Program 1

Aim: To Train a ML model to recognize a Person or Object including gestures

Description:

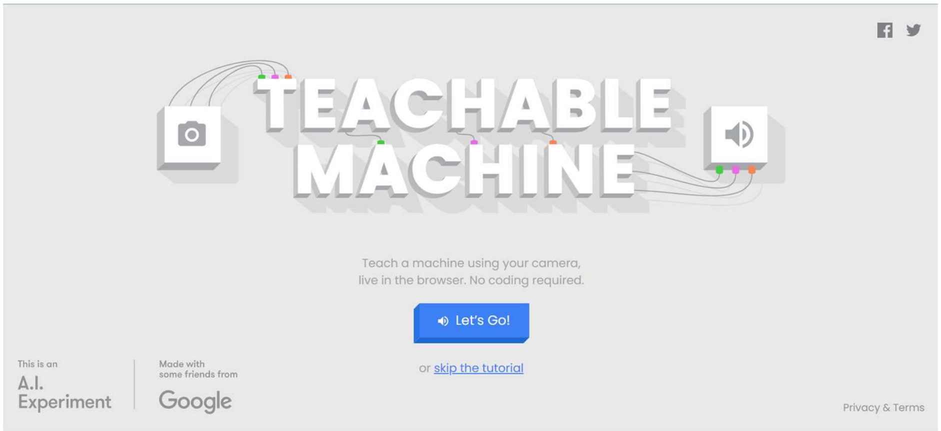
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key grth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.

Teachable Machine is flexible – use files or capture examples live. It's respectful of the way you work. You can even choose to use it entirely on-device, without any webcam or microphone data leaving your computer. The models we make with Teachable Machine are real TensorFlow.js models that work anywhere javascript runs, so they play nice with tools like Glitch, P5.js, Node.js & more.

Screenshots:



Teach a machine using your camera,
live in the browser. No coding required.

Conclusion:

In this ML training model, we trained the model with three different classes. In the first class, hand was shown. In the second class, victory symbol was shown whereas in the third class head was tilted to the left. As the model is trained, it was able to recognise and classify the input from the camera into different classes and show the corresponding GIF images. Thus, when the gestures are changed the output images are changed correspondingly which means the ML model has been trained and will give the result with a very good accuracy

Program 2

Aim: To Train a ML model to recognize various sound bytes and speech

Description:

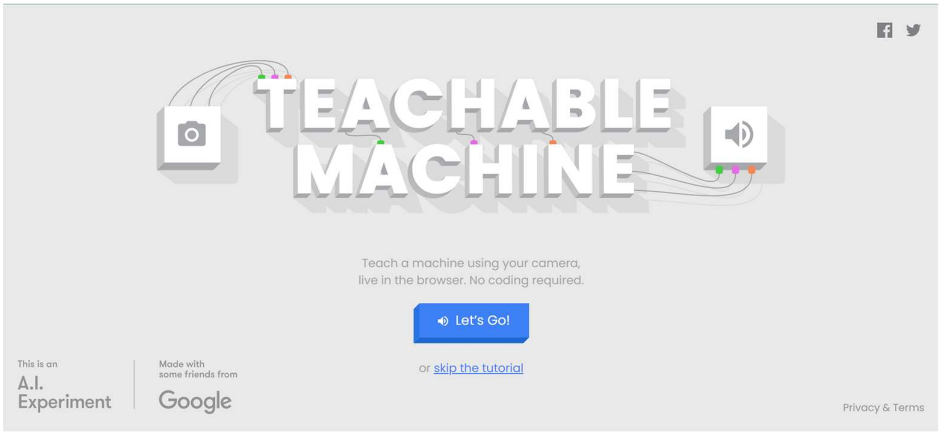
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.

Teachable Machine is flexible – use files or capture examples live. It's respectful of the way you work. You can even choose to use it entirely on-device, without any webcam or microphone data leaving your computer. The models we make with Teachable Machine are real TensorFlow.js models that work anywhere javascript runs, so they play nice with tools like Glitch, P5.js, Node.js & more.

Screenshots:



Teach a machine using your camera, live in the browser. No coding required.

