

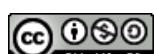
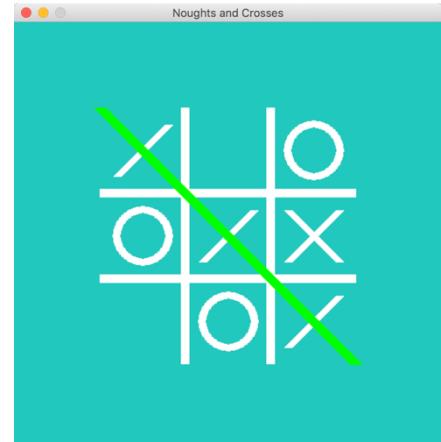
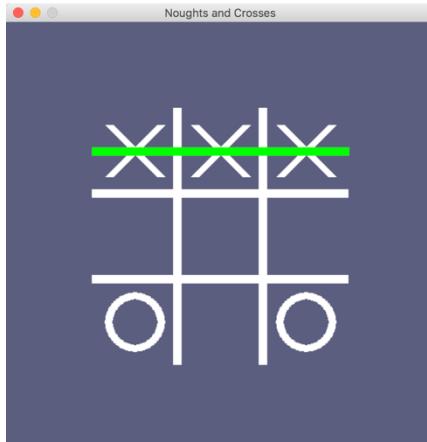
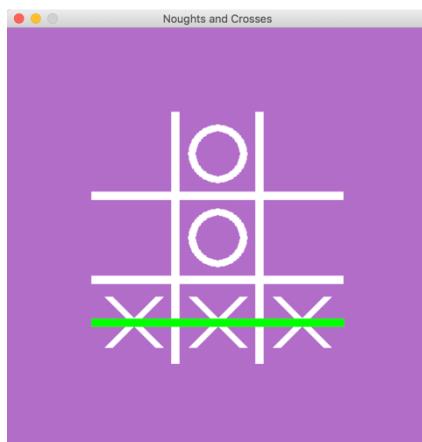
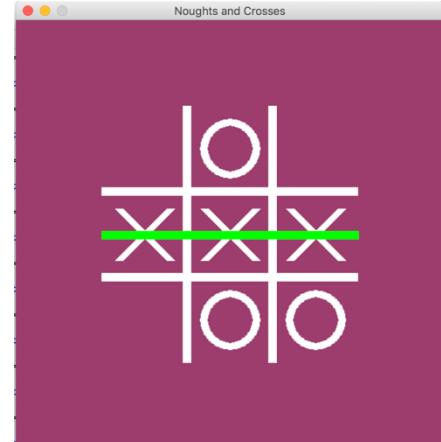
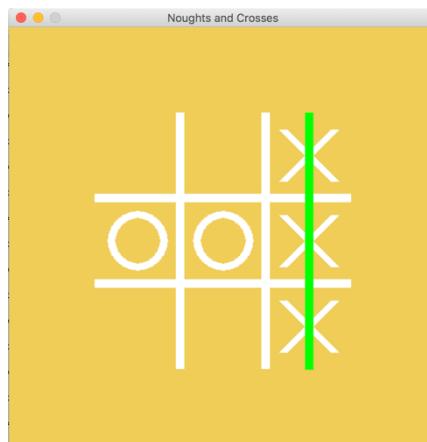
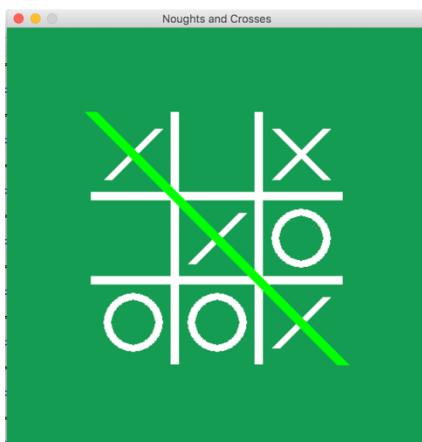


Noughts & Crosses

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

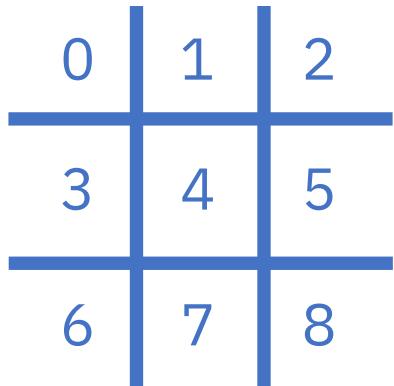
You won't give it instructions for how to play, or tell it what the objective or rules of the game are.

