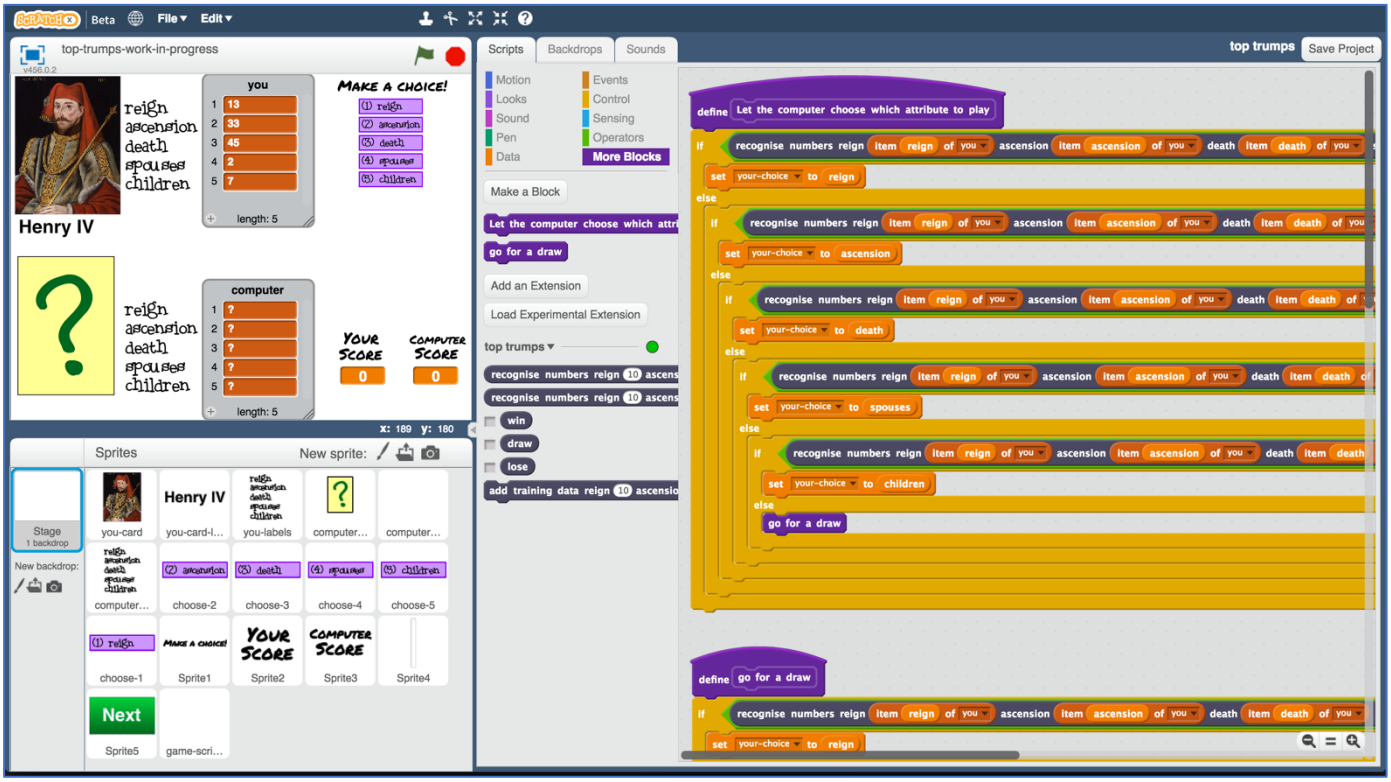
Top Trumps

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

For some values on the cards, you win by having the highest number. For others, you win by having the lowest. The range of numbers for different values will vary.

The aim will be for the computer to learn how to play the game well without you having to give it a list of all the cards or tell it the rules.

Instead, you'll try two different ways of training the computer to play the game by giving it examples of the game being played.



1. You'll need the **top-trumps.sbx** starter file for this project.
If you haven't got this, ask your teacher or group leader.

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

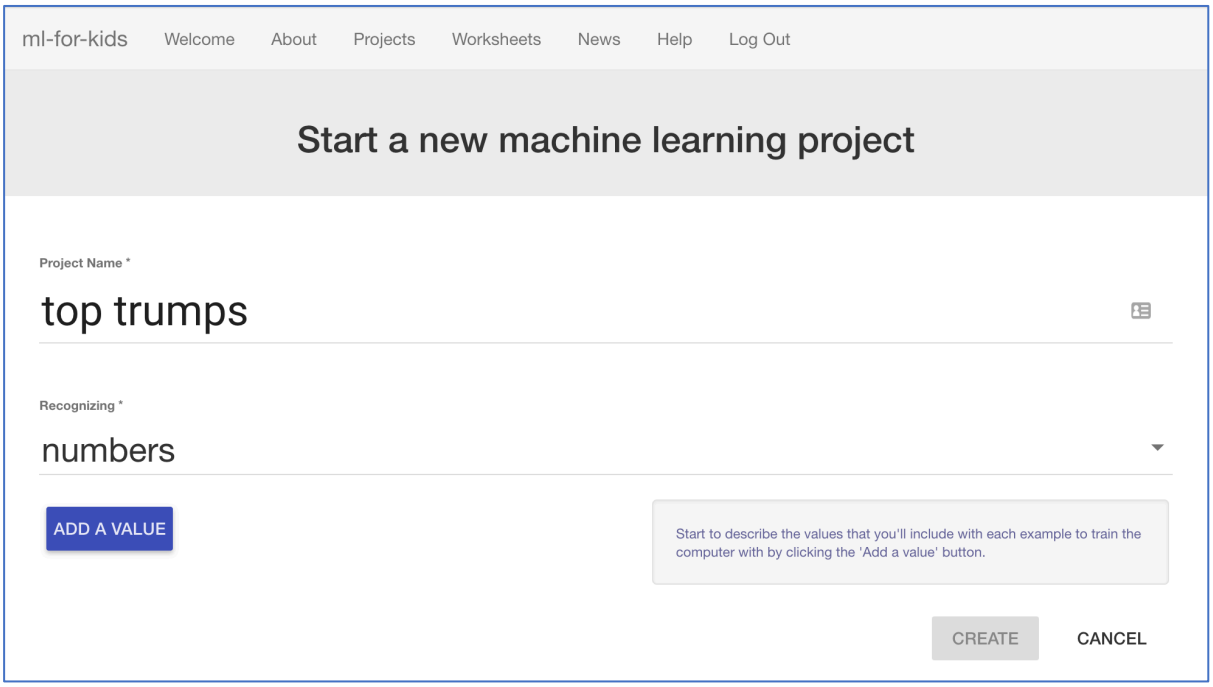
3. Click on "**Get started**"

4. Click on "**Log In**" and type in your username and password
If you don't have a username, ask your teacher or group leader to create one for you.
If you can't remember your username or password, ask your teacher or group leader to reset it for you.

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

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

7. Name your project "top trumps" and set it to learn how to recognise "**numbers**"



8. Click the "**Add a value**" button five times.

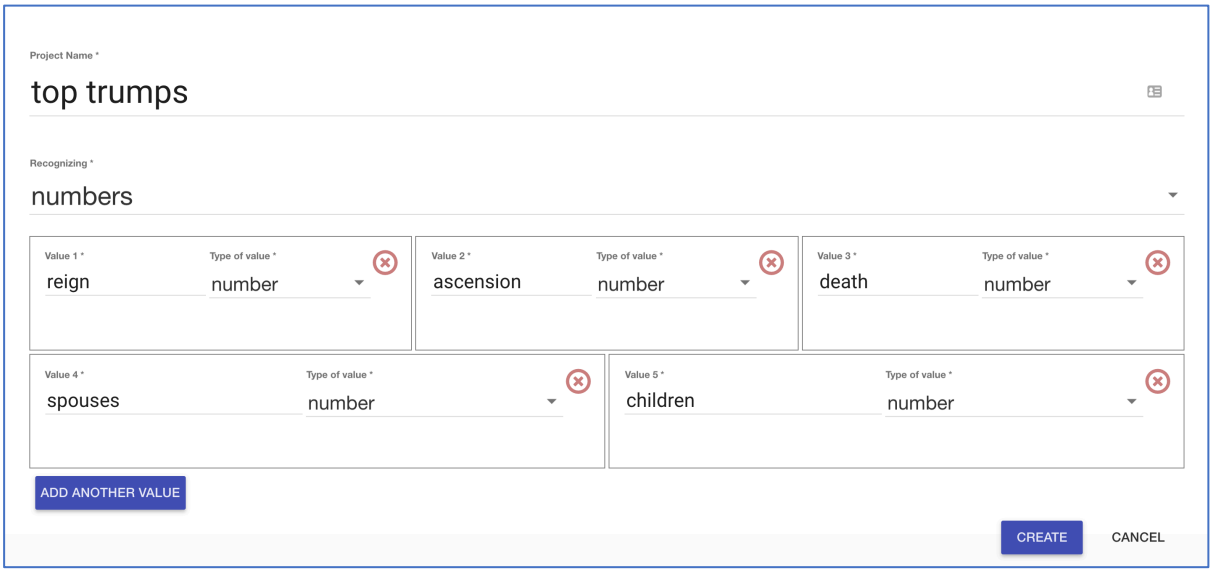
That will give you five text boxes to type names into.

Enter the following names into these boxes, in this order:

* reign
* ascension
* death
* spouses
* children

Set the type for all of these to "number"

These will be where you put the numbers that are on a Top Trumps card.



