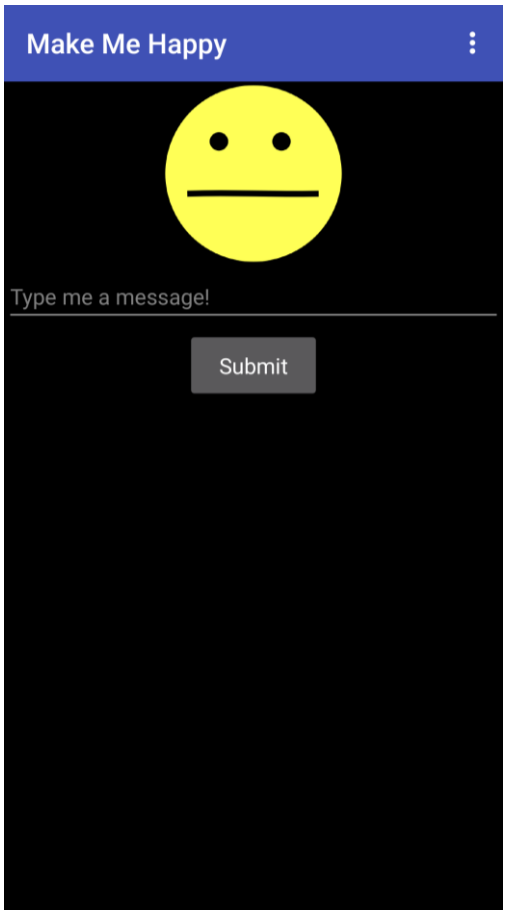
Make me happy (App Inventor)

In this project you will make a character that reacts to what you say.

If you compliment it, it will look happy.
If you insult it, it will look sad.

At first, you'll program a list of rules for what is kind and what is mean, and learn why that approach isn't very good.

Next, you will teach the computer to recognise kind messages and mean messages by giving it examples of each.



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

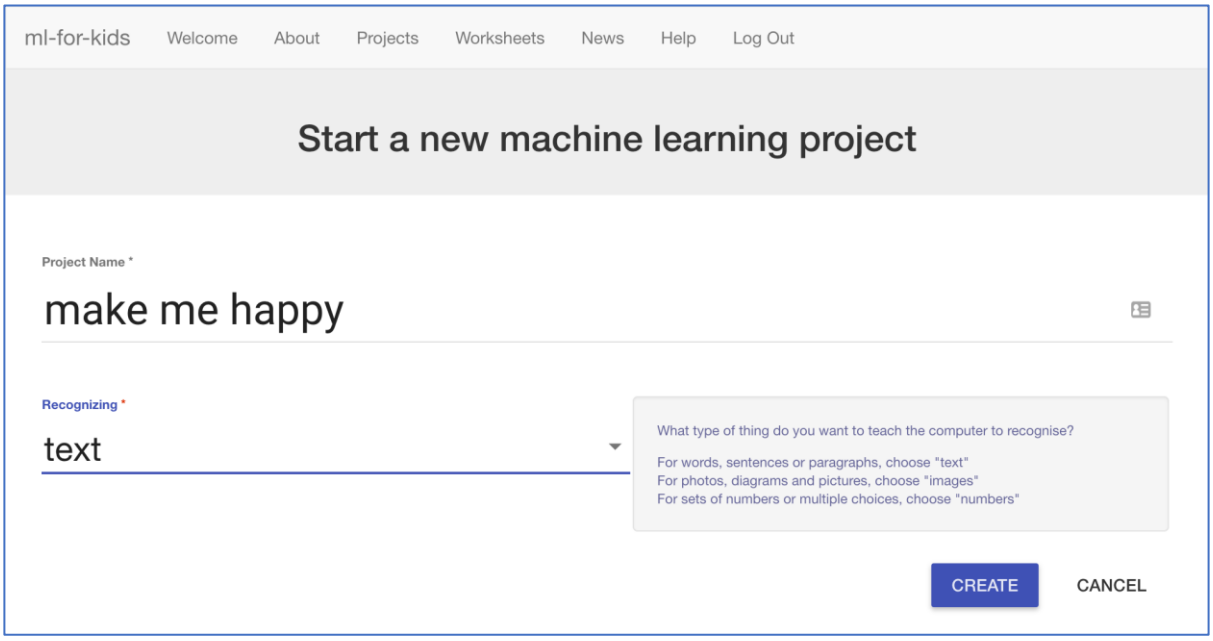
2. Click on "**Get started**"

3. Click on "**Log In**" and type in your username and password
If you don't have a username, ask your teacher 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.

