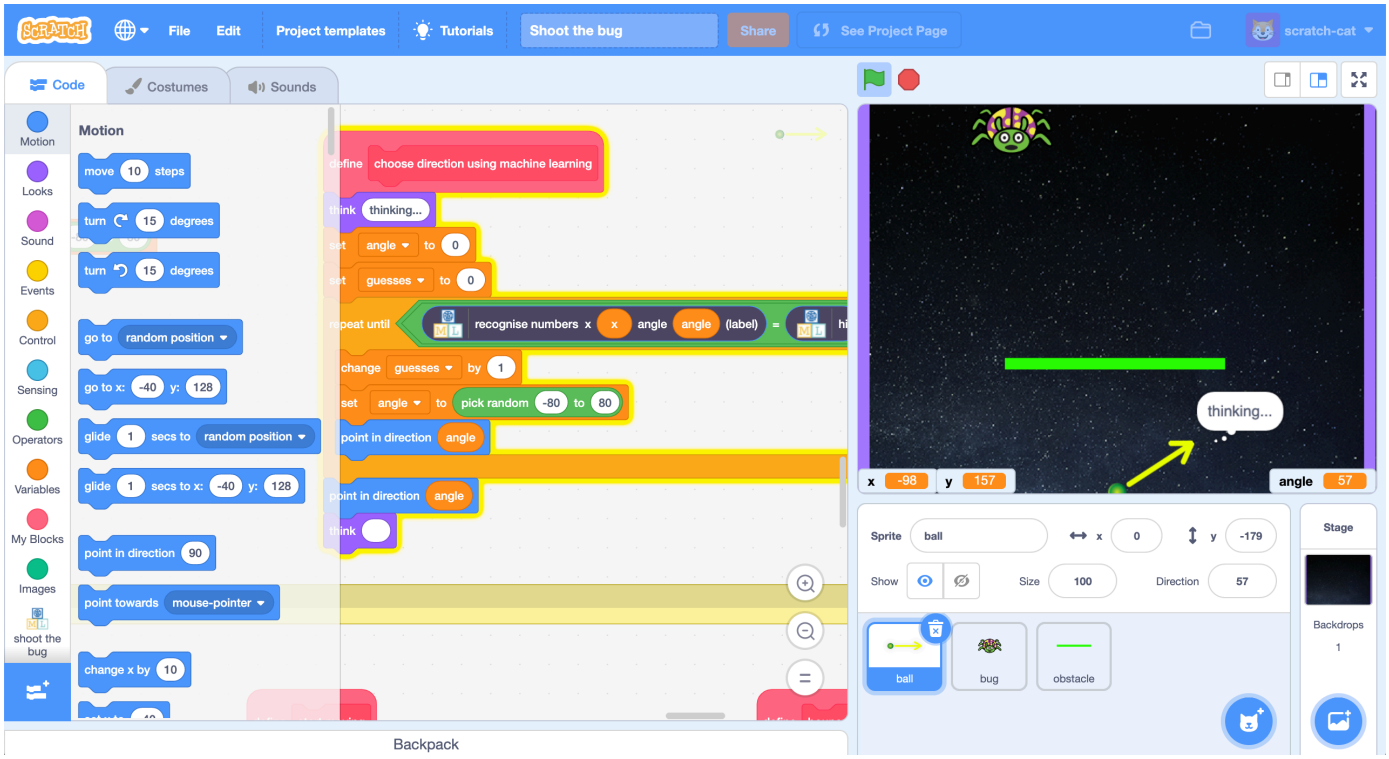
Shoot the bug

In this project you will train a computer to play a simple arcade game.

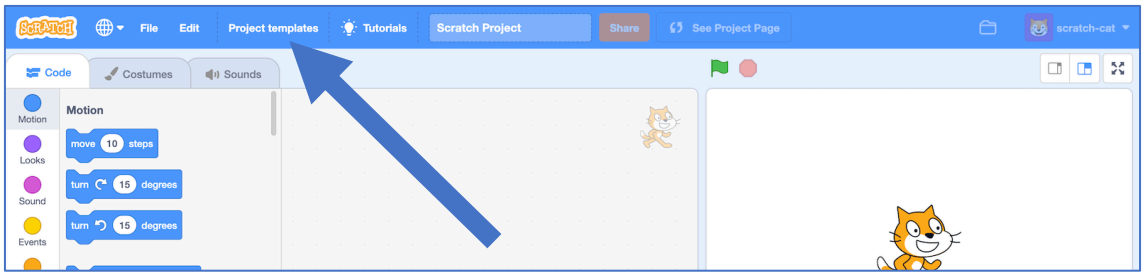
The game is based on shooting balls at a target. You can't aim at the target directly because there is a wall in the way, so you need to bounce the ball off a wall to do it.

You will teach the computer to be able to play this game by collecting examples of shots that hit and miss, so that it can learn to make predictions about the shots it can take.



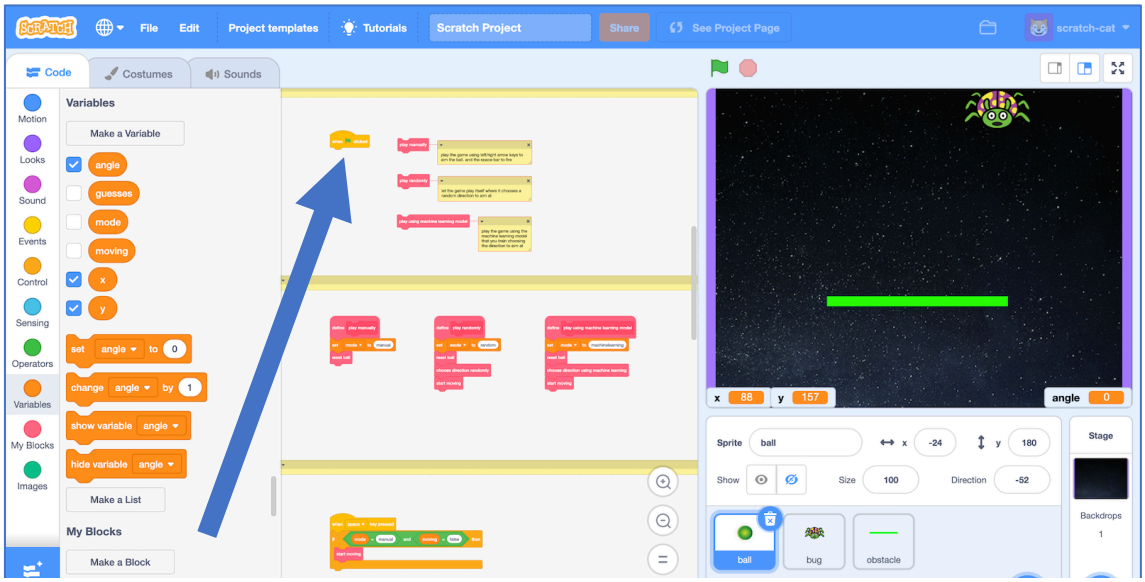
1. Go to https://machinelearningforkids.co.uk/scratch3

