



Shoot the bug

In diesem Projekt wirst du einen Computer trainieren, ein einfaches Arcade-Spiel zu spielen.

Das Spiel basiert darauf, Bälle auf ein Ziel zu schießen. Du kannst das Ziel nicht direkt treffen, weil eine Wand im Weg ist. Also musst du den Ball an einer Wand abprallen lassen, um das Ziel zu treffen.

Du bringst dem Computer bei, dieses Spiel automatisch zu spielen. Dafür sammelst du Beispiele für Treffer und Fehlschüsse, so dass er lernen kann, Vorhersagen über die möglichen Schüsse zu treffen.

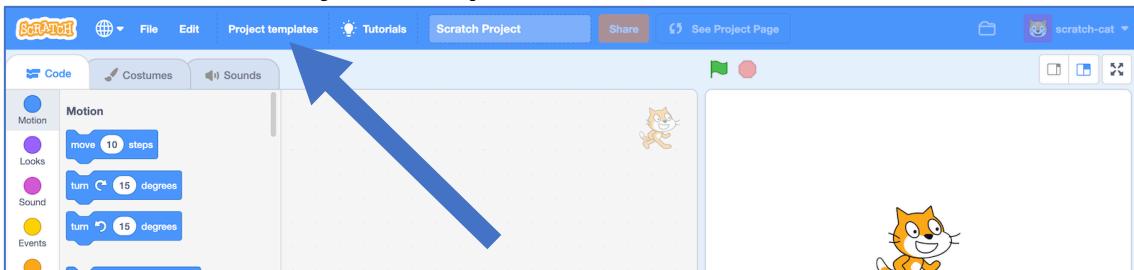
The image shows the Scratch programming environment. On the left, the script palette is open, displaying the script for the 'choose direction using machine learning' define block. This script uses a 'repeat until' loop to continuously check for hits ('recognise numbers x angle angle (label) = hit') and update a 'guesses' counter ('change guesses by 1'). It also sets a random angle ('set angle to pick random -80 to 80') and points the sprite in that direction ('point in direction angle'). A 'think' block is used to show the sprite thinking while it waits for hits. On the right, the stage view shows a green ball sprite moving towards a purple bug sprite, which is positioned behind a grey rectangular obstacle. The ball has a speech bubble saying 'thinking...'. The stage properties panel at the bottom right shows the ball's sprite settings (x: 0, y: 179, angle: 57).



This project worksheet is licensed under a Creative Commons Attribution Non-Commercial Share-Alike License
<http://creativecommons.org/licenses/by-nc-sa/4.0/>
Teile ins Deutsche übersetzt von Steffi Rudel mit Hilfe von deepl.com (Februar 2021)

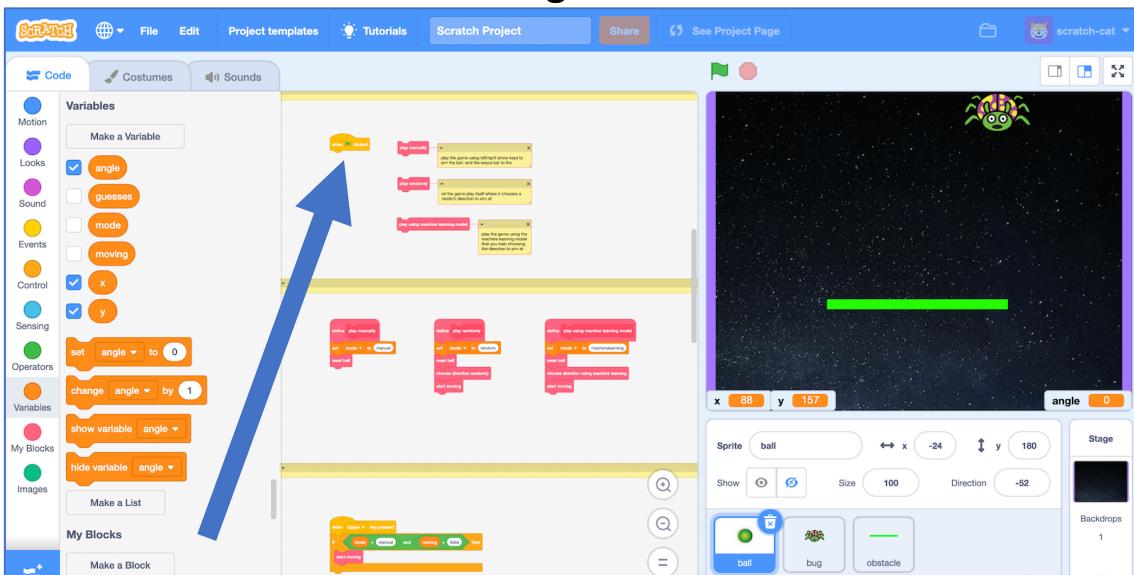
1. Go to <https://machinelearningforkids.co.uk/scratch3>

2. Click on “Project templates”

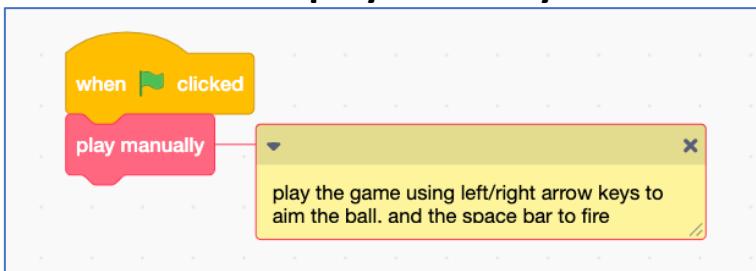


3. Click on the “Shoot the bug” template

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.*

Was hast du bis hierher gemacht?

Du hast ein Spiel in Scratch gespielt. Jedes Mal, wenn du spielst, bewegt sich der Käfer (=Bug) an eine zufällige Stelle. Das Ziel des Spiels ist es, einen Ball auf den Käfer zu schießen.