Conclusion:

In this ML training model, we trained the model with three different classes. In the first class, hand was shown. In the second class, victory symbol was shown whereas in the third class head was tilted to the left. As the model is trained, it was able to recognise and classify the input from the camera into different classes and play the corresponding music. Thus, when the gestures are changed the music is changed correspondingly which means the ML model has been trained and will give the result with a very good accuracy

Program 3

Aim: To develop an app to implement Talk to Me

Description: MIT App Inventor is an intuitive, visual programming environment that allows everyone – even children – to build fully functional apps for Android phones, iPhones, and Android/iOS tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

Sources:

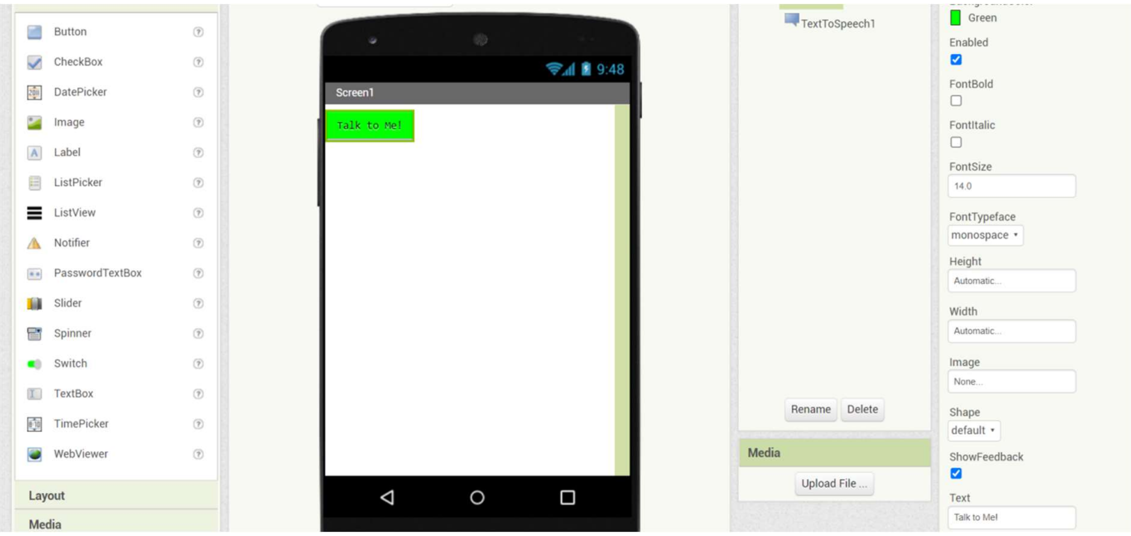
https://appinventor.mit.edu/

http://appinventor.mit.edu/explore/sites/all/files/hourofcode/TalkToMePart1.pdf

https://www.youtube.com/watch?v=Vdo8UdkgDD8

https://www.youtube.com/watch?v=0hikoCvM3oc

Screenshots:



🟩 Green

Enabled
☑

FontBold
☐

FontItalic
☐

FontSize
14.0

FontTypeface
monospace ▾

Height
Automatic...

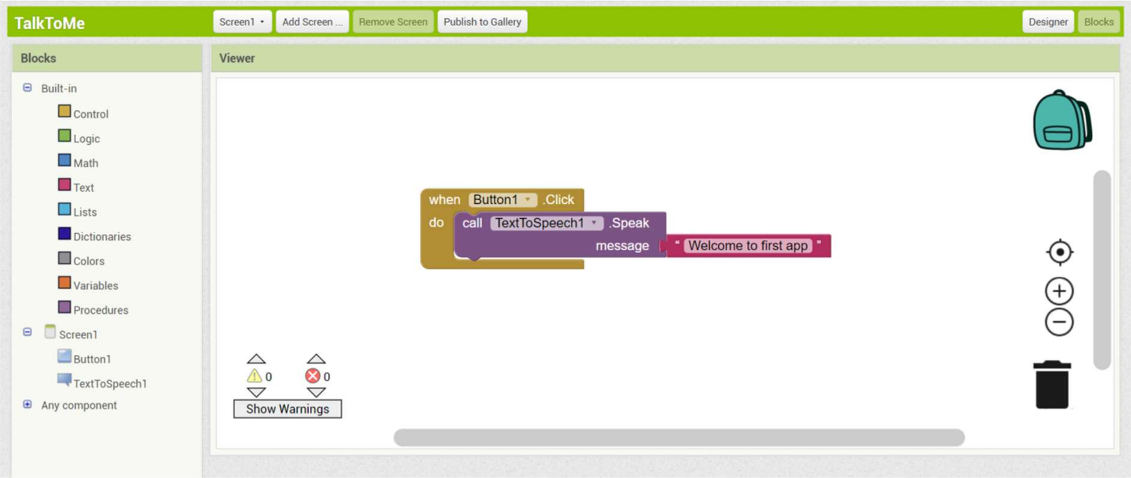
Width
Automatic...

Image
None...