2. Click on "**Project templates**"

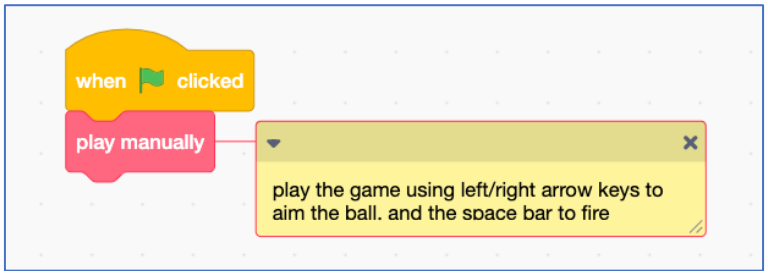


3. Click on the "**Shoot the bug**" template

4. Find the "**when Green Flag clicked**" block on the canvas



5. Attach the "**play manually**" block to the green flag block



6. Click the **Green Flag** and try to shoot the bug!

Use the arrow keys to aim, then press the space bar when you're ready.
Try playing a few times to get used to how the game works.

What have you done so far?

You played a game in Scratch. Each time you play, the bug moves to a random location. The aim of the game is to shoot a ball at the bug.

An obstacle is in the way, so you need to bounce off a side wall to get around the obstacle.

The x,y coordinates of the bug are displayed on the game screen in the bottom-left corner.
The angle you launch the ball at is displayed in the bottom-right corner.

In this project, you are going to get the computer to decide what angle it should shoot at, based on the location of the bug.

You could do this by writing code to calculate the correct angle to launch at, based on the location. (If you have time, give this a try to compare!)

But, for this project, you're going to train the computer so that it learns for itself how to shoot at the bug.

You'll collect examples of the game being played and use that to train a machine learning "model" that can predict if a shot at a certain angle will hit or miss.

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

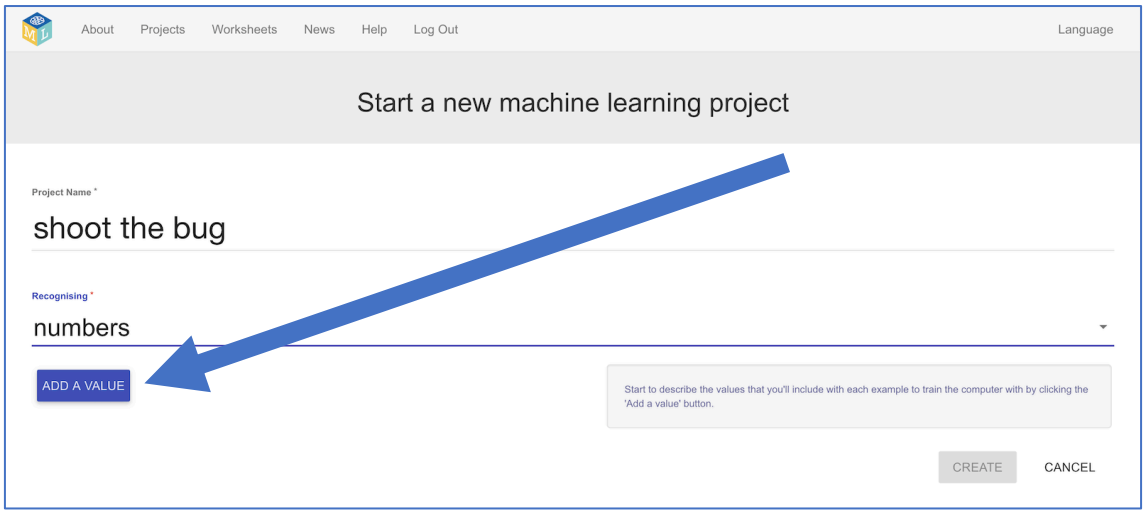
8. Click on "**Get started**"

9. Click on "**Try it now**"

10. Click the **"+ Add a new project**" button.

11. Name your project “shoot the bug” and set it to learn how to recognise “**numbers**”.

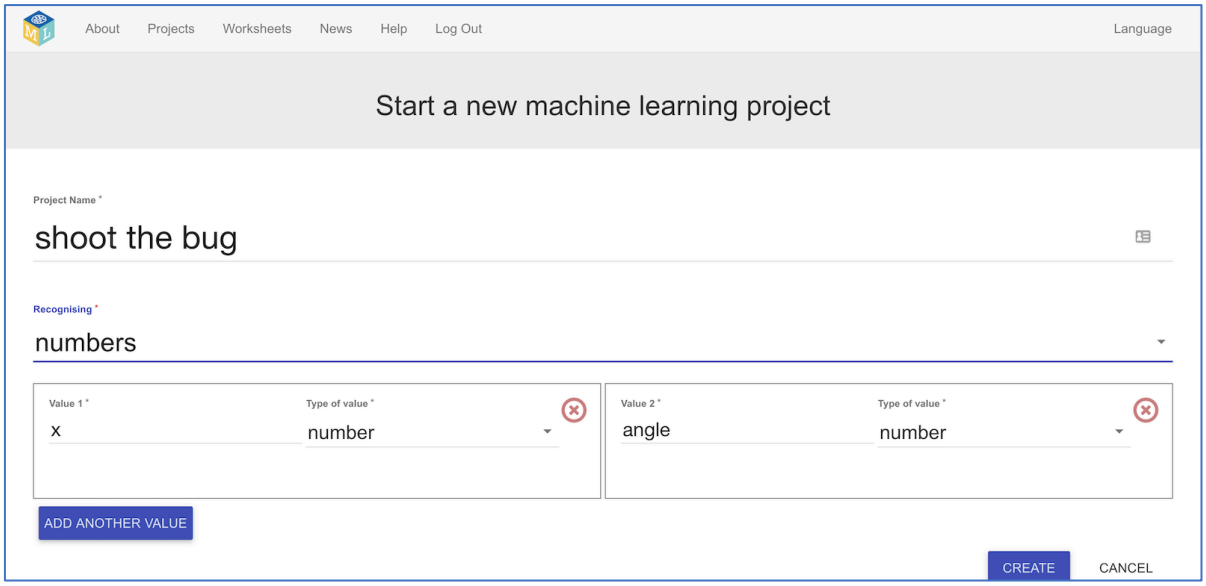
12. Click on “**Add a value**”



13. Create a “**number**” value called “x”, then click “**Add another value**”

14. Create a “**number**” value called “angle”.

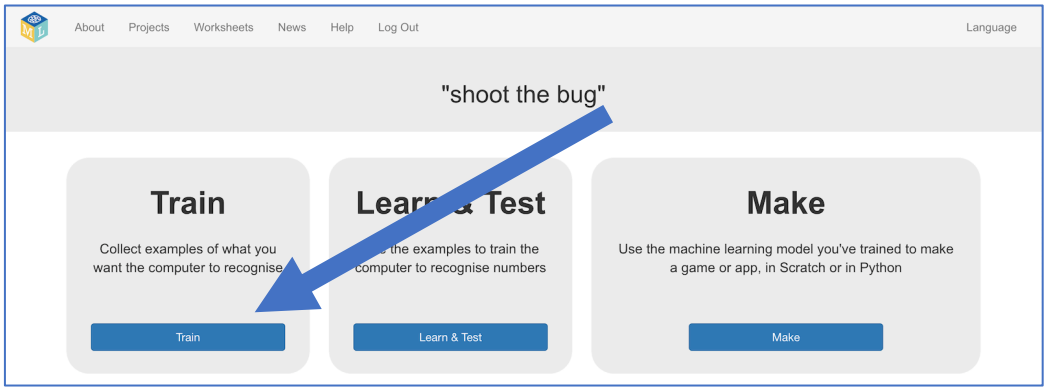
The form should look like this now.