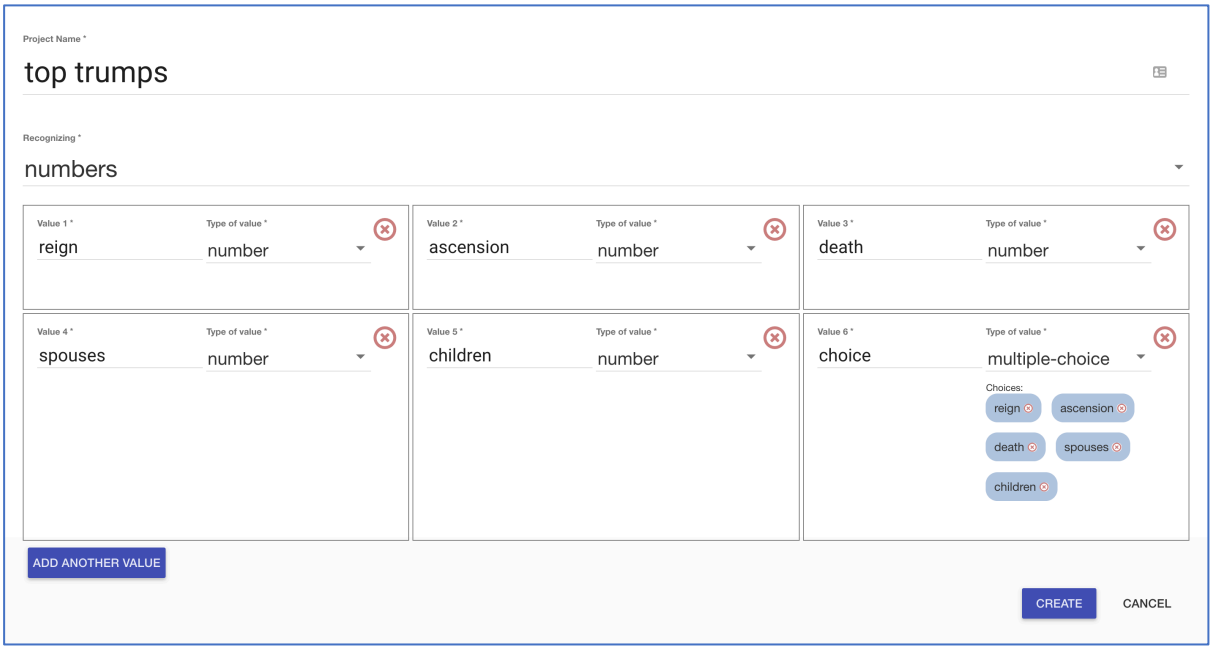
9. Click “**Add another value**” one more time

Call this new value “choice”

Set it’s type to “**multiple-choice**”

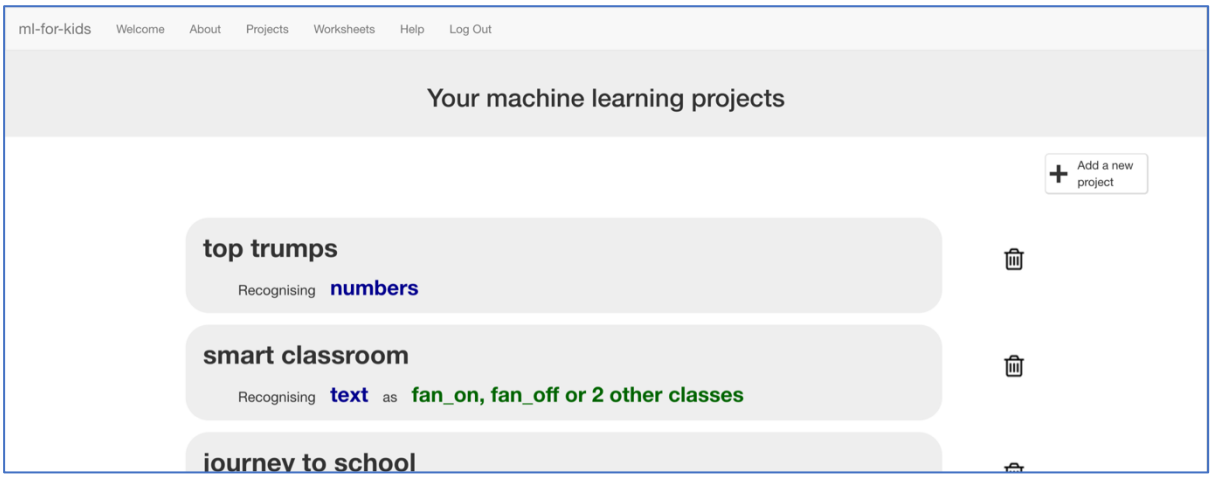
Add “reign”, “ascension”, “death”, “spouses”, “children” as the choices for this value.

This is where you will put which value you choose from the Top Trumps card.

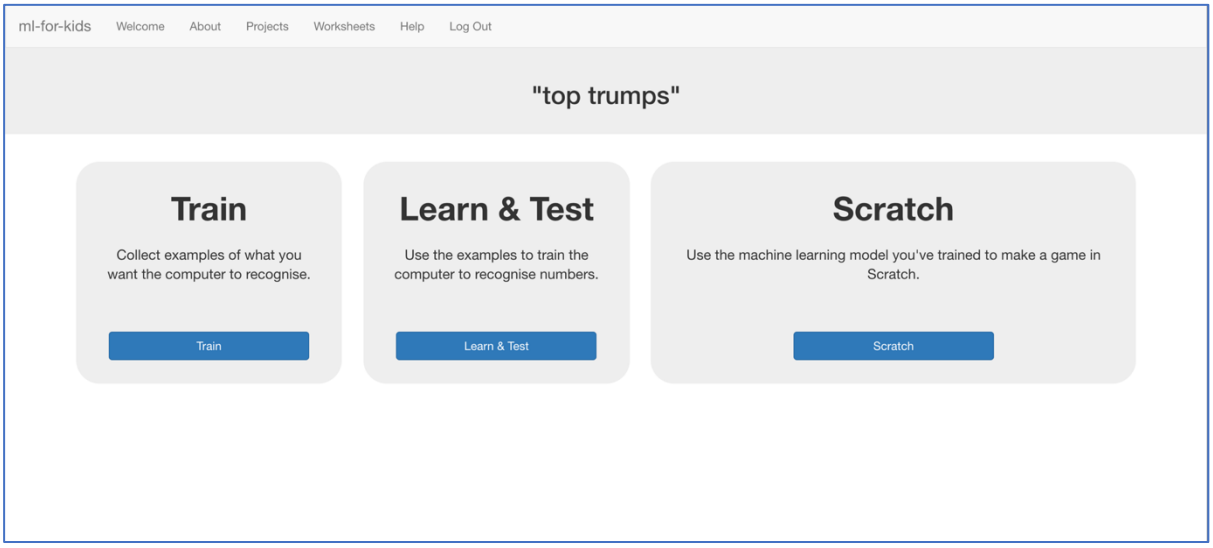


10. Click the “**Create**” button

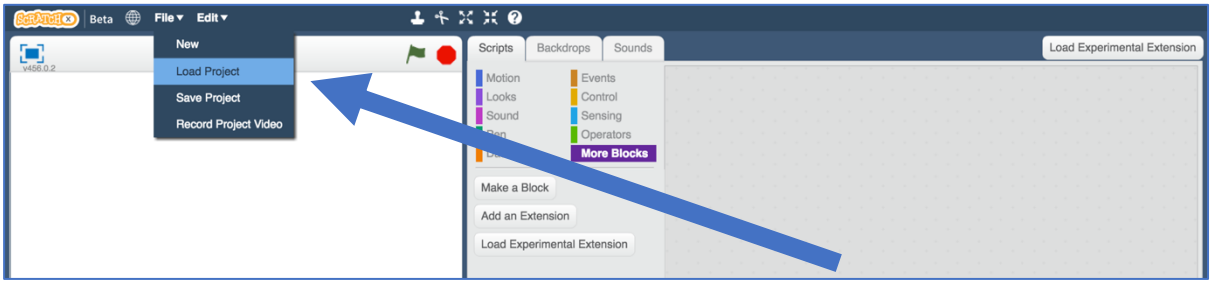
11. You should now see "**top trumps**" in the list of your projects. Click on it.



12. Start by getting a project ready in Scratch. Click the **Scratch** button.
*The next page will warn you that you haven't done any machine learning yet, but clicking on the **Scratch by itself** link will launch Scratch.*

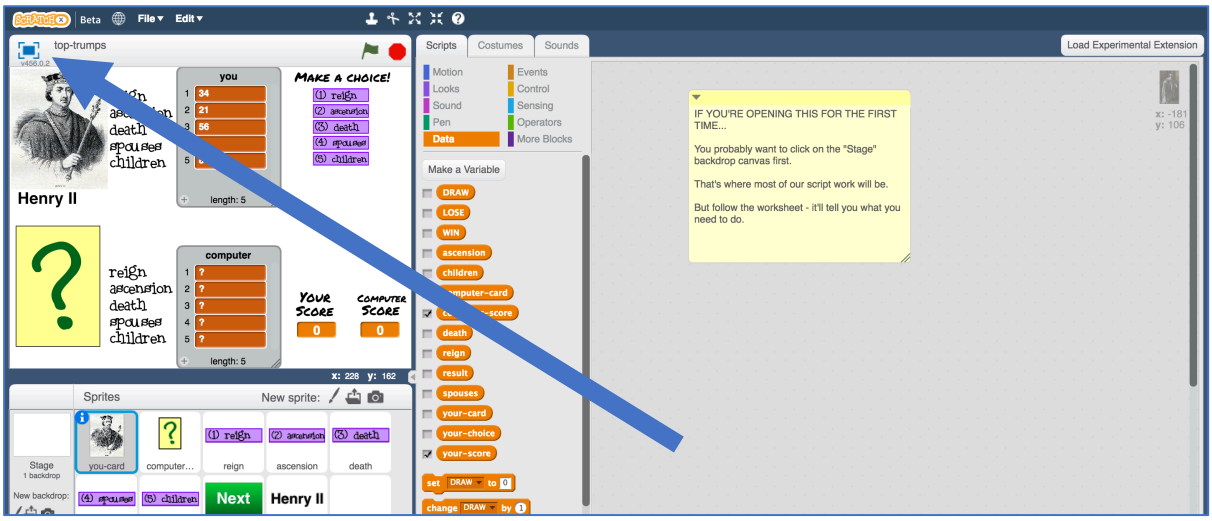


13. Load the top-trumps.sbx file
*Use **File -> Load Project** as shown below*
Click OK when it asks if you want to replace the current project.