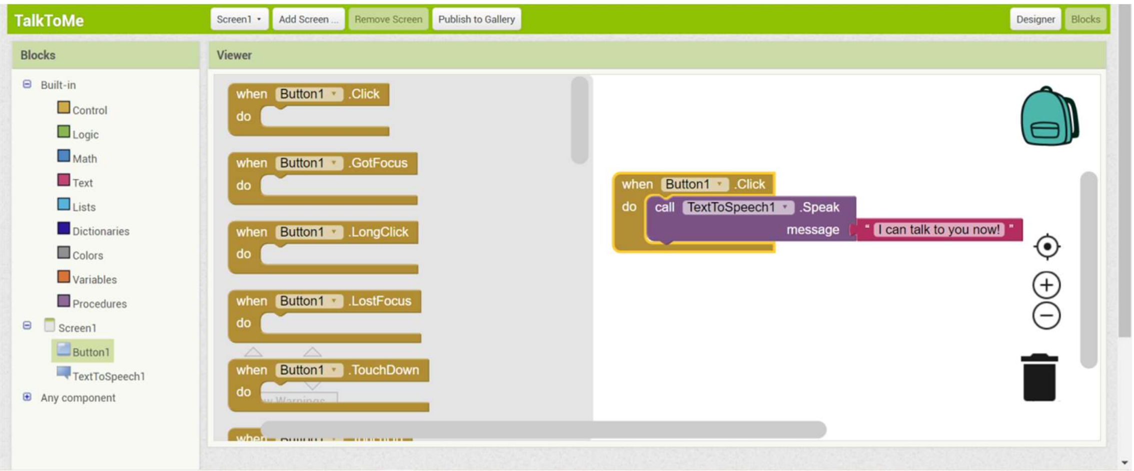
Shape
default ▾

ShowFeedback
☑

Text
Talk to Me!

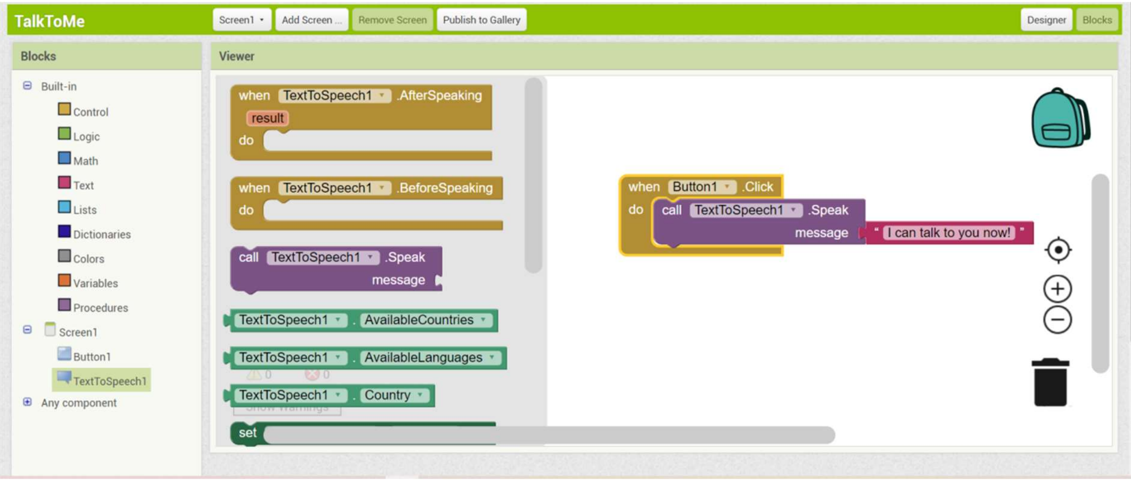


```
when Button1 .Click
do  call TextToSpeech1 .Speak
                  message " Welcome to first app "
```



