4.3 Data Collection

The eye defects that are being detected are Exophthalmos, Cataracts, Strabismus, Glaucoma, Uveitis, Conjunctivitis, Blepharitis, Keratitis, and Pterygium.

Data was web scraped using a python script.

The script uses Selenium which is a tool to be used to automate web browser interaction with Python (also other languages). In layman’s term, selenium pretends to be a real user, it opens the browser, “moves” the cursor around and clicks buttons if you tell it to do so. The initial idea behind Selenium is automated testing. However, Selenium is equally powerful when it comes to automating repetitive web-based tasks.

4.3.1 Searching for a particular phrase & get the image links

The function fetch\_image\_urls expects three input parameters:

1. query : Search term, like Dog
2. max\_links\_to\_fetch : Number of links the scraper is supposed to collect
3. webdriver : instantiated Webdriver



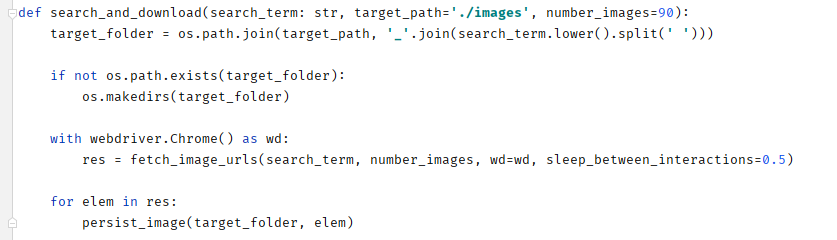
4.32 Downloading the images

Using PIL/Pillow, the persist\_image function grabs an image URL, url and downloads it into the folder\_path. The function will assign the image a random 10-digit id.



4.3.3 Searching and Downloading

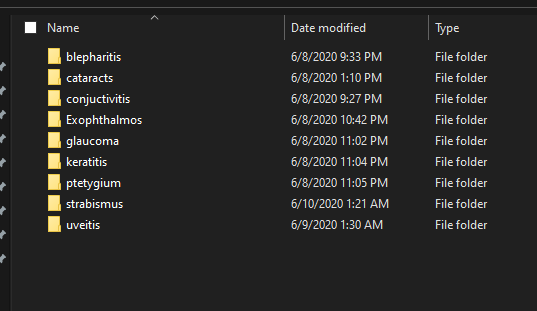
The following function search\_and\_download combines the previous two functions and adds some resiliency to how we use the ChromeDriver. More precisely, we are using the ChromeDriver within a with context, which guarantees that the browser closes down ordinarily, even if something within the with context raises an error. search\_and\_download allows you to specify number\_images, which by default is set to 5, but can be set to whatever number of images you want to download.



