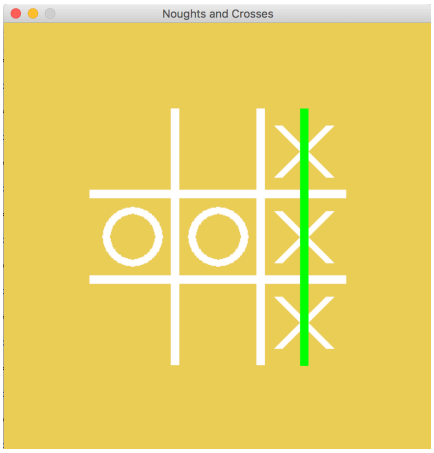
Owen Green

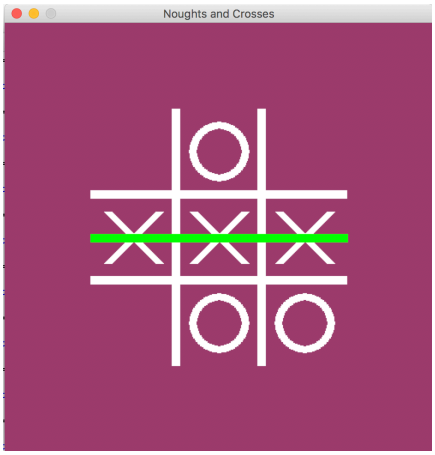
Noughts & Crosses

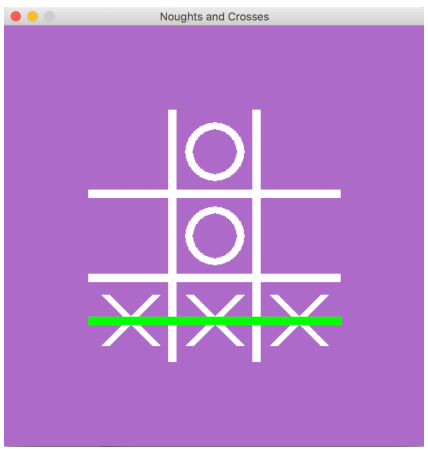
In this project you will create a noughts and crosses game in Python that is able to learn from how you play.

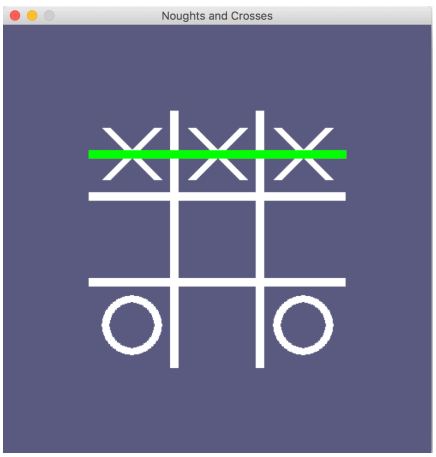
You won't give it instructions for how to play, or tell it what the objective or rules of the game are.

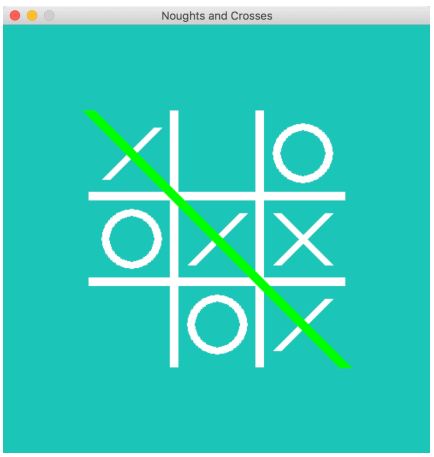
Instead, you'll show it examples of you playing the game. When it's seen enough examples to start trying to play for itself, you'll tell when it beats you.