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.



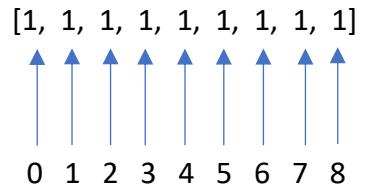
This project worksheet is licensed under a Creative Commons Attribution Non-Commercial Share-Alike License
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

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.

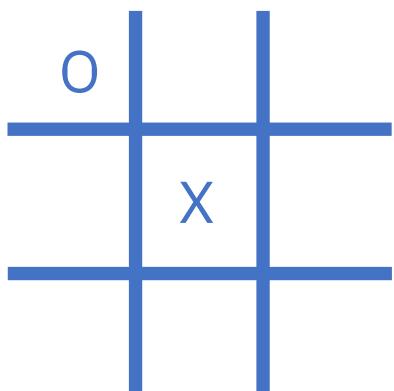


Empty = 1
O = 2
X = 3

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

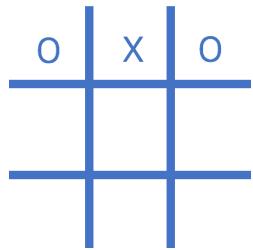
For example, at this point:

[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.

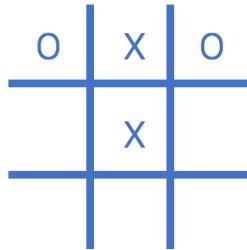


Imagine you decide to put your X in the centre space.

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

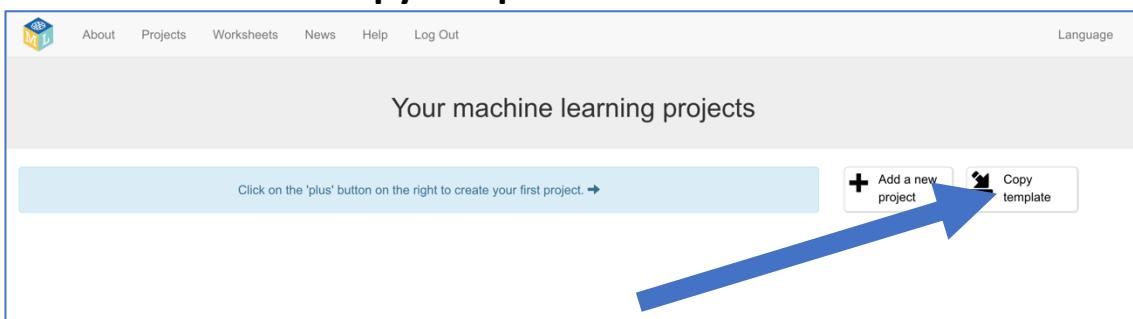
Using “opponent” and “player” instead of “nought” and “cross” means the computer can learn from both nought and cross moves.

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

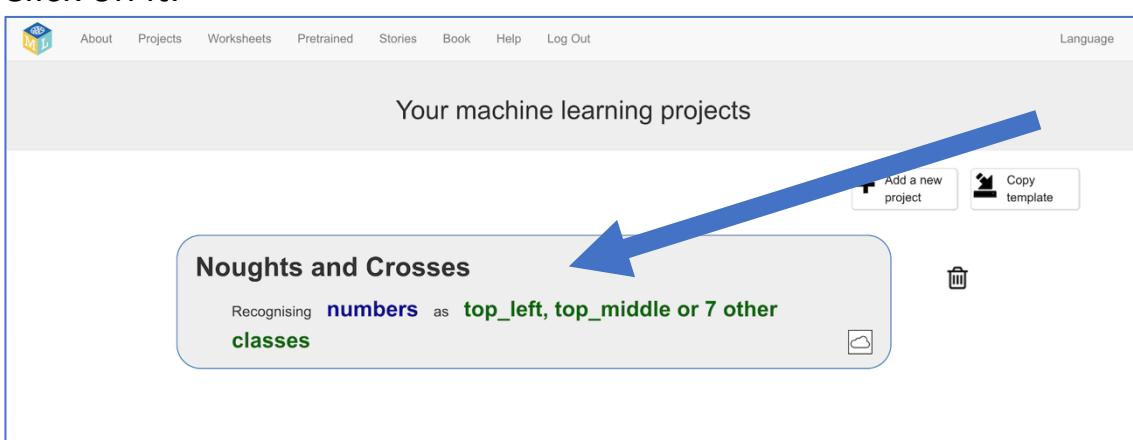
If you (X) win, you'll use your moves as examples to train the computer. If the computer (O) wins, you'll use the computer's moves to train with.