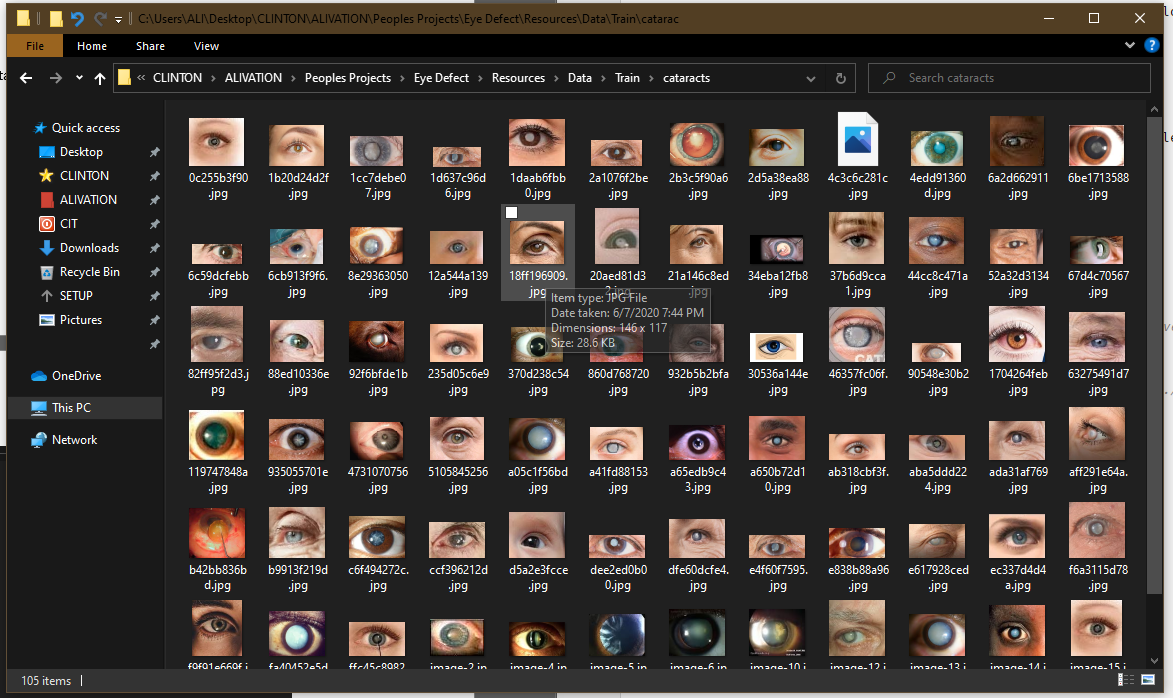
4.33 Resultant Data

From the search, the script obtained a total of 1100 images across the 9 classes of defects.

Below is the directory with all images downloaded



Below is a sample data of the cataracts images downloaded



4.4 Data Preprocessing

4.4.1 Image Cropping

The images had to be manually cropped to remove parts of the images that did not include the eyes that will be required for training of the model

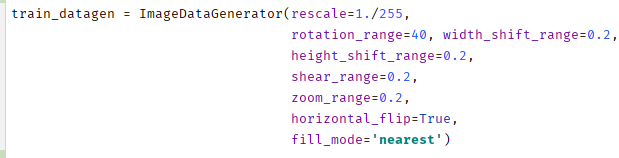
A sample of an image before and after cropping is shown below



4.4.2 Image Augmentation

Image Augmentation is to create more images by resizing, zooming, rotating images, etc to construct new images. With this approach, the model will able to capture more features than before and will able to generalize well for unseen data. Adding image augmentation is really easy with the TensorFlow image generator. When image augmentation is applying, the original dataset will be untouched and all the manipulations will be done in the memory.

The following code segment will show how to add this functionality.



In here image rotating, shifting, zooming and few other image manipulation techniques are applied to generate new samples in the training set.

This enables the model to not be overfitted and with a very small dataset like this, this accuracy is impressive.

4.5 Model Development

4.5.1 Model construction

A Convolutional Neural Network is constructed with multiple convolution layers, pooling layers, and dense layers.

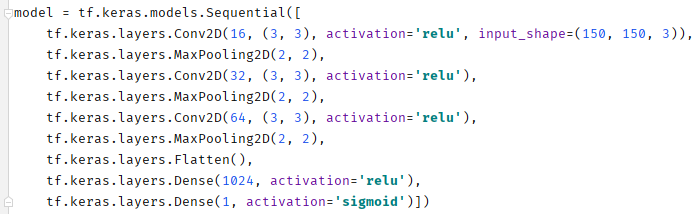
Since the data is ready, the model is built with the following layers:

3 convolutional layers

3 max-pooling layers.

a Flatten layer and finally,

2 dense layers.



NB: GOOGLE EACH OF THE LAYERS AND WRITE A SHORT NOTE ON WHAT THE LAYERS DO… JUST A COUPLE OF SENTENCES

In the first convolution layer, there are 16 kernels which have the size of 3x3. Once the image is convoluted with kernel it will be passed through relu activation to obtain non-linearity. The input shape of this layer should be 150x150 since we resized images for that size. Since all the images are colored images, they have 3 channels for RGB.

In the max-pooling layer, there is a 2x2 kernel such that the max value will be taken when reducing the image size by 50%.

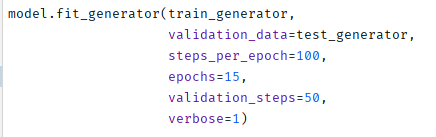
The Flatten layer then takes the output from the previous max-pooling layer and convert it to a 1D array such that it can be feed into the Dense layers. A dense layer is a regular layer of neurons in a neural network. This is where the actual learning process happens by adjusting the weights. Here we have 2 such dense layers and since this is a binary classification there is only 1 neuron in the output layer. The number of neurons in the other layer is adjusted as a hyperparameter to obtain the best accuracy.