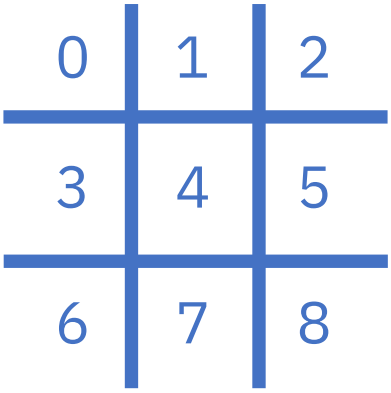






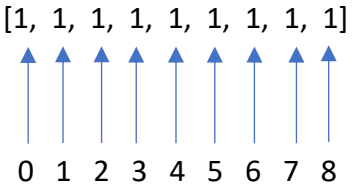


Representing noughts and crosses in Python



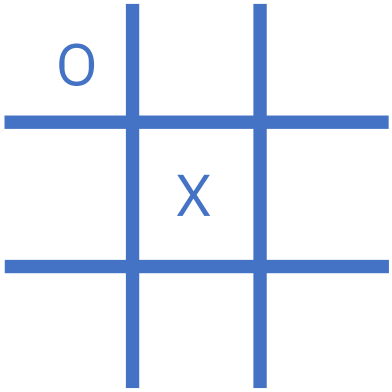
The positions of spaces on the noughts and crosses board are numbered from 0 to 8.

They are then stored in a list.



Empty = 1
O = 2
X = 3

An empty space is shown as a 1.
A nought O is shown as a 2.
A cross X is shown as a 3.



The board is represented as a list of 9 1's, 2's and 3's, one for each space

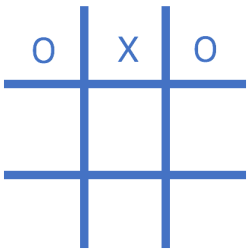
For example, at this point:

[2, 1, 1, 1, 3, 1, 1, 1, 1]

What are you going to do?

You're going to train a computer to play noughts and crosses. You'll do this by showing it examples of how you play the game.

Imagine the board looks like this and it's X's turn.

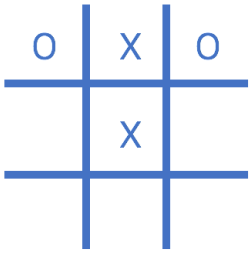


Imagine you decide to put your X in the centre space.

top-left	opponent
top-middle	player
top-right	opponent
middle-left	empty
middle-middle	empty
middle-right	empty
bottom-left	empty
bottom-middle	empty
bottom-right	empty

choice : middle-middle

Imagine the board looks like this and it's O's turn.



Imagine you decide to put your O in the bottom middle space.

top-left	player
top-middle	opponent
top-right	player
middle-left	empty
middle-middle	opponent
middle-right	empty
bottom-left	empty
bottom-middle	empty
bottom-right	empty

choice : bottom-middle

Using "opponent" and "player" instead of "nought" and "cross" means the computer can learn from both nought and cross moves.

You'll use examples of moves from the player that wins the game to train the computer.

If you (X) win, you'll use your moves as examples to train the computer. If the computer (O) wins, you'll use the computer's moves to train with.

These **examples of moves that lead to winning** will teach the computer how to play to win!

1. Go to https://machinelearningforkids.co.uk in a browser.

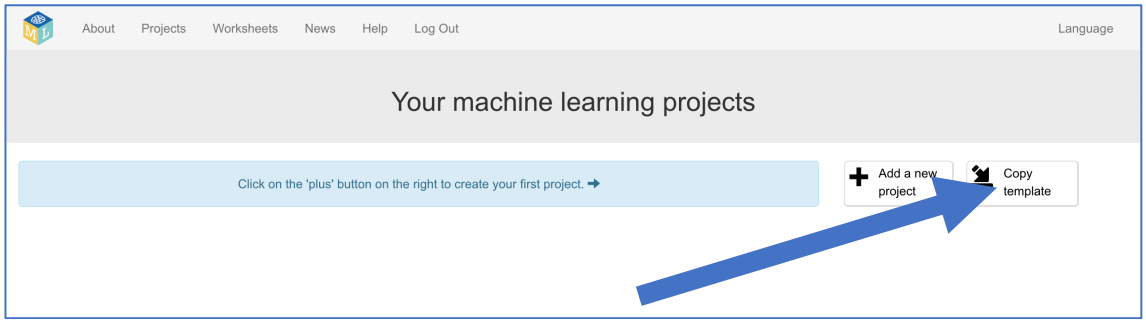
2. Click on "**Get started**"

3. Click on "**Log In**" and type in your username and password

If you don't have a username, ask your teacher to create one for you.
If you can't remember your password, ask your teacher to reset it for you.