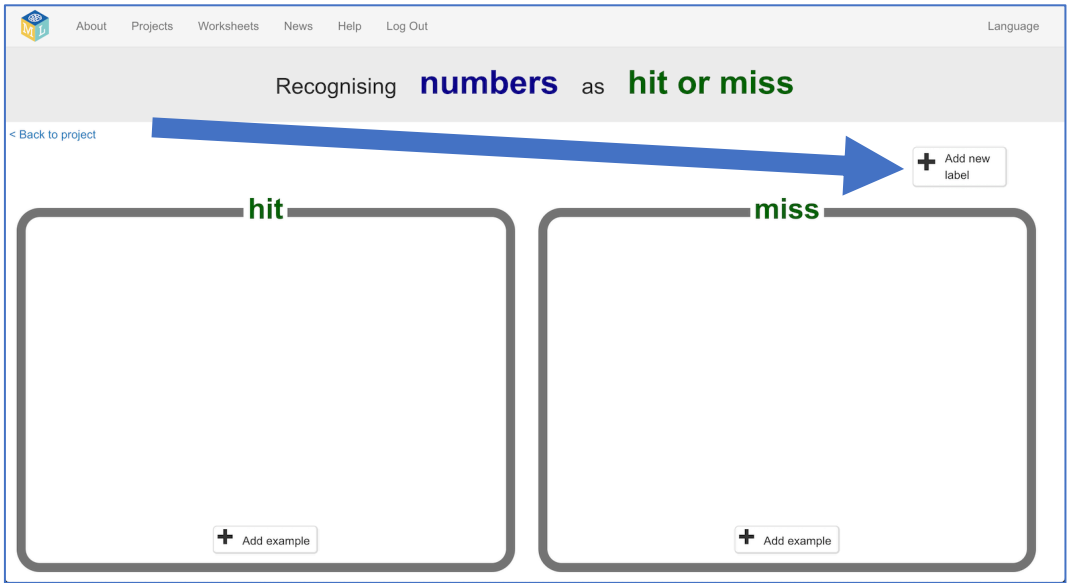
15. Click on the “**Create**” button

16. “shoot the bug” will be added to your list of projects. Click on it.

17. You need to prepare the types of prediction you want the computer to make. Click the "**Train**" button.



18. Click on "**+ Add new label**" and call it "hit".
Do that again, and create a second bucket called "miss".



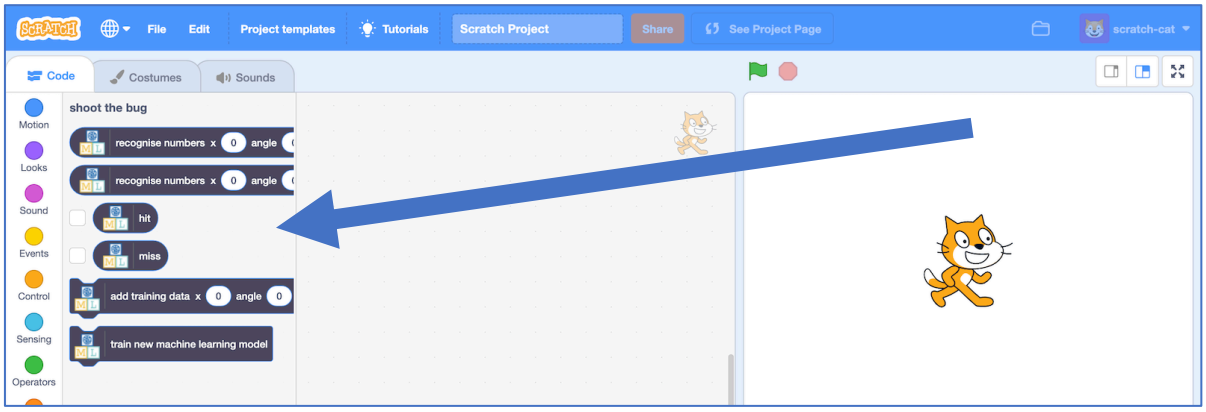
19. Click on the "**< Back to project**" link in the top-left

20. Click on the "**Make**" button

21. Click on the "**Scratch 3**" button

22. Click on the "**straight into Scratch**" button
The page will warn you that you haven't trained a model yet, but that's okay as you'll be using Scratch to collect training examples first.

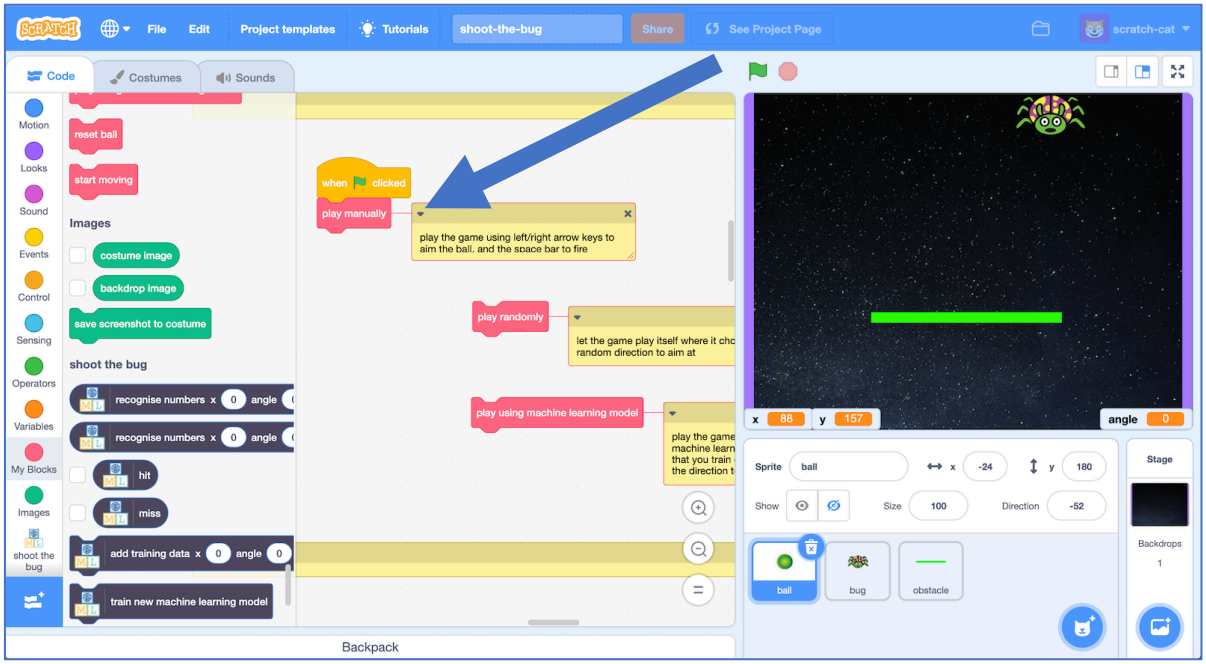
Scratch will be opened, with additional blocks added to the toolbox for your "shoot the bug" project.