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

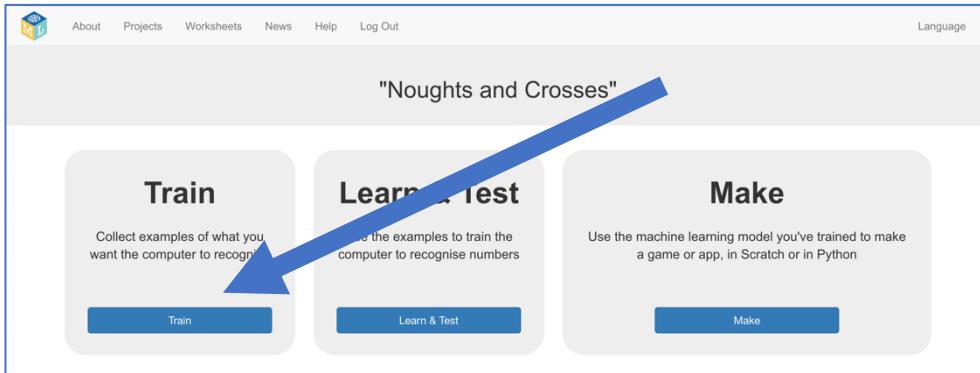
1. Go to <https://machinelearningforkids.co.uk> in a browser.
2. Click on “**Get started**”
3. Click on “**Log In**” and type in your username and password
If you don't have a username, ask your teacher to create one for you.
If you can't remember your password, ask your teacher to reset it for you.
4. Click on “**Projects**” on the top menu bar
5. Click on the “**Copy template**” button.



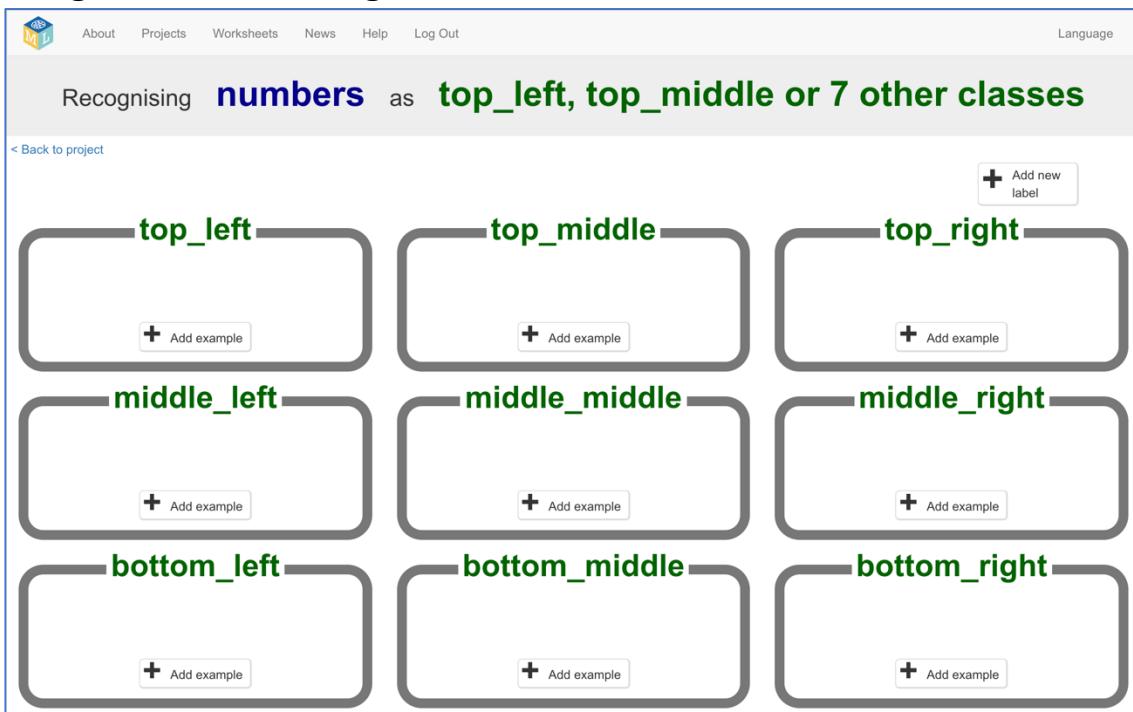
6. Import the “**Noughts and Crosses**” project template.
7. You should see “**Noughts and Crosses**” in your list of projects.
Click on it.