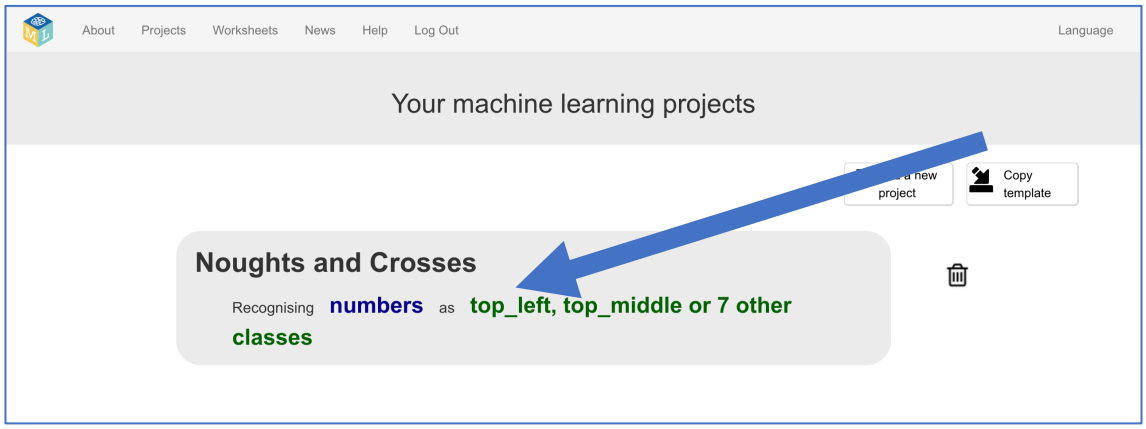
4. Click on "**Projects**" on the top menu bar

5. Click on the **"Copy template**" button.

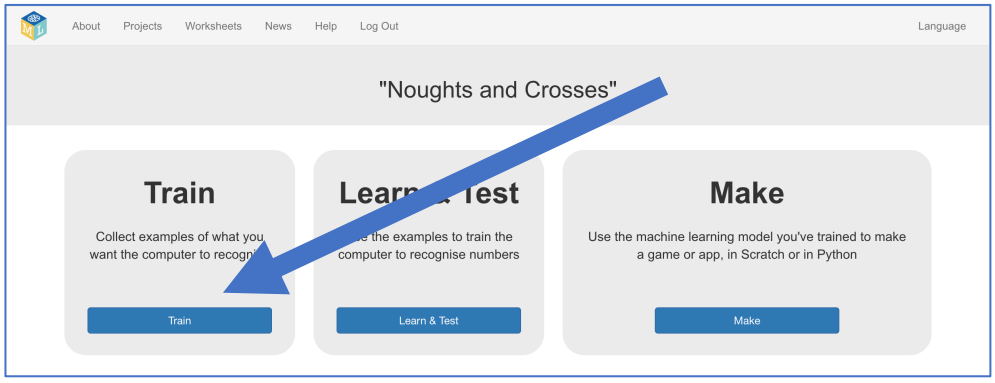


6. Import the "**Noughts and Crosses**" project template.

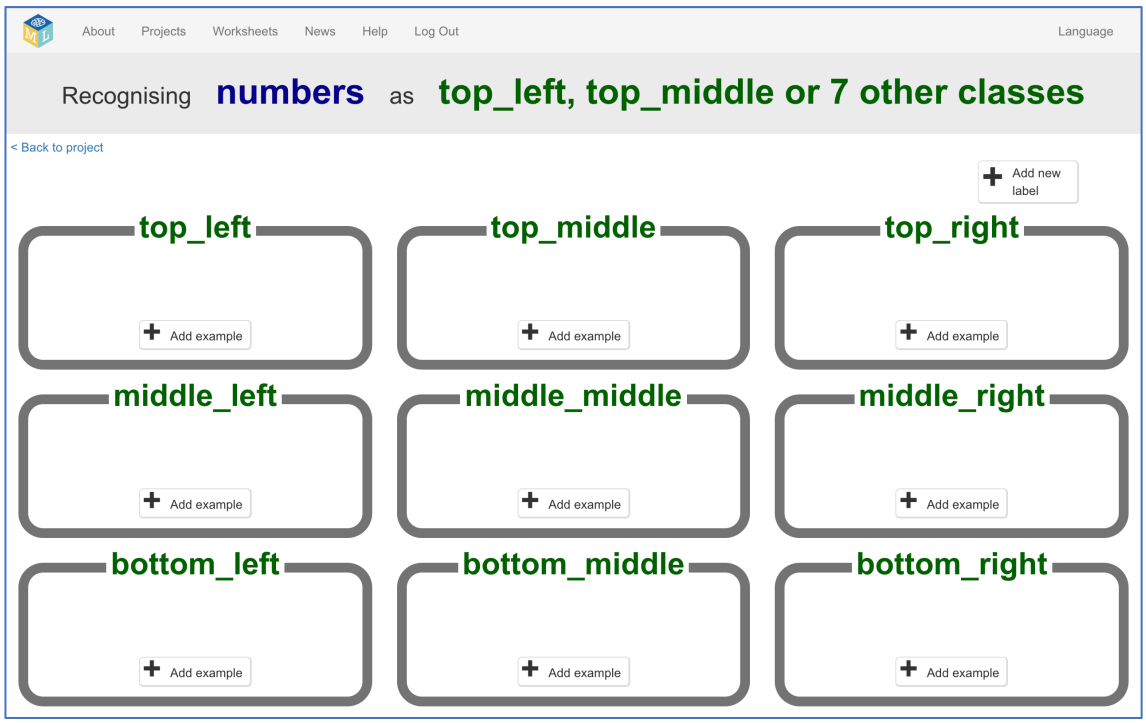
7. You should see "**Noughts and Crosses**" in your list of projects. Click on it.