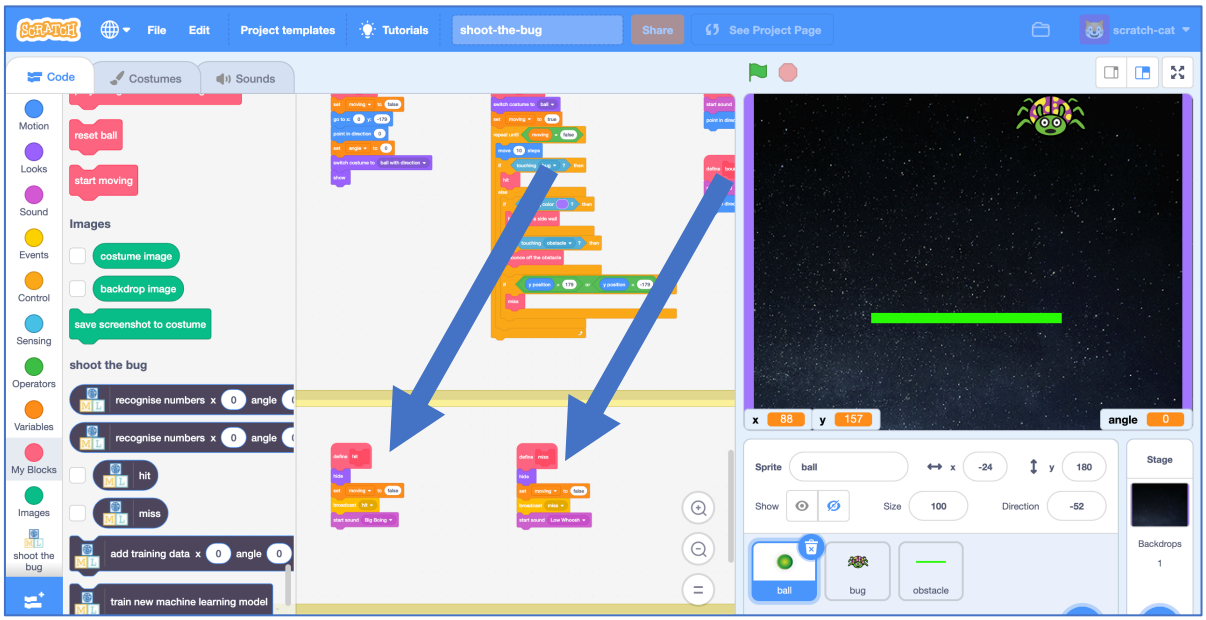
23. Click on the "**Project templates**" button.

24. Open the "shoot the bug" project template again.

25. Connect "play manually" to the "when Green Flag clicked" block again, as you did before.



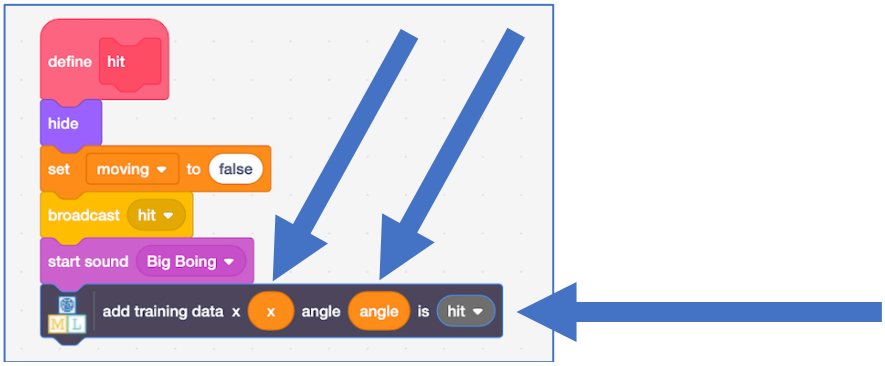
26. Find the scripts for “**hit**” and “**miss**”



27. Add an “**add training data**” block to the “**hit**” script

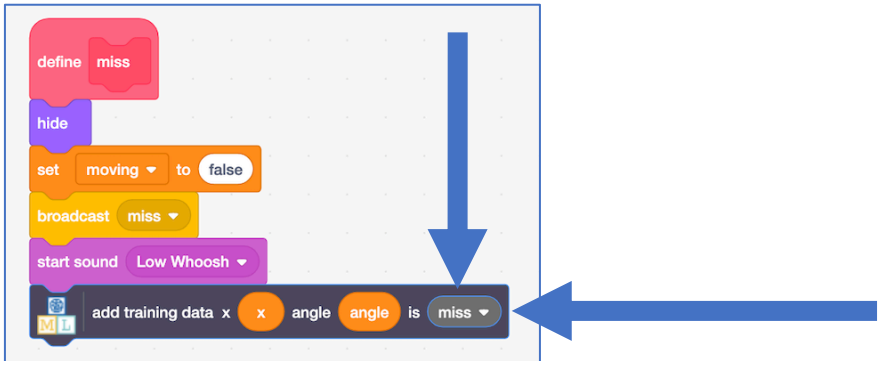
This will a training example to your “hit” bucket every time you make a shot that hits the bug.