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

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

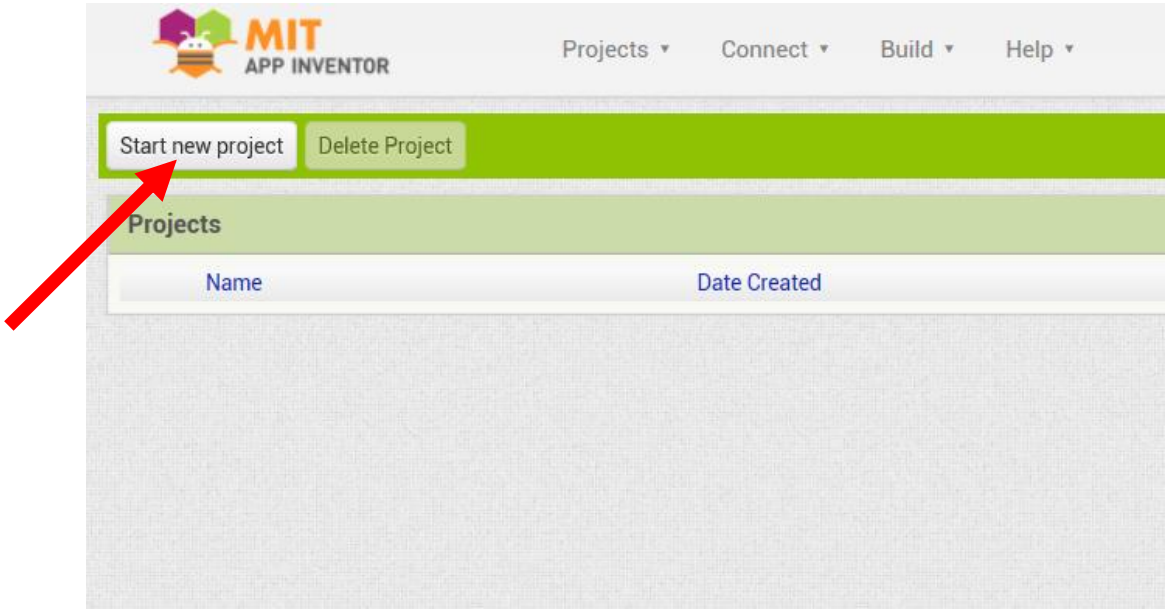
6. Name your project "make me happy" and set it to learn how to recognise "**text**".
Click the "**Create**" button



7. You should now see "**make me happy**" in the list of your projects. Click on it.

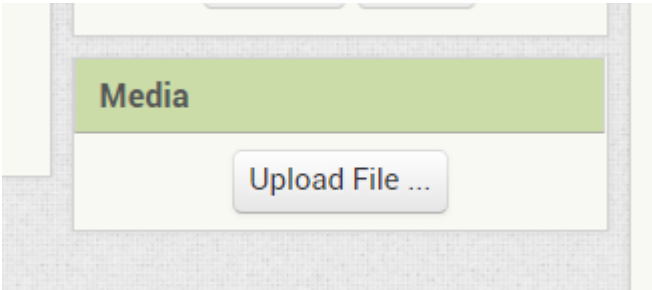
8. Start by getting a project ready in MIT App Inventor. Create an App Inventor project at http://ai2.appinventor.mit.edu

Don't have a Google account? Use http://code.appinventor.mit.edu

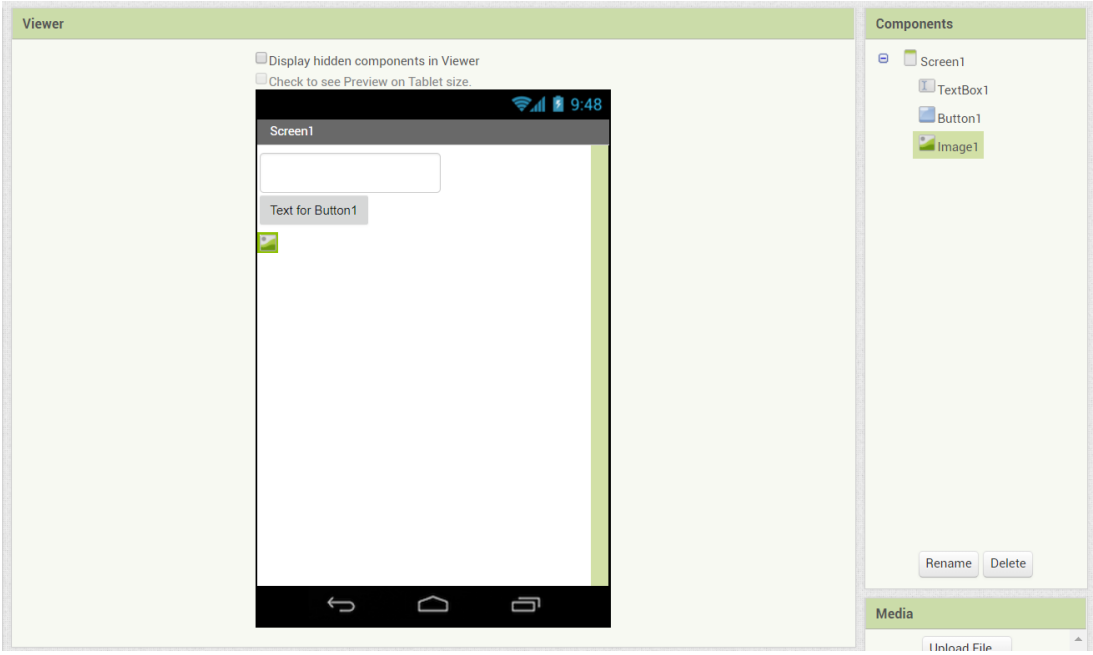


9. Upload a "Happy", "Sad", and "Not Sure" picture to the App Inventor project **Media** assets:

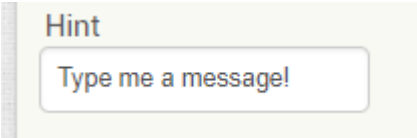
Visit Google Image search to find picture. See the ones below as an example.



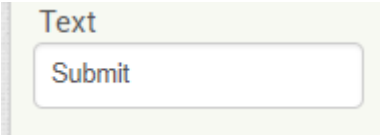
10. Drag a **TextBox**, a **Button**, and an **Image** object from the **Palette** to the **Viewer** to add them to your project.