Ein Hindernis ist im Weg, also muss der Ball an einer Seitenwand abprallen, um das Hindernis zu umgehen.

Die **x,y-Koordinaten des Käfers** werden auf dem Spielbildschirm in der **unteren linken Ecke** angezeigt.

Der **Winkel (=angle)**, in dem der Ball geschossen wird, wird in der **rechten unteren Ecke** angezeigt.

In diesem Projekt wirst du den Computer dazu bringen, zu entscheiden, in welchem Winkel er (basierend auf der Position des Käfers) schießen soll.

Du könntest dazu einen Code schreiben, der den korrekten Abschusswinkel auf der Grundlage der Position berechnet. (Wenn du Zeit hast, probiere das zum Vergleich gerne aus!)

Aber für dieses Projekt wirst du den Computer so trainieren, dass er selbst lernt wie er auf den Käfer schießen soll.

Du wirst Beispiele des Spiels sammeln und damit ein "Modell" für maschinelles Lernen trainieren, das vorhersagen kann, ob ein Schuss in einem bestimmten Winkel trifft oder nicht.

- 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**”

The screenshot shows a 'Start a new machine learning project' dialog. The 'Project Name' field contains 'shoot the bug'. The 'Recognising' dropdown is set to 'numbers'. A blue arrow points to the 'ADD A VALUE' button. A tooltip box says: 'Start to describe the values that you'll include with each example to train the computer with by clicking the 'Add a value' button.' At the bottom right are 'CREATE' and 'CANCEL' buttons.

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

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

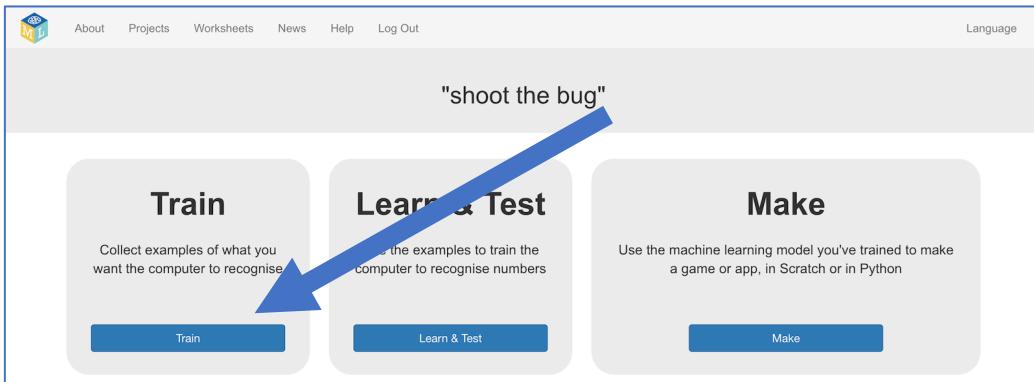
The form should look like this now.

The screenshot shows the 'Add another value' dialog with two entries: 'Value 1' is 'x' and 'Type of value' is 'number'; 'Value 2' is 'angle' and 'Type of value' is 'number'. Both entries have a red 'X' icon to their right. At the bottom left is an 'ADD ANOTHER VALUE' button, and at the bottom right are 'CREATE' and 'CANCEL' buttons.

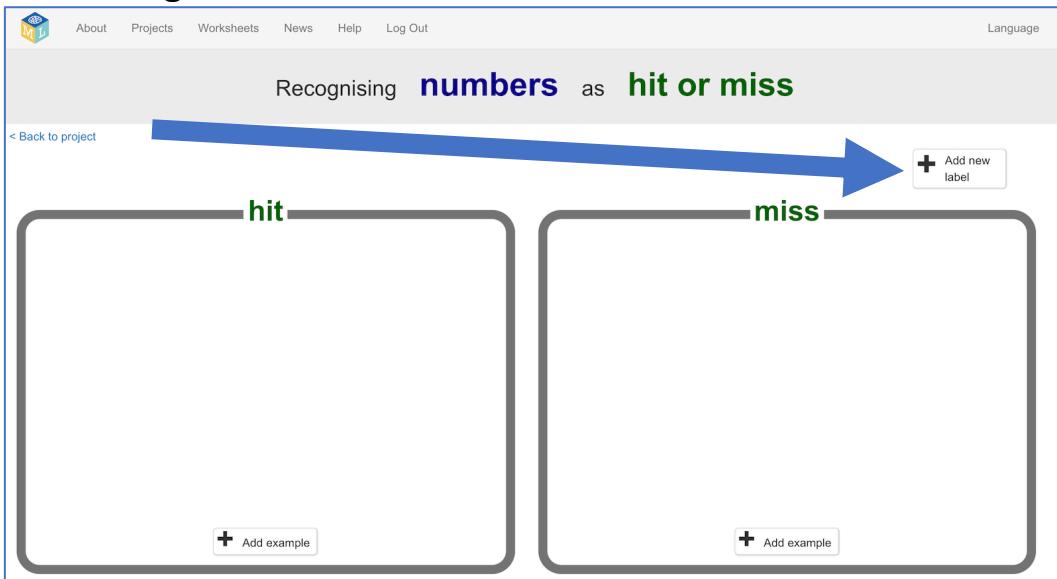
15. Click on the “**Create**” button

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.

