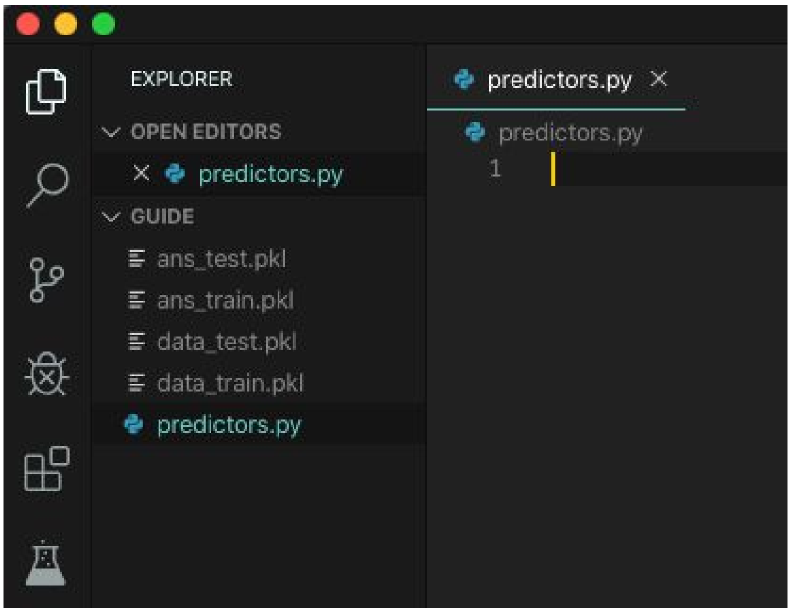
Most people will easily know the answer to this, but how about these two? Which is the predator?

Unless you are an animal expert, this isn't so easy (the answer is both)! In this project we are going to make a programme that takes in information about animals and predicts if the animals are predators or not automatically. To do this we are going to use a technique called **machine learning** to teach the computer how to recognise a predator. At the end we will have a competition to see who is the biggest animal expert, you or your computer!

Let's get started!

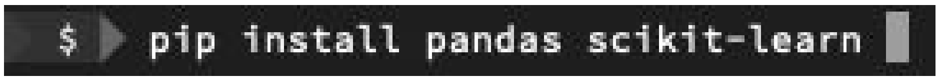
Step One: Getting started

- ✓ First things first, let's create a new python file in VS Code called `predictors.py`
- ✓ Next we need to download the data for the project, this is available from the Python-resources section of the Google Drive, there should be 4 files, called `data_train.pkl`, `data_test.pkl` `ans_train.pkl` and `ans_test.pkl`
- ✓ When you have downloaded these files, put them in the same folder as the python file you just made, your VS Code should look something like this:



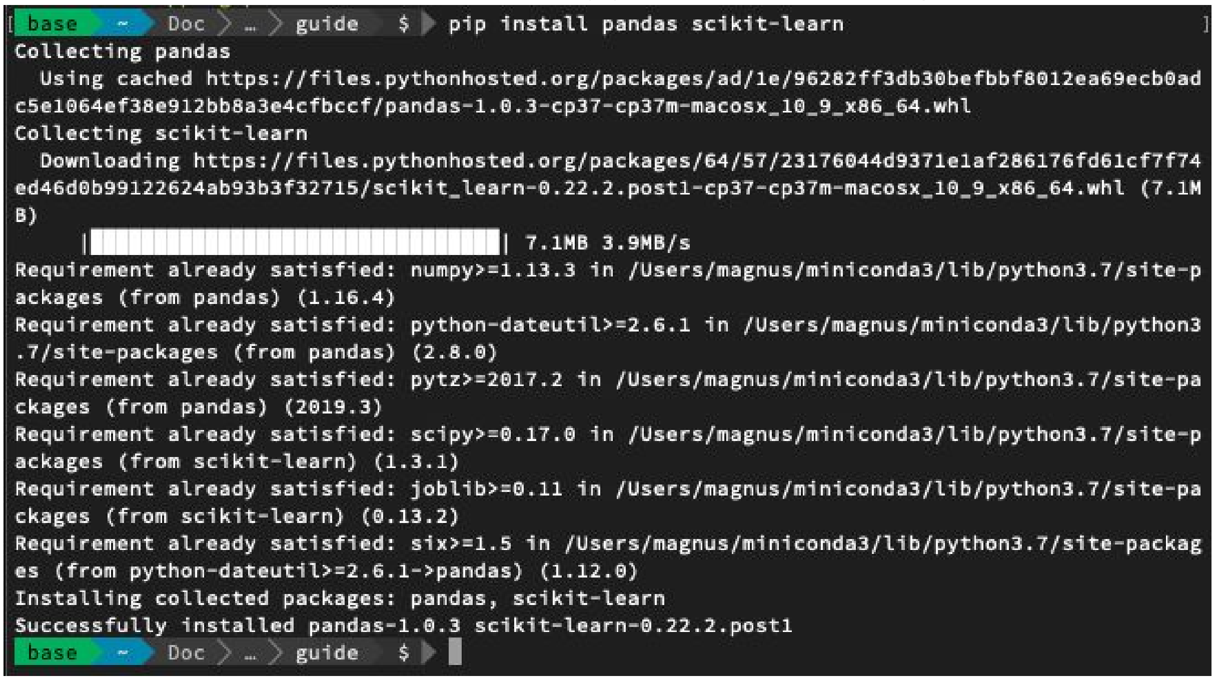
- ✓ We will need 2 extra python packages for this project, the first called pandas will help us manage our data, the second called scikit-learn will help us to do machine learning. To install these packages we need to use the terminal, open your terminal and type this command: `pip install pandas scikit-learn`