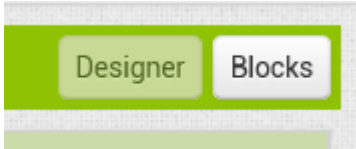
11. Click on **TextBox1** in the **Component** tree. This will reveal the **TextBox** properties on the right. Change the **Hint** property to *Type me a message!*

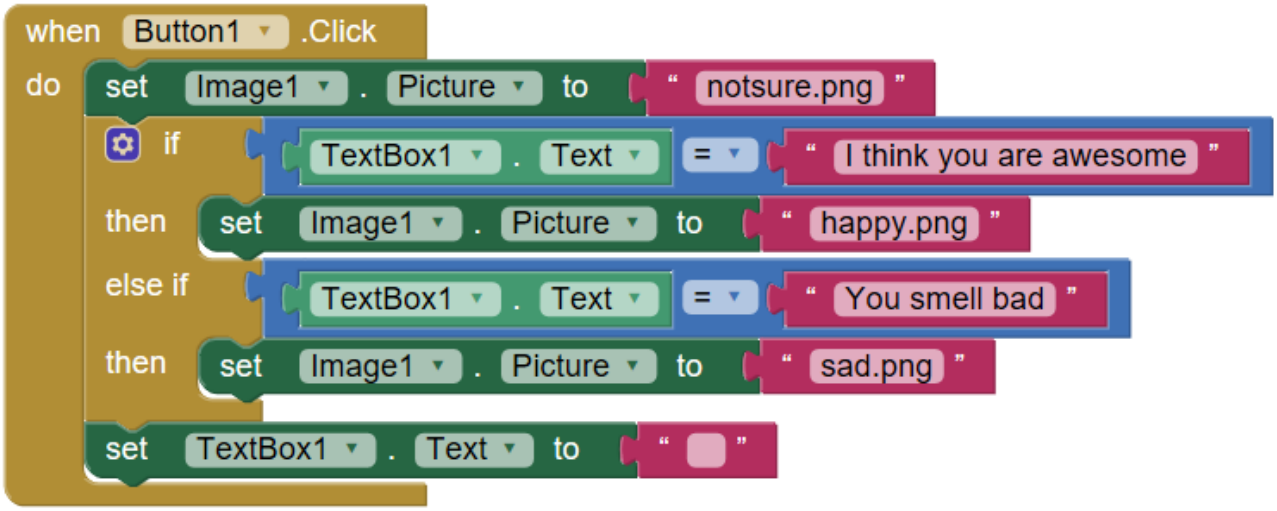


12. Click on **Button1** in the **Component** tree. Change the **Text** property to *Submit*.



13. Switch from Designer view to Blocks. Create the following algorithm for when the button is clicked.



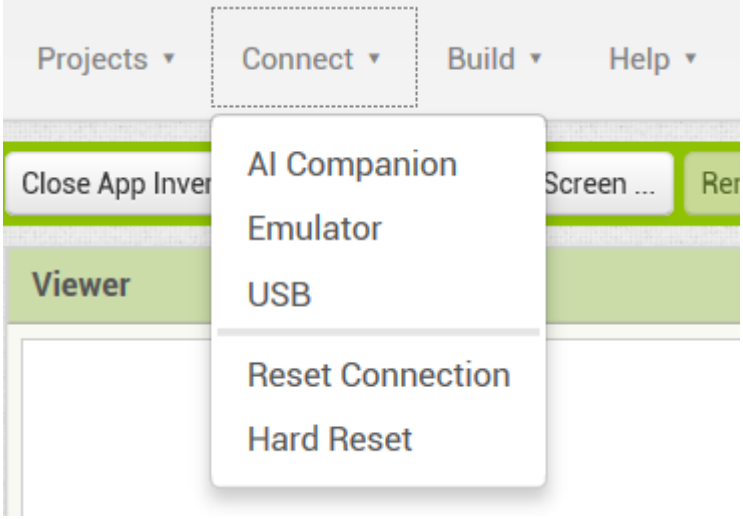


What you type to set the picture property of the image depends on your file's name and file extension.

14. Save your project.

*Click on **Projects** -> **Save Project** to save the project to your account.*
*Click on **Projects** -> **Export selected project (.aia) to my computer** to save the project file to your computer.*

15. Click the **Connect** to test your app with your preferred method.



16. Type in a message and watch it react!

Type "I think you are awesome" and click submit. The character smiles. Click the green flag again and type "You smell bad". The character is sad. Type anything else, and the character's face is the not sure face.

What have you done so far?

You've created a character that should react to what people type, and programmed it using a simple rules-based approach.

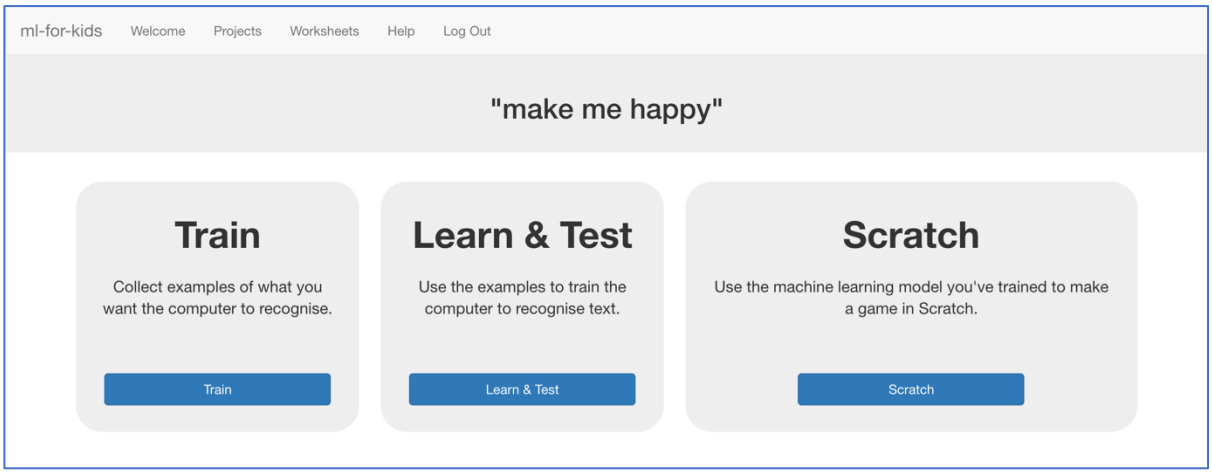
If you want it to react to other messages, you will need to add extra **if** blocks.