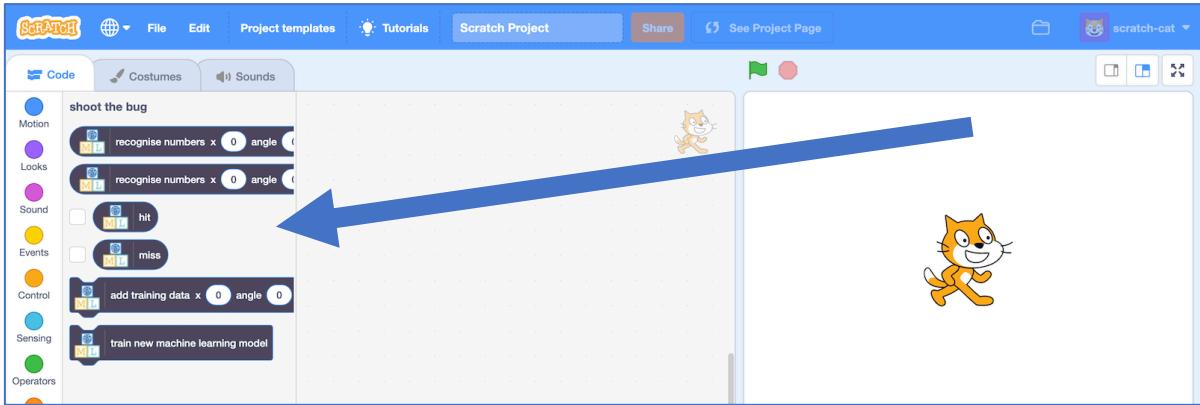


- 18.** Click on “+ Add new label” and call it “hit”.
Do that again, and create a second bucket called “miss”.

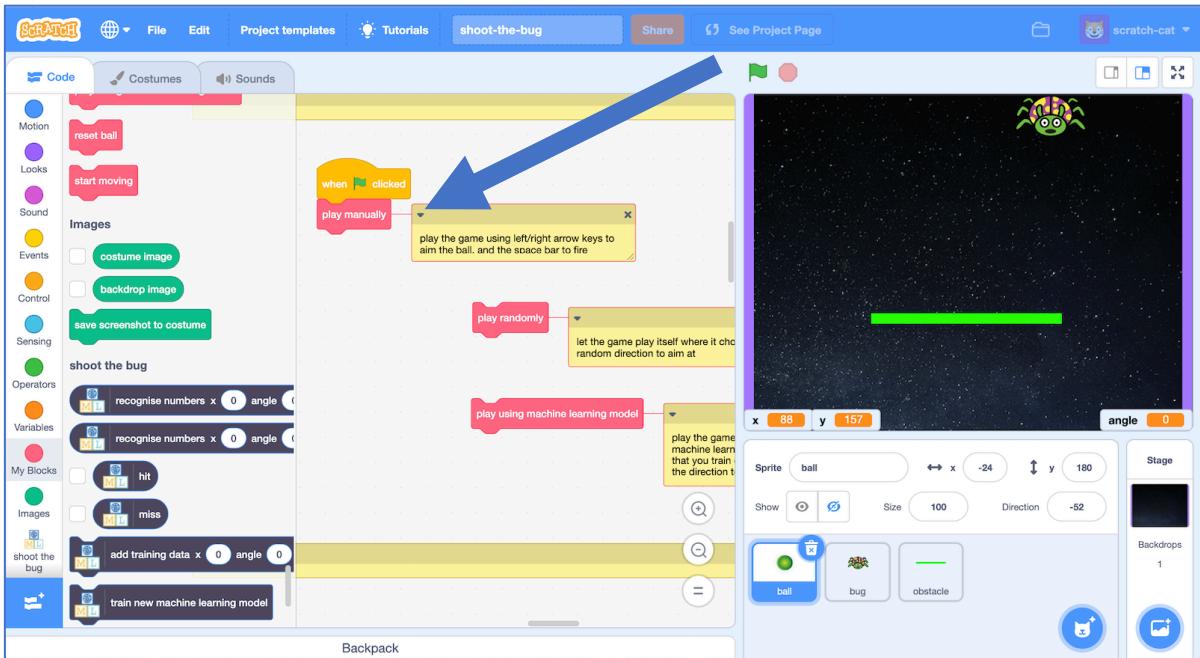


- 19.** Click on the “< Back to project” link in the top-left
- 20.** Click on the “Make” button
- 21.** Click on the “Scratch 3” button
- 22.** Click on the “straight into Scratch” button
The page will warn you that you haven’t trained a model yet, but that’s okay as you’ll be using Scratch to collect training examples first.

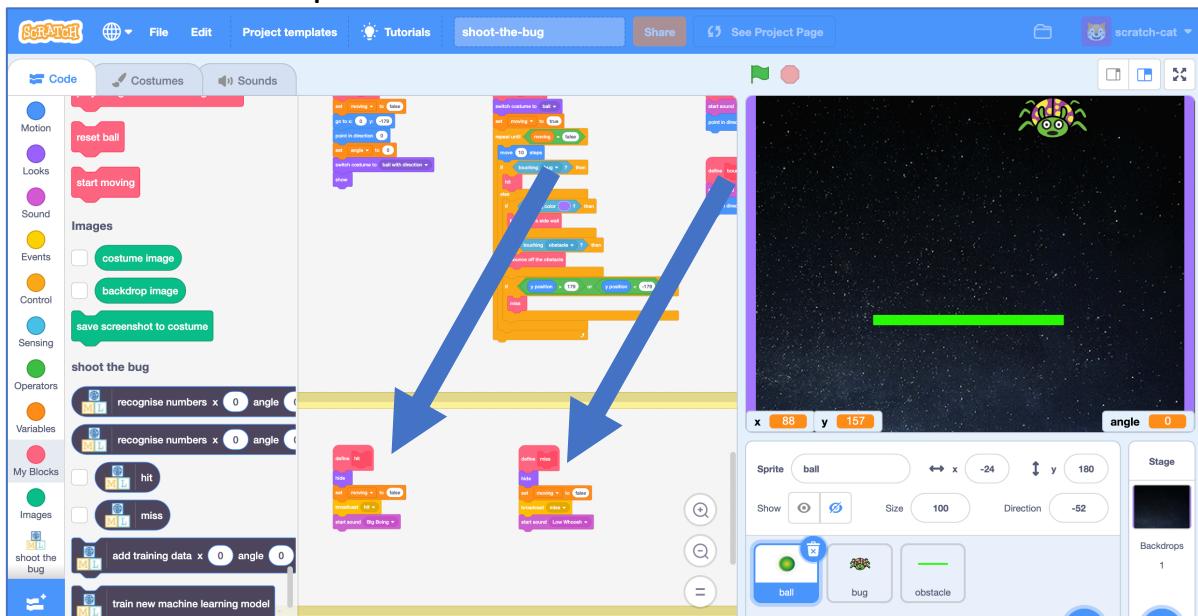
Scratch will be opened, with additional blocks added to the toolbox for your “shoot the bug” project.