  It should look something like this.



```
$ pip install pandas scikit-learn
```

Press enter to run the command, hopefully you will see something like this:



```
[ base  ~  Doc 〉 … 〉 guide  $ 〉 pip install pandas scikit-learn                                    ]
Collecting pandas
  Using cached https://files.pythonhosted.org/packages/ad/1e/96282ff3db30befbbf8012ea69ecb0ad
c5e1064ef38e912bb8a3e4cfbccf/pandas-1.0.3-cp37-cp37m-macosx_10_9_x86_64.whl
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/64/57/23176044d9371e1af286176fd61cf7f74
ed46d0b99122624ab93b3f32715/scikit_learn-0.22.2.post1-cp37-cp37m-macosx_10_9_x86_64.whl (7.1M
B)
      |████████████████████████████████| 7.1MB 3.9MB/s
Requirement already satisfied: numpy>=1.13.3 in /Users/magnus/miniconda3/lib/python3.7/site-p
ackages (from pandas) (1.16.4)
Requirement already satisfied: python-dateutil>=2.6.1 in /Users/magnus/miniconda3/lib/python3
.7/site-packages (from pandas) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in /Users/magnus/miniconda3/lib/python3.7/site-pa
ckages (from pandas) (2019.3)
Requirement already satisfied: scipy>=0.17.0 in /Users/magnus/miniconda3/lib/python3.7/site-p
ackages (from scikit-learn) (1.3.1)
Requirement already satisfied: joblib>=0.11 in /Users/magnus/miniconda3/lib/python3.7/site-pa
ckages (from scikit-learn) (0.13.2)
Requirement already satisfied: six>=1.5 in /Users/magnus/miniconda3/lib/python3.7/site-packag
es (from python-dateutil>=2.6.1->pandas) (1.12.0)
Installing collected packages: pandas, scikit-learn
Successfully installed pandas-1.0.3 scikit-learn-0.22.2.post1
 base  ~  Doc 〉 … 〉 guide  $ 〉
```

It won’t look exactly the same, but somewhere it should say ‘Successfully installed’. Note, if you already have the packages it will say something like ‘Requirement already satisfied’ this is also fine!

We should now be ready, let’s get going!