*Make sure you add the **x** and **angle** variables*

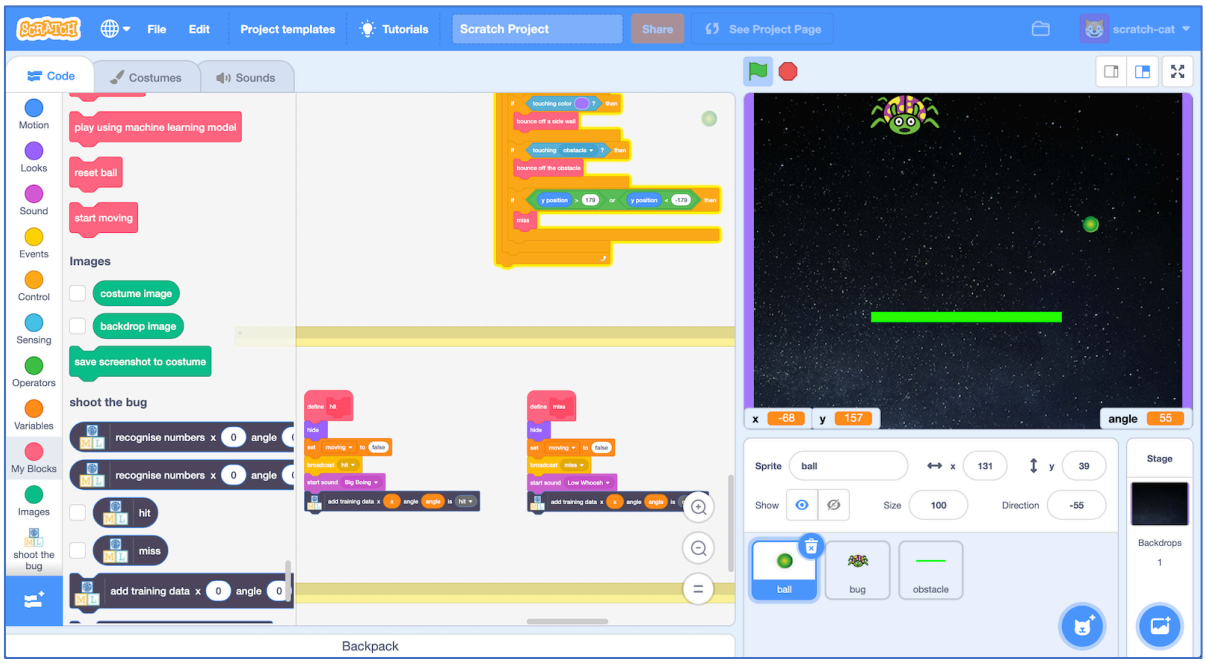


28. Add an “**add training data**” block to the “**miss**” script

Make sure you update the final option to “miss” so it adds examples to your “miss” bucket.



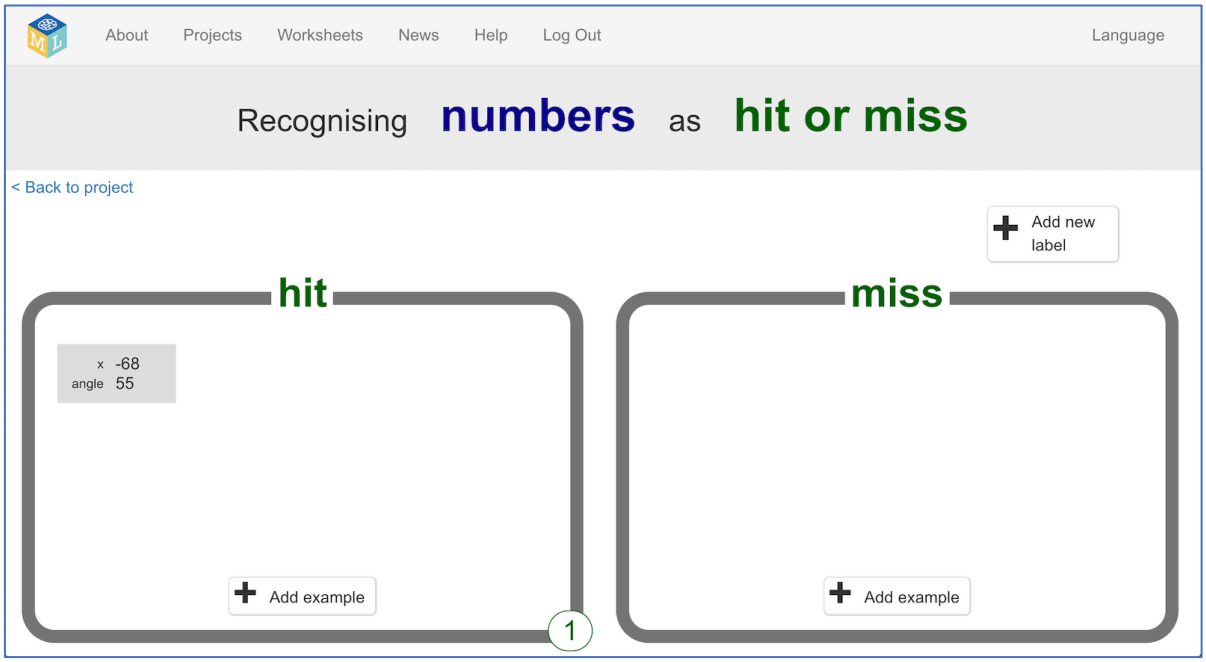
29. Click on the "**Green Flag**" and play the game. Try to hit the bug!



30. In your other web browser window still on the machine learning tool, click on the "**< Back to project**" link in the top-left corner.

31. Click on the "**Train**" button.

32. Check that the shot you just took has been added to the training examples you will use to train the computer.

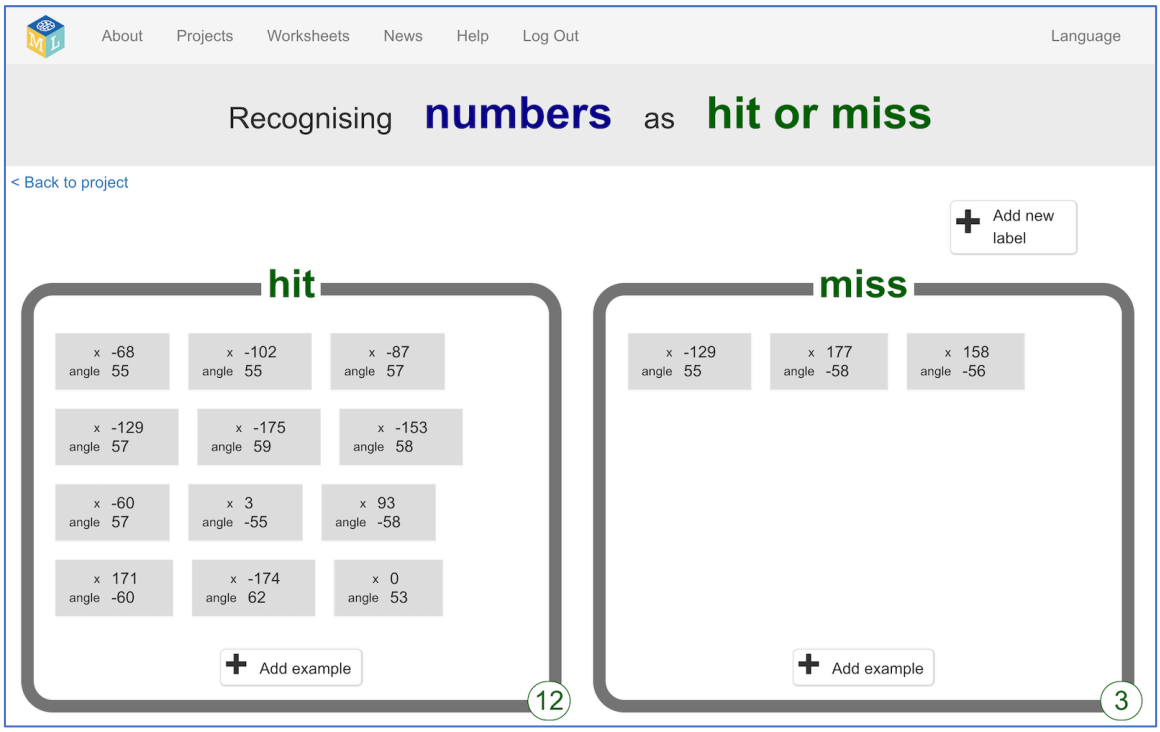


33. Go back to Scratch, and play the game again **fourteen** more times.

You might find it easier to play the game in full-screen mode.

