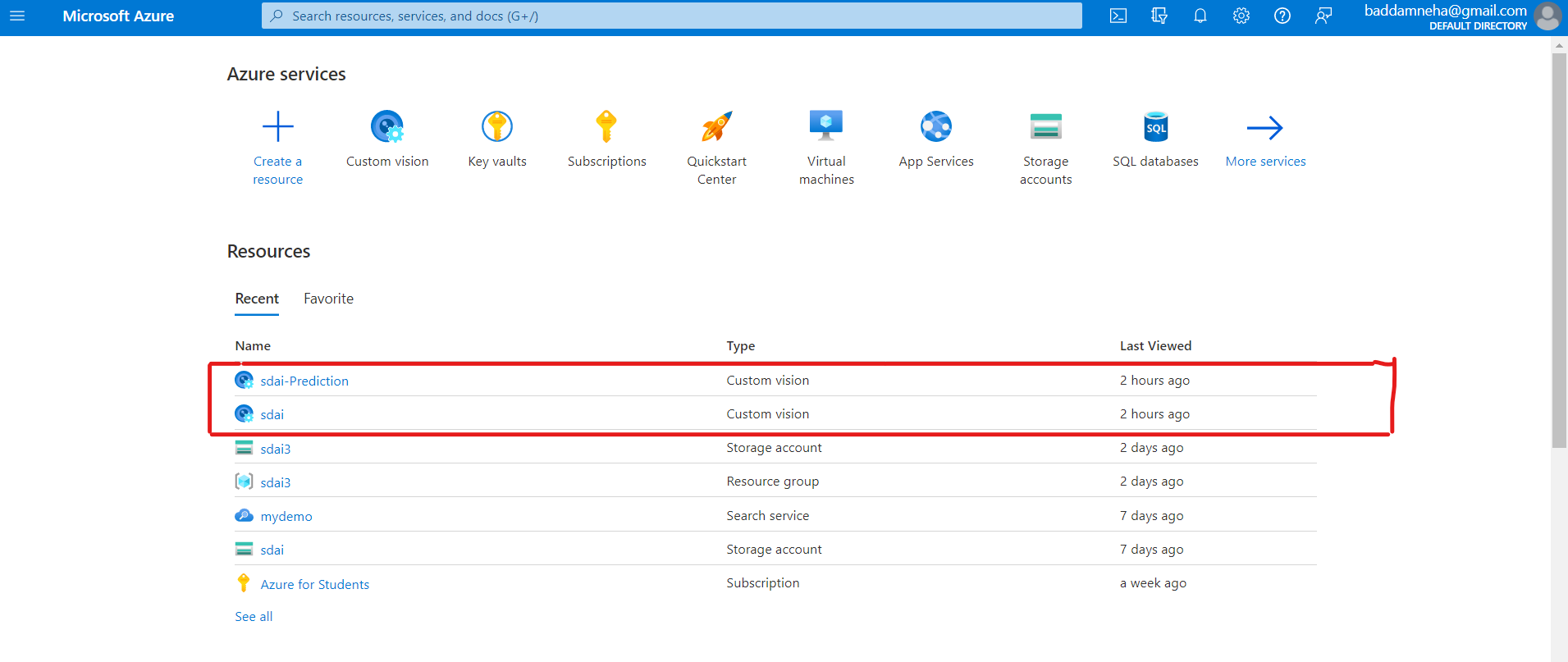
# Pre-requisites

## Azure Setup

Firstly, we must create a Custom Vision Project with training and prediction resources.



Then note down the keys and endpoints of training and prediction resources.

Training key and Endpoint:

Graphical user interface, application, Teams

Description automatically generated

Prediction key, endpoint, and Resource ID:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Dataset setup

On your local storage create a Data folder and upload images for training and prediction.

Graphical user interface, chart, application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

## Package Setup

Run the below command in the command prompt to install custom vision packages for code execution.

pip install azure-cognitiveservices-vision-customvision

**Image classification with the Azure Custom Vision**

Now, import all the packages required for image classification. Paste all the keys and endpoints in the code.

Graphical user interface, text, application

Description automatically generated

We are validating the API key of both training and prediction and using the key and endpoint to create Custom Vision client objects for both training and prediction.

Text

Description automatically generated with medium confidence

After creating the custom vision project, we now create a custom vision project using the “create\_project” function.

Graphical user interface, text, application, email

Description automatically generated

After creating the project, we created two tags namely mango and apple.



Now, we upload the images from the local storage in the system by giving the directory path of the folder that contains the Dataset. Each image is loaded one after the other into the respective tags. If the images are not uploaded successfully, an error with status is printed.Graphical user interface, text, application

Description automatically generated

Now we train the project with the above-loaded images using the train\_project function. Once the training is completed, the training status is printed as completed.

Graphical user interface, text, application, email

Description automatically generated

Background pattern

Description automatically generated with low confidence

Once the training is completed, we must publish the current completed iteration to the project endpoint using the “publish\_iteration” function.

A picture containing text

Description automatically generated

Once the training is published, we now use the test images to predict and classify the image. In the below code, I have used the image of a mango and the prediction shows that it’s 100% mango.

Graphical user interface, text, application

Description automatically generated

In the below code, I have used the image of an apple and the prediction shows that it’s 100% apple.

Graphical user interface, text, application

Description automatically generated

Now, I have written a code to upload the image from the local drive-in runtime. Here, we must copy and paste the directory of the actual test image folder and the test image filename. Once the input is given, the image is classified, and prediction is displayed.

Text

Description automatically generated

Below is one more way to dynamically upload images in the runtime. Here we can directly select the images from the dropdown as shown below from the file directory that has been passed to the “filedir” variable in the below code.

Graphical user interface, text, application

Description automatically generated

We can download and save any number of images into the file directory mentioned in the code to display in the drop-down. We can select any image from the drop-down. When we dynamically change the images, that image is classified and a prediction is printed for that image that has been selected.

Graphical user interface, text, application

Description automatically generated

A picture containing fruit, apple, cherry

Description automatically generatedA picture containing fruit, apple, orange

Description automatically generated

Graphical user interface, application, Word

Description automatically generated