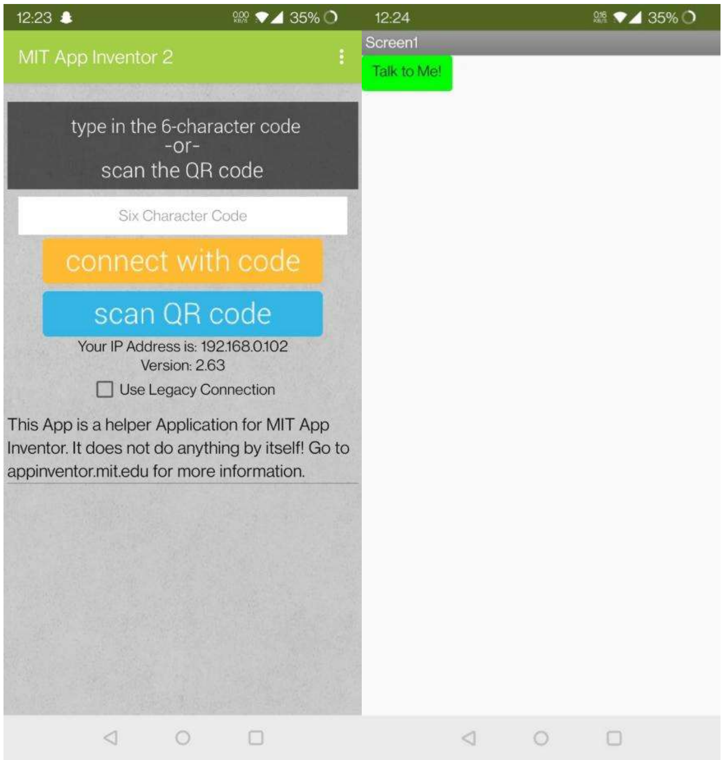
```
when Button1 .Click
do

when Button1 .GotFocus
do

when Button1 .LongClick
do

when Button1 .LostFocus
do

when Button1 .TouchDown
do
```

```
when Button1 .Click
do  call TextToSpeech1 .Speak
                  message " I can talk to you now! "
```



```
when TextToSpeech1 .AfterSpeaking
  result
do

when TextToSpeech1 .BeforeSpeaking
do

call TextToSpeech1 .Speak
                message

TextToSpeech1 . AvailableCountries
TextToSpeech1 . AvailableLanguages
TextToSpeech1 . Country
set
```

```
when Button1 .Click
do  call TextToSpeech1 .Speak
                  message " I can talk to you now! "
```



type in the 6-character code
-or-
scan the QR code

Your IP Address is: 192.168.0.102
Version: 2.63

□ Use Legacy Connection

This App is a helper Application for MIT App Inventor. It does not do anything by itself! Go to appinventor.mit.edu for more information.

Conclusion:

The ML model has been trained to convert “Text to Speech” which is one the greatest features of Machine Learning. In the MIT inventor website, the interface of the app is designed in the Designer section with all the necessary buttons. In the Blocks section, the app is built by connecting the necessary blocks and hence forming the final code for the app to run and for the model to convert speech to text. The text can be entered in the designated space in the blocks section and when the button is clicked in the AI companion app, the entered text is converted into audio. Various sentences have been given as the inputs and the app worked pretty well in the cases

Program 4

Aim: To develop an app to recognize objects using image classification

Description:

Artificial Intelligence (AI) and Machine Learning (ML) are fields of computation used to describe machines or systems that simulate human methods of cognition, such as learning or problem-solving. Examples of machine learning include Google's search engine, social media content suggestions, and the current development of self-driving cars.

Image classification is a very important component of machine learning. For example, a self-driving car would have to instantly classify any images it sees, and the difference between a pedestrian and a green traffic light is critical.

LookExtension is an example of a machine learning tool that implements image classification. One popular way to create machine learning programs is by training them to classify images into some number of classes. The LookExtension uses supervised learning, which is a common way of classifying known categories. Supervised learning is when you have inputs (images) and you want to match them to known outputs (the labels "cat" and "dog"). This way, during training, it is easy to tell when an image is classified correctly or incorrectly.

Unsupervised learning is when you have inputs but no known outputs and you want to use machine learning to find patterns that are not already known. This could be used in a large collection of medical data if one is trying to find patterns that humans may have missed. Generally, image recognition programs will use supervised learning since we already know what images we are looking for.

About LookExtension

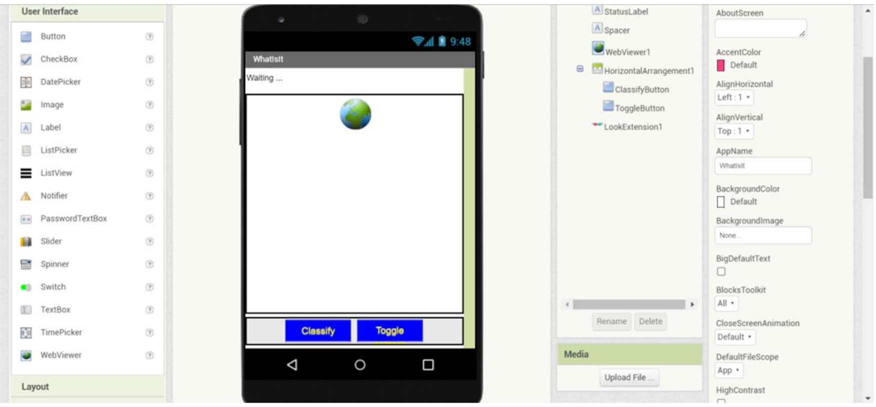
This extension uses a kind of neural network called a mobilenet that is specifically designed to work well for image classification on smartphones. The mobilenet in the Look Extension has been pre-trained to recognize 999 classes, based on training with millions of images. You can check all of the classes here, and note what labels are included. You can also use the App Inventor LookExtension.knownClasses block to obtain the list of classes for use in new apps. When you use the WhatIsIt app, you're taking advantage of all that training. But you're not doing any additional training yourself, so the image recognition never gets any better.