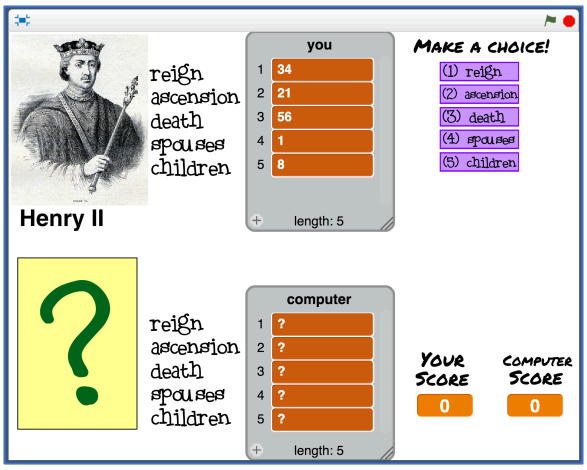


14. This is Top Trumps based on the Kings and Queens of England.

Click the **full-screen button**.



15. Click the **green flag** to start



The top half of the screen is you.

The bottom half is the computer.

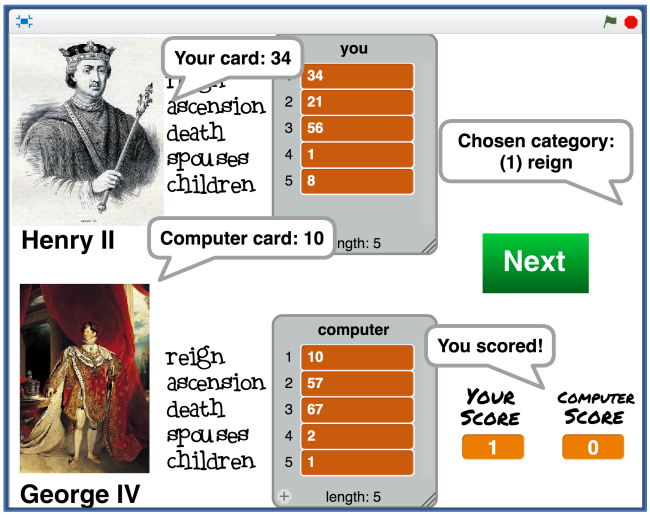
When you click the Green Flag to start the game, you can't see the computer's card yet.

It's all just question marks.

Choose a value from your king or queen by **clicking the purple button next to it**

For example, in the screenshot above, my card is Henry II.

1) He reigned for **34** years.
   (If I choose this and he was King longer than the computer's card, I'll win)
2) He ascended to the throne when he was **21**.
   (If I choose this and he become King quicker than the computer's card, I'll win)
3) He died when he was **56**.
   (If I choose this and he lived longer than the computer's card, I'll win)
4) He had **1** spouse.
   (If I choose this and he had more spouses than the computer's card, I'll win)
5) He had **8** children.
   (If I choose this and he had more children than the computer's card, I'll win)



When you choose a value, the computer card is revealed, and you see if you win or lost.

The score in the bottom right corner is updated.

Click on the green Next button to move onto the next card and play again.

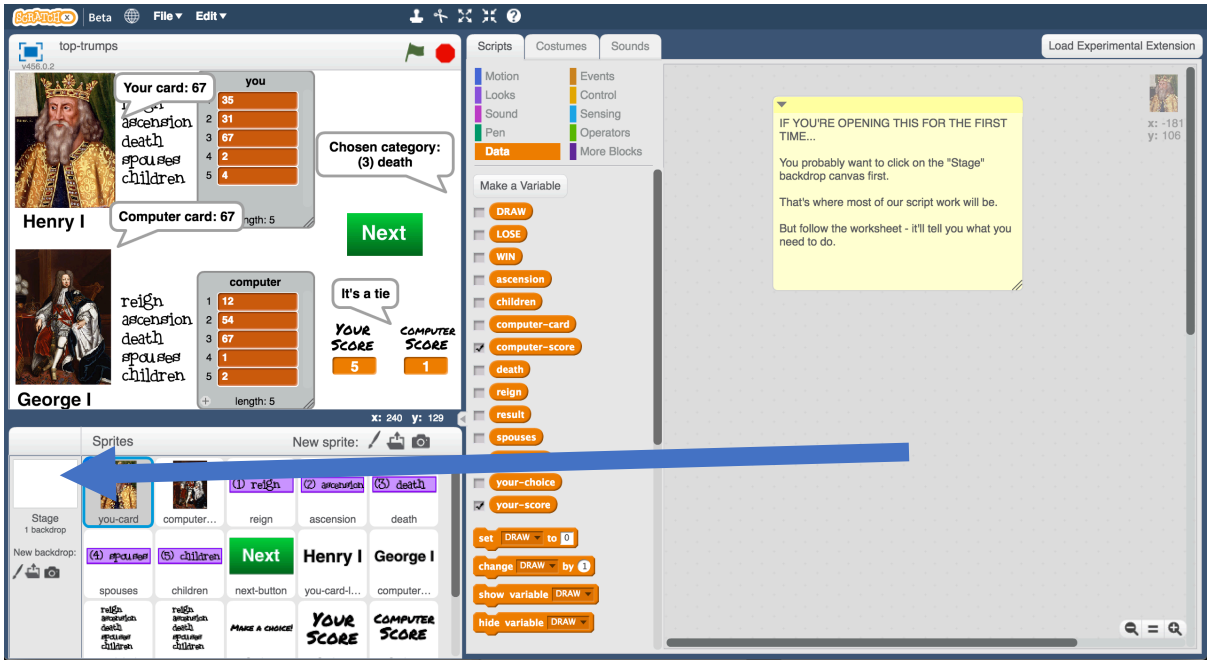
If you win or tie, it's your turn again.
If you lose, the computer will get to choose the next value instead.

16. Play a few rounds of the game against the computer.

Try to work out how the computer is choosing values to play.
When you think you've worked out how the computer is playing, move onto the next step.

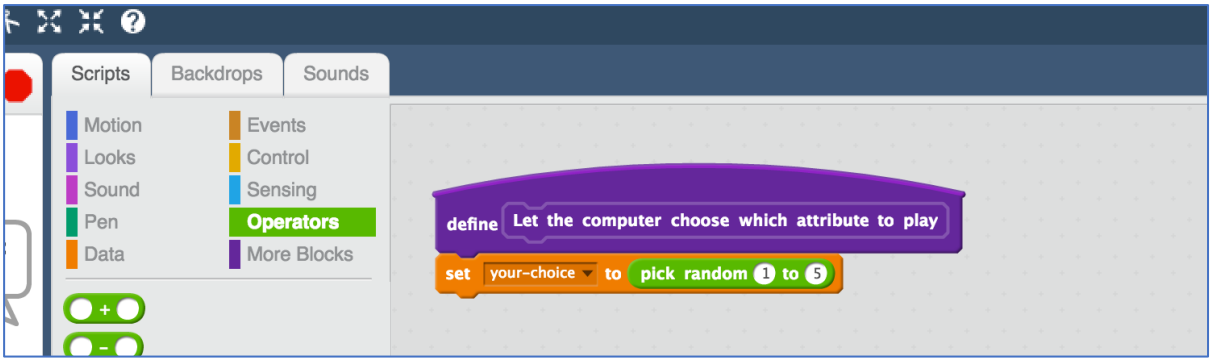
17. Click on the **full-screen** button again to go back to normal view.

Then click on the **Stage**
This is the white rectangle above "Stage" shown with the arrow below.



18. The script on the **Stage** shows how the computer has been coded.
The computer always picks "reign".
Did you get it right?

19. Change the script so that the computer chooses a value at random when it's the computer's turn.
Choosing from 1 (reign) to 5 (children) at random.



20. Click the **green flag** to reset the scores to 0. Go back to full-screen and play the game again.
Stop when either you or the computer reaches 10 points. Who won?

What have you done so far?

You've set up a bot to play Top Trumps and given it a simple strategy: choose values at random.

But people don't play like that. We learn how to choose which value would give us the best chance of winning. We do this based on the cards we've seen before, and on our understanding of the rules.

Next, you'll try a simple way to train the computer using a few experiences of seeing how the game is played.

21. Go back to Scratch.

Play the game, and write down what happens. You can do this by filling in the tables on the next page.

*Only write down the numbers from **your** cards.*

You need:
5 examples of a round where you won.
5 examples of a round where you drew.
5 examples of a round where you lost.

An example row is included in each table to show you what I mean.