The problem with this is that you need to predict exactly what messages the character will receive. Making a list of every possible message would take forever!

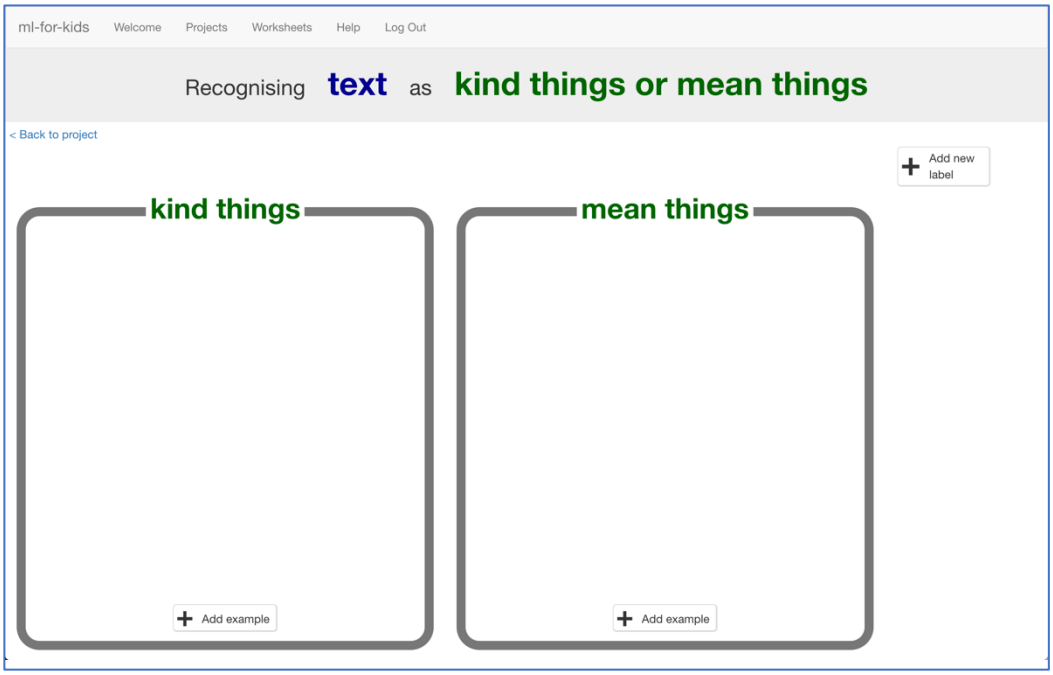
Next, we'll try a better approach – teaching the computer to recognise messages for itself.

17. Go back to Machine Learning for Kids to https://machinelearningforkids.co.uk/.

18. You need examples to train the computer. Click the **"< Back to project"** link. Then click the **Train** button.



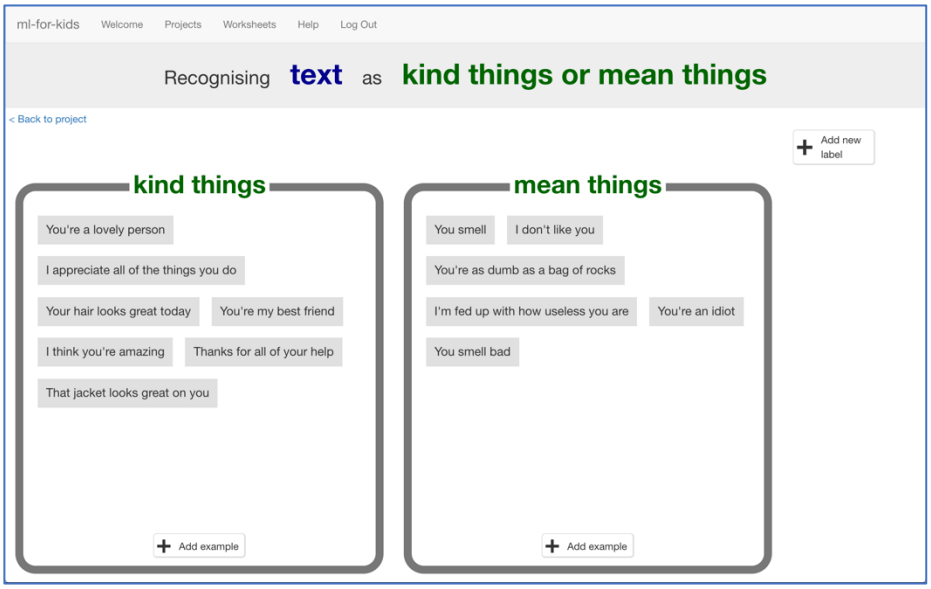
19. Click on **"+ Add new label"** and call it "kind things".
Do that again, and create a second bucket called "mean things".