Sources:

https://appinventor.mit.edu/explore/resources/ai/image-classification-look-extension

https://www.youtube.com/watch?v=G6z6irE0CjU

Screenshots:



AboutScreen

AccentColor: Default

AlignHorizontal: Left : 1

AlignVertical: Top : 1

AppName: WhatIsIt

BackgroundColor: Default

BackgroundImage: None...

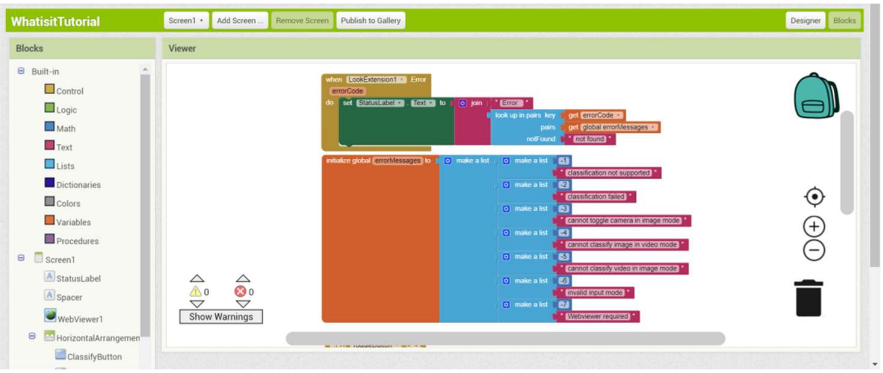
BigDefaultText: □

BlocksToolkit: All

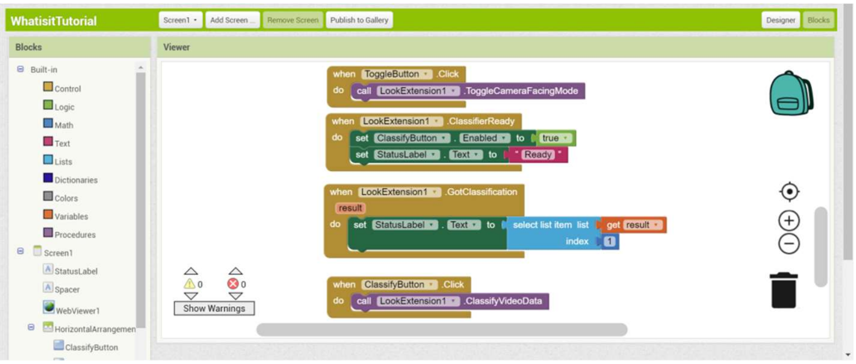
CloseScreenAnimation: Default

DefaultFileScope: App

HighContrast



```
when LookExtension1 .Error
  errorCode
do  set StatusLabel . Text to  join  " Error "
                                     look up in pairs  key  get errorCode
                                                       pairs  get global errorMessages
                                                    notFound  " not found "

initialize global errorMessages to  make a list  make a list  -1
                                                              " classification not supported "
                                                 make a list  -2
                                                              " classification failed "
                                                 make a list  -3
                                                              " cannot toggle camera in image mode "
                                                 make a list  -4
                                                              " cannot classify image in video mode "
                                                 make a list  -5
                                                              " cannot classify video in image mode "
                                                 make a list  -6
                                                              " invalid input mode "
                                                 make a list  -7
                                                              " Webviewer required "
```



```
when ToggleButton .Click
do  call LookExtension1 .ToggleCameraFacingMode

when LookExtension1 .ClassifierReady
do  set ClassifyButton . Enabled to  true
    set StatusLabel . Text to  " Ready "

when LookExtension1 .GotClassification
  result
do  set StatusLabel . Text to  select list item  list  get result
                                                index  1

when ClassifyButton .Click
do  call LookExtension1 .ClassifyVideoData
```

Conclusion:

Various objects like laptop, bottle, clock, mouse, safety pin, scissors, keys and switch are identified and classified using the Image classification app. In the MIT inventor website, the interface of the app is designed in the Designer section with all the necessary buttons. In the Blocks section, the app is built by connecting the necessary blocks and hence forming the final code for the app to run and for the object to the identified. Toggle button is used to change the camera in our phone and classify button is to identify the object. In some of the cases when the angle of the image is change, the model is not able to recognize the objects. From this we can observe that not all the objects can be identified perfectly since the objects are not always in the same shape

Program 5

Aim: To develop an Expression Match app using the trained ML model for facial expressions