Step Two: Exploring the Data

✓ First we need to import the packages we just installed so we can make use of them. Type these lines into `predictors.py`:

```
import pandas
from sklearn import tree
```

✓ Next we need to load our data, there are 2 sets of data files and 2 sets of answer files, one with ‘train’ in the name and one with ‘test’. We will use the train data to teach the computer about the animals, and the test data to see how much it knows at the end of the project. We can’t let it see the test yet as that would make it too easy, imagine if your teacher let you see the answers to an exam before you took it! Let’s load the 2 train files, type these lines:

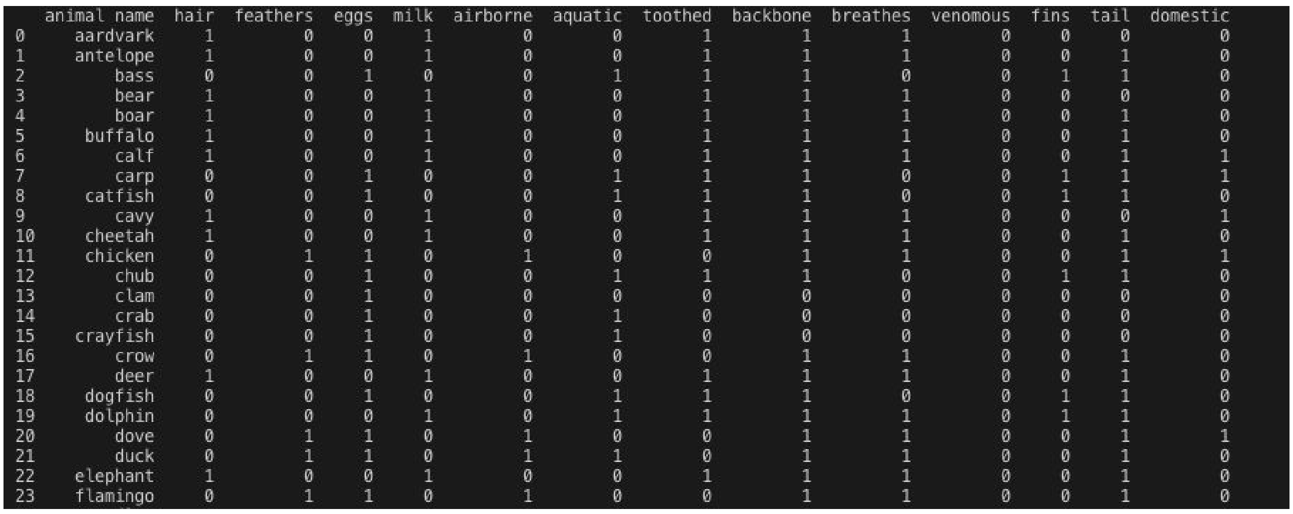
```
data_train = pandas.read_pickle('data_train.pkl')
ans_train = pandas.read_pickle('ans_train.pkl')
```

✓ Let’s have a look at the data, type:

```
print(data_train)
```

✓ Run the code, we should get something that looks a bit like this!

Test your code!

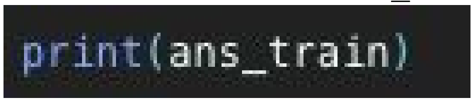


```
    animal name  hair  feathers  eggs  milk  airborne  aquatic  toothed  backbone  breathes  venomous  fins  tail  domestic
0      aardvark     1         0     0     1         0        0        1         1         1         0     0     0         0
1      antelope     1         0     0     1         0        0        1         1         1         0     0     1         0
2          bass     0         0     1     0         0        1        1         1         0         0     1     1         0
3          bear     1         0     0     1         0        0        1         1         1         0     0     0         0
4          boar     1         0     0     1         0        0        1         1         1         0     0     1         0
5       buffalo     1         0     0     1         0        0        1         1         1         0     0     1         0
6          calf     1         0     0     1         0        0        1         1         1         0     0     1         1
7          carp     0         0     1     0         0        1        1         1         0         0     1     1         1
8       catfish     0         0     1     0         0        1        1         1         0         0     1     1         0
9          cavy     1         0     0     1         0        0        1         1         1         0     0     0         1
10      cheetah     1         0     0     1         0        0        1         1         1         0     0     1         0
11      chicken     0         1     1     0         1        0        0         1         1         0     0     1         1
12         chub     0         0     1     0         0        1        1         1         0         0     1     1         0
13         clam     0         0     1     0         0        0        0         0         0         0     0     0         0
14         crab     0         0     1     0         0        1        0         0         0         0     0     0         0
15     crayfish     0         0     1     0         0        1        0         0         0         0     0     0         0
16         crow     0         1     1     0         1        0        0         1         1         0     0     1         0
17         deer     1         0     0     1         0        0        1         1         1         0     0     1         0
18      dogfish     0         0     1     0         0        1        1         1         0         0     1     1         0
19      dolphin     0         0     0     1         0        1        1         1         1         0     1     1         0
20         dove     0         1     1     0         1        0        0         1         1         0     0     1         1
21         duck     0         1     1     0         1        1        0         1         1         0     0     1         0
22     elephant     1         0     0     1         0        0        1         1         1         0     0     1         0
23     flamingo     0         1     1     0         1        0        0         1         1         0     0     1         0
```