Examples where you **won**, and the computer lost

The values from your card					which value did you pick?
reign	**ascension**	**death**	**spouses**	**children**	
23	*24*	*48*	*1*	*9*	*ascension*

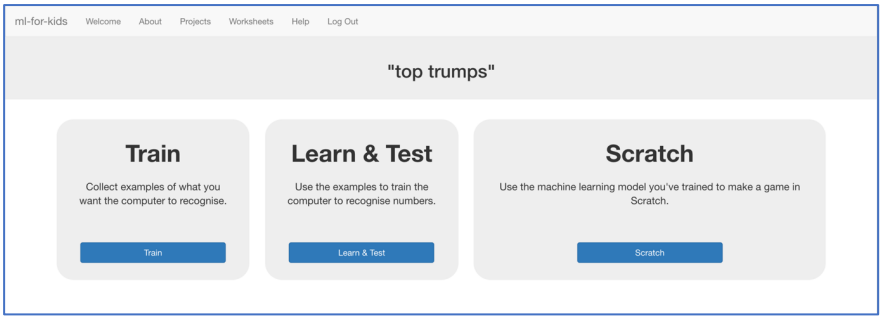
Examples where it was a **draw**

The values from your card					which value did you pick?
reign	**ascension**	**death**	**spouses**	**children**	
21	*36*	*58*	*1*	*9*	*spouses*

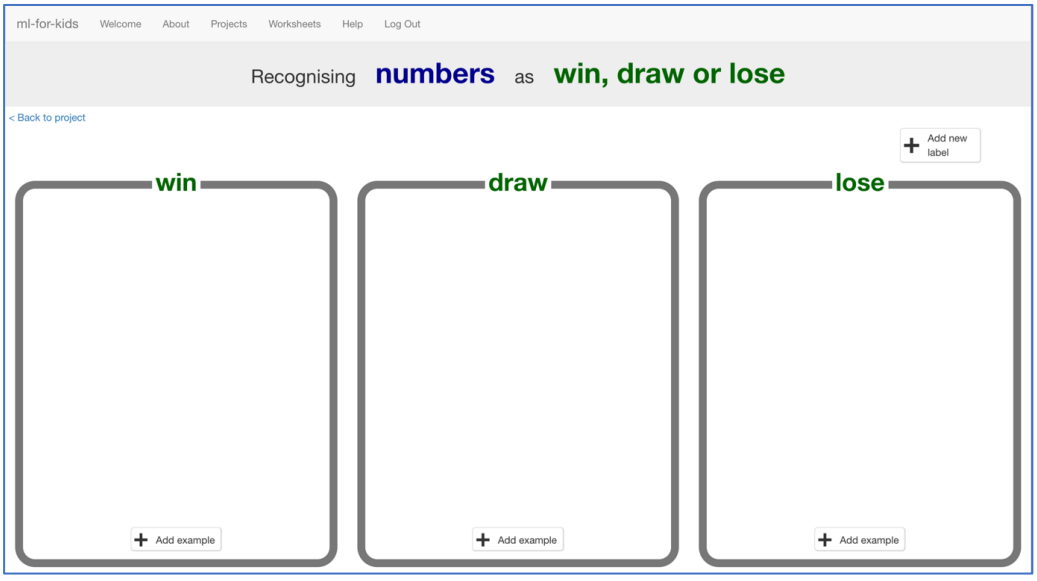
Examples where you **lost**, and the computer won

The values from your card					which value did you pick?
reign	**ascension**	**death**	**spouses**	**children**	
13	*38*	*51*	*1*	*3*	*death*

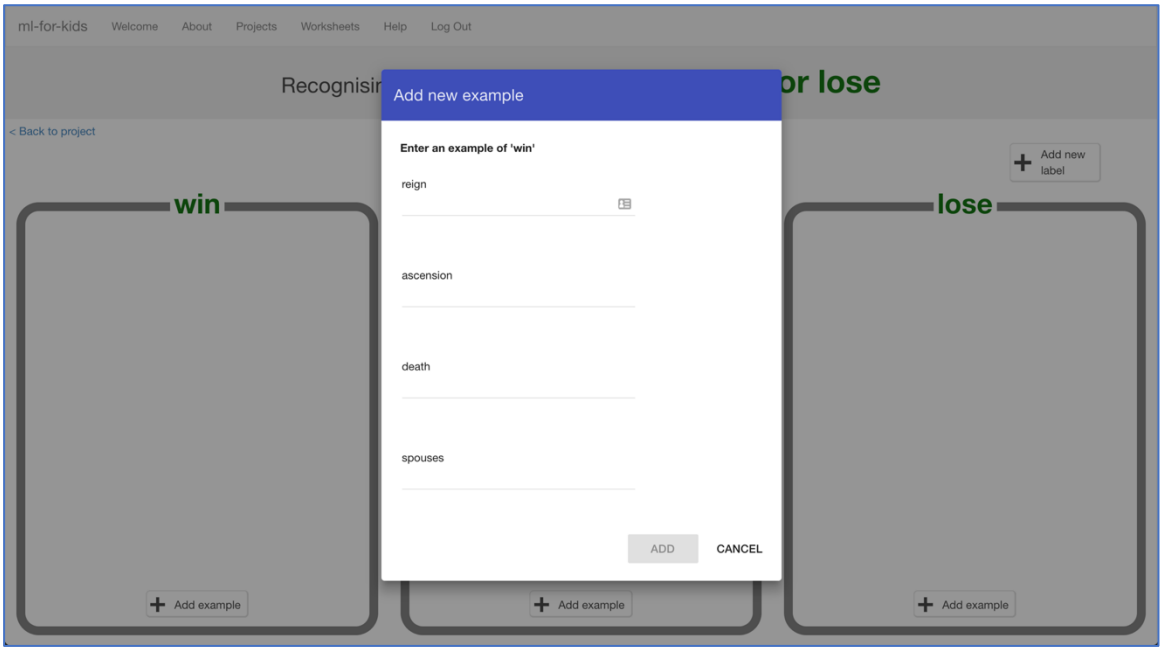
22. Close Scratch. Click the **"< Back to project**" link.
Click on the **Train** button.



23. Click on **"+ Add new label**" and call it "win".
Do that again, and create a second bucket called "draw".
Do that again, and create a third bucket called "lose".



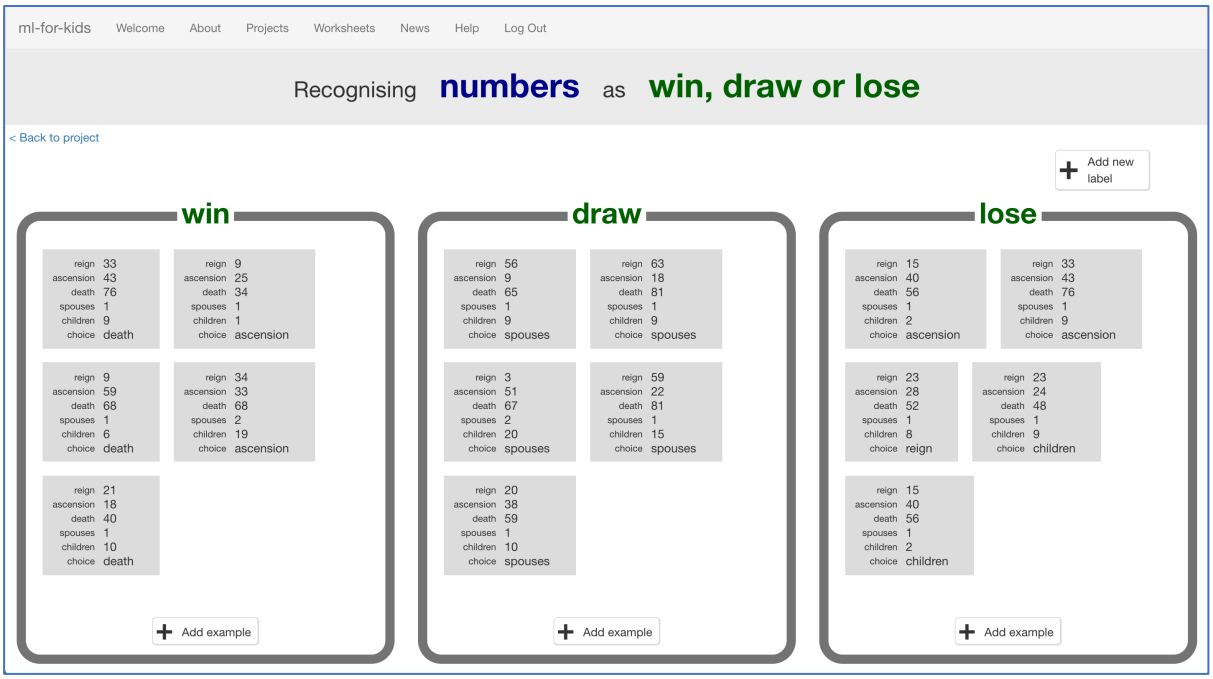
24. Click the “**Add example**” button in the “win” bucket, and type in the values from the first row in your “win” table.



25. Do this for all the examples in the “win” table.

Now click on the “**Add example**” button in the “draw” bucket, and do the same for all the rows in the “draw” table.

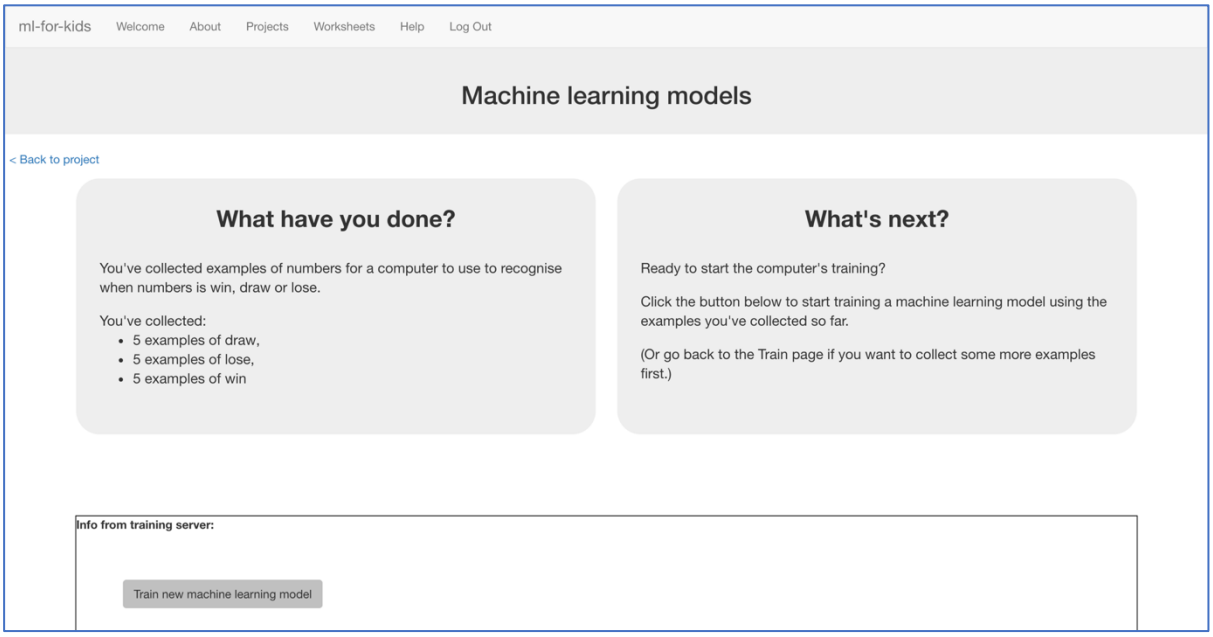
Then type in all the examples from the “lose” table into the “lose” bucket.