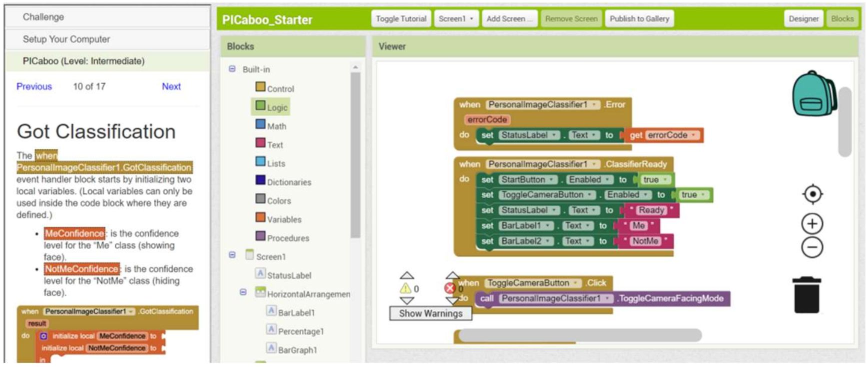
Description:

How can a machine learn about the world? In some ways computers are like very young babies, always soaking up new examples and trying to put what they sense into different buckets — dog, cat, familiar face, stranger.

In this two-part tutorial, we will learn about a type of artificial intelligence (AI) called machine learning (ML), exploring an example called “image classification” — a way for computers to put what they see into various buckets. We will make a “Peekaboo” game with our very own Personal Image Classification (PIC) model. A baby shown in the app will smile when we show our face and cry when we hide our face.

Sources:
https://appinventor.mit.edu/explore/resources/ai/picaboo
https://www.youtube.com/watch?v=b0pvffWfVBY
https://mit-cml.github.io/yrtoolkit/yr/images/PICaboo/PICaboo_Part1.pdf

Screenshots:



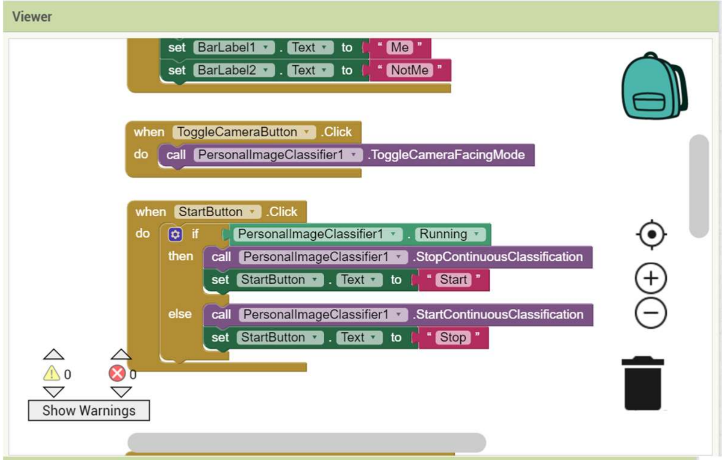
Got Classification

The when PersonalImageClassifier1.GotClassification event handler block starts by initializing two local variables. (Local variables can only be used inside the code block where they are defined.)