- 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.



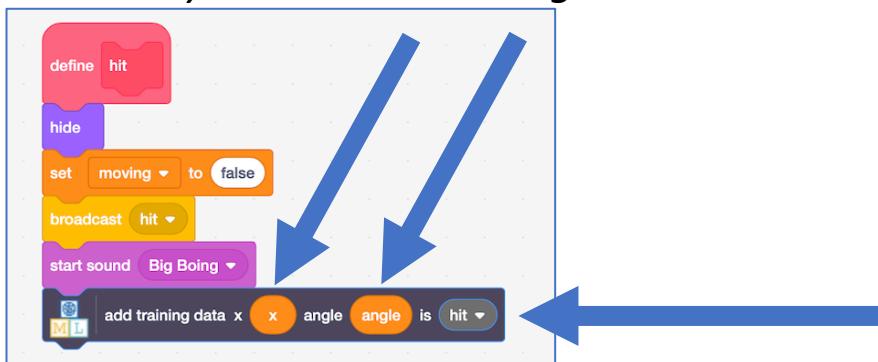
26. Find the scripts for “hit” and “miss”



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

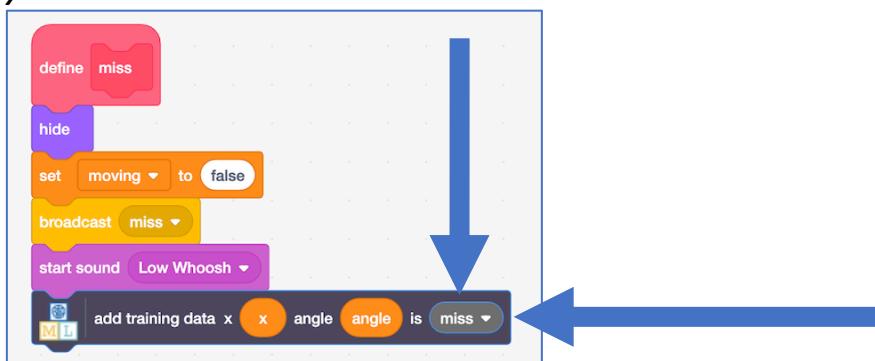
This will add a training example to your “hit” bucket every time you make a shot that hits the bug.