8. Click the “**Train**” button

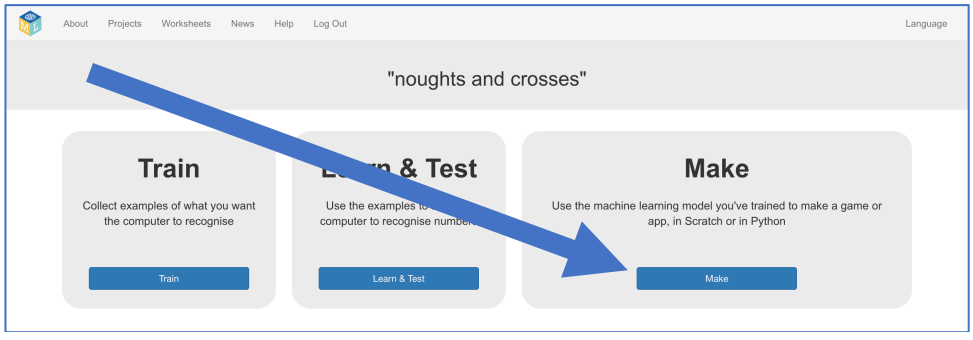


9. The template project has training buckets to store examples of noughts and crosses game moves.



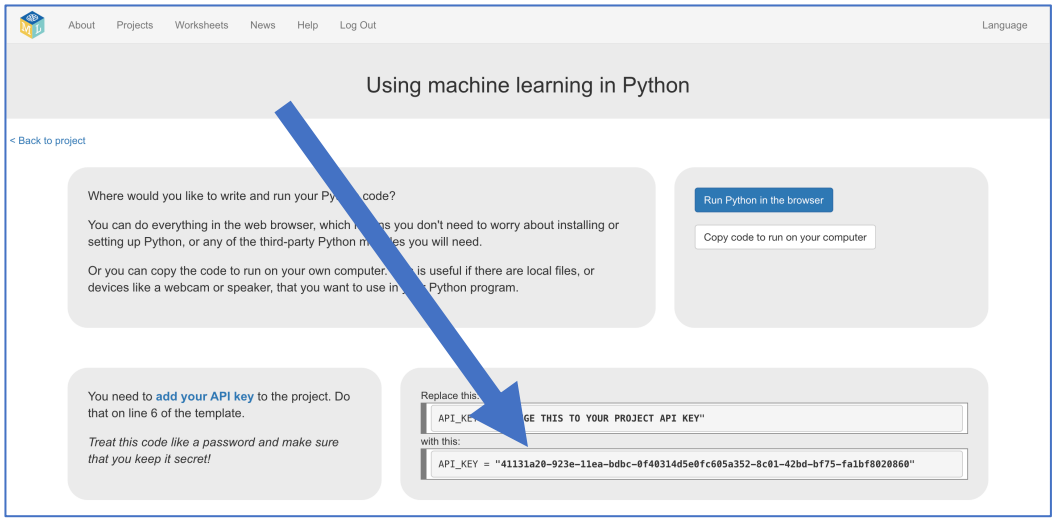
10. Click the “**< Back to project**” link.