Predicting Predators

Your table will be bigger than this, it should have rows from 0-49. The table gives us lots of information about different animals, it looks confusing at first but the table is quite simple to read. Look at the first row that starts with ‘aardvark’, if we look at the first column for ‘aardvark’, called ‘hair’ we can see that it has a ‘1’ in. A 1 means that an aardvark has hair! Look at the next column, called ‘feathers’, this column has a 0 in it, this means that an aardvark does not have feathers. Look at the final column called ‘domestic’, it has a 0 in which means aardvarks are not domestic, have you ever seen a pet aardvark?!

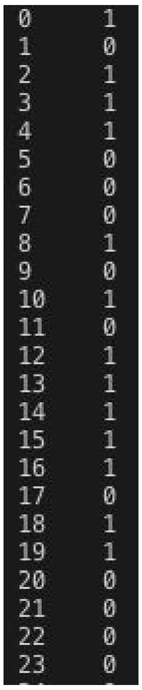
Now we have some information about our animals, we need to find out which are predators and which are not, so we can teach the computer. Print the `ans_train` variable, which tells us the answer to which of these animals is a predator. In the print function, change `data_train` to `ans_train`. It should now look like this:



```
print(ans_train)
```

Run this code! The output will look like this:

Test your code!



```
0     1
1     0
2     1
3     1
4     1
5     0
6     0
7     0
8     1
9     0
10    1
11    0
12    1
13    1
14    1
15    1
16    1
17    0
18    1
19    1
20    0
21    0
22    0
23    0
```

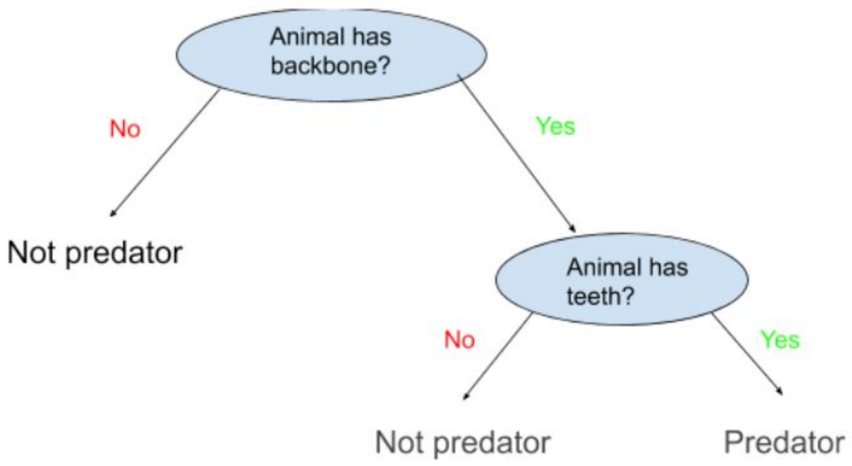
This tells us which of the animals from before are predators, the number on the left tells us the column of our `data_train` table. We can see that the number 3 has a 1 next to it, now look in row number 3 from our data table. This is the row for a bear, the 1 in the answers table means **a bear is a predator.** Next to the number 23 there is a 0, now look at row number 23 in the data table, it should be the row for flamingo, this means **a flamingo is not a predator.**

Challenge time!

Answer these animal questions using the data:
Does a dolphin have teeth?
Are fleas airborne?
Is a boar a predator?