20. Click the "**Add example**" button in the "kind things" bucket, and type in a kind message.

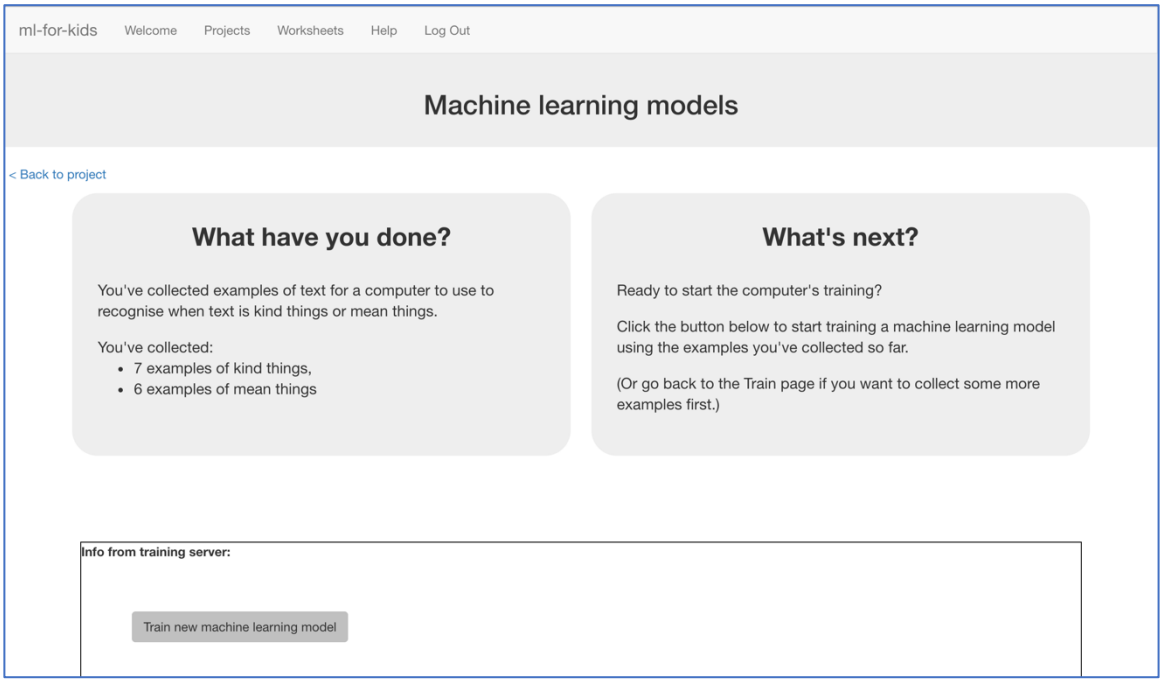
21. Click on the "**Add example**" button in the "mean things" bucket, and type in a mean message.

22. Repeat steps 20 and 21 until you've written at least **ten** examples of each.

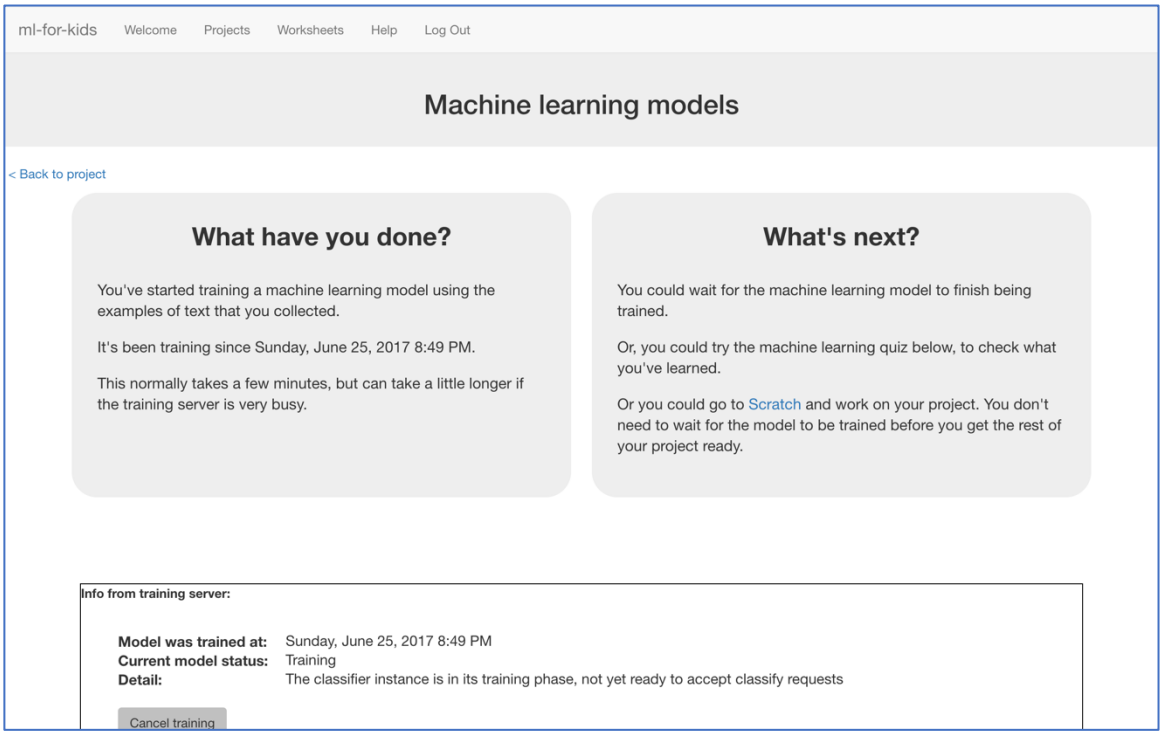


23. Click on the **"< Back to project"** link.
Then click on the "**Learn & Test**" button.

24. 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 messages from the examples you've given to it.