4.5.2 Model Training

Since we are using a binary classification we can use binary\_crossentropy method for calculating loss/error.

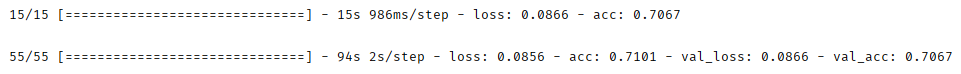
the metrics parameter will be used to estimate how good our model is and here we use the accuracy.

Code is below



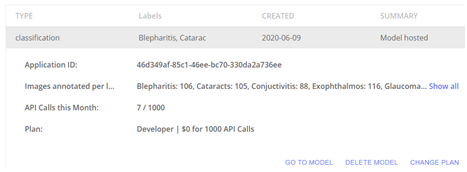
4.5.3 Model Accuracy

Accuracy is 71%

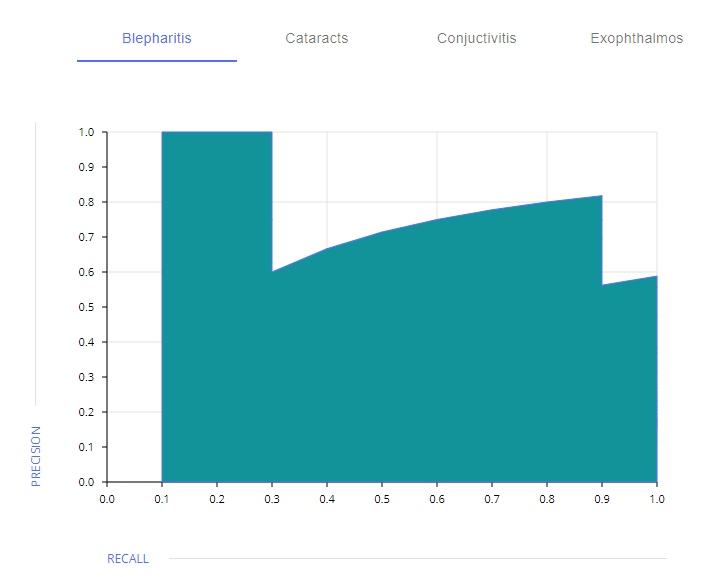


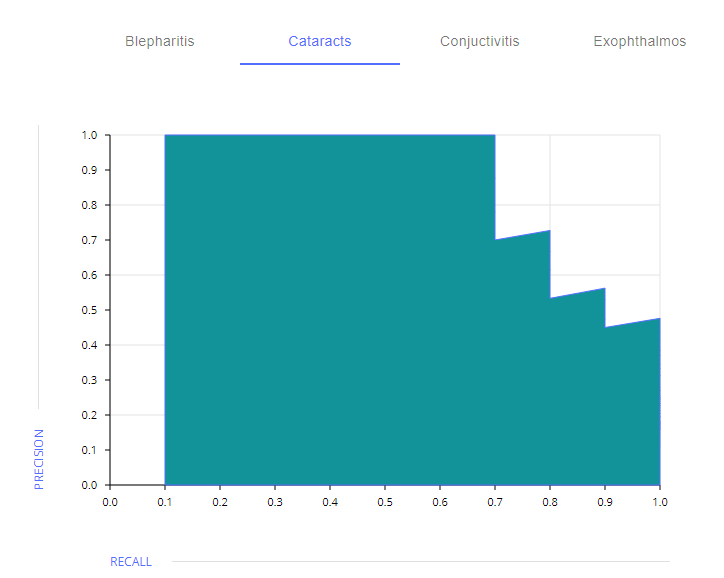
4.5.4 Model Hosting

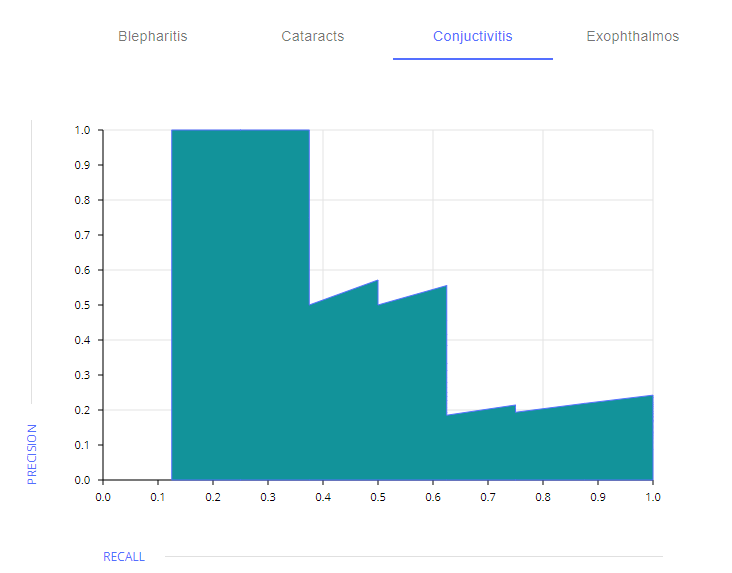
The Machine Learning model is hosted on Nanonets API as a classification model.

