

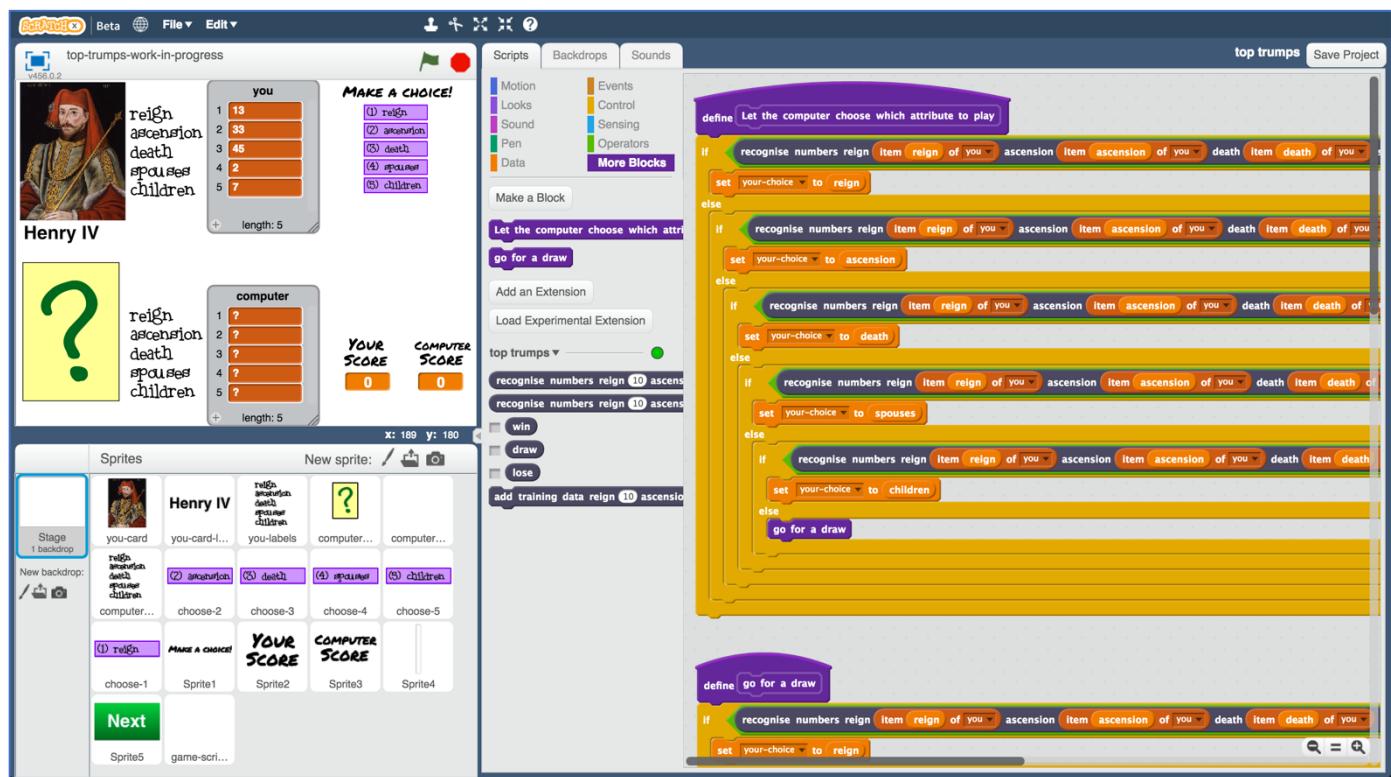
# Top Trumps

In this project you will train a computer to play a card game like “Top Trumps”.

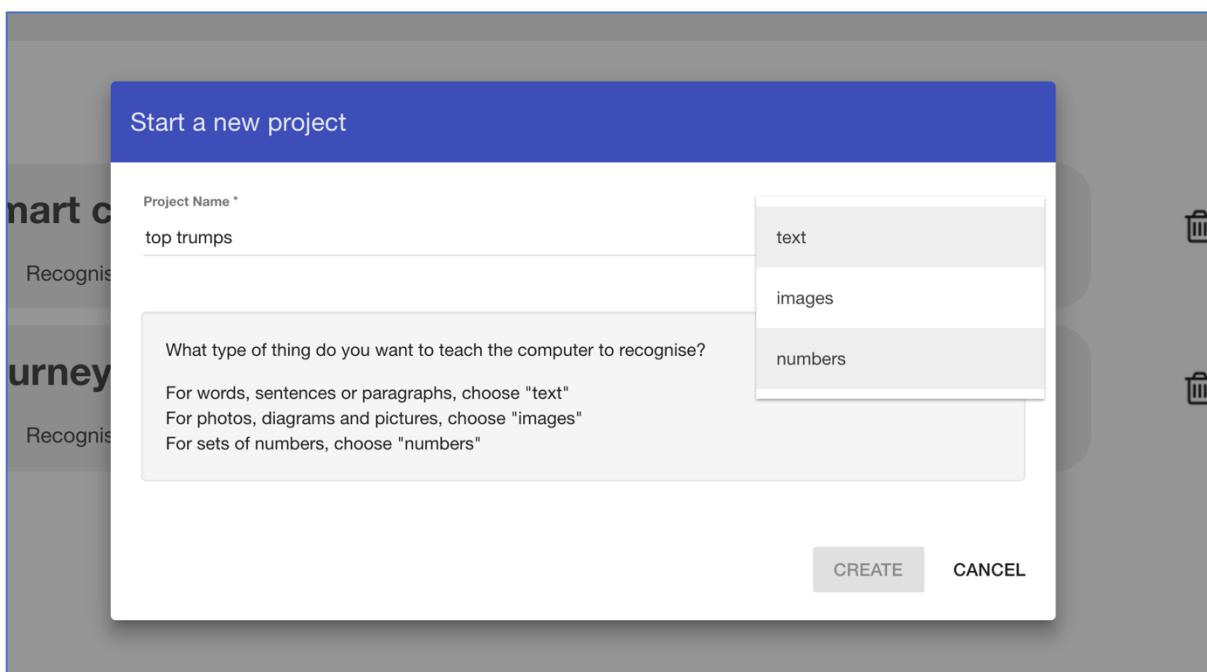
For some values on the cards, you win by having the highest number. For others, you win by having the lowest. The range of numbers for different values will vary.