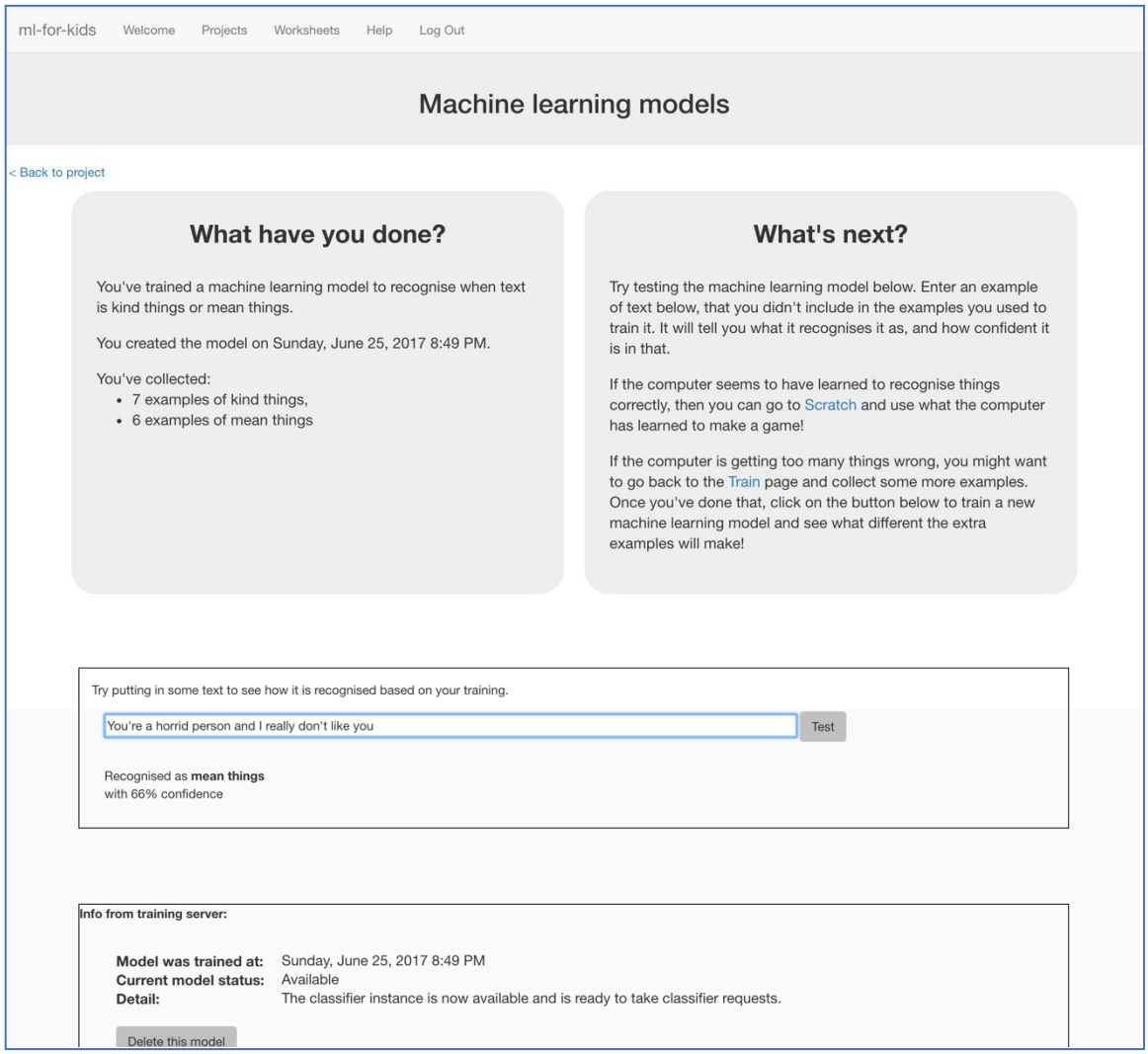


25. Wait for the training to complete. This might take a few minutes.



26. Once the training has completed, a Test box will be displayed.
Try testing your machine learning model to see what the computer has learned.
Type something kind, and press enter. It should be recognised as kind.
Type something mean, and press enter. It should be recognised as mean.

Test it with examples that you haven't shown the computer before.
If you're not happy with how the computer recognises the messages, go back to step 21, and add some more examples.
Make sure you repeat step 25 to train with the new examples though!



What have you done so far?

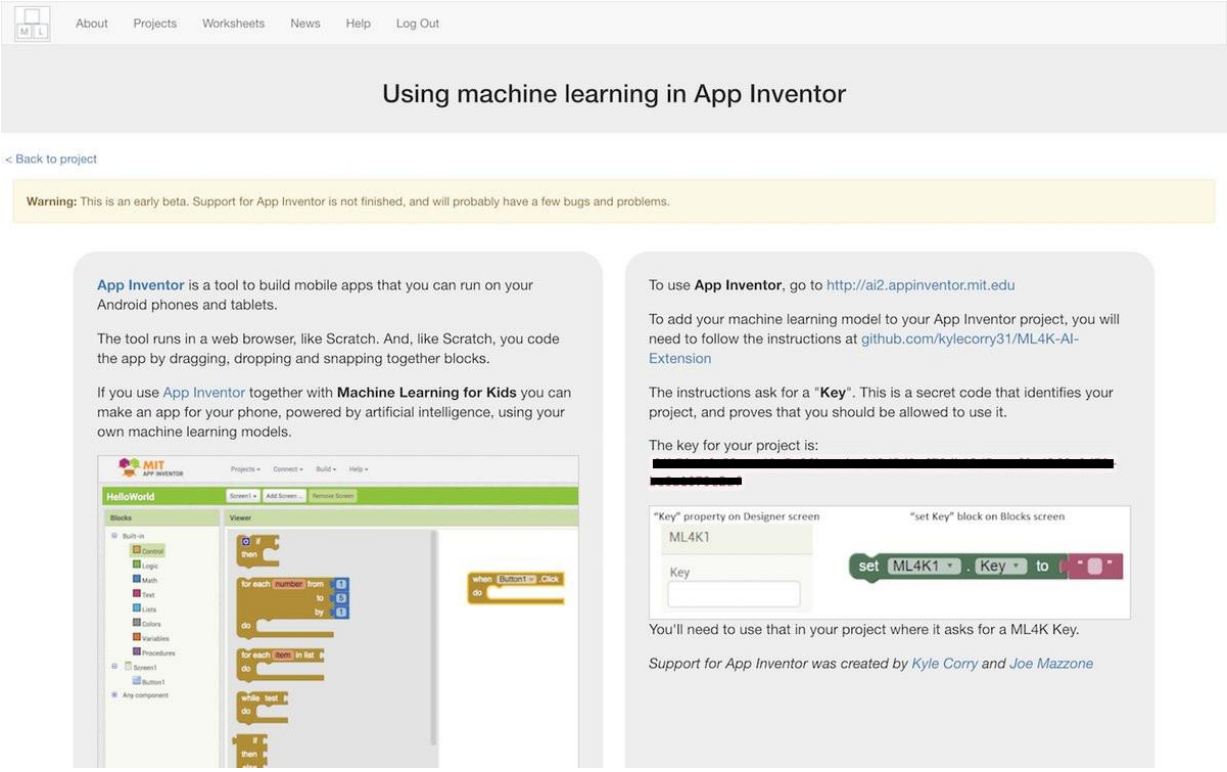
You've started to train a computer to recognise text as being kind or mean. Instead of trying to write rules to be able to do this, you are doing it by collecting examples. These examples are being used to train a machine learning "model".