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

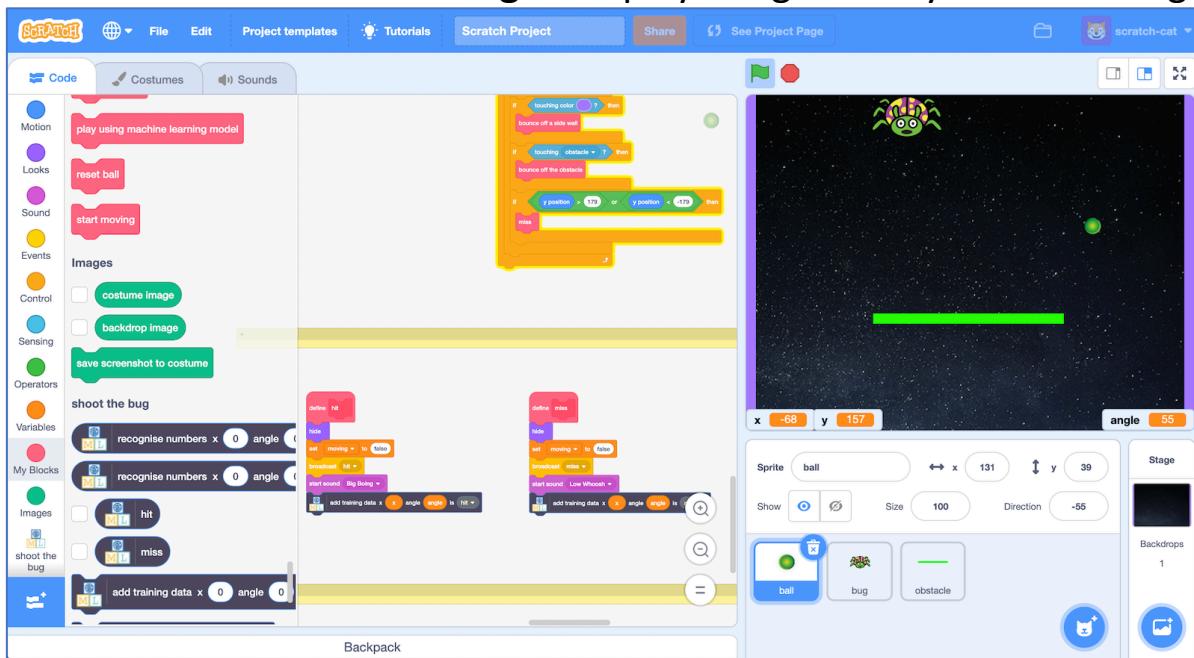


28. Add an “add training data” block to the “miss” script

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



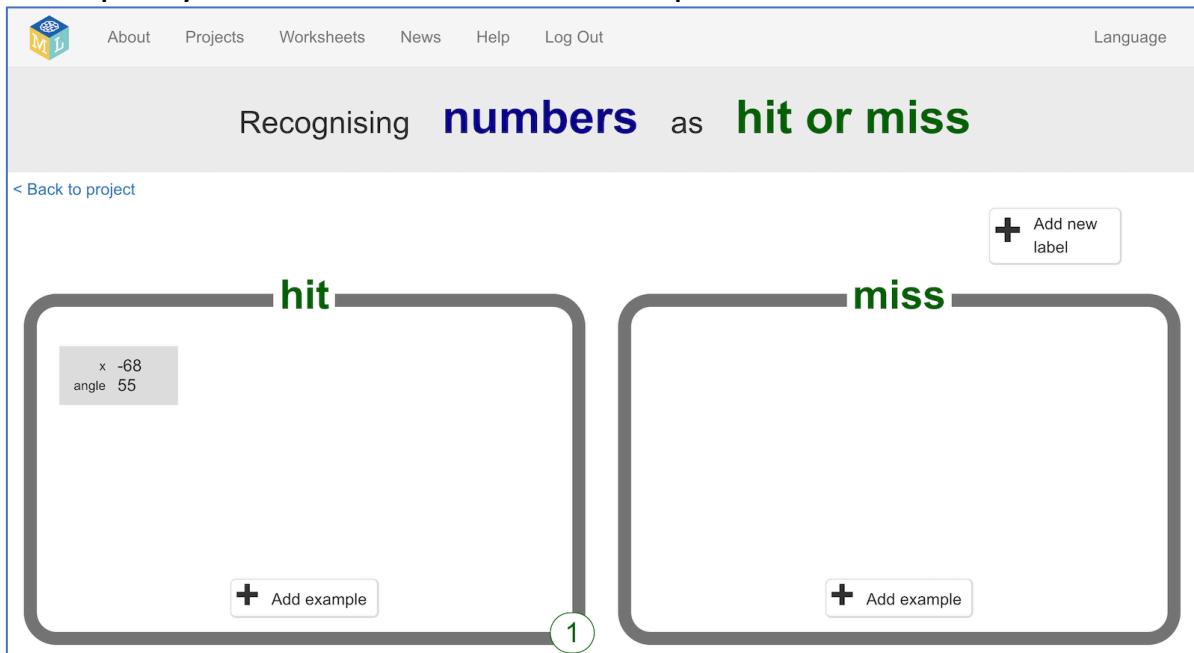
29. Click on the “**Green Flag**” and play the game. Try to hit the bug!



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!

The screenshot shows a web-based application for training a machine learning model. The title is "Recognising numbers as hit or miss". There are two main sections: "hit" and "miss".

hit section: Contains 12 training examples. Each example is represented by a box with coordinates (x, angle). The examples are:

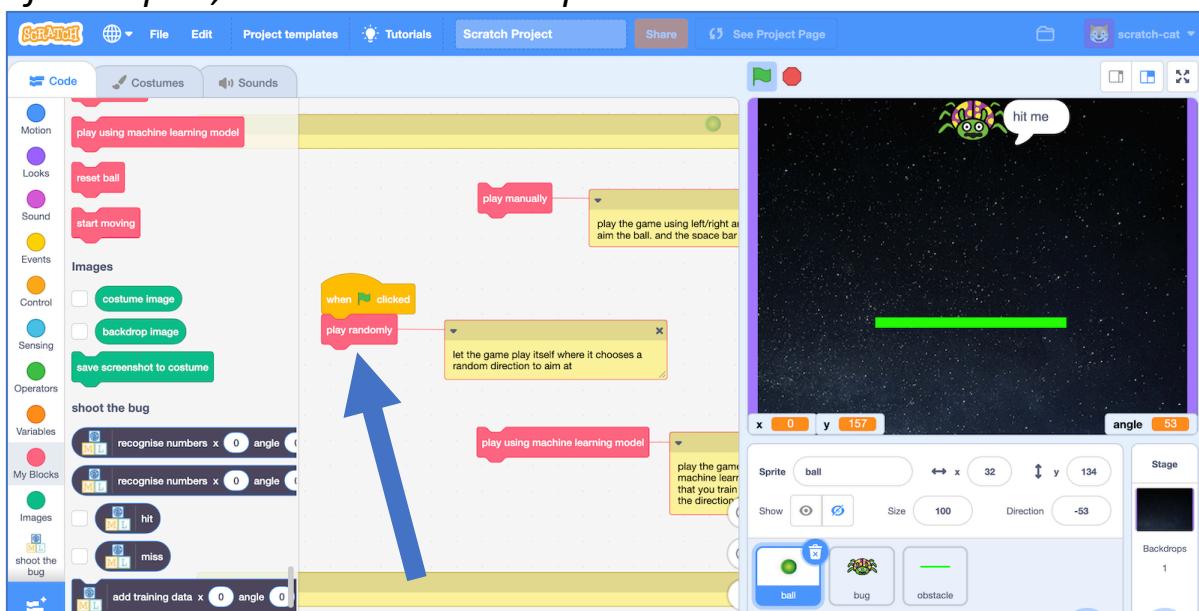
- x -68 angle 55
- x -102 angle 55
- x -87 angle 57
- x -129 angle 57
- x -175 angle 59
- x -153 angle 58
- x -60 angle 57
- x 3 angle -55
- x 93 angle -58
- x 171 angle -60
- x -174 angle 62
- x 0 angle 53

miss section: Contains 3 training examples. Each example is represented by a box with coordinates (x, angle). The examples are:

- x -129 angle 55
- x 177 angle -58
- x 158 angle -56

Buttons for "Add example" are located at the bottom of each section. A "Language" dropdown menu is at the top right.

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!