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. Set “how many numbers” to 6.

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

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

- \* reign
- \* ascension
- \* death
- \* spouses
- \* children
- \* choice

*It should look like the screenshot below when you click on “Create”.*

*This won’t make any sense to you yet.*

*Don’t worry. It will in a moment!*

The screenshot shows a user interface for creating a new project. At the top, it says "Start a new project". Below that, there are two input fields: "Project Name \*" with "top trumps" typed in, and "Recognizing \*" with "numbers" selected from a dropdown. Underneath, it asks "How many numbers in each example?" with the number "6" chosen from a dropdown. There are six text input fields labeled "Field 1 name \*", "Field 2 name \*", "Field 3 name \*", "Field 4 name \*", "Field 5 name \*", and "Field 6 name \*". The first five fields contain "reign", "ascension", "death", "spouses", and "children" respectively. The sixth field, "Field 6 name \*", contains "choice" and is highlighted with a blue underline. At the bottom, there is a note: "Each example for numbers projects can contain a set of numbers. For example, if you want to train the computer to recognise the time of year based on weather data, you might want a few numbers - for temperature, amount of rain, and windspeed." Finally, at the bottom right, there are "CREATE" and "CANCEL" buttons.

- 9.** You should now see “**top trumps**” show up in the list of your projects. Click on it.

Your machine learning projects

**top trumps**  
Recognising **numbers**

**smart classroom**  
Recognising **text** as **fan\_on, fan\_off or 2 other classes**

**journey to school**

+ Add a new project

- 10.** 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.*

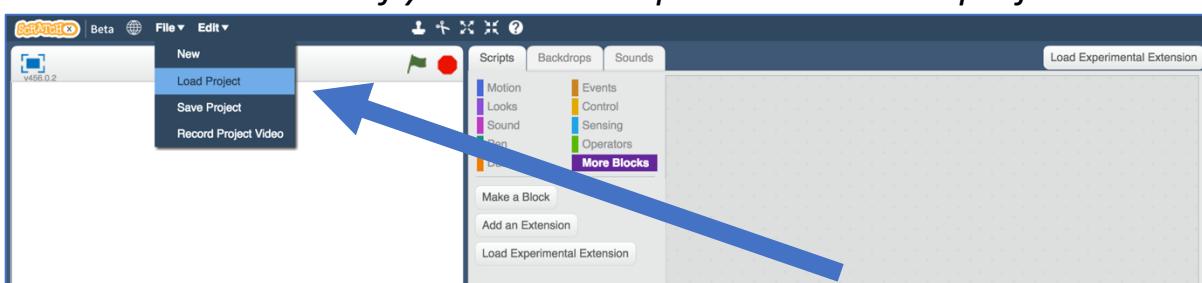
"top trumps"

**Train**  
Collect examples of what you want the computer to recognise.

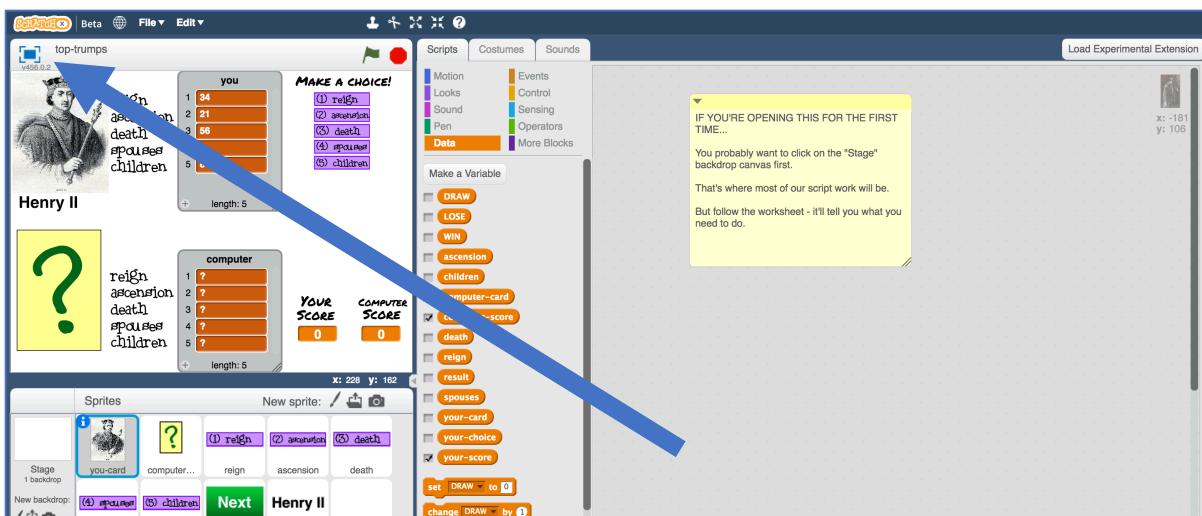
**Learn & Test**  
Use the examples to train the computer to recognise numbers.

**Scratch**  
Use the machine learning model you've trained to make a game in Scratch.

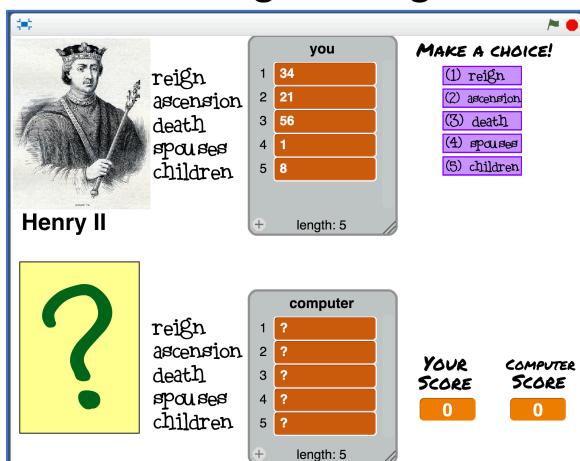
- 11.** Load the top-trumps.sbx file  
Use **File -> Load Project** as shown below  
Click **OK** when it asks if you want to replace the current project.



**12.** This is Top Trumps based on the Kings and Queens of England.  
Click the full-screen button.



**13.** 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 became King quicker than the computer's card, I'll win)

3) He died when he was **56**.