11. Click on “**Make**”



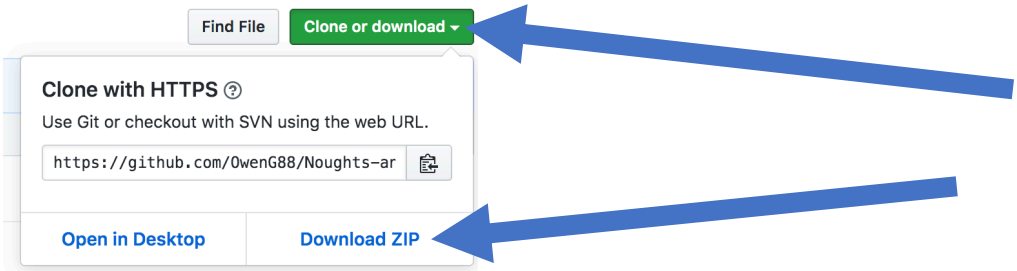
12. Click on “**Python**”

13. Find your project “API key”. You’ll need this code later.



14. **Visit** https://github.com/dalelane/Noughts-and-Crosses

15. Click the “**Clone or download**” button, then click “**Download ZIP**”

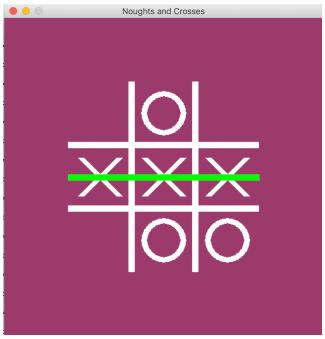


16. Unpack the ZIP and open the Python code in your preferred editor.

17. Paste the API key from Step 22 into the KEY variable

18. Make sure you have **requests** and **pygame** installed.
Ask your teacher if you don't know how to do this.

19. Run the Python program.
Play a game of noughts and crosses against the computer.



Your machine learning project will be choosing where to make it's moves at random.

20. Look at the output from the Python program after the game ends.
Next, you will edit the Python program so that the moves made by whoever wins the game are used to train your machine learning model.

```
HUMAN won the game!
Maybe the computer could learn from HUMAN's experience?

At the start of move 1 the board looked like this:
['EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY']
And HUMAN decided to put their mark in bottom_left

At the start of move 2 the board looked like this:
['COMPUTER', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'EMPTY', 'HUMAN', 'EMPTY', 'EMPTY']
And HUMAN decided to put their mark in middle_middle

At the start of move 3 the board looked like this:
['COMPUTER', 'EMPTY', 'EMPTY', 'EMPTY', 'HUMAN', 'EMPTY', 'HUMAN', 'EMPTY', 'COMPUTER']
And HUMAN decided to put their mark in top_right
```

21. Find the `learn_from_this` function
It is printing out the moves made by whoever won the game.

```
185
186 # Someone won the game.
187 #  A machine learning model could learn from this...
188 #
189 #  winner          : who won - either HUMAN or COMPUTER
190 #  boardhistory    : the contents of the game board at each stage in the game
191 #  winnerdecisions : each of the decisions that the winner made
192 def learn_from_this(winner, boardhistory, winnerdecisions):
193     print("%s won the game!" % (winner))
194     print("Maybe the computer could learn from %s's experience?" % (winner))
195     for idx in range(len(winnerdecisions)):
196         print("\nAt the start of move %d the board looked like this:" % (idx + 1))
197         print(boardhistory[idx])
198         print("And %s decided to put their mark in %s" % (winner, winnerdecisions[idx]))
199
200
```