This is called "supervised learning" because of the way you are supervising the computer's training.

The computer will learn from patterns in the examples you've given it, such as the choice of words, and the way sentences are structured. These will be used to be able to recognise new messages.

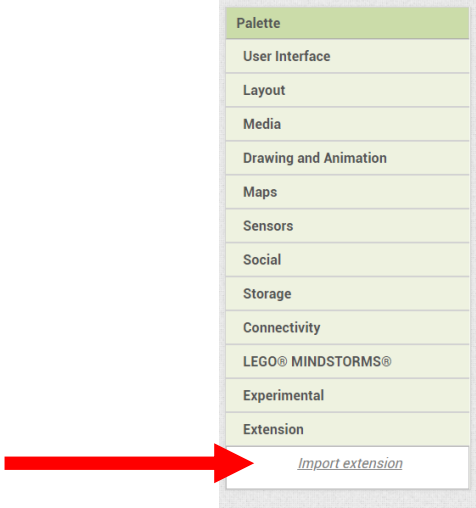
27. Click the **"< Back to project"** link, then the "**Advanced**" button to select **"App Inventor"**.

This page will provide you with your API Key and instructions on how to use the App Inventor Extension. Keep the page open for easy reference.

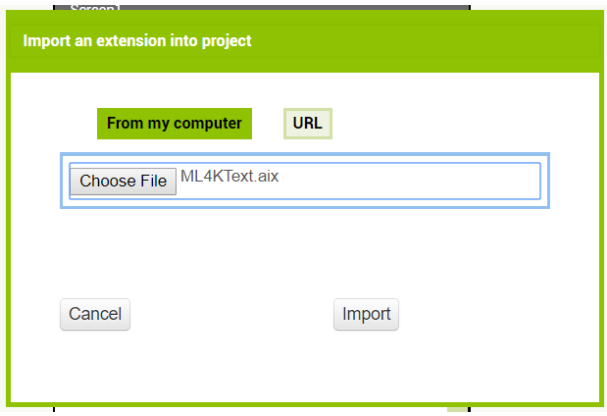


28. Download the App Inventor Extension (ML4K.aix) and go back to your App Inventor project.

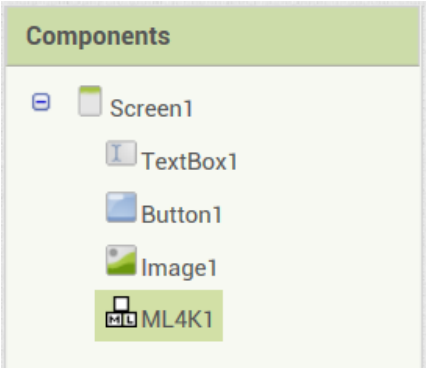
29. We need to add the App Inventor Extension to your App Inventor project. The last category in the "**Pallet**" is "**Extensions**". Click this section to reveal the "**Import Extension**" button.



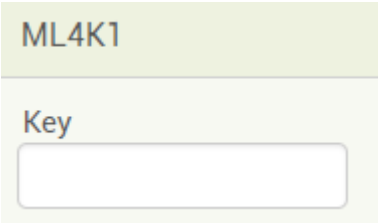
30. Select the "ML4KText.aix" file you downloaded to your computer.



31. Drag the "**ML4K**" component into your project so it shows up in the "**Components**" tree.



32. Click on "**ML4K1"** in the "**Components"** tree to reveal its properties. Copy and paste your Key from the Machine Learning for Kids site into the ML4K1's "**Key"** property field. (Your Key will be unique for your project)



Tips

More examples!

The more examples you give it, the better the computer should get at recognising whether a message is kind or mean.

Try and be even

Try and come up with roughly the same number of examples for kind and mean.

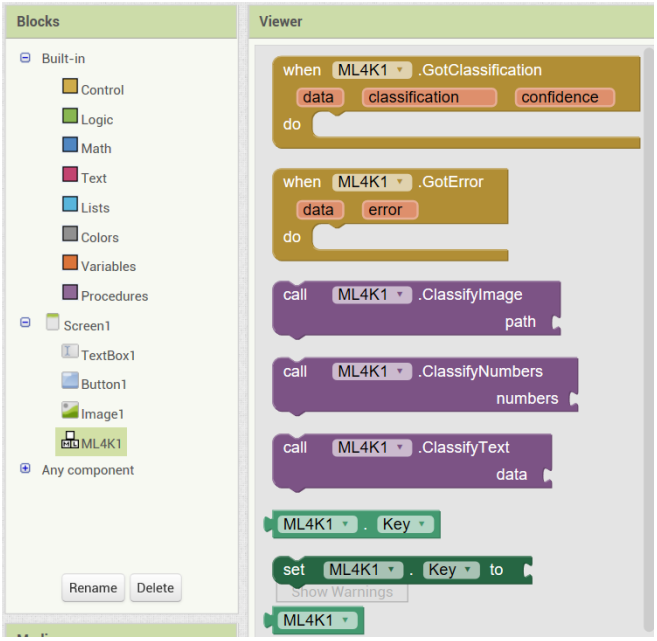
If you have a lot of examples for one type, and not the other, the computer might learn that type is more likely, so you'll affect the way that it learns to recognise messages.