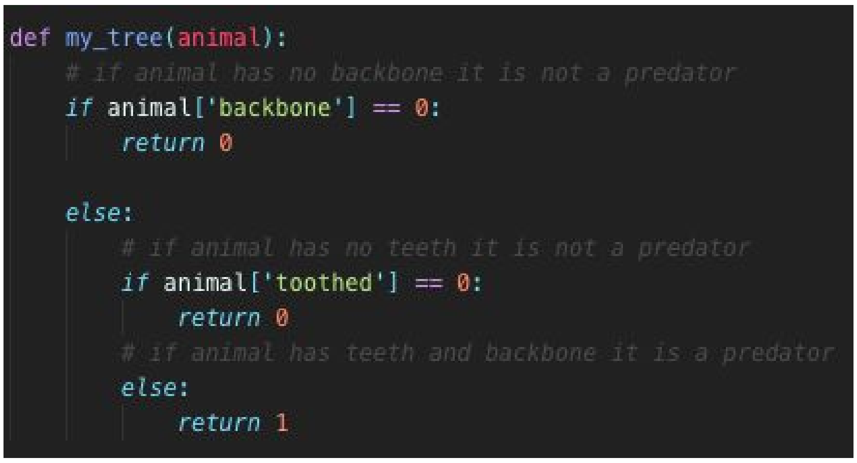
Step Three: Making a Tree

To help the computer decide which animal is a predator we are going to use something called a ‘decision tree’. A decision tree lets us write down a set of rules about our animals in a simple way. Here is an example:



By answering the questions in the ovals we can decide if an animal is a predator. Try to use this tree to decide if an aardvark is a predator using the data? Can you check you have the right answer using `ans_train`?

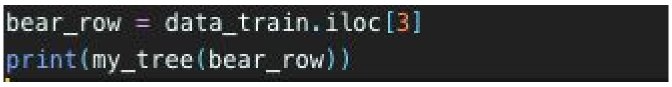
Now we are going to convert the tree above into a python function. We want the function to take the information about the animal (the rows of the table) as an argument and return 1 if the animal is a predator and 0 if not. We can access properties about the animal like this, `animal['backbone']` will be 1 if the animal has a backbone. The tree above looks like this:



```python
def my_tree(animal):
    # if animal has no backbone it is not a predator
    if animal['backbone'] == 0:
        return 0

    else:
        # if animal has no teeth it is not a predator
        if animal['toothed'] == 0:
            return 0
        # if animal has teeth and backbone it is a predator
        else:
            return 1
```

This may look a bit daunting but we can see that we just turn each question from the diagram above into an if statement. Put this code into your file, you don’t need to copy the comments, they are just to help explain what’s going on!

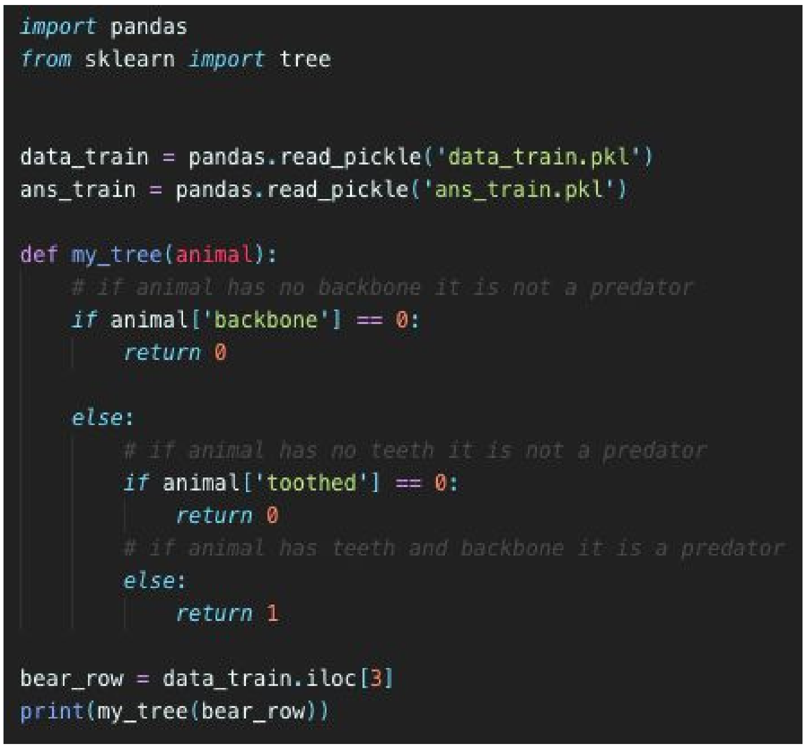
✓ Let's test our tree! We can try it on bear since we know that it should be a predator. The bear is in row 3 of the table, you can check this by printing the table again or looking on page 4 of the guide. Below your function write this code:



```
bear_row = data_train.iloc[3]
print(my_tree(bear_row))
```

Don't be confused by the iloc, that just gets the row from our table. Delete the print from the previous section, we don't need it anymore.