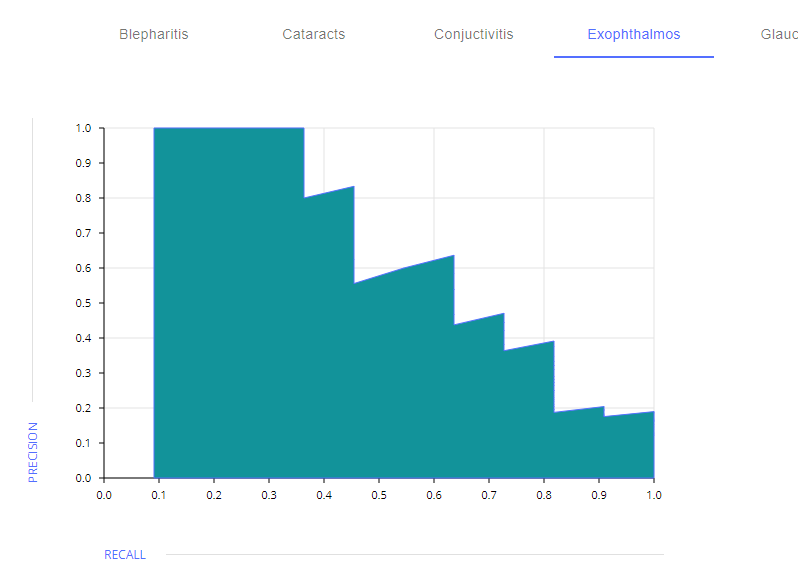


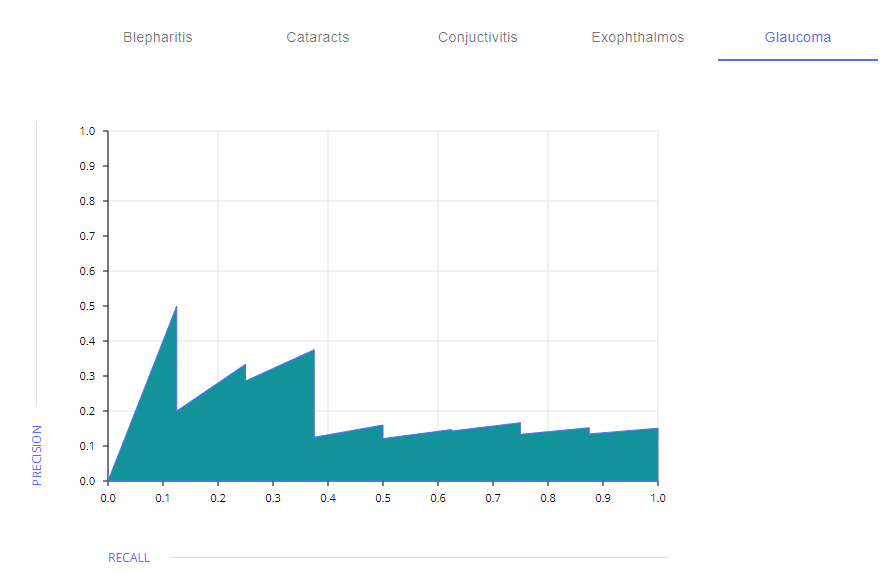
4.5.5 Model Metrics

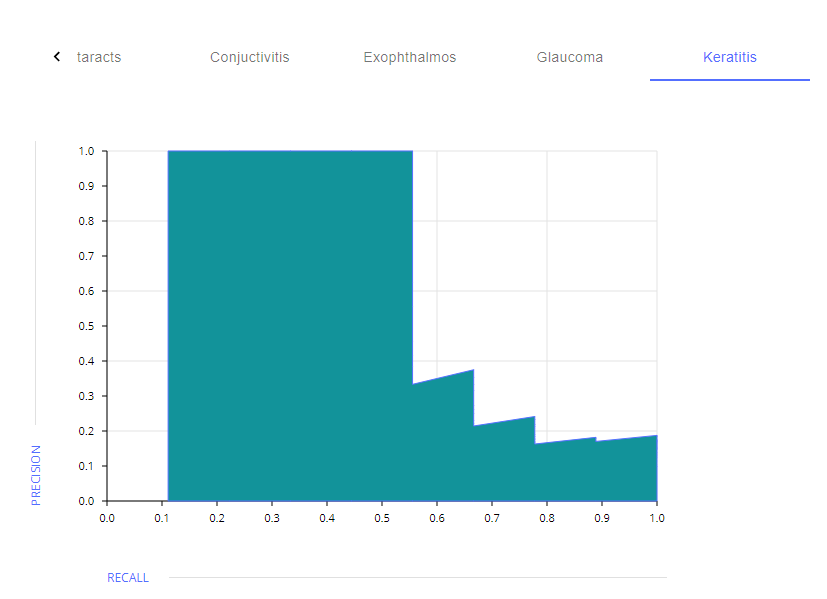
i) Receiver operating characteristic curve for each