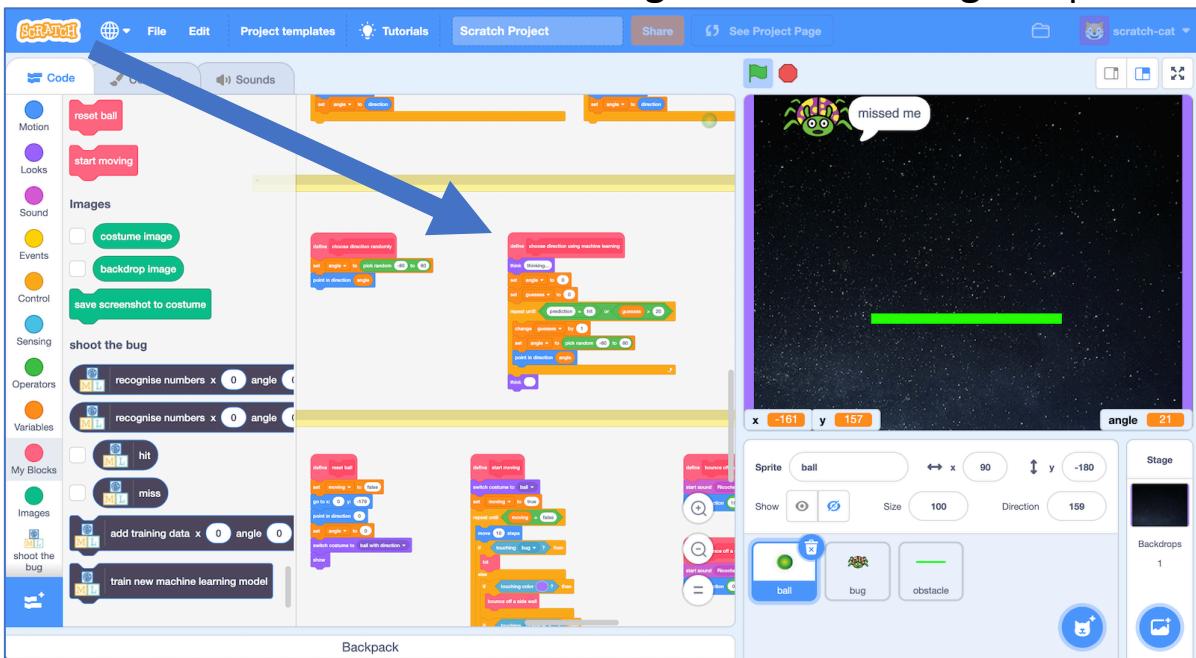
36. Click on the **Green Flag** to collect another example.
Do this at least **thirty** more times.

hit	miss
x: -68, angle: 55	x: 65, angle: 77
x: -102, angle: 55	x: -108, angle: 46
x: -87, angle: 57	x: 127, angle: -72
x: -129, angle: 57	x: 86, angle: -52
x: -175, angle: 59	x: -150, angle: -71
x: -153, angle: 58	x: 90, angle: -46
x: -60, angle: 57	x: 18, angle: 73
x: 3, angle: -55	x: 116, angle: 55
x: 93, angle: -58	x: -71, angle: 49
x: 171, angle: -60	x: 106, angle: -44
x: -174, angle: 62	x: -170, angle: -61
x: 0, angle: 53	x: -113, angle: 42
x: 58, angle: -54	x: 183, angle: 75
x: 137, angle: 47	x: 56, angle: 24
x: -36, angle: -49	x: 159, angle: -4
x: 163, angle: -64	x: 171, angle: 50
x: 125, angle: -57	x: 144, angle: -56
x: -161, angle: 69	x: 109, angle: -48

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.

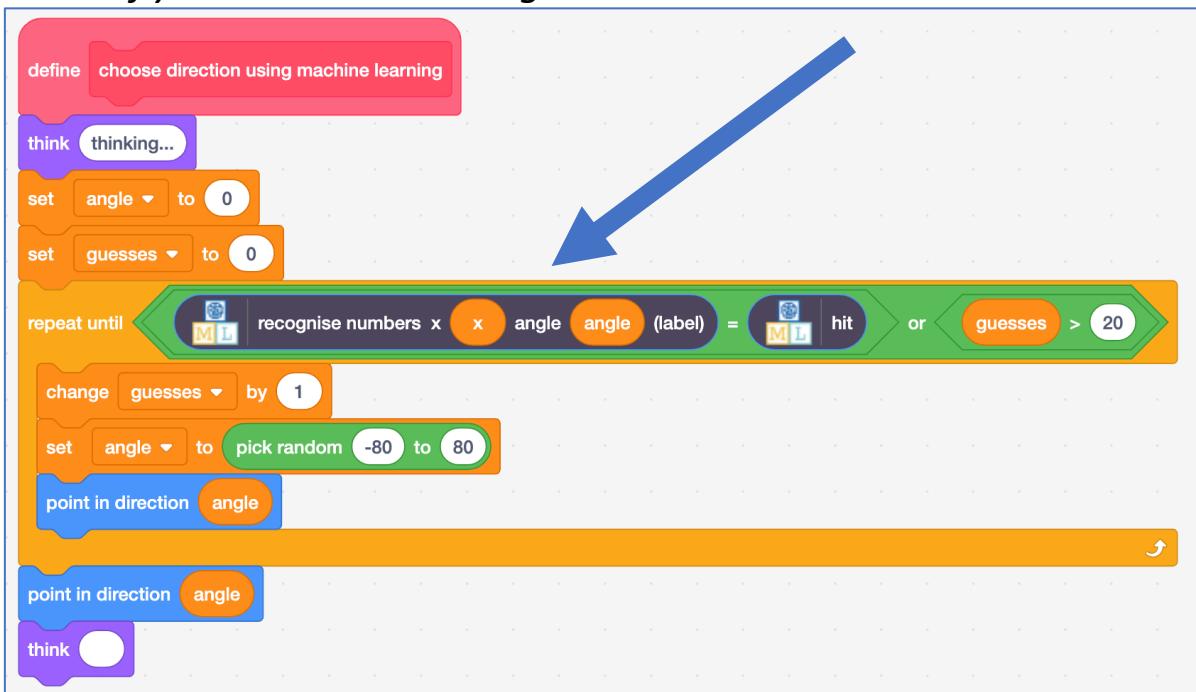
38. Add a “train new machine learning model” block to the “when Green Flag clicked” script

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?