✓ All together, your file should look like this (you don't need the comments):



```
import pandas
from sklearn import tree


data_train = pandas.read_pickle('data_train.pkl')
ans_train = pandas.read_pickle('ans_train.pkl')

def my_tree(animal):
    # if animal has no backbone it is not a predator
    if animal['backbone'] == 0:
        return 0

    else:
        # if animal has no teeth it is not a predator
        if animal['toothed'] == 0:
            return 0
        # if animal has teeth and backbone it is a predator
        else:
            return 1

bear_row = data_train.iloc[3]
print(my_tree(bear_row))
```

▷ Test your code!

Run the code, it should print the number 1, this means we have predicted that a bear is a predator, we got it right!

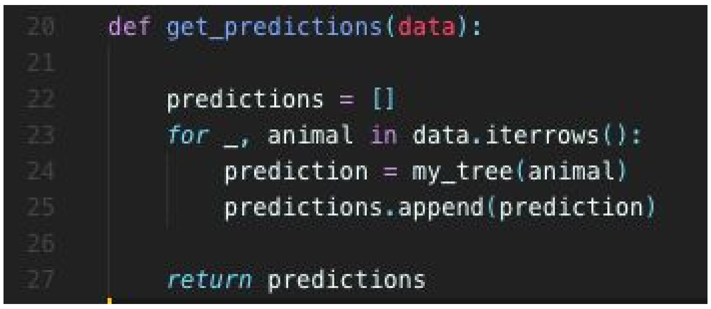
Challenge time!

Use the tree to predict which other animals are predators, how many do you get right?

Extra challenge: Can you make your own tree using different rules? Make it into a python function and see if it works!

Step Three: Testing our tree

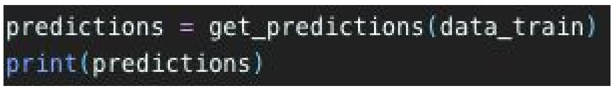
In the last section we created our first decision tree and used it on a few animals. To see how well our tree is doing we need to see what it predicts for all the different animals. Then we can quickly check how many animals we are getting right. First let's write some a function to get the prediction for every animal and put it in a list To do this we need to write a for loop like this:
Let's go through this code line by line. First on line 20 we define the function, remember it needs to take our data in. On line 22 we make an empty list, this is where we will put our predictions. Line 23 loops through our data, `data.iterrows()` is just like a list with each element containing one row, and that rows index in the table we don't need the index so we throw it away using the _ variable name. Don't worry if this is confusing, you just need to know that this line



```python
20   def get_predictions(data):
21
22       predictions = []
23       for _, animal in data.iterrows():
24           prediction = my_tree(animal)
25           predictions.append(prediction)
26
27       return predictions
```

loops through all our data. On like 24 we predict for each animal, and then put the prediction in our list on line 25. Finally we return the list.

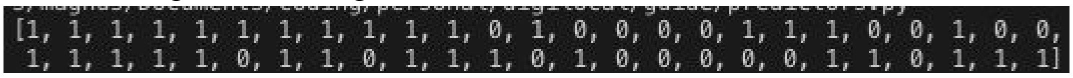
Let's test this code. Print the predictions for our data:



```python
predictions = get_predictions(data_train)
print(predictions)
```

Test your code!