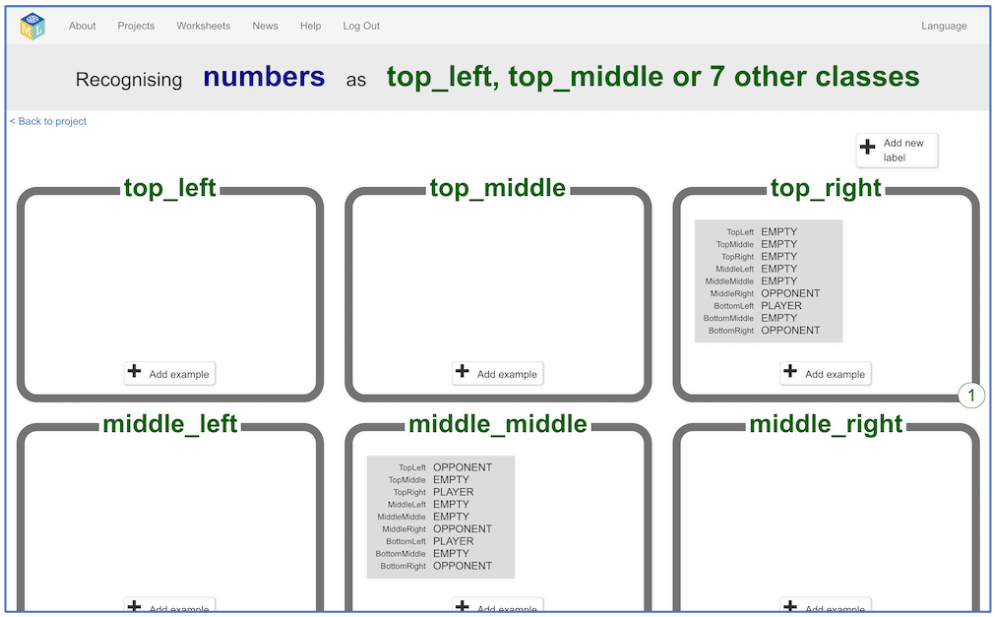
22. Add a line in the `for` loop so each move is added to your project training data

```
184
185
186   # Someone won the game.
187   #  A machine learning model could learn from this...
188   #
189   #  winner          : who won - either HUMAN or COMPUTER
190   #  boardhistory    : the contents of the game board at each stage in the game
191   #  winnerdecisions : each of the decisions that the winner made
192   def learn_from_this(winner, boardhistory, winnerdecisions):
193       print("%s won the game!" % (winner))
194       print("Maybe the computer could learn from %s's experience?" % (winner))
195       for idx in range(len(winnerdecisions)):
196           print("\nAt the start of move %d the board looked like this:" % (idx + 1))
197           print(boardhistory[idx])
198           print("And %s decided to put their mark in %s" % (winner, winnerdecisions[idx]))
199           add_to_train(boardhistory[idx], winner, winnerdecisions[idx])
200
201
```

23. Run your Python program again and play another game.

24. Go back to the training page

Leave the Python window to go back to the training tool window.
*Click the "**< Back to project**" link and then click "**Train**"*



25. Look at your training so far

Each item is a move made by the winning player.
The details in each item describe the state of the board at the time the winning player made that move.
This is collecting training data, but you still need to use it to train a machine learning model.

26. Edit the `learn_from_this` function again to add a new line

The new line will use the training data collected so far to create a new machine learning model.

Make sure you get the indenting right so you only train a model after adding all of the moves.

```
189   #  winner          : who won - either HUMAN or COMPUTER
190   #  boardhistory    : the contents of the game board at each stage in the game
191   #  winnerdecisions : each of the decisions that the winner made
192   def learn_from_this(winner, boardhistory, winnerdecisions):
193       print("%s won the game!" % (winner))
194       print("Maybe the computer could learn from %s's experience?" % (winner))
195       for idx in range(len(winnerdecisions)):
196           print("\nAt the start of move %d the board looked like this:" % (idx + 1))
197           print(boardhistory[idx])
198           print("And %s decided to put their mark in %s" % (winner, winnerdeci
199           add_to_train(boardhistory[idx], winner, winner
200       train_new_model()
201
202
```

27. Play several more games.

Your changes to the code mean that after every game finishes, the moves made by the winner are added to the training data as examples of how to win, and a new model is trained using all examples collected so far.

The longer you play, the more the computer can learn from you, and the better it should get.