26. Click the **“< Back to project**” link. Click the “**Learn & Test**” button.

27. Click on the "**Train new machine learning model**" button.

As long as you've collected enough examples, the computer should start to learn how to recognise commands from the examples you've written.

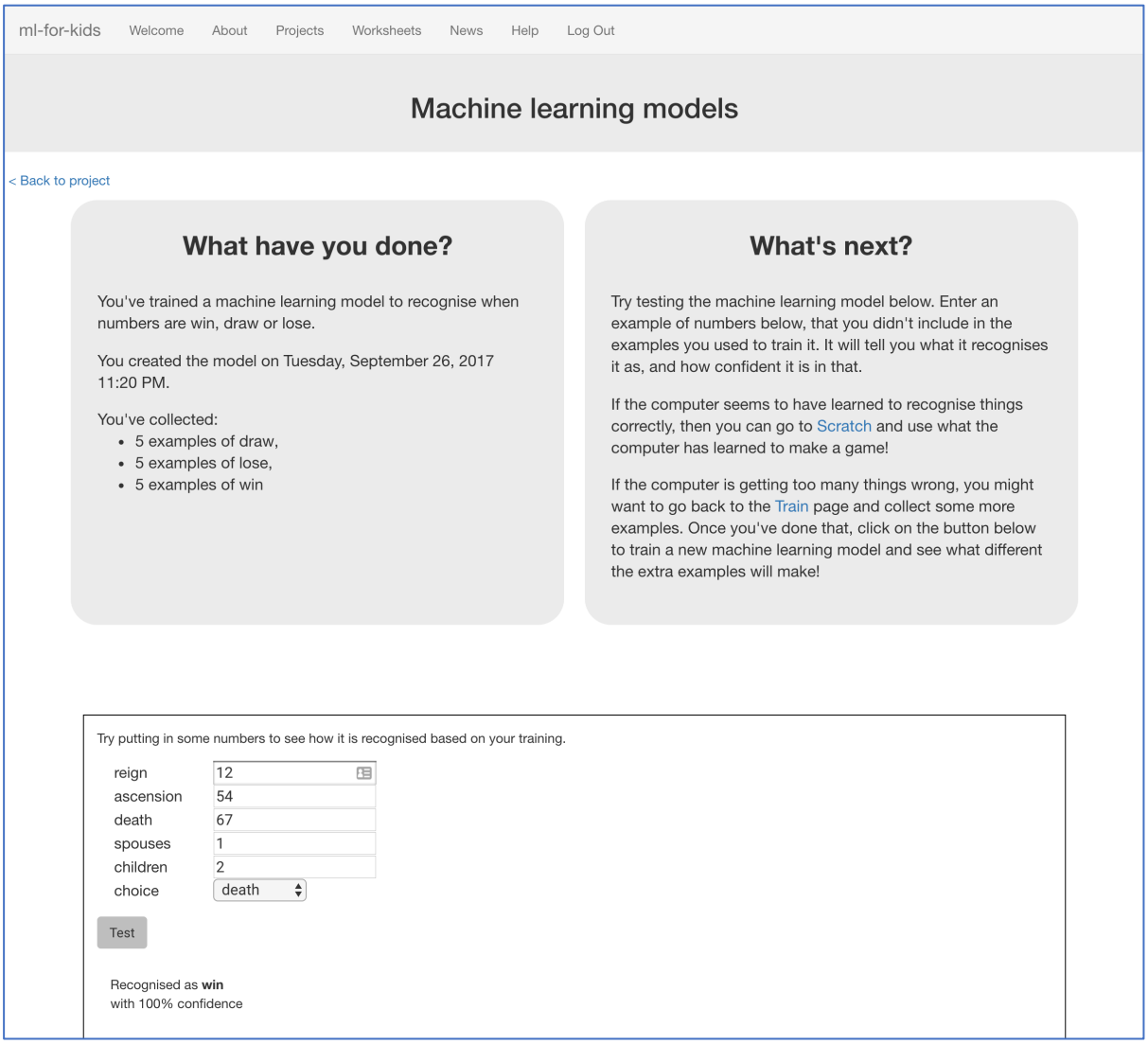


28. Once the training has completed, a Test box will be displayed.

Test your machine learning model to see what the computer has learned.

Type in all the values from a card into the 'reign', 'ascension', 'death', 'spouses', and 'children' boxes.
Then pick the value you would choose from the 'choice' drop-down list.

The computer will tell you whether it thinks you will win, lose or draw based on that choice.



What have you done so far?

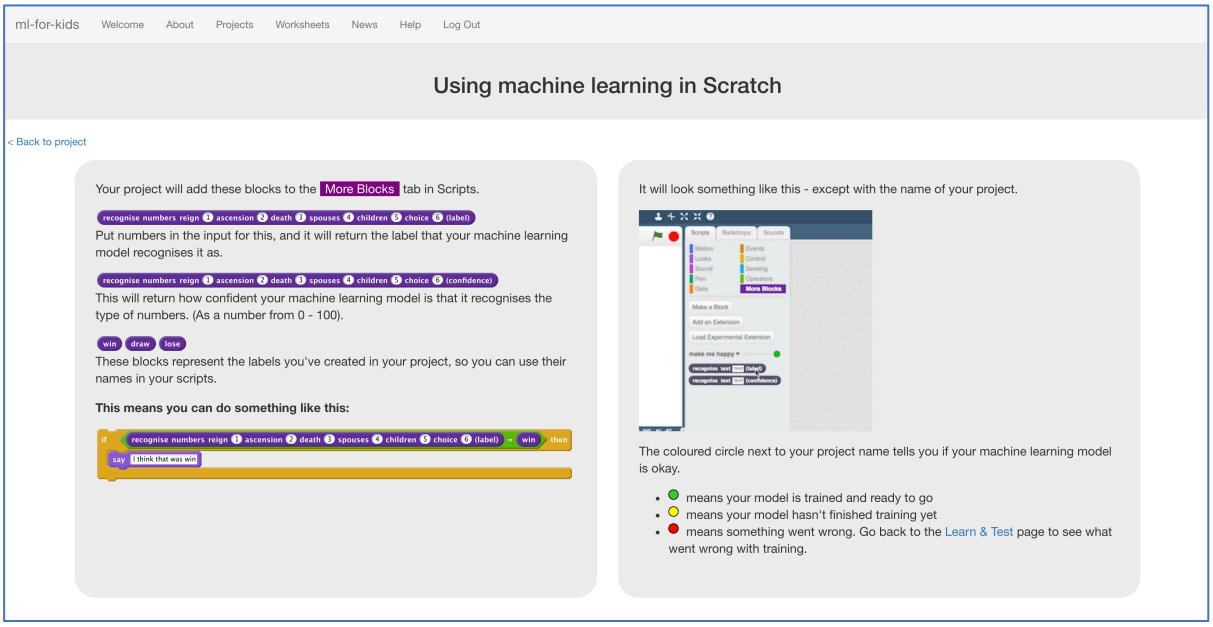
You've started to train a computer to learn about Top Trumps.

The examples help the computer learn what values to expect in cards: the range of numbers for each value, how often it should expect to see high values, how often it should expect to see low values.

The examples also help the computer to learn what numbers will cause it to win or lose, without you needing to tell it what the rules are.

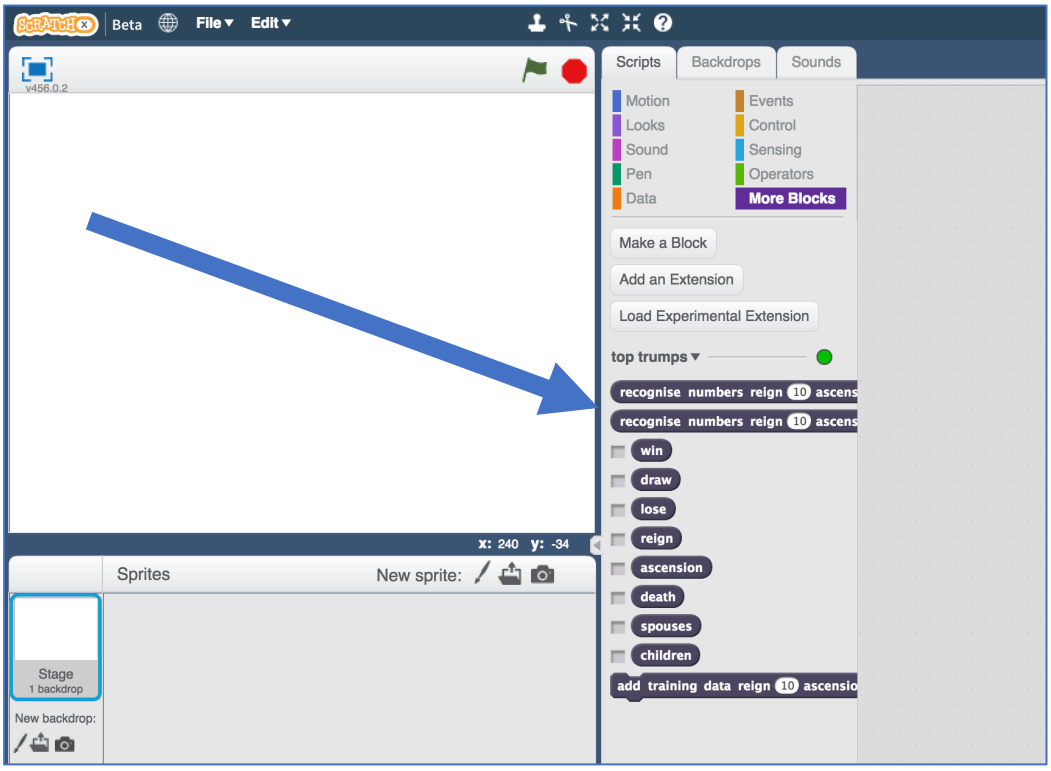
29. Click the **“< Back to project**” link, then back to the “**Scratch**” button.