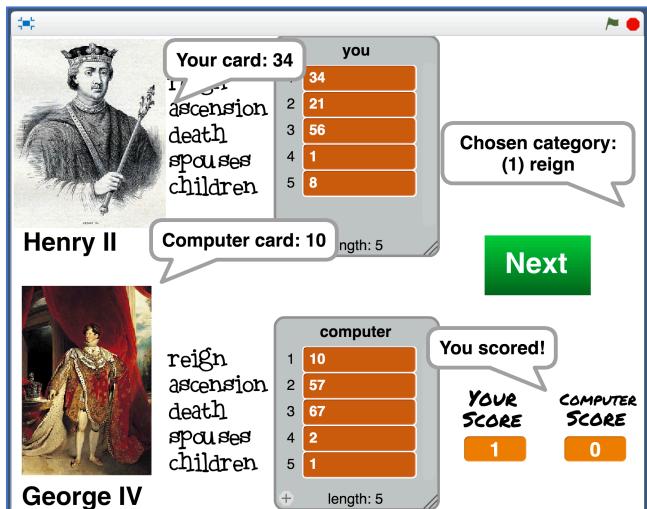
(If I choose this and he lived longer than the computer's card, I'll win)

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.**

## 14. Play a few rounds of the game against the computer.

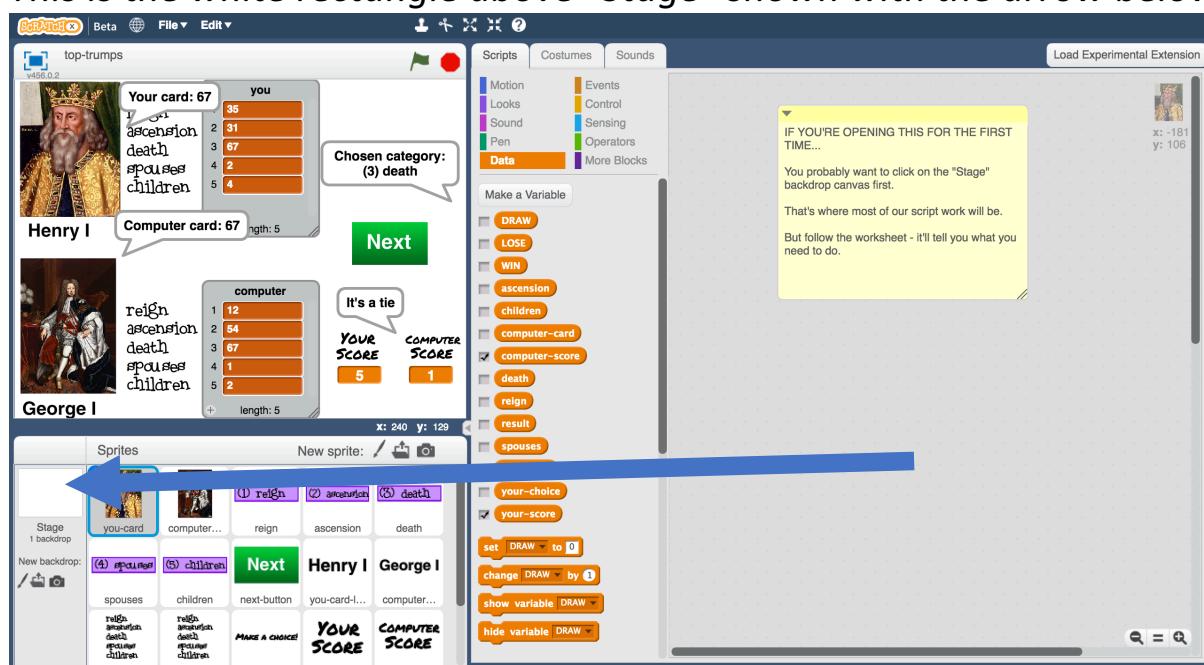
*Try to work out how the computer is choosing values to play.*

*When you think you've worked out how the computer is playing, move onto the next step.*

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

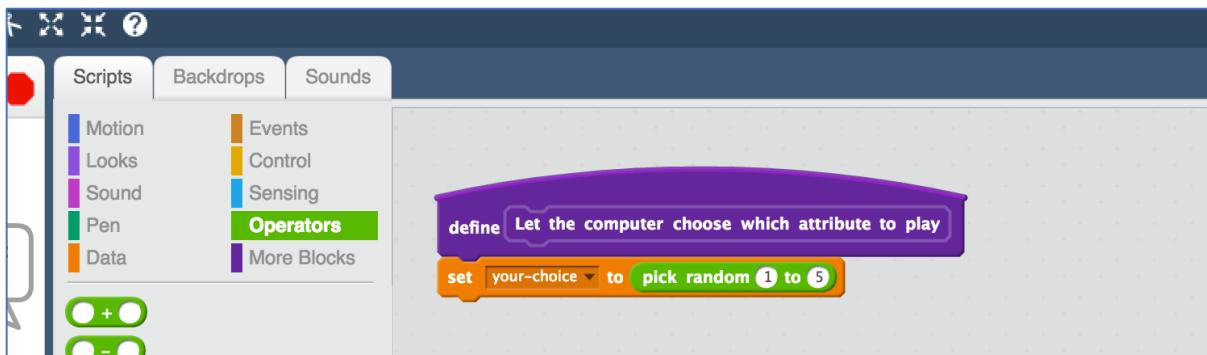
Then click on the **Stage**

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



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

**17.** 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.*



**18.** 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 we 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.

## 19. 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 attribute did you pick?
reign	ascension	death	spouses	children	
23	24	48	1	9	ascension (2)

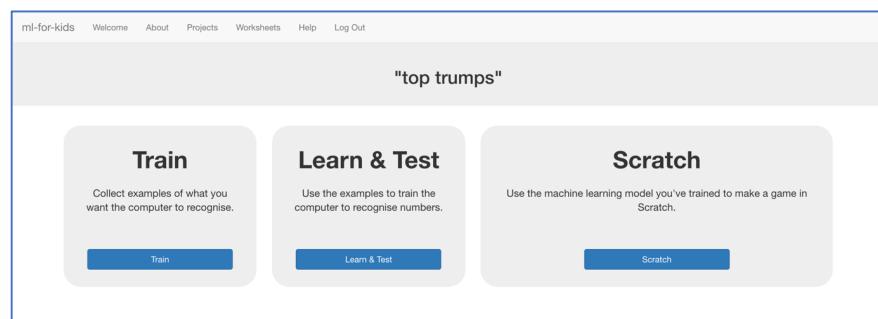
Examples where it was a **draw**

The values from your card					which attribute did you pick?
reign	ascension	death	spouses	children	
21	36	58	1	9	spouses (4)