8. Click the “Train” button

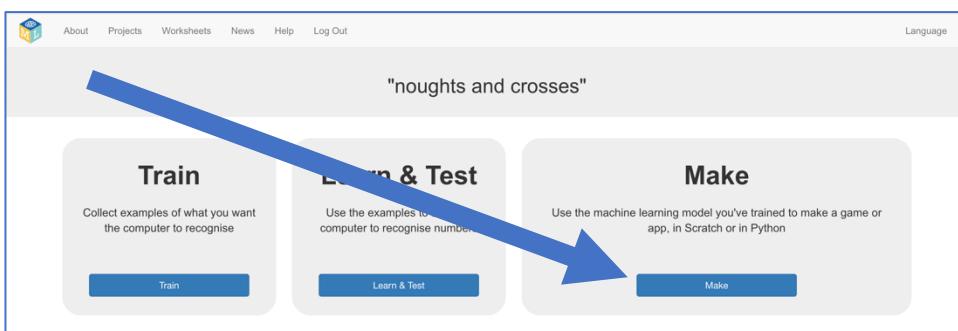


9. The template project has training buckets to store examples of noughts and crosses game moves.



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

11. Click on “Make”



12. Click on “Python”

13. Find your project “key” in the sample code. You’ll need this later.

The screenshot shows a web-based machine learning project interface. At the top, there's a navigation bar with links for About, Projects, Worksheets, Pretrained, Stories, Book, Help, Log Out, and Language. The main title is "Using machine learning in Python". Below the title, there's a link "< Back to project". On the left, there's a sidebar with dropdown menus for "TopLeft", "TopMiddle", "TopRight", "MiddleLeft", "MiddleMiddle", and "MiddleRight", all currently set to "EMPTY". A text box says "You can use this code to add numbers to your training data." and "Enter the numbers below:". To the right, the Python code is displayed:

```
from mlforkidsnumbers import MLforKidsNumbers
project = MLforKidsNumbers(
    key="43487c20-f6c8-11ee-b24a-bfccddc19de725208319-6864-4de4-94d7-114ece46745a"
)
# CHANGE THIS to something you want to
# add to your training examples
trainingvalue = [
    "EMPTY",
    "EMPTY",
    "EMPTY",
    "EMPTY",
    "EMPTY",
    "EMPTY",
]
```

A large blue arrow points from the sidebar dropdowns towards the "key" variable in the Python code.

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

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

The screenshot shows a GitHub repository page for "OwenG88/Noughts-and-Crosses". At the top, there's a "Find File" input field and a green "Clone or download" button with a dropdown arrow. Below that, there's a "Clone with HTTPS" section with a URL "https://github.com/OwenG88/Noughts-and-Crosses" and a copy icon. At the bottom, there are two buttons: "Open in Desktop" and "Download ZIP". Two blue arrows point from the right side of the image towards the "Clone or download" button and the "Download ZIP" button respectively.

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

17. Update the `project` variable using the sample code from Step 13

The screenshot shows a code editor with Python code. The code includes imports for random and MLforKidsNumbers, and defines a project variable:

```
# this module is used to choose a random colour for the user interface and
# make random choices about moves the computer should make
import random
# this module is used to interact with your machine learning project
from mlforkidsnumbers import MLforKidsNumbers
project = MLforKidsNumbers(
    key="43487c20-f6c8-11ee-b24a-bfccddc19de725208319-6864-4de4-94d7-114ece46745a"
)
```

A large blue arrow points from the "key" value in the code back up towards the "key" variable in the code from Step 13.