Was hast du bis hierher gemacht?

Du hast begonnen, einen Computer zu trainieren, um ein Spiel zu spielen. Anstatt dafür Regeln zu schreiben oder die Gleichung zur Berechnung des Winkels, in dem der Ball abgefeuert werden soll, zu erarbeiten, hast du Beispiele gesammelt. Diese Beispiele werden vom Computer verwendet, um ein maschinelles Lernmodell zu trainieren.

Der Computer lernt aus den Mustern in den Beispielen. Er verwendet sie, um Vorhersagen darüber zu treffen, ob die Position des Käfers und ein Winkel zu einem Treffer oder einem Fehlschuss führen werden.

Da du immer noch die Blöcke "Trainingsdaten hinzufügen" in deinem Skript hast, sammelst du bei jedem Spiel weitere Trainingsbeispiele. Das bedeutet, je häufiger du das maschinelle Lernmodell spielen lässt, desto erfolgreicher sollte es werden.

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?

The screenshot shows a Scratch project titled "Recognising numbers as hit or miss". The interface includes a menu bar with "About", "Projects", "Worksheets", "News", "Help", "Log Out", and a "Language" dropdown. Below the title, there's a link "[< Back to project](#)". The main area is divided into two sections: "hit" and "miss".
hit: This section contains 51 examples. Each example is a card with a number and an angle. The cards are arranged in a grid-like structure. Some examples include:

- angle 55, angle 55, angle 57, angle 57, angle 59
- x -153, angle 58, x -60, angle 57, x 3, angle -55, x 93, angle -56, x 171, angle -60
- x -174, angle 62, x 0, angle 53, x 58, angle -54, x 137, angle 47, x -36, angle -49
- x 163, angle -64, x 125, angle -57, x -161, angle -69, x 147, angle -66, x 142, angle -62
- x 101, angle -66, x -41, angle -53, x -102, angle 56, x -175, angle 62, x 7, angle -53
- x -38, angle -58, x 167, angle -57, x 24, angle 40, x -180, angle -40, x 110, angle -57

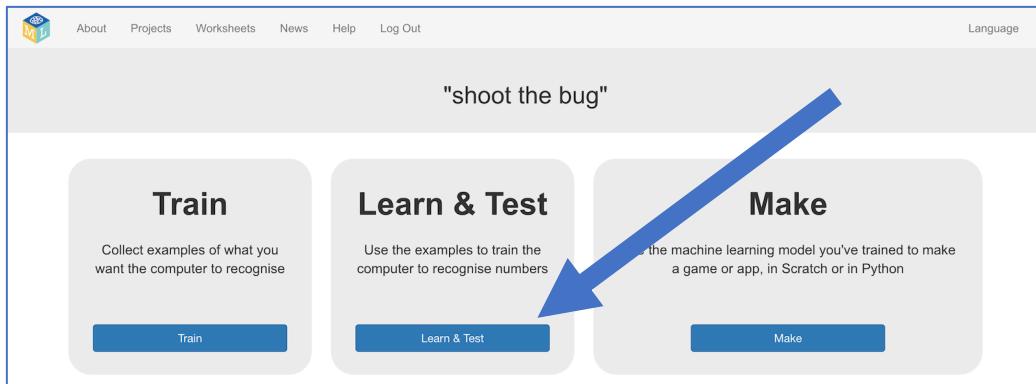
miss: This section contains 61 examples. Each example is a card with a number and an angle. The cards are arranged in a grid-like structure. Some examples include:

- x -20, angle -66, x -100, angle -38, x -100, angle -20, angle 77, angle 46
- x 127, angle -72, x 86, angle -52, x -150, angle -71, x 90, angle -46, x 18, angle 73
- x 116, angle 55, x -49, angle 63, x -71, angle 36, x 106, angle -44, x -170, angle -61
- x -113, angle 42, x 183, angle 75, x 56, angle 24, x -161, angle -32, x 144, angle -56
- x 109, angle -48, x 159, angle -4, x 171, angle 50, x 103, angle -35, x 7, angle -28
- x -77, x -161, x 49, x 70, x -172

At the bottom of each section, there are buttons "+ Add example".