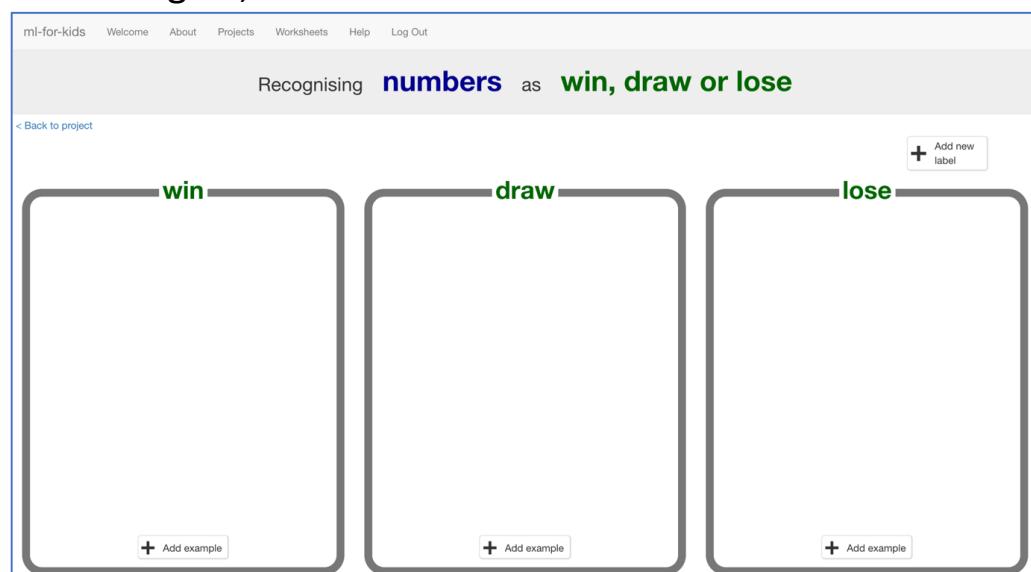
Examples where you **lost**, and the computer won

The values from your card					which attribute did you pick?
reign	ascension	death	spouses	children	
13	38	51	1	3	<i>death (3)</i>

- 20.** Close Scratch. Click the “< Back to project” link.  
Click on the **Train** button.



- 21.** 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”.



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

*For “choice”, type in the number (1 – 5) not the word that goes with it.*

The screenshot shows a web application titled "Recognising numbers as win, draw or lose". At the top, there are navigation links: ml-for-kids, Welcome, About, Projects, Worksheets, Help, and Log Out. Below the title, there are three main buckets: "win", "draw", and "lose". The "win" bucket contains several examples of numbers separated by commas: reign 23, ascension 24, death 48, spouses 1, children 9, choice 2; reign 33, ascension 43, death 76, spouses 1, children 9, choice 1; reign 21, ascension 18, death 40, spouses 1, children 10, choice 5; reign 63, ascension 18, death 81, spouses 1, children 9, choice 3; reign 37, ascension 17, death 55, spouses 6, children 10, choice 4; reign 23, ascension 24, death 55, spouses 1, children 9, choice 4. The "draw" bucket contains: reign 21, ascension 36, death 58, spouses 1, children 9, choice 4; reign 63, ascension 18, death 81, spouses 1, children 9, choice 4; reign 10, ascension 57, death 67, spouses 2, children 1, choice 5; reign 23, ascension 28, death 52, spouses 1, children 8, choice 4. The "lose" bucket contains: reign 13, ascension 38, death 51, spouses 1, children 3, choice 3; reign 0, ascension 38, death 65, spouses 2, children 3, choice 1; reign 13, ascension 33, death 45, spouses 2, children 7, choice 5; reign 10, ascension 57, death 67, spouses 1, children 1, choice 2; reign 12, ascension 54, death 67, spouses 1, children 2, choice 1. In the center, a modal window titled "Add new example" is open, prompting the user to "Enter an example of 'win'". It has four input fields: "reign", "ascension", "death", and "spouses", each with a corresponding text input field below it. At the bottom of the modal are "ADD" and "CANCEL" buttons. There is also a "+ Add new label" button in the top right corner of the modal.

**23.** 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.

The screenshot shows the same web application interface. The "draw" bucket now contains examples from the original "win" table: reign 21, ascension 36, death 58, spouses 1, children 9, choice 4; reign 63, ascension 18, death 81, spouses 1, children 9, choice 4; reign 10, ascension 57, death 67, spouses 2, children 1, choice 5; reign 23, ascension 28, death 52, spouses 1, children 8, choice 4. The "lose" bucket now contains examples from the original "draw" table: reign 13, ascension 38, death 51, spouses 1, children 3, choice 3; reign 0, ascension 38, death 65, spouses 2, children 3, choice 1; reign 13, ascension 33, death 45, spouses 2, children 7, choice 5; reign 10, ascension 57, death 67, spouses 1, children 1, choice 2; reign 12, ascension 54, death 67, spouses 1, children 2, choice 1. The "win" bucket remains the same as in the previous screenshot.

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

**25.** 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.*

The screenshot shows a web page titled "Machine learning models". At the top, there is a navigation bar with links: ml-for-kids, Welcome, About, Projects, Worksheets, Help, and Log Out. Below the title, there is a link "< Back to project". The main content area is divided into two sections: "What have you done?" and "What's next?". The "What have you done?" section contains text about collecting examples and a list of items: "• 5 examples of draw, • 5 examples of lose, • 5 examples of win". The "What's next?" section contains text about being ready to start training and a button labeled "Train new machine learning model". At the bottom left, there is a box labeled "Info from training server:" which is currently empty. A small note at the bottom right says "(Or go back to the Train page if you want to collect some more examples first.)".