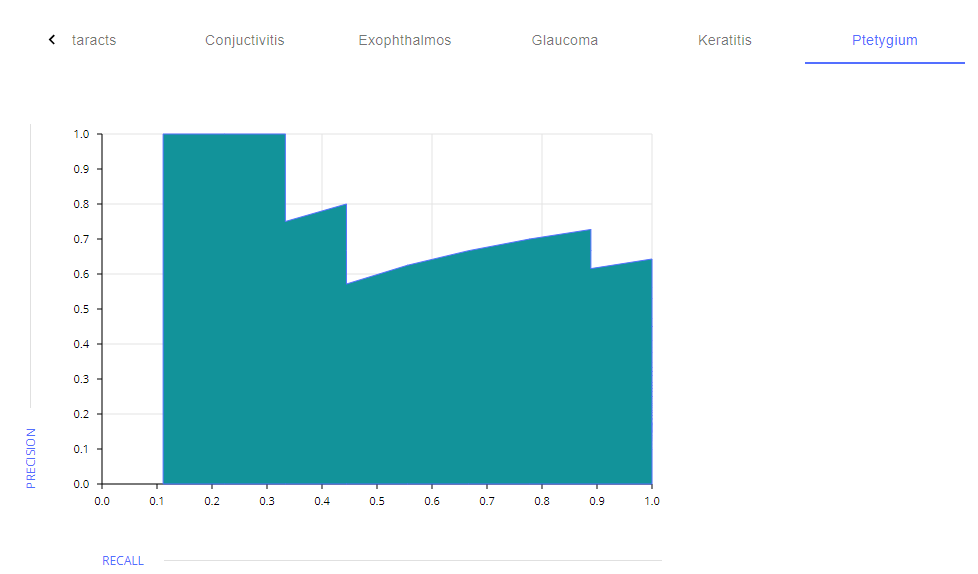


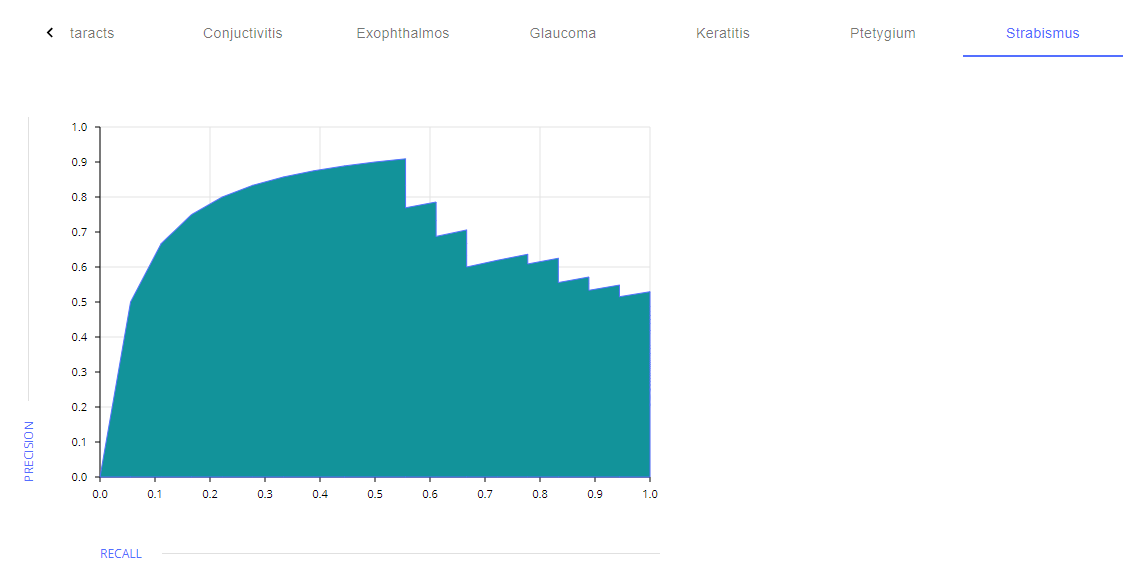