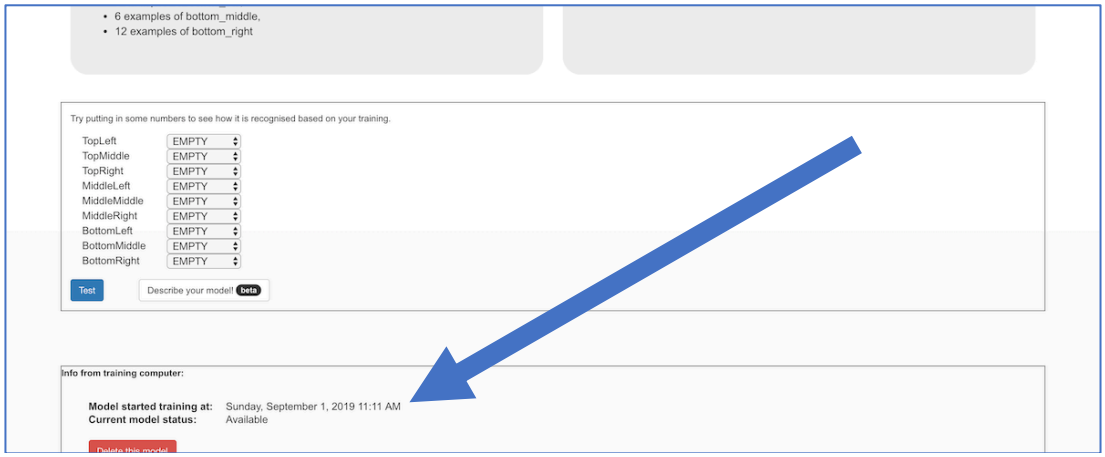
28. Go back to training page and look at all of your training examples

You will need to refresh the page to see them.