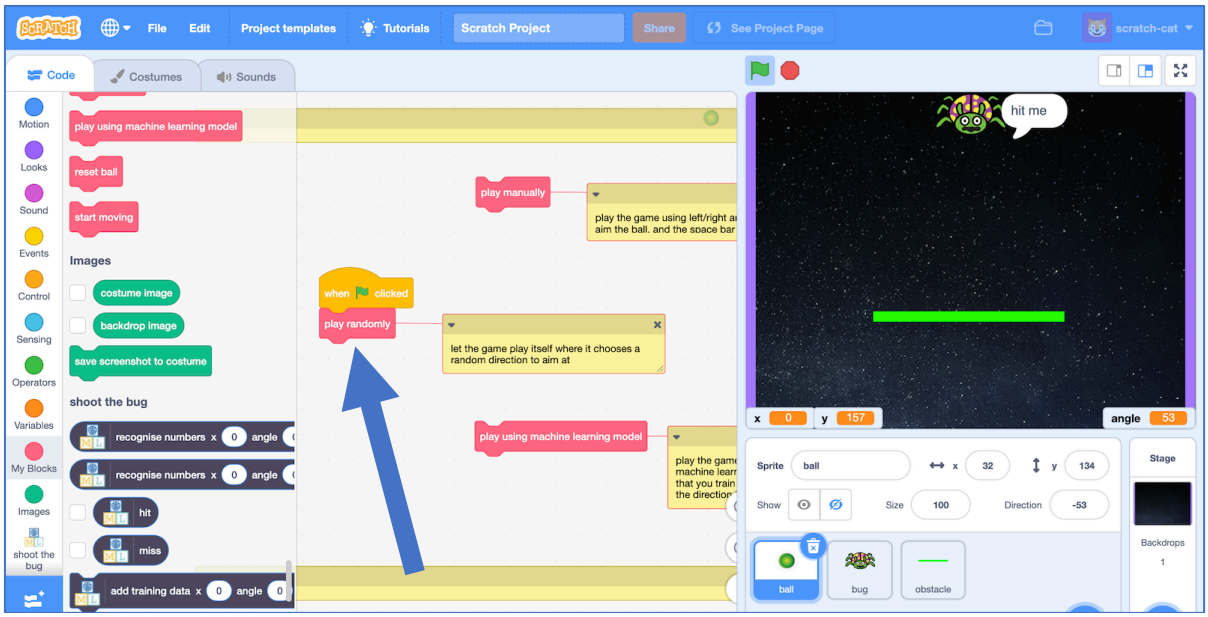
34. Check how many training examples you've collected

Try to hit as many shots as you can, but don't worry if you miss a few!



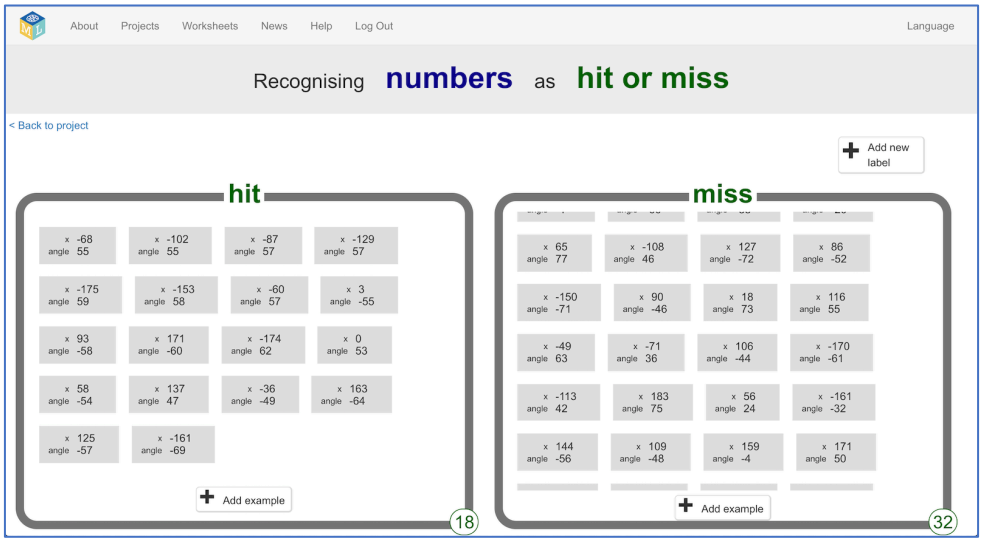
35. Update the "**when Green Flag clicked**" script so that it uses "**play randomly**" (instead of "play manually")

Using random angles for your training examples will get you a better mix of examples, and it will make it quicker and easier to collect them!



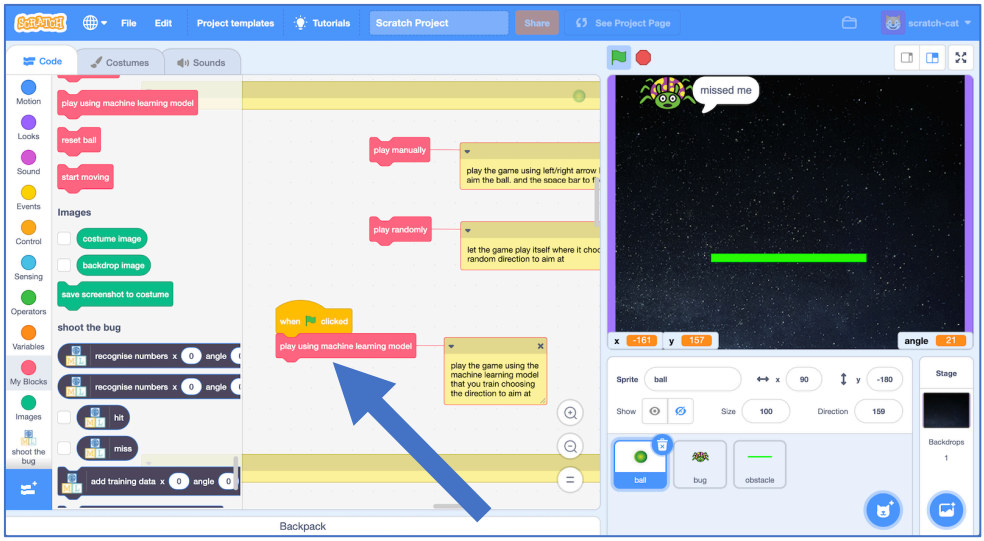
36. Click on the **Green Flag** to collect another example.