**26.** 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 type in which value you would choose (1 for reign, 2 for ascension, 3 for death, 4 for spouses, 5 for children) in the ‘choice’ box.*

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

## Machine learning models

< Back to project

### What have you done?

You've trained a machine learning model to recognise when numbers is win, draw or lose.

You created the model on .

You've collected:

- 5 examples of draw,
- 5 examples of lose,
- 5 examples of win

### What's next?

Try testing the machine learning model below. Enter an example of numbers below, that you didn't include in the examples you used to train it. It will tell you what it recognises it as, and how confident it is in that.

If the computer seems to have learned to recognise things correctly, then you can go to [Scratch](#) and use what the computer has learned to make a game!

If the computer is getting too many things wrong, you might want to go back to the [Train](#) page and collect some more examples. Once you've done that, click on the button below to train a new machine learning model and see what different the extra examples will make!

Try putting in some numbers to see how it is recognised based on your training.

reign  
ascension  
death  
spouses  
children  
choice


Test

Info from training server:

Model started training at:  
Current model status: Available

Delete this model

Train new machine learning model

## 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.

## 27. 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 can check back on how to use them.*

Your project will add these blocks to the **More Blocks** tab in Scripts.

`recognise numbers reign 1 ascension 2 death 3 spouses 4 children 5 choice 6 dabeli`  
Put numbers in the input for this, and it will return the label that your machine learning model recognises it as.

`recognise numbers reign 1 ascension 2 death 3 spouses 4 children 5 choice 6 (confidence)`  
This will return how confident your machine learning model is that it recognises the type of numbers. (As a number from 0 - 100).

`win draw lose`  
These blocks represent the labels you've created in your project, so you can use their names in your scripts.

This means you can do something like this:

`if [recognise numbers reign 1 ascension 2 death 3 spouses 4 children 5 choice 6 (label) != win] then  
[say [I think that was win]]`

It will look something like this - except with the name of your project.

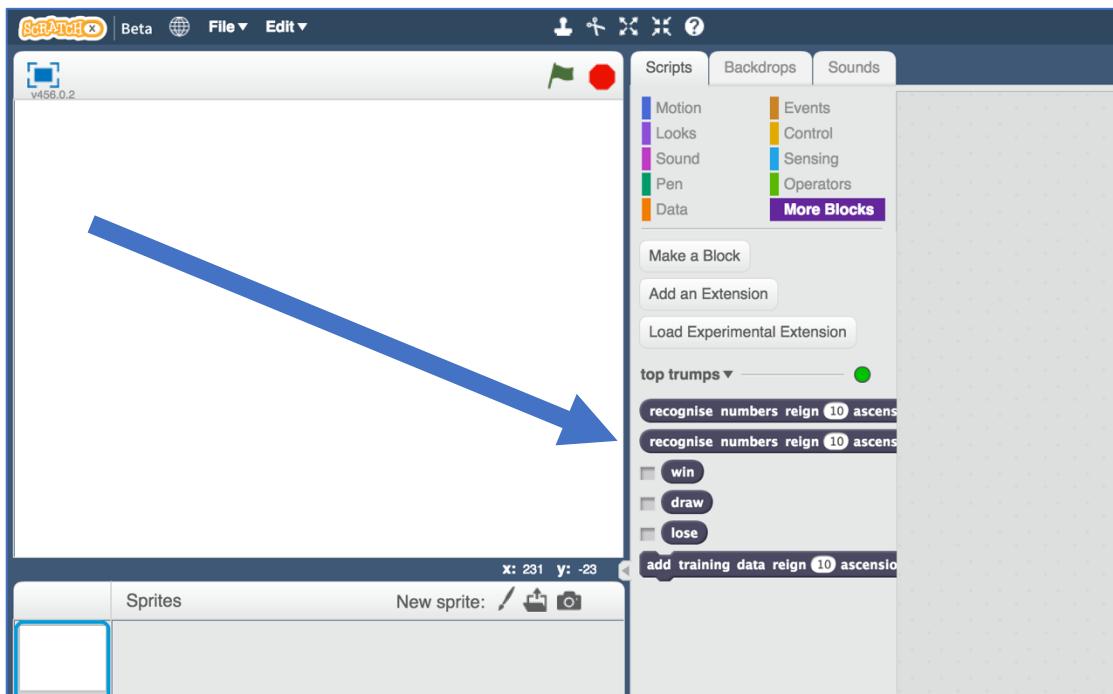
The coloured circle next to your project name tells you if your machine learning model is okay.

- Green means your model is trained and ready to go
- Yellow means your model hasn't finished training yet
- Red means something went wrong. Go back to the [Learn & Test](#) page to see what went wrong with training.

[Open in Scratch](#)

## 28. Click on 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.*



**29.** Load the top-trumps.sbx Scratch project you opened before.

*Click on **File** -> **Load Project***

**30.** Click on the **Stage** (as you did before) to go back to the Script for how the computer chooses it's moves.

**31.** 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 it's 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)

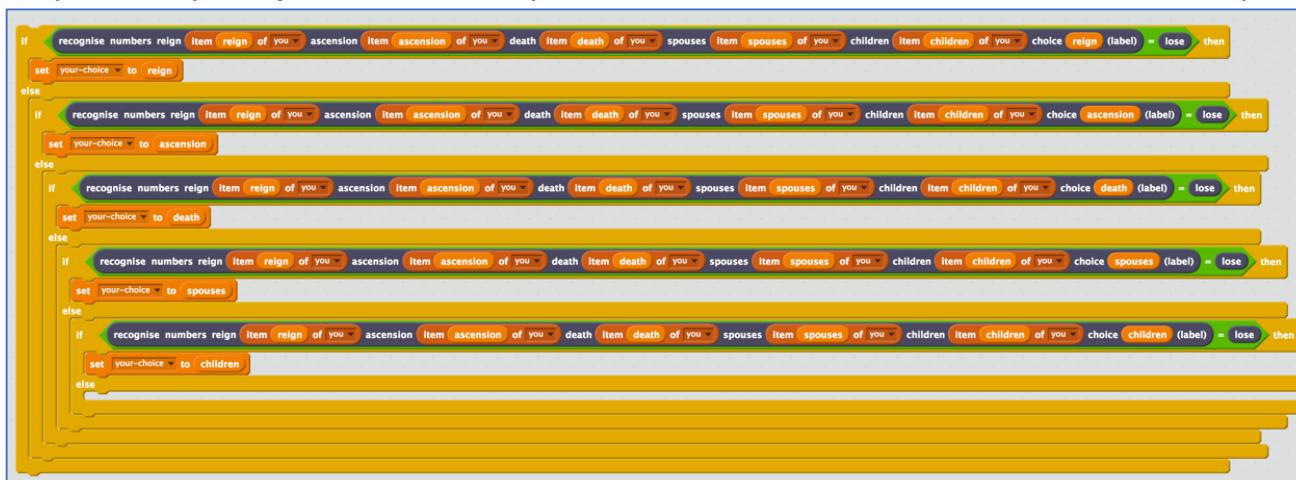


## Step 2 – Duplicate that. Put the copy into the else block.

Change the choice and the set-your-choice value from “reign” to “ascension”. You should end up with this:



## Step 3 – Repeat for “death”, “spouses” and “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”*

```
define go for a draw
  if [recognise numbers reign item reign of you v ascension item ascension of you v death]
    set [your-choice v] to [reign]
  else
    if [recognise numbers reign item reign of you v ascension item ascension of you v dead]
      set [your-choice v] to [ascension]
    else
      if [recognise numbers reign item reign of you v ascension item ascension of you v death]
        set [your-choice v] to [death]
      else
        if [recognise numbers reign item reign of you v ascension item ascension of you v spouses]
          set [your-choice v] to [spouses]
        else
          if [recognise numbers reign item reign of you v ascension item ascension of you v children]
            set [your-choice v] to [children]
          else
            set [your-choice v] to [pick random 1 to 5]
```

**Step 5**

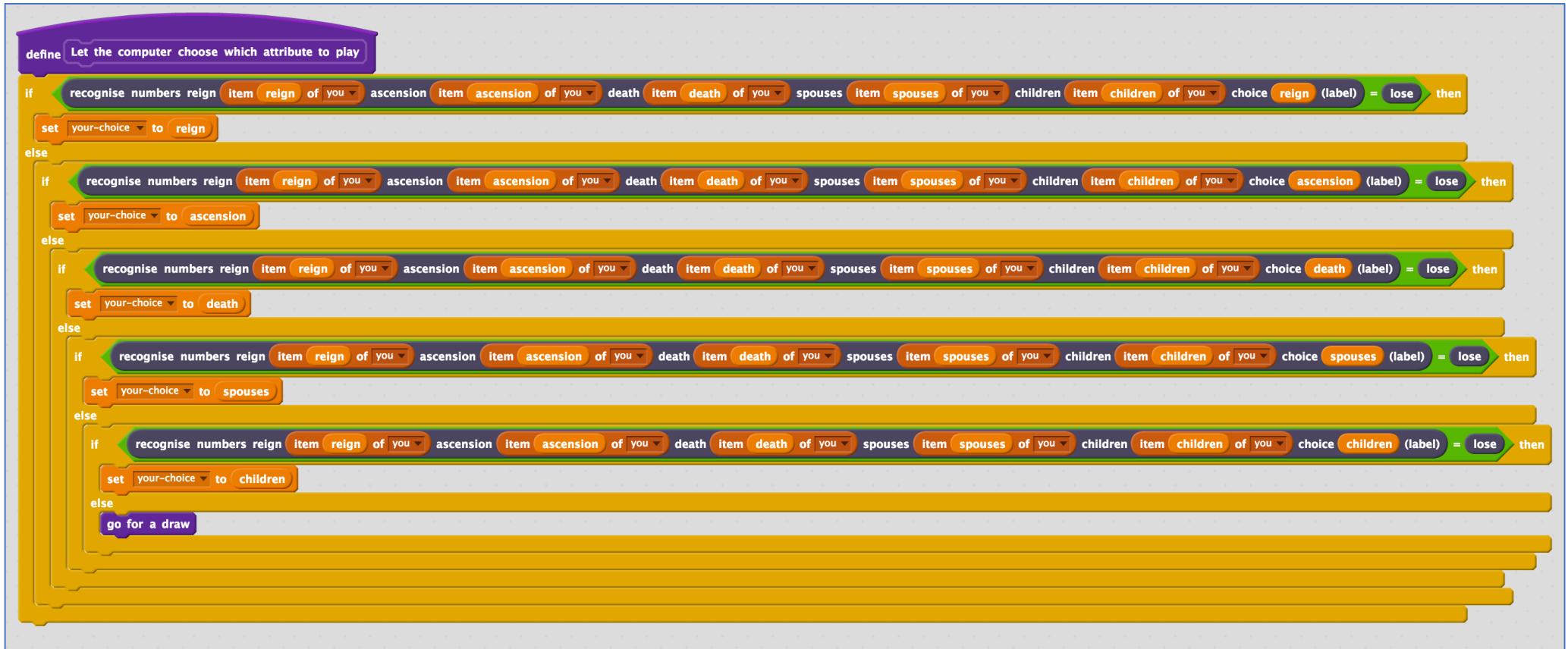
```
define Let the computer choose which attribute to play
  if [recognise numbers reign item reign of you v ascension item ascension]
    set [your-choice v] to [reign]
  else
    if [recognise numbers reign item reign of you v ascension item ascension]
      set [your-choice v] to [ascension]
    else
      if [recognise numbers reign item reign of you v ascension item ascension]
        set [your-choice v] to [death]
      else
        if [recognise numbers reign item reign of you v ascension item ascension]
          set [your-choice v] to [spouses]
        else
          if [recognise numbers reign item reign of you v ascension item ascension]
            set [your-choice v] to [children]
          else
            go for a draw
```

**Step 6**

```
define go for a draw
  if [recognise numbers reign item reign of you v ascension item ascension]
    set [your-choice v] to [reign]
  else
    if [recognise numbers reign item reign of you v ascension item ascension]
      set [your-choice v] to [ascension]
    else
      if [recognise numbers reign item reign of you v ascension item ascension]
        set [your-choice v] to [death]
      else
        if [recognise numbers reign item reign of you v ascension item ascension]
          set [your-choice v] to [spouses]
        else
          if [recognise numbers reign item reign of you v ascension item ascension]
            set [your-choice v] to [children]
          else
            go for a draw
```