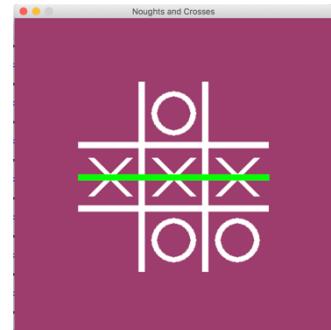
18. Install the requirements listed in `requirements.txt`

Ask your teacher if you need help with 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', 'HUMAN', 'EMPTY', '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', '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.

```
161
162
163     # Someone won the game.
164     # A machine learning model could learn from this...
165
166     # winner      : who won - either HUMAN or COMPUTER
167     # boardhistory : the contents of the game board at each stage in the game
168     # winnerdecisions : each of the decisions that the winner made
169 def learn_from_this(winner, boardhistory, winnerdecisions):
170     print("%s won the game!" % (winner))
171     print("Maybe the computer could learn from %s's experience?" % (winner))
172     for idx in range(len(winnerdecisions)):
173         print("\nAt the start of move %d the board looked like this:" % (idx + 1))
174         print(boardhistory[idx])
175         print("And %s decided to put their mark in %s" % (winner, winnerdecisions[idx]))
```

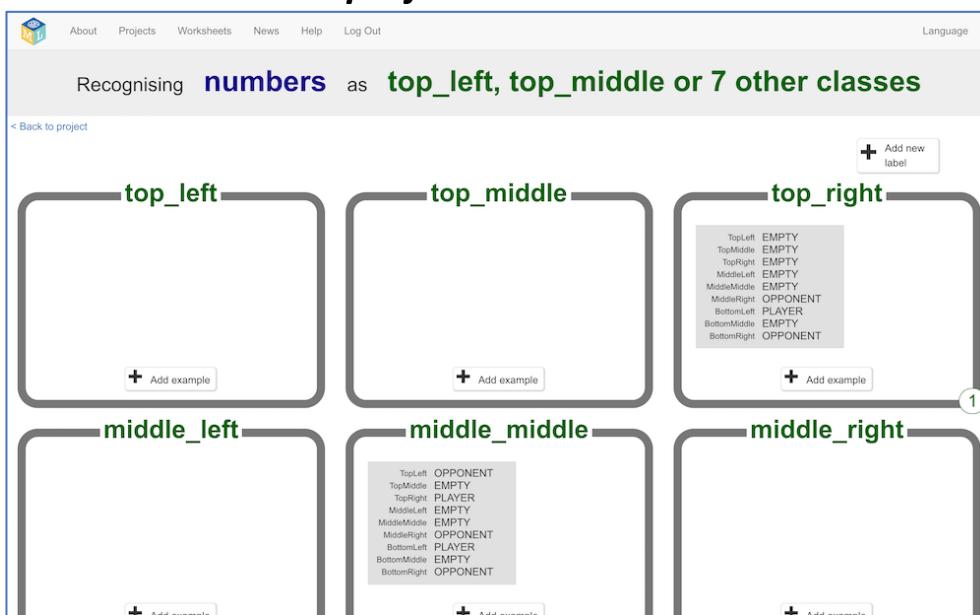
22. Add a line in the `for` loop so each move is added to your project training data

```
164 # A machine learning model could learn from this...
165 #
166 # winner : who won - either HUMAN or COMPUTER
167 # boardhistory : the contents of the game board at each stage in the game
168 # winnerdecisions : each of the decisions that the winner made
169 def learn_from_this(winner, boardhistory, winnerdecisions):
170     print("%s won the game!" % (winner))
171     print("Maybe the computer could learn from %s's experience?" % (winner))
172     for idx in range(len(winnerdecisions)):
173         print("\nAt the start of move %d the board looked like this:" % (idx + 1))
174         print(boardhistory[idx])
175         print("And %s decided to put their mark in %s" % (winner, winnerdecisions[idx]))
176         add_to_train(boardhistory[idx], winner, winnerdecisions[idx])
177
178
```

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. Go to the Learn & Test page

Click the “< Back to project” link and then click “Learn & Test”

You will see that you don’t have enough examples to train a machine learning model yet.