*Do this at least **thirty** more times.*

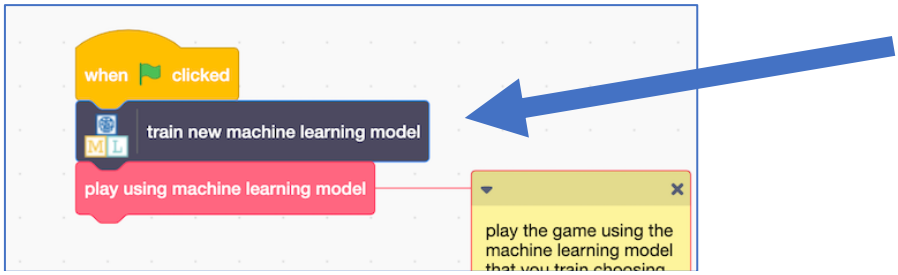


37. Update the "**when Green Flag clicked**" script so that it uses "**play using machine learning model**" (instead of "play randomly")

You should have enough examples now to try using a machine learning model to predict the right angles to fire at.



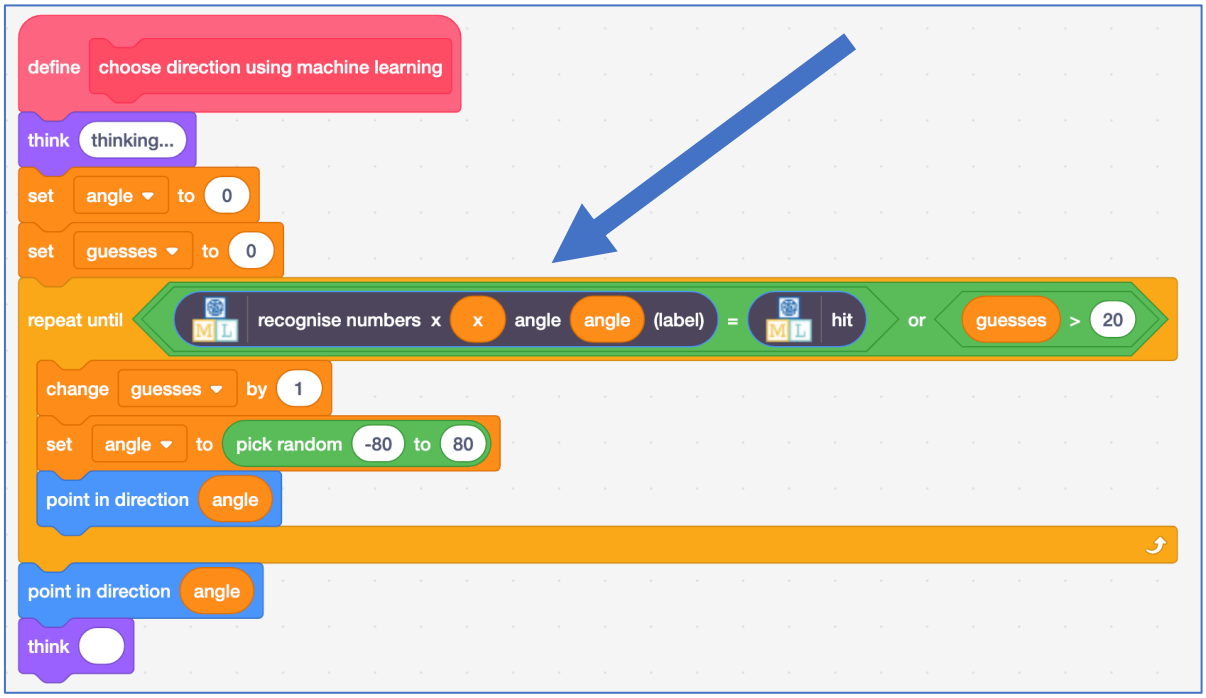
38. Add a "**train new machine learning model**" block to the "**when Green Flag clicked**" script



39. Find the “**choose direction using machine learning**” script.

40. Update the script to use your machine learning model

This will make random choices for angles to fire at, but only use a random choice if your machine learning model thinks it will hit.



41. Click on the **Green Flag** again

How good is your machine learning model at choosing angles that will hit the bug?

What have you done so far?

You've started to train a computer to play a game. Instead of writing rules to be able to do this or working out the equation to calculate the angle to fire the projectile at, you are doing it by collecting examples. These examples are being used to train a machine learning model.

The computer will learn from patterns in the examples. It will use these to make predictions whether a location and angle will result in a hit or miss.

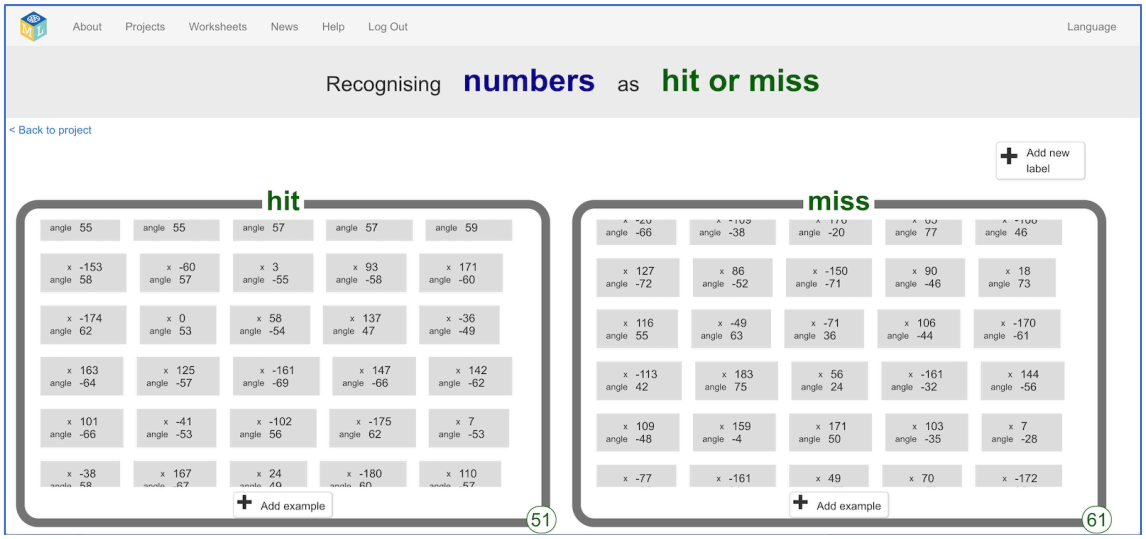
Because you still have the "add training data" blocks in your script, you are still collecting more training examples every time you play. This means the more time you let your machine learning model play the game, the better it should get at playing.

42. How many times is your machine learning model missing?

If it is missing too often, it might be because you haven't given it enough examples of a hit.