**Step 7**

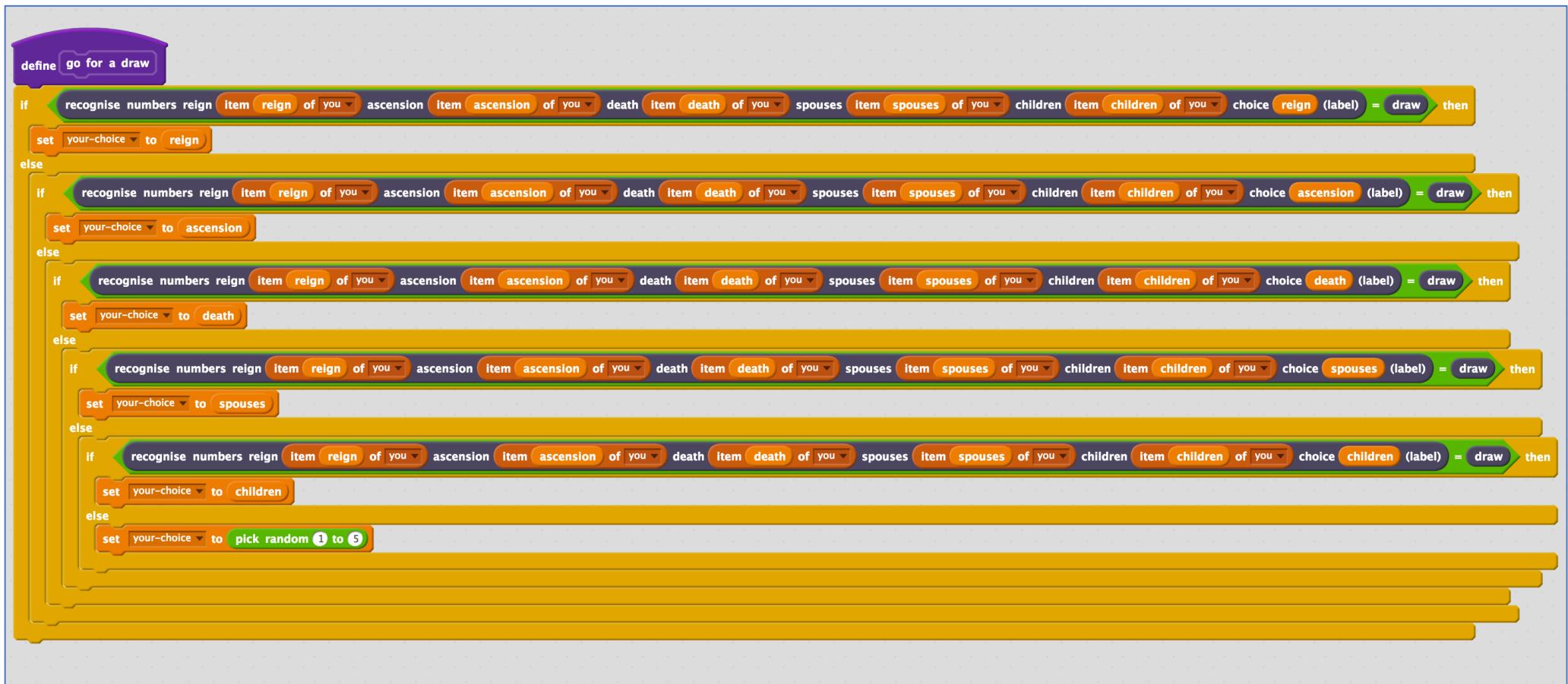
The final script looks like this (part 1):



A Scratch script titled "Let the computer choose which attribute to play". The script uses a "define" block to set up a loop that repeatedly asks the computer to choose between four attributes: reign, ascension, death, or spouses. If the computer's choice matches one of these attributes, it sets the variable "your-choice" to that attribute and then ends the loop. If the computer's choice does not match any of the four, it goes for a draw.

```
define Let the computer choose which attribute to play
  if [recognise numbers reign item reign of you v ascension item ascension of you v death item death of you v spouses item spouses of you v children item children of you v choice reign (label) = lose] then
    set [your-choice v] to [reign]
  else
    if [recognise numbers reign item reign of you v ascension item ascension of you v death item death of you v spouses item spouses of you v children item children of you v choice ascension (label) = lose] then
      set [your-choice v] to [ascension]
    else
      if [recognise numbers reign item reign of you v ascension item ascension of you v death item death of you v spouses item spouses of you v children item children of you v choice death (label) = lose] then
        set [your-choice v] to [death]
      else
        if [recognise numbers reign item reign of you v ascension item ascension of you v death item death of you v spouses item spouses of you v children item children of you v choice spouses (label) = lose] then
          set [your-choice v] to [spouses]
        else
          if [recognise numbers reign item reign of you v ascension item ascension of you v death item death of you v spouses item spouses of you v children item children of you v choice children (label) = lose] then
            set [your-choice v] to [children]
          else
            go for a draw
  end
end
```

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's easy to make a mistake.

**32.** Click the **fullscreen** button, and then click on the **green flag** again.

Play the game again.

*Keep a note of your score.*

*How good is the computer after learning from just five examples of each possible outcome?*