Mix things up with your examples

Try to come up with lots of different types of examples.

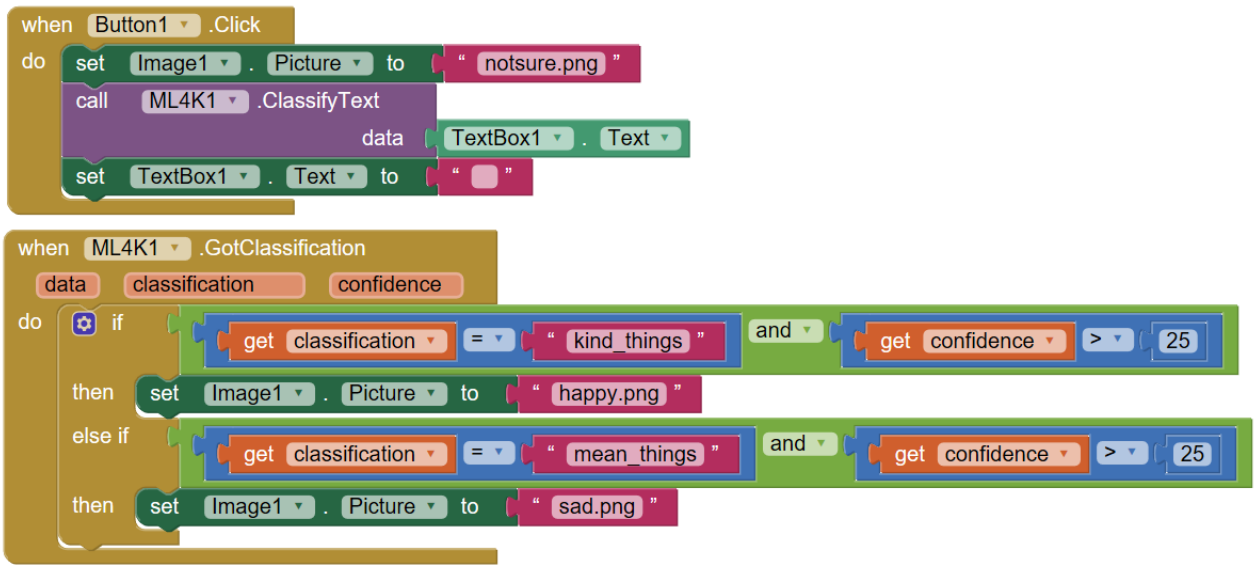
For example, make sure that you include some long examples and some very short ones.

33. Switch from Designer view to Blocks. Notice the code blocks available to you now that you added the extension.

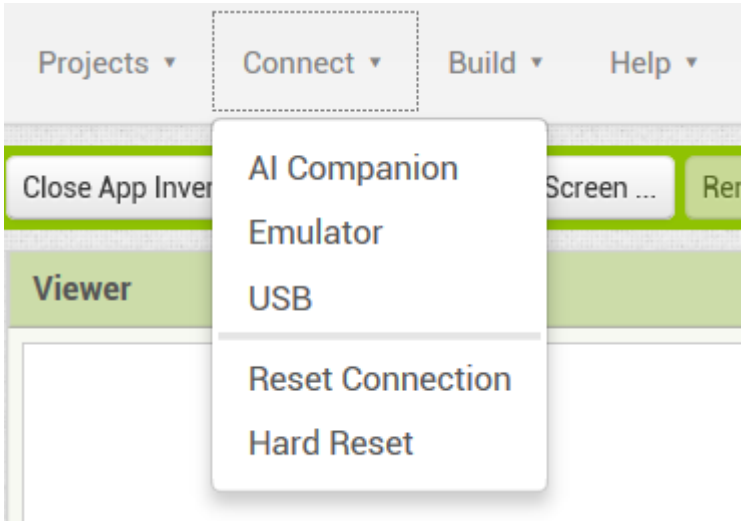


34. Update your algorithm to use your machine learning model instead of the rules you made before.

*The **"ClassifyText"** method block is an important new block. If you give it some text, it will return either "kind things" or "mean things" based on the training you've given to the computer. It returns this information with the "**GotClassification**" event block. You can use this to choose an image to switch to.*



35. Click the **Connect** to test your app with your preferred method.



36. Test your project

Type a kind message and click submit. The character should smile.
Type a mean and unkind message and click submit. The character should look sad.
This should work for messages that you didn't include in your training.

37. Save your project.

*Click on **Projects** -> **Save Project** to save the project to your account.*
*Click on **Projects** -> **Export selected project (.aia) to my computer** to save the project file to your computer.*

What have you done?

You've modified your character to use machine learning instead of your earlier rules-based approach.

Training the computer to be able to recognise messages for itself should be much quicker than trying to make a list of every possible message.

The more examples you give it, the better it should get at recognising messages correctly.

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?

Write a reply

Instead of just changing the way they look, make your character reply, based on what it recognises in the message!

Try a different character

Instead of a person's face, why not try something different, like an animal?

It could react in different ways, instead of smiling.

For example, you could make a dog that wags their tail if you say something kind to it!

Different emotions

Instead of kind and mean, could you train the character to recognise other types of message?

Real world sentiment analysis

Can you think of examples where it's useful to be able to train a computer to recognise the emotion in writing?