This page will be updated with instructions on how to use the new blocks in Scratch from your project. Keep this page open so you can check back on how to use them.



30. Click the “**Open in Scratch**” button at the bottom of that page to launch the Scratch editor.

You should see new blocks in the “More blocks” section from your “top trumps” project.



31. Load the "top-trumps.sbx" Scratch project you opened before.
*Click on **File** -> **Load Project***
Click 'OK' when it asks if you want to replace the current project.

32. Click on the **Stage** (as you did before) to go back to the Script for how the computer chooses its moves.

33. You need to change the script so that the computer uses the machine learning model that you've started to train.

Your script needs to:

*1) **Make a prediction for all the possible choices on its card***
(reign / ascension / death / spouses / children)

*2) Choose the one that the computer predicts will give it a **win***

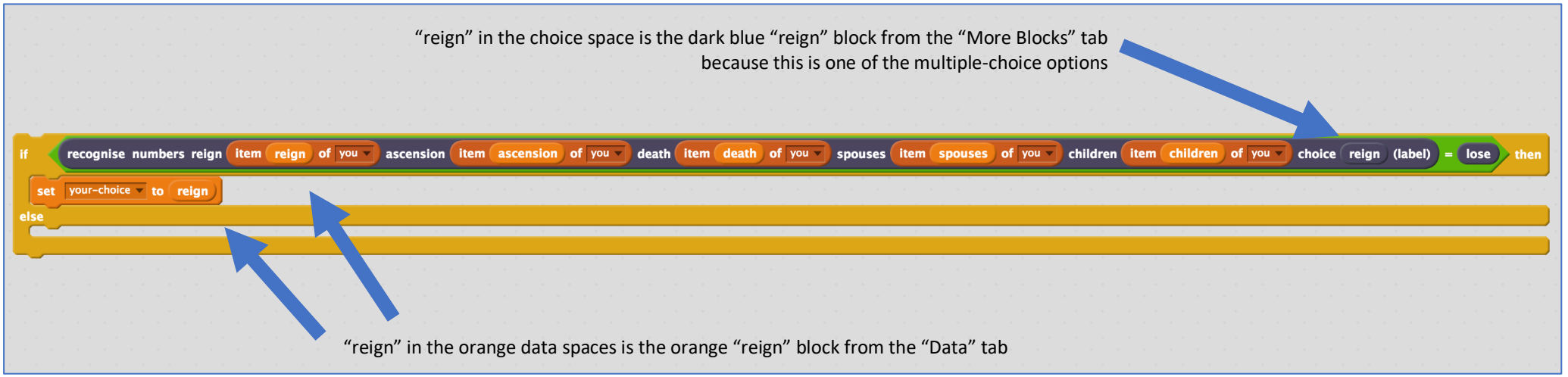
*3) If the computer doesn't predict any of the choices will give it a win, then choose the one that the computer predicts will give it a **draw**.*

*4) If the computer doesn't predict any choice will give it a win or draw, then choose one at **random** and hope for the best!*

This is a long and complicated script.

Take it slowly - we'll build it up in stages.

Step 1 – Create this – to check for a win for a single value (reign)
(Use the label "lose" because the computer is going to choose which value to go for based on your card. For it to win, it wants to choose a value that it has learned will make you lose).

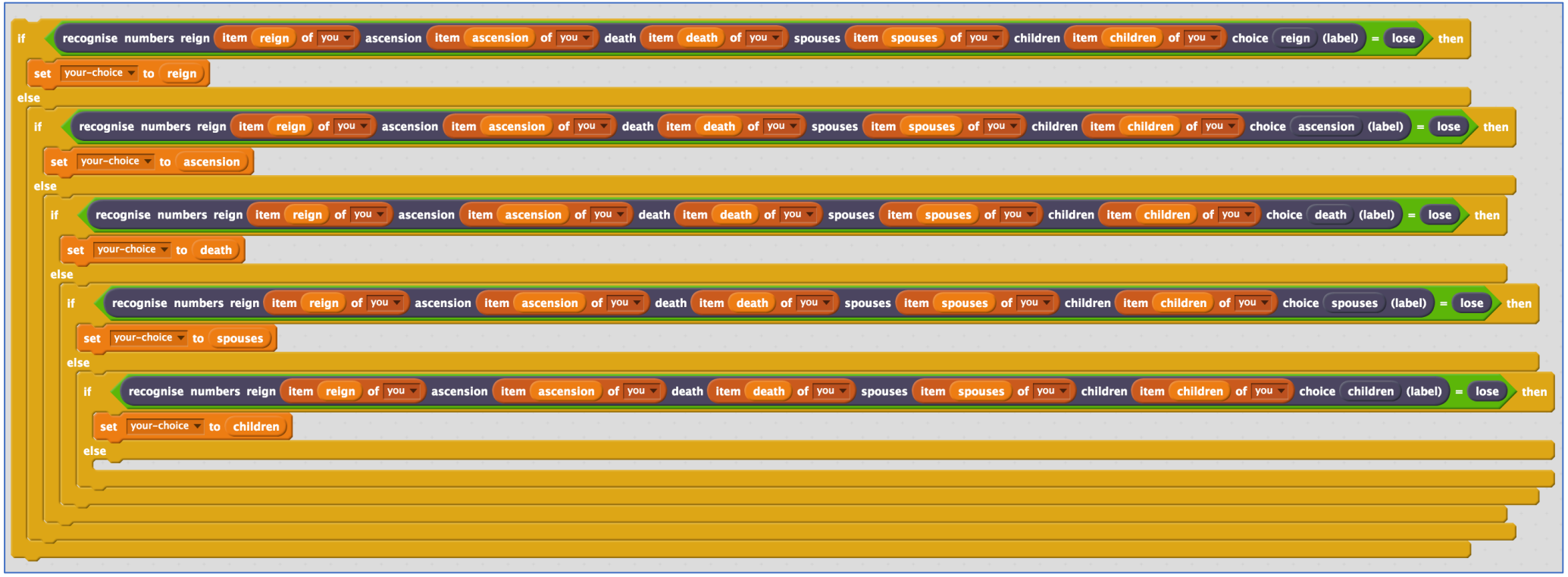


Step 2 – Duplicate that. Put the copy into the else block.
Change the choice and the set-your-choice value from "reign" to "ascension".
You should end up with this:

Step 3 – Duplicate again and change the choice and set-your-choice to “death”. Put that copy into the else block.

Repeat for “reign”
Repeat again for “children”.

You should end up with this:

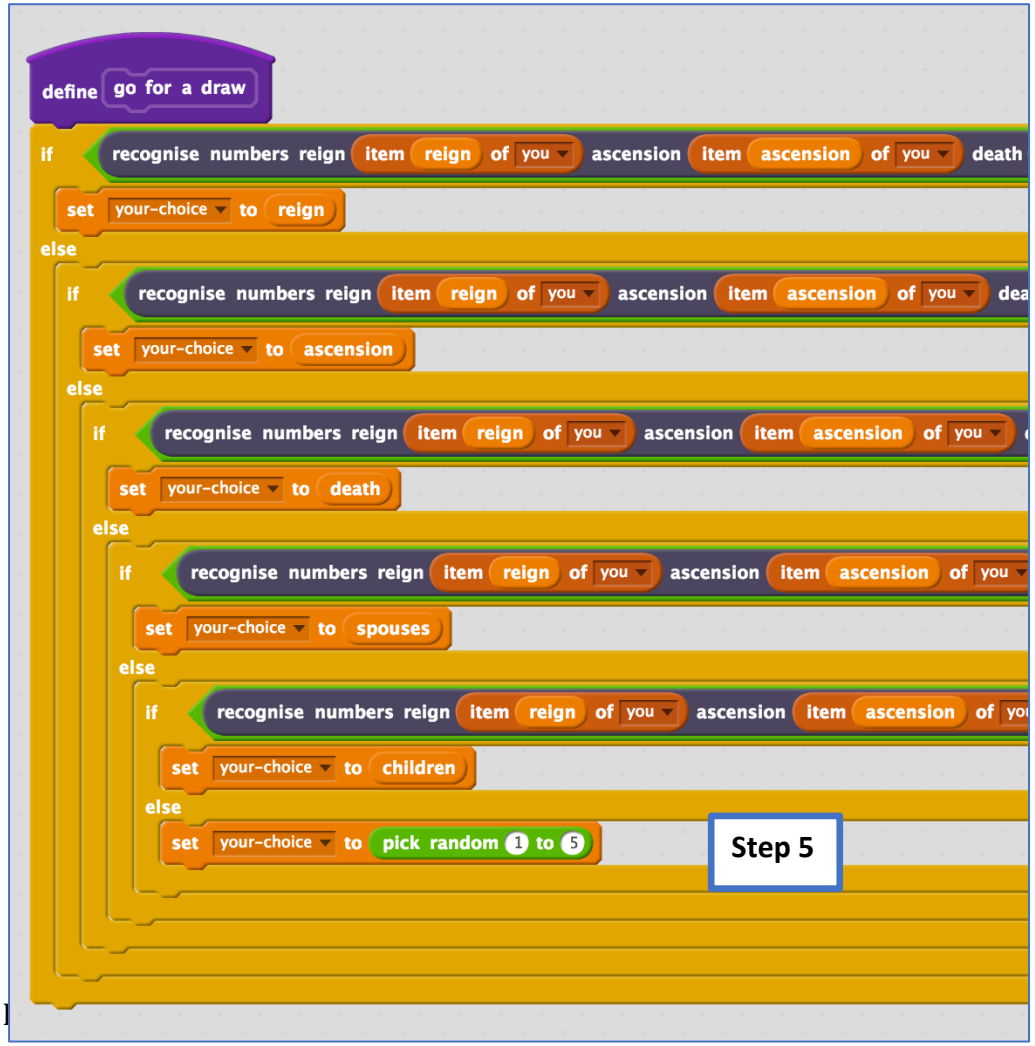


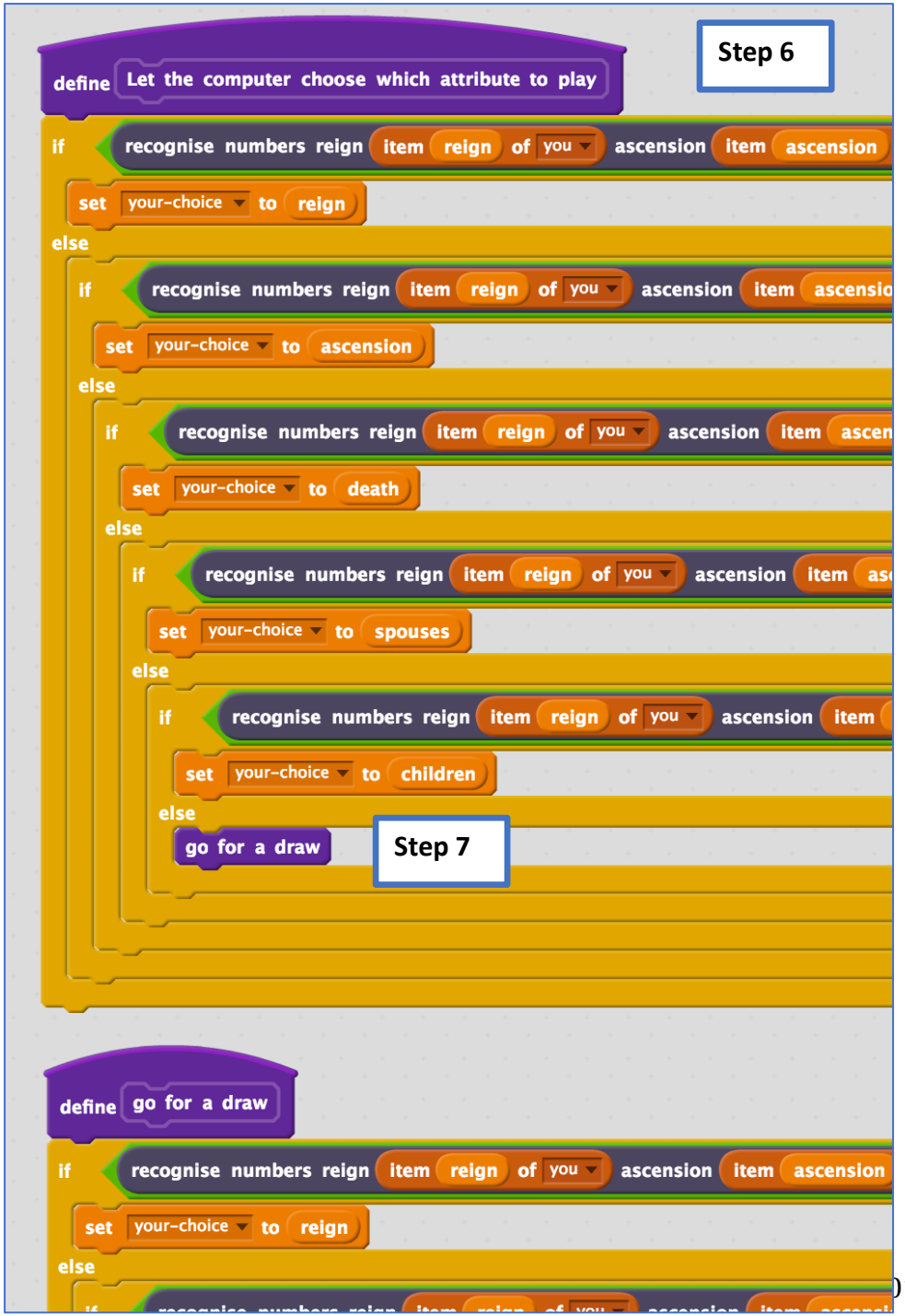
Step 4 – Click “Make a Block” and create a block called “go for a draw”.
Duplicate everything we just did into this new “go for a draw” block.
Change all the “lose” values to “draw”.
You should end up with this new block:

Step 5 – Add a random choice block in the last else of "go for a draw"

Step 6 – Join up the first script you made to "Let the computer choose which attribute to play"

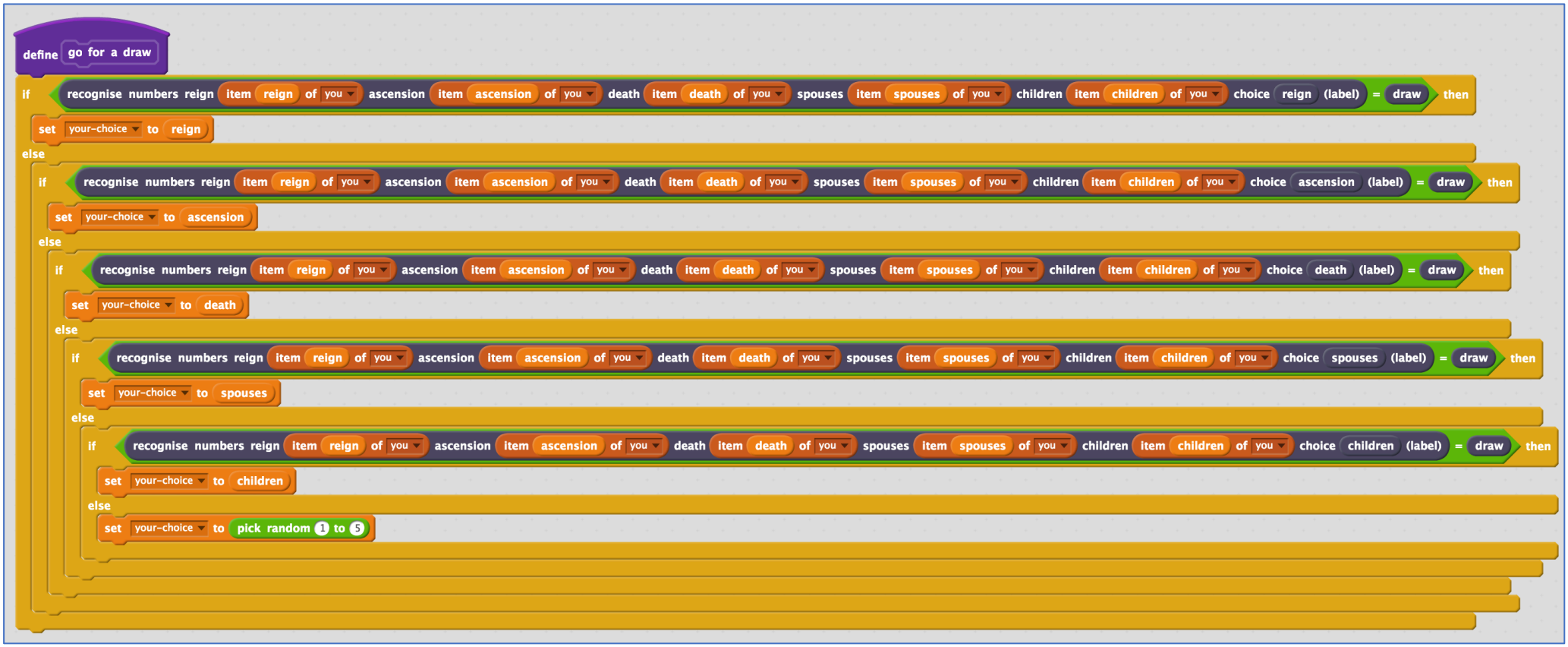
Step 7 – Add "go for a draw" to the last else in "Let the computer choose which attribute to play"





The final script looks like this (part 1):

The final script looks like this (part 2):



That's it – well done!

It might be good to get someone to double-check your script at this point. It is easy to make a mistake.

34. Click the **fullscreen** button, and then click on the **green flag** again. Play the game again.
Make a note of the score.
How good is the computer after learning from just five examples of each possible outcome?

35. Save your project.
*Click on **File** -> **Save Project***

What have you done so far?

You've modified your Scratch Top Trumps bot to use machine learning instead of your earlier random approach.