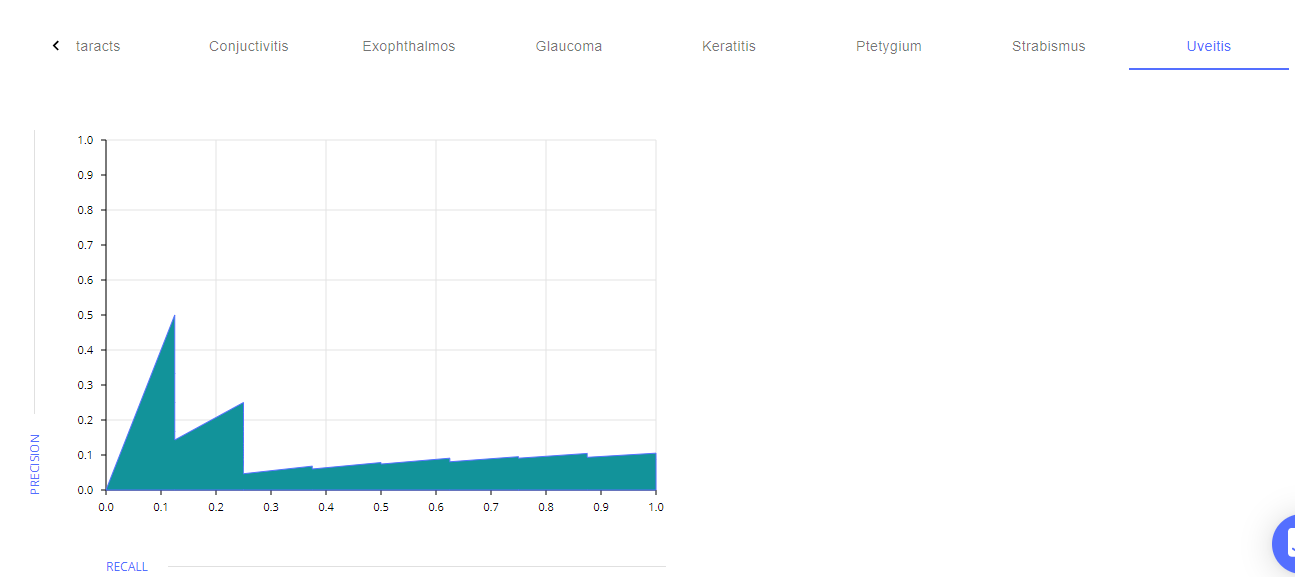




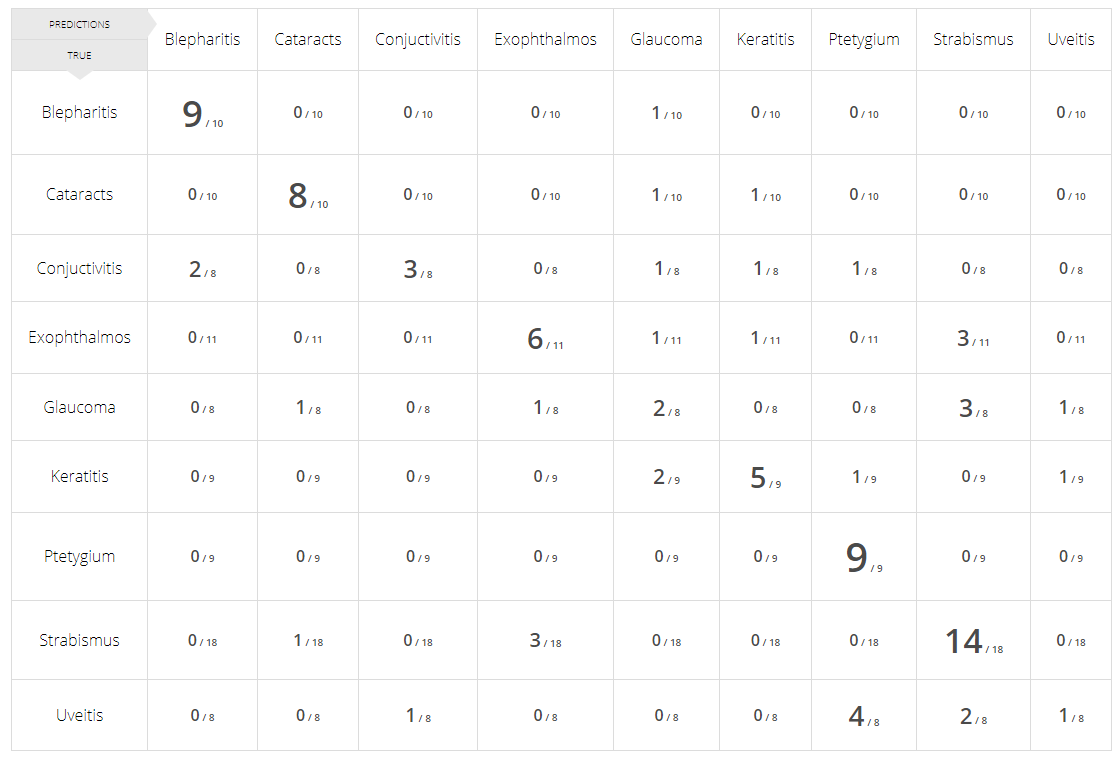






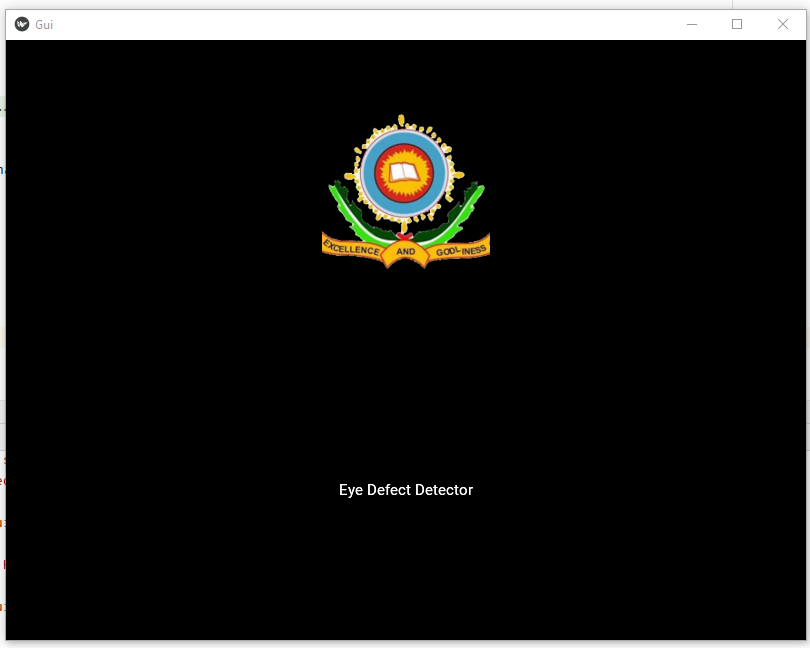


ii) Confusion Matrix



4.6 GENERAL WORKING OF THE SYSTEM

4.4.1 the home page



The main page