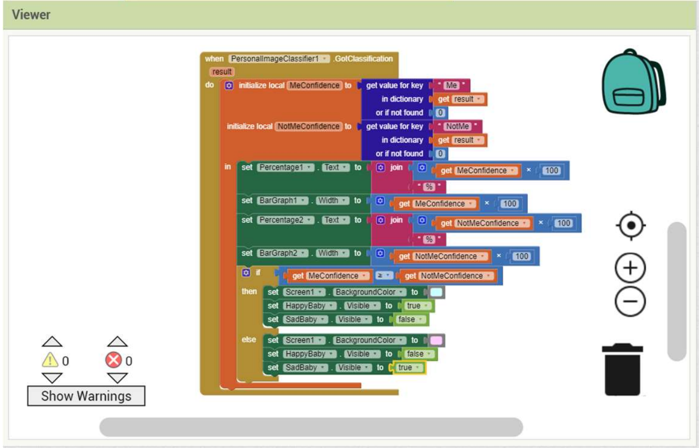
- MeConfidence: is the confidence level for the “Me” class (showing face).
- NotMeConfidence: is the confidence level for the “NotMe” class (hiding face).

```
when PersonalImageClassifier1 .GotClassification
 result
do  initialize local MeConfidence to
    initialize local NotMeConfidence to
    in
```

```
when PersonalImageClassifier1 .Error
 errorCode
do  set StatusLabel . Text to get errorCode

when PersonalImageClassifier1 .ClassifierReady
do  set StartButton . Enabled to true
    set ToggleCameraButton . Enabled to true
    set StatusLabel . Text to " Ready "
    set BarLabel1 . Text to " Me "
    set BarLabel2 . Text to " NotMe "

when ToggleCameraButton .Click
do  call PersonalImageClassifier1 .ToggleCameraFacingMode
```



```
    set BarLabel1 . Text to " Me "
    set BarLabel2 . Text to " NotMe "

when ToggleCameraButton .Click
do  call PersonalImageClassifier1 .ToggleCameraFacingMode

when StartButton .Click
do  if PersonalImageClassifier1 . Running
    then call PersonalImageClassifier1 .StopContinuousClassification
         set StartButton . Text to " Start "
    else call PersonalImageClassifier1 .StartContinuousClassification
         set StartButton . Text to " Stop "
```



```
when PersonalImageClassifier1 .GotClassification
  result
do  initialize local MeConfidence to get value for key " Me "
                                     in dictionary get result
                                     or if not found 0
    initialize local NotMeConfidence to get value for key " NotMe "
                                        in dictionary get result
                                        or if not found 0
    in  set Percentage1 . Text to join get MeConfidence × 100
                                       " % "
        set BarGraph1 . Width to get MeConfidence × 100
        set Percentage2 . Text to join get NotMeConfidence × 100
                                       " % "
        set BarGraph2 . Width to get NotMeConfidence × 100
        if get MeConfidence ≥ get NotMeConfidence
        then set Screen1 . BackgroundColor to
             set HappyBaby . Visible to true
             set SadBaby . Visible to false
        else set Screen1 . BackgroundColor to
             set HappyBaby . Visible to false
             set SadBaby . Visible to true
```

Conclusion:

In this model, two classes are trained. In the first class "Me" the face is not hidden whereas in the second class "NotMe", the face is hidden. More than 50 examples are given for each class to train them. In the testing page, the model is tested and the accuracy is shown on the right. After this, the app is built using the MI inventor website. The app can be viewed on the AI companion app. Since the model is trained now, from the input from the mobile camera, if the face is shown, a happy baby image is displayed and when the face is hidden a sad baby image is displayed. The background colour of the app is also changed accordingly. The accuracy of the prediction is showed on the top section of the app using bar graphs

Program 6

Aim: To develop a Voice Authentication app that uses a trained audio model of the user using audio classification.

Description: This program shows us how to use the Personal Audio Classifier website to train an audio model using short 1-2 second recordings. We can then use the model and the Personal Audio Classifier extension in App Inventor to build an app that can distinguish between voices.

In this case, the app is a Diary app. Our voice will authenticate the app and only allow our voice to open the diary.

The Personal Audio Classifier uses a Spectogram of an audio recording in creating the model that we can use in our app. The Spectogram of different voices will differ and the Classifier will be able to distinguish between different voices or sounds to classify them according to labels.

Sources:

https://appinventor.mit.edu/explore/resources/ai/personal-audio-classifier
https://c1.appinventor.mit.edu/
https://appinventor.mit.edu/assets/files/voice-authentication-app-tutorial.pdf

STEPS TO BE FOLLOWED:

1. Navigate browser to https://c1.appinventor.mit.edu/

You're going to train an audio classifier from scratch! First, you have to teach it what you want to classify. This is the "training" phase. You want to train two voices, so find a friend or family member to be your partner for this activity.

2. Click on the + icon to add labels for you and your partner's voice.

In this example, there are two possible classifications for the model: Nikhil and Natalie. Your model is going to learn to distinguish between you and your partner saying the word "Hello." You will record some voice clips for the model to train on.

3. Test your computer's microphone and get a sense of how long each voice clip will be. Practice by pressing record and saying "Hello" loudly and clearly (you may need to grant microphone access). Make sure you only speak while the red recording bar is active.

4. Clear any practice recordings and record 5-10 good examples of you and your partner saying "Hello". Click the "RECORDING FOR" button to change which person you are recording.

5. Using your examples, you will now "train" the model. Press the "train" button, and then press "train model" using the default parameters.

6. Now that your model is trained, it's time to test it! Press record to record a voice clip and see how the model classifies the clip!