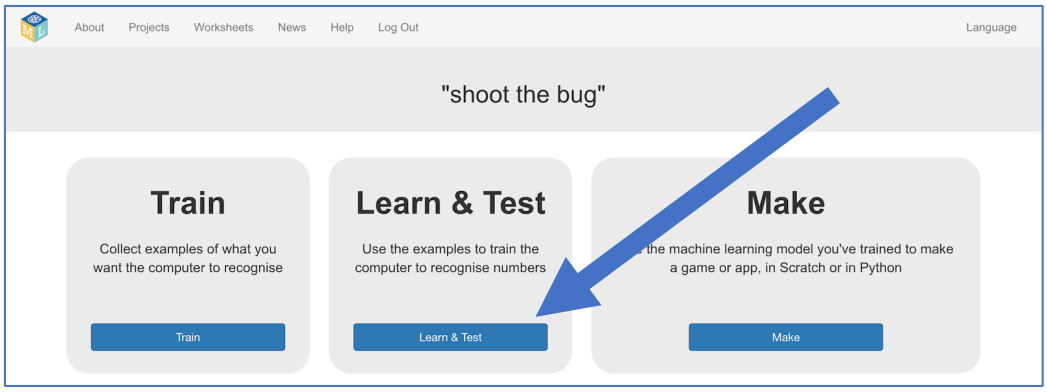
*Try changing the game back to "play manually" mode and use the arrow keys again. Collect another **ten** examples of "hit". Then change back to "play using machine learning model" and see if that helped.*

43. Keep collecting training examples until your machine learning model starts getting good at the game. How many does it take for your model?



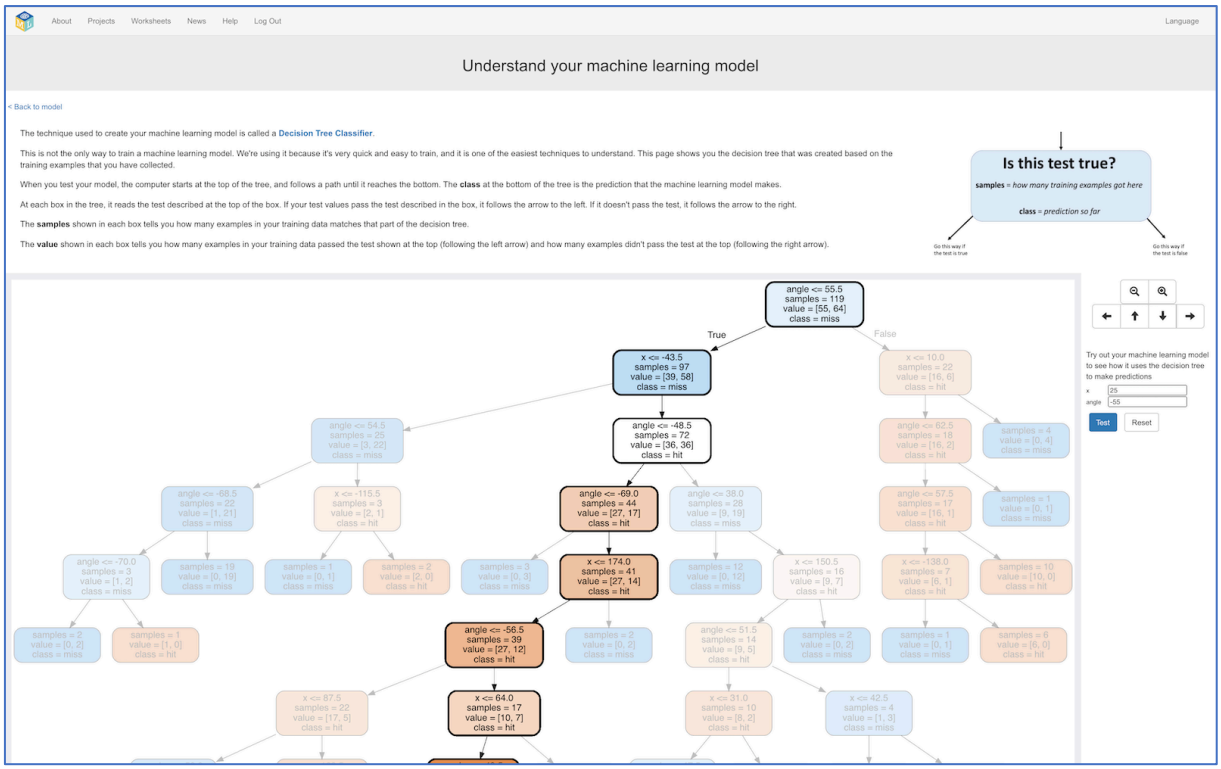
44. Click on the "**< Back to project**" link

45. Click on "**Learn & Test**"



46. Click on "**Describe your model**"

*This page will show you a picture of your machine learning model. Read the page to understand what it means. Try putting in values for the x-coordinate of the bug, and an angle to fire at, then click **Test** to see how your machine learning model makes a prediction about what will happen.*



47. Use this visualisation, and the game in Scratch in "play manually" mode, to see what predictions your machine learning model is making, and whether they are correct.

What have you done?

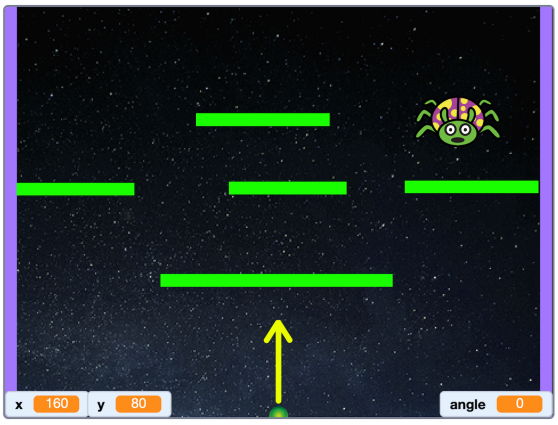
The type of machine learning model you've trained is a "**decision tree classifier**". The visualisation lets you see how your model makes predictions. It's a good way to see what patterns the computer found in the training data you collected.

Is this a good use of machine learning?

We use machine learning when we want computers to do things that are too complicated for us to be able to write the instructions for it to follow.

We avoid machine learning when the time it takes to collect training examples of a task would be longer than just writing the instructions for how to do the task.

Compare the effort to collect the training examples to train the computer to play this game, with the effort it would've taken you to work out the angle to fire at. Do you think this game is a good use of machine learning?



What if the game was made harder? What if there were two obstacles to get around? Or three? Or five? What if the bug could appear at random heights, not just at the top?

These sorts of changes make it a harder task to know what will happen to the ball when you fire.

The equations you would need to calculate the right angle to fire at would be much more complicated.

This makes it a better use of machine learning than playing the game with only one obstacle.

(But it would likely need more training examples for the computer to learn how to play this because it's a more complex task than learning to get around a single obstacle. Try it and see for yourself!)

Ideas and Extensions

Now that you've finished, why not give one of these ideas a try?

Or come up with one of your own?

Add additional obstacles

Try making the game more challenging by adding additional obstacles to the game screen.

You will need to update the "start moving" script so that the ball knows to bounce off your new obstacles.

Use x and y coordinates

To reduce the amount of training needed, we only used one coordinate (x coordinate) and only let the bug move left/right.

Try doing the project again where the bug can move to a random height (y position) as well. You will need to add a new number value to store these y coordinates when you create the machine learning project.

Make it competitive!

Try adding a variable to keep score and see if your machine learning model can get a higher score than you can.