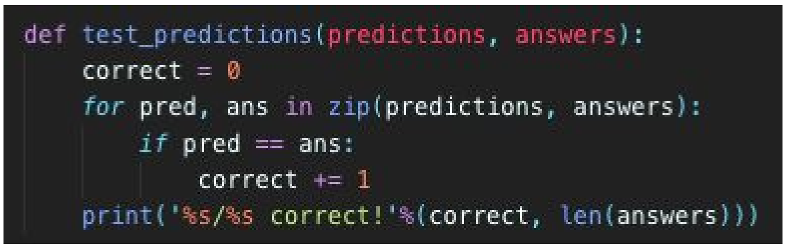
You should get something like this:



```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1]
```

This is the prediction for every animal, remember 1 is predator, 0 is not!

It's still pretty hard to tell how well our tree is doing, we need to check if our predictions match the answers. Let's write a function to do this:



```python
def test_predictions(predictions, answers):
    correct = 0
    for pred, ans in zip(predictions, answers):
        if pred == ans:
            correct += 1
    print('%s/%s correct!'%(correct, len(answers)))
```

All this code does is check if each prediction is the same as each answer, if it is it adds one to the variable `correct`, finally it prints the number we got right and the total number of answers.

Predicting Predators

Write this function in your file, then let's test it out. You can delete the other print statements from before then write this at the bottom of your code:

```
predictions = get_predictions(data_train)
test_predictions(predictions, ans_train)
```

Remember ans_train contains the answers! All together you should have this:

Run this code.

Test your code!

You should get an output like this, telling us that our tree got 27 animals out of 50 correct. Not bad!

```
27/50 correct!
```

```
import pandas
from sklearn import tree


data_train = pandas.read_pickle('data_train.pkl')
ans_train = pandas.read_pickle('ans_train.pkl')

def my_tree(animal):
    # if animal has no backbone it is not a predator
    if animal['backbone'] == 0:
        return 0

    else:
        # if animal has no teeth it is not a predator
        if animal['toothed'] == 0:
            return 0
        # if animal has teeth and backbone it is a preda
        else:
            return 1

def get_predictions(data):

    predictions = []
    for _, animal in data.iterrows():
        prediction = my_tree(animal)
        predictions.append(prediction)

    return predictions


def test_predictions(predictions, answers):
    correct = 0
    for pred, ans in zip(predictions, answers):
        if pred == ans:
            correct += 1
    print('%s/%s correct!'%(correct, len(answers)))



predictions = get_predictions(data_train)
test_predictions(predictions, ans_train)
```

Challenge time!

Change some of the rules in the tree, choose some different attributes (like 'venomous' or 'fins') and see if you can make the score higher. What's the highest score you can get?

Step Three: Automatic Tree Building

- Ok so we have a decision tree that can say if an animal is a predator or not, but it is only getting about half of the answers correct! That's obviously not great, so how can we do better? One way is to use an automatic algorithm to build the best decision tree for our data. We can use `sklearn` to do this. If you are interested in learning more about how this automatic building algorithm works you can follow this link.
- First let's create the tree:

```
auto_tree = tree.DecisionTreeClassifier()
```

- Next we need to show our data to the tree, so it can work out what the best decisions are to make:

```
auto_tree = auto_tree.fit(data_train.drop(columns='animal name'), ans_train)
```

The drop here removes the names column from our data, the computer only knows how to use numbers to classify things, and can't use names, so we have to get rid of them.

- Now let's predict using our new tree:

```
auto_preds = auto_tree.predict(data_train.drop(columns='animal name'))
```

Remember the drop again!

- Finally let's test the new tree's predictions

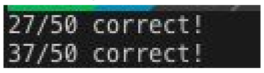
```
test_predictions(auto_preds, ans_train)
```

- Let's run the code, it should look like this below you three functions:

```
predictions = get_predictions(data_train)
test_predictions(predictions, ans_train)

auto_tree = tree.DecisionTreeClassifier()
auto_tree = auto_tree.fit(data_train.drop(columns='animal name'), ans_train)

auto_preds = auto_tree.predict(data_train.drop(columns='animal name'))

test_predictions(auto_preds, ans_train)
```

Test your code!

✓ When you run the code , you should get something like this:



```
27/50 correct!
37/50 correct!
```

The top score is for your tree, the bottom score is for the automatically built tree, it does much better!

Challenge time!

You should have 2 files in your folder that we haven't used yet, called `data_test.pkl` and `ans_test.pkl`, load up these files and write some code to test both your tree and the auto tree on the new data. How well do the trees perform on the new data?