7. Once you're confident in your model, press the "Export" button to export your new model!

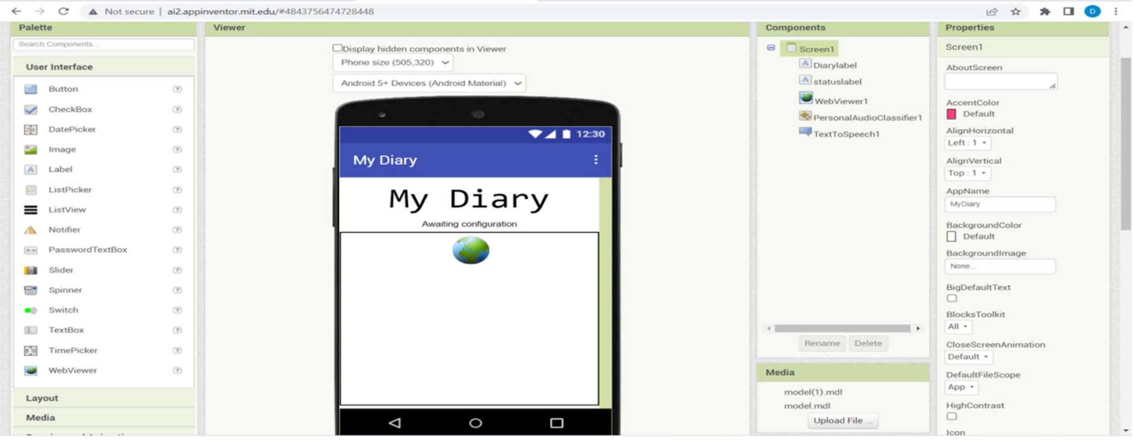
8. Now it's time to build your Voice Authentication App using this exported model!

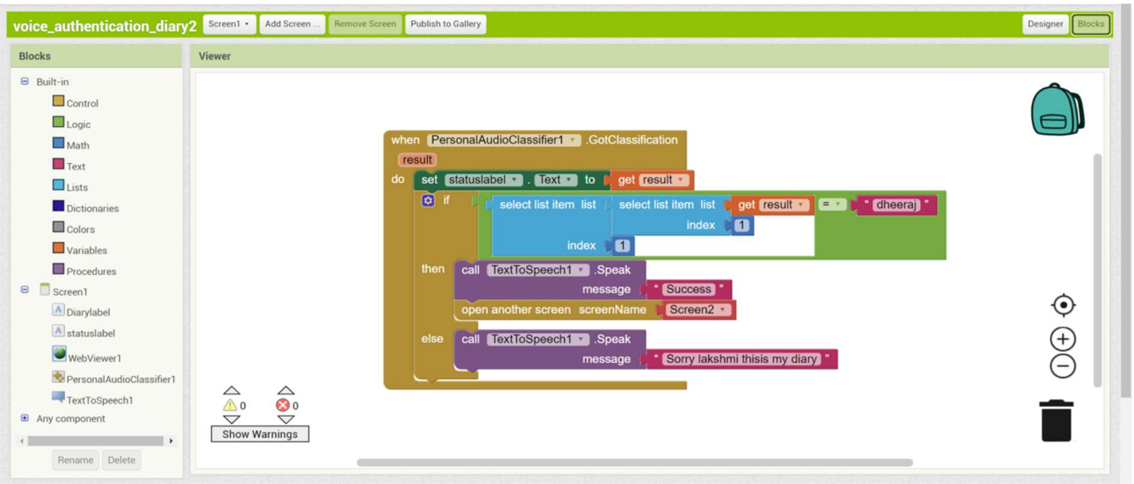
9. Upload your model from the PAC website! Click on PersonalAudioClassifier1, and in the Properties window, click on Model where it says "None" and then upload your model.mdl file. Click OK.

10. Now that your model is uploaded, edit the blocks to tell the diary who is allowed in!

11. And you're done! Take a quick look at Screen2 provided in the template. It simply allows you to type whatever you want in your diary. There is a back button to return to Screen1. The task is left to you to build out the diary aspect of this app. One recommendation is to use TinyDB and store your diary entries on your device so you can save and read your previous diary entries.

Screenshots:





```
when PersonalAudioClassifier1 .GotClassification
  result
do  set statuslabel . Text to get result
    if  select list item list  select list item list  get result  =  " dheeraj "
                                                index 1
                               index 1
    then  call TextToSpeech1 .Speak
                         message " Success "
          open another screen screenName Screen2
    else  call TextToSpeech1 .Speak
                         message " Sorry lakshmi thisis my diary "
```

Conclusion:

In this model, two classes are trained. In the first class audio of 1st person is stored and in the second class audio of 2nd person is stored. About 10 clips are given for each class to train them. Here we have taken the audio of "Hello" from 2 persons. On the testing page, the model is tested and accuracy is shown on the right. After this, the app is built using the MI inventor website. Since the model is trained now, Title named "My Diary" will be displayed on the Diary app with an option of record button. Audio input will be taken on clicking the button and analysed based on our trained model. If it matches then the diary can be accessed along with "success" audio sound or else audio saying "Sorry, This is my diary" is heard as the output with the accuracy scales of audio matching with person 1 and person 2 being displayed at the top. If the right person's voice is recognized, the diary is shown and can be read or edited