44. Click on the “< Back to project” link

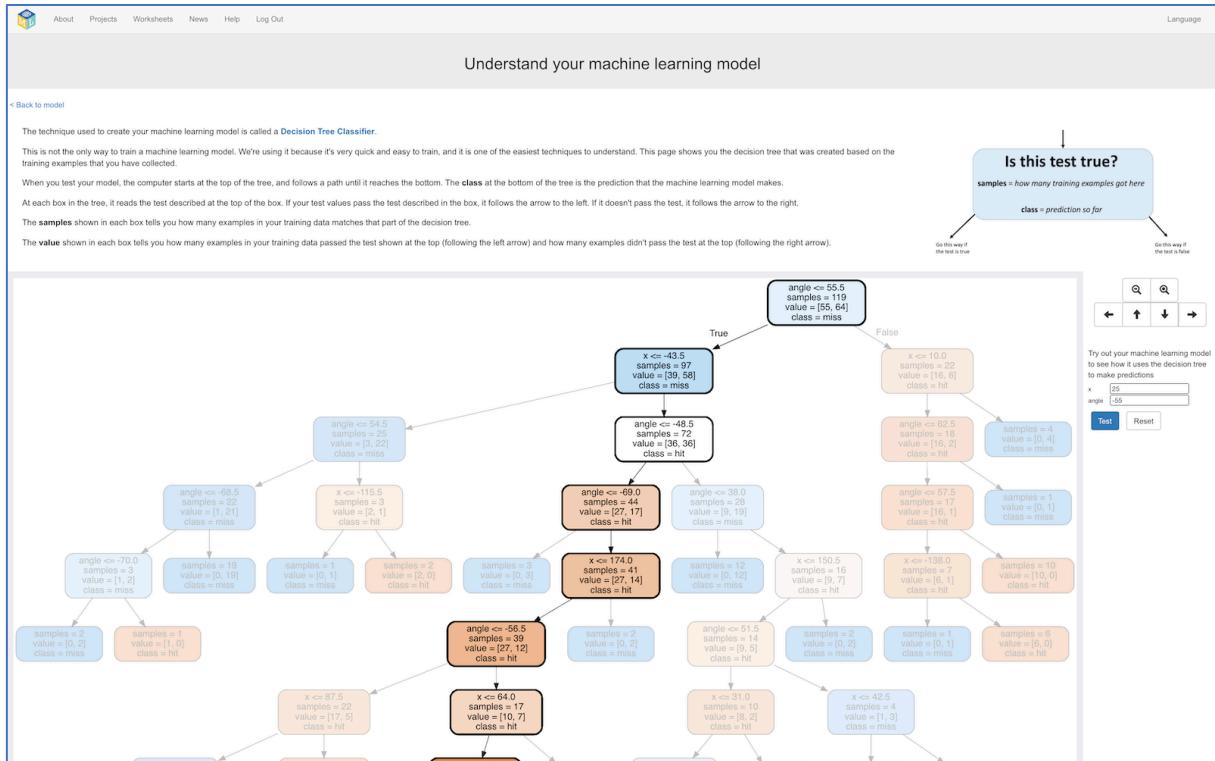
45. Click on “Learn & Test”



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.

Was hast du gemacht?

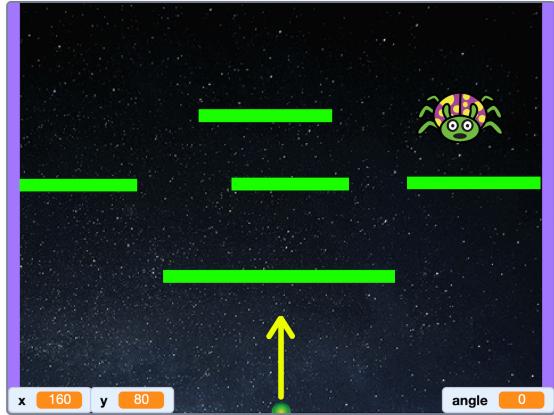
Der Typ des maschinellen Lernmodells, das du trainiert hast, ist ein "**Entscheidungsbaum-Klassifizierer**" (=“**decision tree classifier**”). Mit der Visualisierung kannst du sehen, wie das Modell Vorhersagen erstellt. Es ist eine gute Möglichkeit zu sehen, welche Muster der Computer in den gesammelten Trainingsdaten gefunden hat.

Ist das eine gute Anwendung für maschinelles Lernen?

Wir verwenden maschinelles Lernen, wenn wir wollen, dass ein Computer Dinge „von selbst lernt“. Das machen wir so, wenn es zu kompliziert wäre, die Anweisungen in einem Programm zu schreiben.

Weniger gut ist maschinelles Lernen geeignet, wenn der Zeitaufwand für das Sammeln von Trainingsbeispielen für eine Aufgabe länger dauern würde als das Schreiben eines Programms.

Vergleich mal den Aufwand für das Sammeln von Beispielen, um den Computer zu trainieren, mit dem Aufwand, den Winkel zum Schießen in einem Programm selbst zu schreiben Denkst du, dass dieses Spiel eine gute Anwendung von maschinellem Lernen ist?



Was wäre, wenn das Spiel schwieriger gemacht worden wäre? Was wäre, wenn es zwei Hindernisse gäbe, die umgangen werden müssten? Oder drei? Oder fünf? Was wäre, wenn der Käfer in zufälligen Höhen erscheinen könnte, nicht nur oben?

Diese Art von Änderungen machen es schwieriger zu wissen, was mit dem Ball passiert, wenn du schießt. Die Gleichungen, die du benötigen würdest, um den richtigen Winkel zum Feuern zu berechnen, wären viel komplizierter.

Das macht es zu einer besseren Anwendung des maschinellen Lernens als das Spiel mit nur einem Hindernis zu spielen.

(Aber es bräuchte wahrscheinlich mehr Trainingsbeispiele für den Computer, um zu lernen, wie man das spielt, weil es komplexer ist als um ein einzelnes Hindernis herumzukommen. Probieren es ruhig aus und überzeuge dich selbst!)

Ideen und Erweiterungen

Jetzt, wo du fertig bist, wie wäre es mit folgenden Ideen?
Oder fällt dir selbst noch etwas Cooles ein?

Füge weitere Hindernisse ein

Versuche das Spiel herausfordernder zu gestalten, indem du zusätzliche Hindernisse einfügst.

Du wirst das “start moving”-Skript erweitern müssen, so dass der Ball auch von den neuen Hindernissen abprallt.

Benutze x und y Koordinaten

Um den Trainingsaufwand zu reduzieren, haben wir nur eine Koordinate (x-Koordinate) verwendet und den Käfer nur nach links/rechts wandern lassen.

Versuche, das Projekt erneut durchzuführen, wobei sich der Käfer auch auf eine zufällige Höhe (y-Position) bewegen kann. Du musst einen neuen Zahlenwert hinzufügen, um diese y-Koordinaten zu speichern, wenn du das Projekt für maschinelles Lernen erstellst.

Mach es zum Wettbewerb!

Versuche, eine Variable hinzuzufügen, um die Punktzahl festzuhalten. Und schau mal, ob das maschinelle Lernmodell eine höhere Punktzahl erreichen kann als du!