**33.** 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.

**34.** 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 drawed will be added to the training data.*



*This is a bit long, so to make it easier to read...*



**35.** Save your project.

*Click on **File** -> **Save Project***

**36.** Click the **fullscreen** button, and then click on the **green flag** again.

*Play the game again for a while to collect training examples.*

*The longer you play, the more the computer will have to learn from.*

**37.** Close the Scratch window.

**38.** Click the “**< Back to project**” link and then click the “**Train**” button.

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

ml-for-kids   Welcome   About   Projects   Worksheets   Help   Log Out

Recognising **numbers** as **win, draw or lose**

< Back to project

**win**   **draw**   **lose**

+ Add new label

win	draw	lose									
reign 22 ascension 22 death 23 spouses 2 children 2 choice 4	reign 0 ascension 0 death 7 spouses 1 children 0 choice 3	reign 34 ascension 34 death 56 spouses 1 children 8 choice 2	reign 23 ascension 24 death 48 spouses 1 children 8 choice 2	reign 33 ascension 33 death 76 spouses 1 children 9 choice 4	reign 21 ascension 36 death 58 spouses 2 children 1 choice 1	reign 10 ascension 57 death 67 spouses 2 children 1 choice 2	reign 56 ascension 56 death 65 spouses 1 children 9 choice 4	reign 9 ascension 9 death 34 spouses 1 children 1 choice 3	reign 9 ascension 9 death 34 spouses 1 children 1 choice 1	reign 13 ascension 13 death 43 spouses 0 children 0 choice 1	reign 0 ascension 12 death 17 spouses 2 children 5 choice 1
reign 21 ascension 18 death 40 spouses 1 children 10 choice 5	reign 37 ascension 43 death 55 spouses 1 children 10 choice 4	reign 33 ascension 43 death 76 spouses 1 children 9 choice 1	reign 63 ascension 18 death 81 spouses 1 children 9 choice 3	reign 13 ascension 13 death 45 spouses 2 children 7 choice 4	reign 23 ascension 24 death 48 spouses 2 children 9 choice 4	reign 33 ascension 33 death 76 spouses 1 children 9 choice 4	reign 63 ascension 18 death 81 spouses 1 children 9 choice 4	reign 9 ascension 44 death 68 spouses 0 children 6 choice 2	reign 44 ascension 44 death 69 spouses 0 children 0 choice 5	reign 6 ascension 6 death 15 spouses 0 children 0 choice 1	reign 17 ascension 17 death 48 spouses 2 children 5 choice 1
reign 5 ascension 5 death 42 spouses 1 children 0 choice 2	reign 9 ascension 5 death 68 spouses 1 children 6 choice 3	reign 23 ascension 23 death 52 spouses 1 children 8 choice 5	reign 34 ascension 34 death 68 spouses 2 children 19 choice 5	reign 19 ascension 19 death 43 spouses 1 children 4 choice 4	reign 39 ascension 39 death 49 spouses 1 children 4 choice 4	reign 59 ascension 59 death 81 spouses 1 children 15 choice 4	reign 17 ascension 20 death 48 spouses 2 children 5 choice 1	reign 20 ascension 20 death 59 spouses 1 children 5 choice 2	reign 17 ascension 17 death 51 spouses 1 children 3 choice 3	reign 20 ascension 20 death 59 spouses 1 children 10 choice 5	reign 0 ascension 0 death 0 spouses 0 children 0 choice 5
reign 0 ascension 0 death 77 spouses 1 children 0 choice 3	reign 35 ascension 35 death 67 spouses 2 children 4 choice 4	reign 13 ascension 13 death 59 spouses 0 children 0 choice 1	reign 44 ascension 44 death 69 spouses 0 children 0 choice 1	reign 23 ascension 28 death 43 spouses 1 children 10 choice 5	reign 19 ascension 39 death 43 spouses 1 children 4 choice 4	reign 59 ascension 59 death 81 spouses 1 children 15 choice 4	reign 17 ascension 17 death 48 spouses 2 children 5 choice 1	reign 20 ascension 20 death 48 spouses 2 children 5 choice 2	reign 13 ascension 13 death 51 spouses 0 children 3 choice 3	reign 0 ascension 5 death 42 spouses 2 children 0 choice 1	reign 12 ascension 54 death 67 spouses 0 children 2 choice 1
reign 3 ascension 3 death 47 spouses 2 children 20 choice 1	reign 12 ascension 12 death 47 spouses 1 children 10 choice 1	reign 20 ascension 20 death 59 spouses 1 children 4 choice 5	reign 35 ascension 35 death 57 spouses 2 children 4 choice 1	reign 17 ascension 17 death 48 spouses 1 children 5 choice 1	reign 9 ascension 9 death 43 spouses 1 children 0 choice 1	reign 21 ascension 21 death 40 spouses 1 children 1 choice 1	reign 44 ascension 44 death 69 spouses 0 children 4 choice 4	reign 13 ascension 13 death 58 spouses 1 children 5 choice 2	reign 13 ascension 13 death 51 spouses 1 children 3 choice 3	reign 0 ascension 10 death 42 spouses 2 children 1 choice 2	reign 5 ascension 5 death 42 spouses 2 children 0 choice 1
reign 23 ascension 23 death 48 spouses 1 children 9 choice 5	reign 17 ascension 17 death 48 spouses 1 children 5 choice 1	reign 9 ascension 9 death 41 spouses 1 children 0 choice 1	reign 21 ascension 21 death 40 spouses 1 children 10 choice 1	reign 19 ascension 19 death 43 spouses 1 children 5 choice 5	reign 38 ascension 38 death 65 spouses 1 children 3 choice 1	reign 6 ascension 6 death 15 spouses 2 children 0 choice 1	reign 19 ascension 19 death 57 spouses 1 children 5 choice 5	reign 10 ascension 57 death 57 spouses 2 children 1 choice 2	reign 12 ascension 54 death 67 spouses 0 children 2 choice 1	reign 0 ascension 0 death 0 spouses 0 children 0 choice 5	

+ Add example   + Add example   + Add example

## 40. Click the “< Back to project” link and then click the “Learn & Test” button.

## 41. Click on the “Train new machine learning model” button.

ml-for-kids   Welcome   About   Projects   Worksheets   Help   Log Out

Machine learning models

< Back to project

**What have you done?**

You've trained a machine learning model to recognise when numbers is win, draw or lose.

You created the model on .

You've collected:

- 22 examples of draw,
- 52 examples of lose,
- 64 examples of win

**What's next?**

Try testing the machine learning model below. Enter an example of numbers below, that you didn't include in the examples you used to train it. It will tell you what it recognises it as, and how confident it is in that.

If the computer seems to have learned to recognise things correctly, then you can go to [Scratch](#) and use what the computer has learned to make a game!

If the computer is getting too many things wrong, you might want to go back to the [Train](#) page and collect some more examples. Once you've done that, click on the button below to train a new machine learning model and see what different the extra examples will make!

## 42. Click on the “< Back to project” link and then click on the “Scratch” button. Click on the “Open in Scratch” button.

**43.** Open your saved project again.

*Use File -> Load Project*

**44.** Play the game again.

*Is it getting any better? Does the computer win more often now?*

**45.** Repeat steps 37 – 45 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 we 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?*