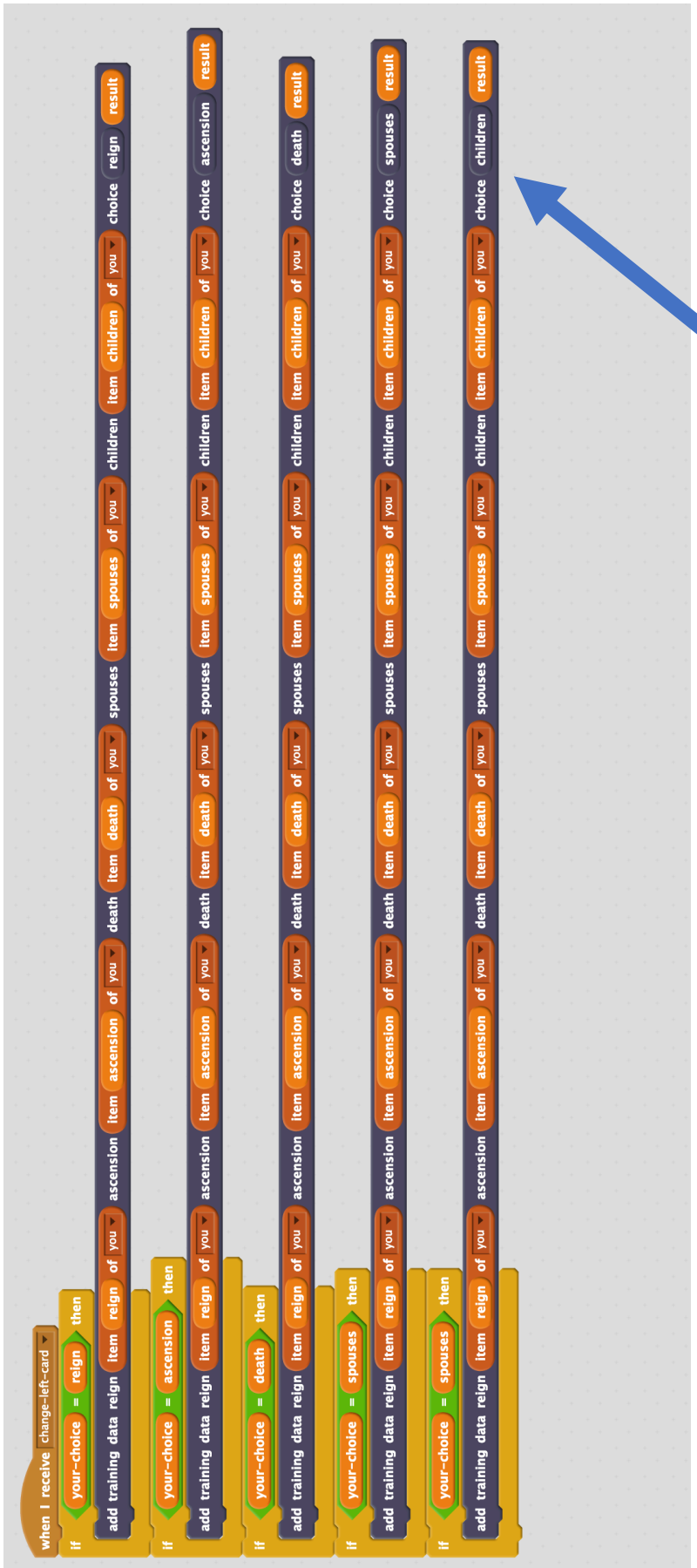
You haven't collected nearly enough examples to train a good model yet. The computer won't have seen enough examples of the game being played to have learned the types of values to expect, or the values that are more likely to win. Its predictions will often be wrong.

It needs more examples. Lots more examples.

Collecting, writing down, and then typing in the training examples was very slow. You probably don't want to have to do that a lot more.

Next, you'll modify the game to collect new training examples as part of playing the game. This will mean the more you play, the more examples the computer will have to learn from, and the better it should get.

36. Still in the “Stage”, in the same place as the scripts we wrote before, add the following script.



After every hand, the outcome of whether you won, lost or it was a draw will be added to the training data.

Choices should use the dark blue blocks from the “More Blocks” tab which represent the multiple-choice options

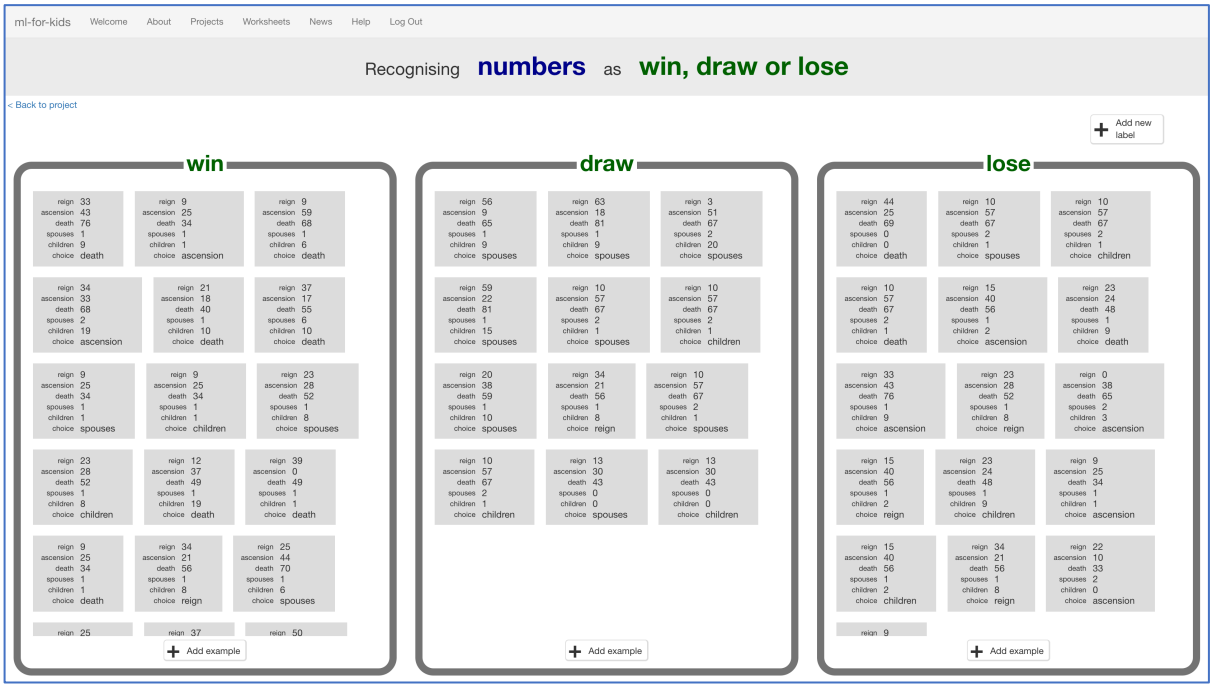
37. Save your project.
*Click on **File** -> **Save Project***

38. Click the **fullscreen** button, and then click the **green flag** again.
Play the game again for a while to collect training examples.
Make a note of the score.
The longer you play the game, the more examples the computer will have to learn from.

39. **Close** the Scratch window.

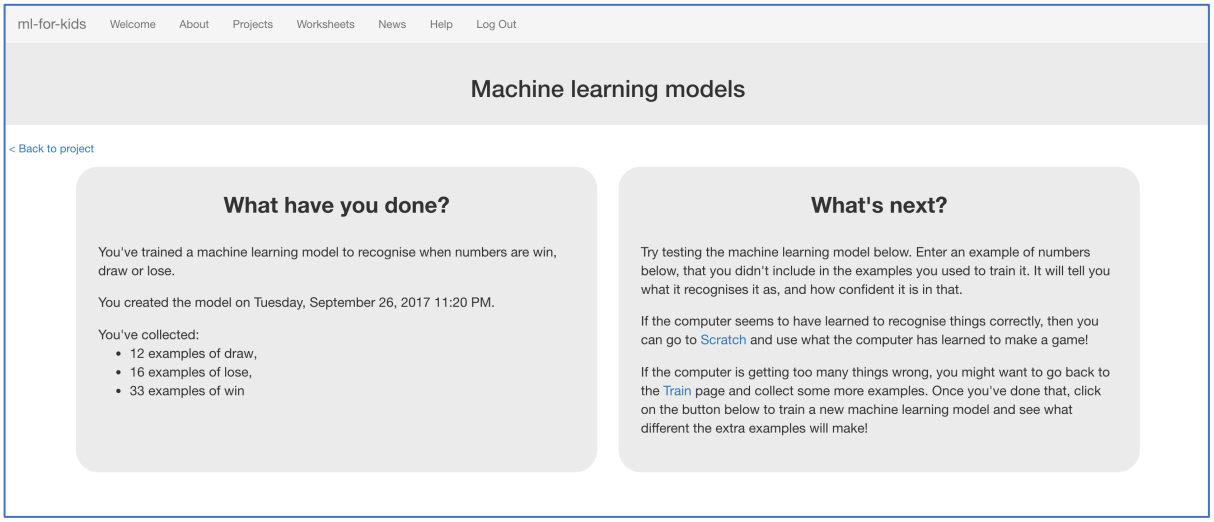
40. Click the "**< Back to project**" link and then click the "**Train**" button.

41. You should see examples from your game in the training buckets.



42. Click the **"< Back to project**" link and then click on the "**Learn & Test**" button.

43. Click on the "**Train new machine learning model**" button.



44. Click on the **"< Back to project**" link and then click the "**Scratch**" button. Click the "**Open in Scratch**" button.

45. Open your saved project again.
*Use **File** -> **Load Project***

46. Play the game again.
Is it getting any better? Does the computer win more often now?

47. Repeat steps 38 – 46 to collect more examples, and then train a new machine learning model with them.
You need to close the Scratch window every time you train a machine learning model for Scratch to start using the new trained model.

What have you done?

You've modified your Scratch Top Trumps bot to use machine learning.

Instead of training the bot being a separate thing, you've made the computer learn by playing the game. This means you don't need to wait for the computer to have learned before it can start playing. It can start playing (even if it loses a lot at first), straight away. And by playing the game, it will learn from those experiences how to get better.

You haven't told the computer what to do, but allowed it to try out different choices and discover what choices are more likely to help it to win.

This is called "reinforcement learning". When it makes a good choice, this is reinforced by the computer being told that it has won.

An example of training a Top Trumps bot

Your results will be different to this.

But these were the results I got from training my bot.

	Score	
	Human	**Computer**
No training – computer choosing at random	72	28
Trained with 100 examples	47	53
Trained with 200 examples	38	62
Trained with 300 examples	29	71
Trained with 400 examples	25	75
Trained with 500 examples	27	73
Trained with 600 examples	29	71
Trained with 700 examples	27	73

In general, more training is better.

There were times where the computer did worse after more training. Why do you think that was?

After a certain point, the computer's scores stopped improving, even after I kept adding more and more training.
Why do you think that was?

Compare these results with the results from your bot. How has your bot learned from the training you've given it?