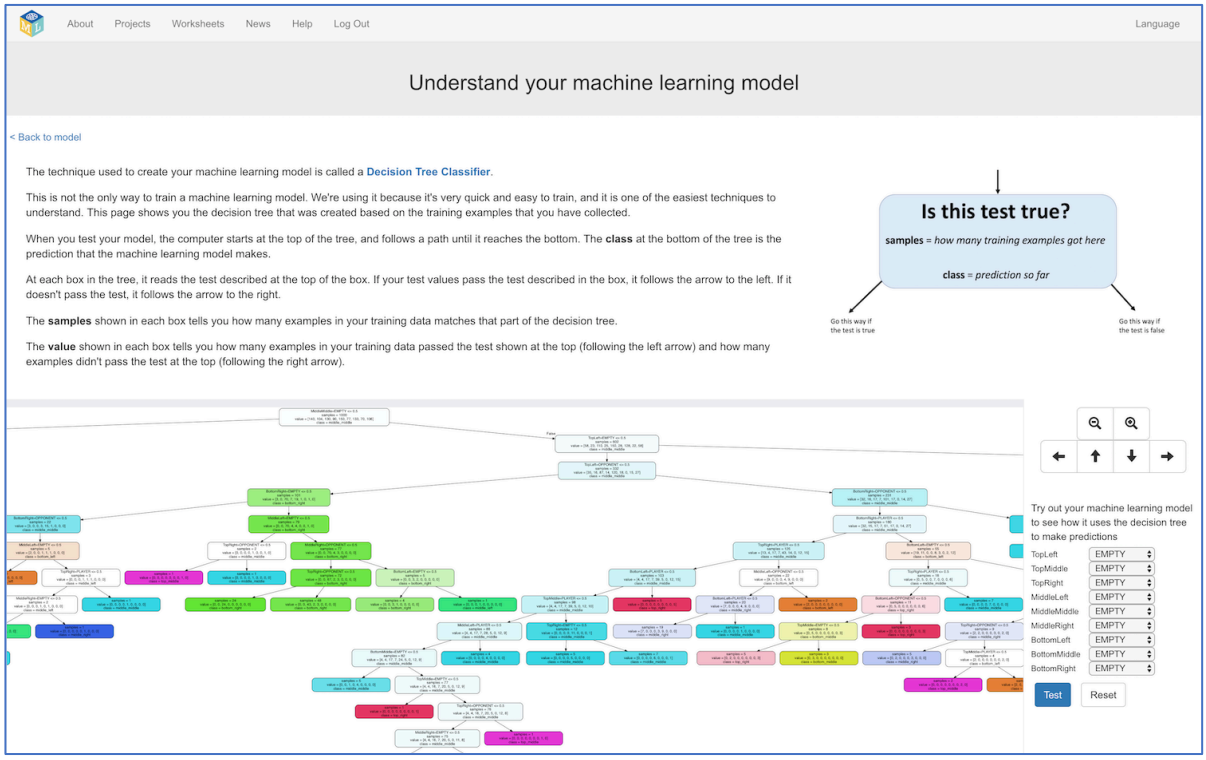
29. Go to the Learn & Test page

*Click the "**< Back to project**" link and then click "**Learn & Test**"*

Check when your code last trained a machine learning model

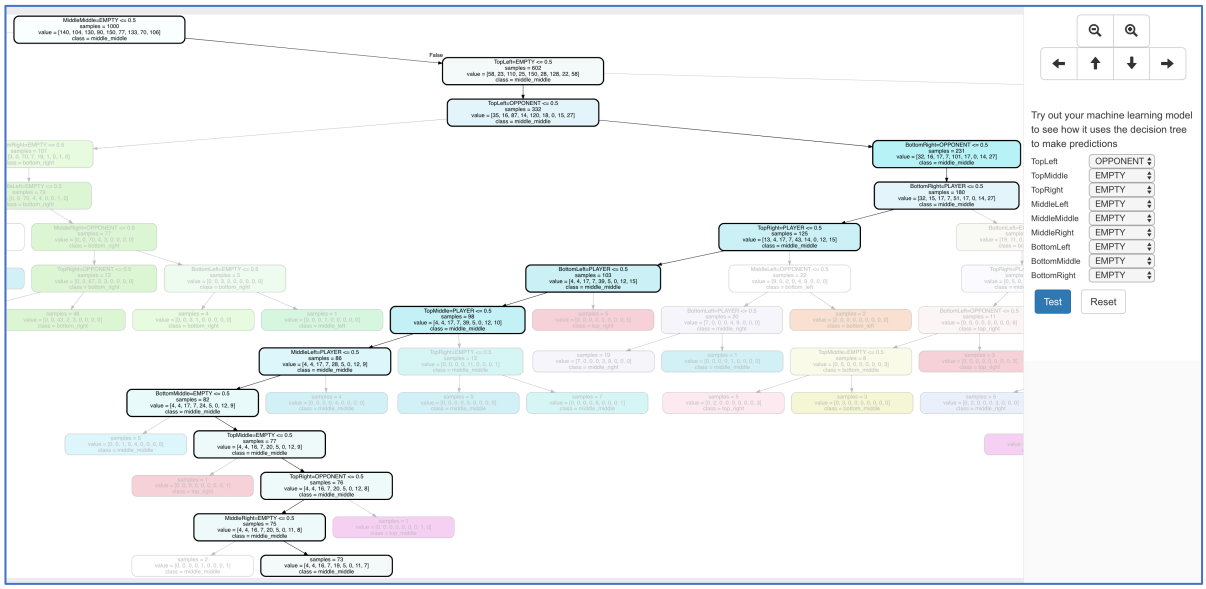


30. Click “**Describe your model**” to see how your model works



31. Describe a state of the board on the right and click “**Test**”

The visualisation describes how your model decides where to move.



32. Keep playing until you start to see the computer get better

How does that change the machine learning model you create?

What have you done?

You've trained a computer to play noughts and crosses.

You didn't have to describe the rules to the computer.
You didn't tell it that it should try to get three noughts in a row.
(The rules are in the Python code so it can be displayed but aren't used in the machine learning model).

You showed it how you play, by collecting examples of decisions that you made when you win. When it makes decisions that leads to it winning, this is added to its training data, so it can be even more confident in that approach in future.

This is called "reinforcement learning" because when it does something good you are "reinforcing" this by rewarding it.

Tips

Don't be kind!

You might be tempted to go easy on the computer when you're playing against it, particularly when it's just starting to learn and is playing very badly.

For example, you might have two crosses-in-a-row next to a blank space and could win. But instead, you might feel sorry for it doing badly and put a cross somewhere else instead to give it a chance.

Don't.

It is learning from the way that you play. If you don't complete a three-in-a-row when you can, you will be teaching it that it should do that.

If you want it to get better quickly, **play as well as you can**.

Mix things up with your examples

Try to come up with lots of different types of examples.

For example, start from a different position on the board on every turn.

Did you know?

People have been learning about machine learning by training a computer to play noughts and crosses for decades!

One famous example was **Donald Michie** – a British artificial intelligence researcher. During World War II, Michie worked at Bletchley Park as a code breaker.

In 1960, he developed "**MENACE**" – the Machine Educable Noughts And Crosses Engine. This was one of the first programs able to learn how to play noughts and crosses perfectly.

As he didn't have a computer he could use, Michie built MENACE using 304 matchboxes and coloured glass beads.

Each matchbox represented a possible state of the board – like the examples that you've been collecting in your training data.

He put beads in the matchboxes to show how often a choice led to a win – the number of beads in the matchbox was like the number of times an example shows up in one of the buckets you created for your training data.