The screenshot shows a web interface for a machine learning project. At the top, there is a navigation bar with links for 'About', 'Projects', 'Worksheets', 'Pretrained', 'Stories', 'Book', 'Help', 'Log Out', and 'Language'. Below the navigation bar, the title 'Machine learning models' is displayed. Underneath the title, there is a link '< Back to project'. The main content area is divided into two sections: 'What have you done?' on the left and 'What's next?' on the right. The 'What have you done?' section contains the following text:

You have collected examples of numbers for a computer to use to recognise when numbers are top_left, top_middle or 7 other classes.

You've collected:

- 0 examples of top_left,
- 0 examples of top_middle,
- 1 example of top_right,
- 0 examples of middle_left,
- 1 example of middle_middle,
- 0 examples of middle_right,
- 1 example of bottom_left,
- 0 examples of bottom_middle,
- 0 examples of bottom_right

The 'What's next?' section contains the following text:

Keep going!

Go back to the [Train](#) page and collect more examples for each of the labels.

The more you can get, the better it should learn, but you need at least five examples of each as an absolute minimum.

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 Learn & Test page and refresh it

Do you have enough examples to train a model yet?

If no, go back to step 27 – play more games

If yes, you can go on to step 29

29. Train a new machine learning model with the example games you've collected so far

Click on the Train new machine learning model button

The screenshot shows the 'Machine learning models' page. On the left, under 'What have you done?', it says: 'You have collected examples of numbers for a computer to use to recognise when numbers are top_left, top_middle or 7 other classes.' Below this is a list of collected items. On the right, under 'What's next?', it says: 'Ready to start the computer's training?' and 'Click the button below to start training a machine learning model using the examples you have collected so far'. At the bottom, there is a box labeled 'Info from training computer:' containing the text 'Train new machine learning model'.

30. Once your model is trained, go back to the **Python** page

Click the “< Back to project” link, then click “Make”, then “Python”

You should see that now you have a model, the sample code has been updated to include the web address of your machine learning model

The screenshot shows the 'Using machine learning in Python' page. On the left, there is a section for entering numbers: 'You can use this code to submit numbers to your machine learning model.' and 'Enter the numbers below:'. It includes four dropdown menus: 'TopLeft : EMPTY', 'TopMiddle : EMPTY', 'TopRight : EMPTY', and 'MiddleLeft : EMPTY'. On the right, there is a code editor window displaying a Python script:

```
from mlforkidnumbers import MLforKidsNumbers
project = MLforKidsNumbers(
    key="43487c20-f6c8-11ee-b24a-bfccddc19de725208319-6864-4de4-94d7-114ece46745a",
    modelurl="https://mlforkids-newnumbers.j8ayd8ayn23.eu-de.codeengine.appdomain.cloud/")
# CHANGE THIS to something you want your
# machine learning model to classify
testvalue = {
    "TopLeft" : "EMPTY",
    "TopMiddle" : "EMPTY",
    "TopRight" : "EMPTY",
    "MiddleLeft" : "EMPTY"
}
```

31. Update the `project` variable in your code to match

```
5 # This module is used to choose a random colour for the user interface and
6 # make random choices about moves the computer should make
7 import random
8 # this module is used to interact with your machine learning project
9 from mlforkidsnumbers import MLforKidsNumbers
10
11
12 project = MLforKidsNumbers(
13     key="43487c20-f6c8-11ee-b24a-bfccddc19de725208319-6864-4de4-94d7-114ece46745a",
14     modelurl="https://mlforkids-newnumbers.j8ayd8ayn23.eu-de.codeengine.appdomain.cloud/saved-models/05ea
15 )
16
17
18
19
20 ######
21 # Constants that match names in your Machine Learning project
```



32. Run the Python program again, and play more games of Noughts and Crosses.

Now that you have a machine learning model, the computer will be using this to decide on where to move, instead of choosing at random

As you play, the winning moves from every game are still being added to your training examples. You are collecting more examples of games that will help your project to learn to be better at playing the game.

33. After playing several more games, go back to the **Learn and Test** page to train a new model

Click the “< Back to project” link, then click “Learn and Test”

34. Click the **Train new machine learning model** button

35. Find the directory where your Python project is, and then delete the `saved-models` folder

Your Python code downloads the machine learning model from the Machine Learning for Kids server. Deleting your local copy will get it to download the new updated model

36. Run the Python program and play again against the updated model *Has the extra training improved the way it plays the game?*

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.

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.

