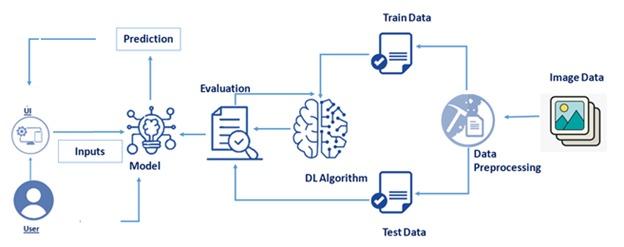
**Action Classification Recognition Using Resnet**

**Introduction to project:**

Human action recognition, or HAR, is a challenging time series classification task. It involves predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to fit a machine learning model. Recently, deep learning methods such as convolutional neural networks and recurrent neural networks have shown capable and even achieve state-of-the-art results by automatically learning features from the raw sensor data.

**Technical Architecture:**



**Pre requisites:**

**To complete this project, you must require following software’s, concepts and packages**

* **Anaconda navigator:**
  + Refer to the link below to download anaconda navigator
  + **Link :** <https://www.youtube.com/watch?v=5mDYijMfSzs>
* **Python packages:**
  + open anaconda prompt as administrator
  + Type “pip install tensorflow” (make sure you are working on python 64 bit)
  + Type “pip install flask”.

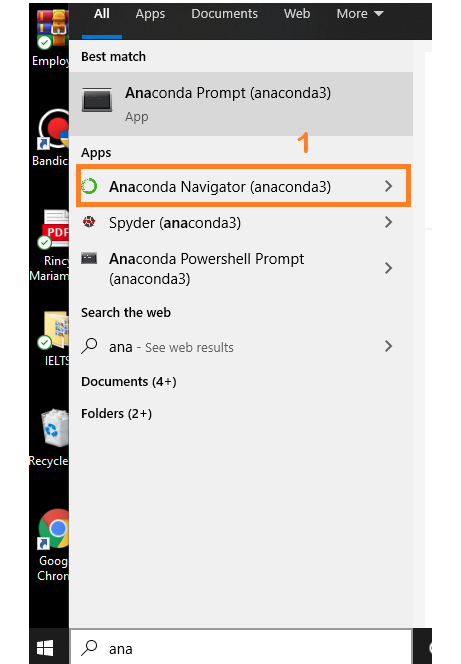
or

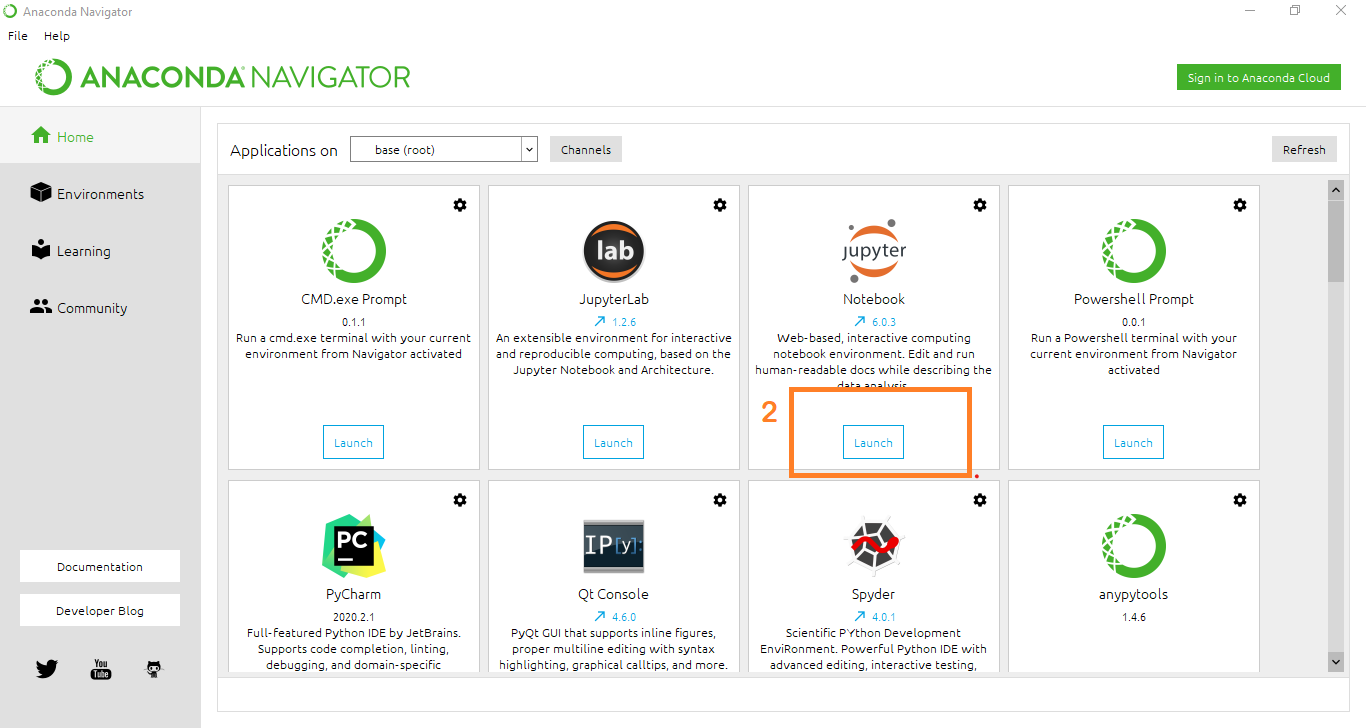
* + Open anaconda prompt as administrator
  + Type “pip install -r requirements.txt” (make sure you are working on python 64 bit)

The above requirements.txt files allow you to install all the required files in the anaconda environment for your project.

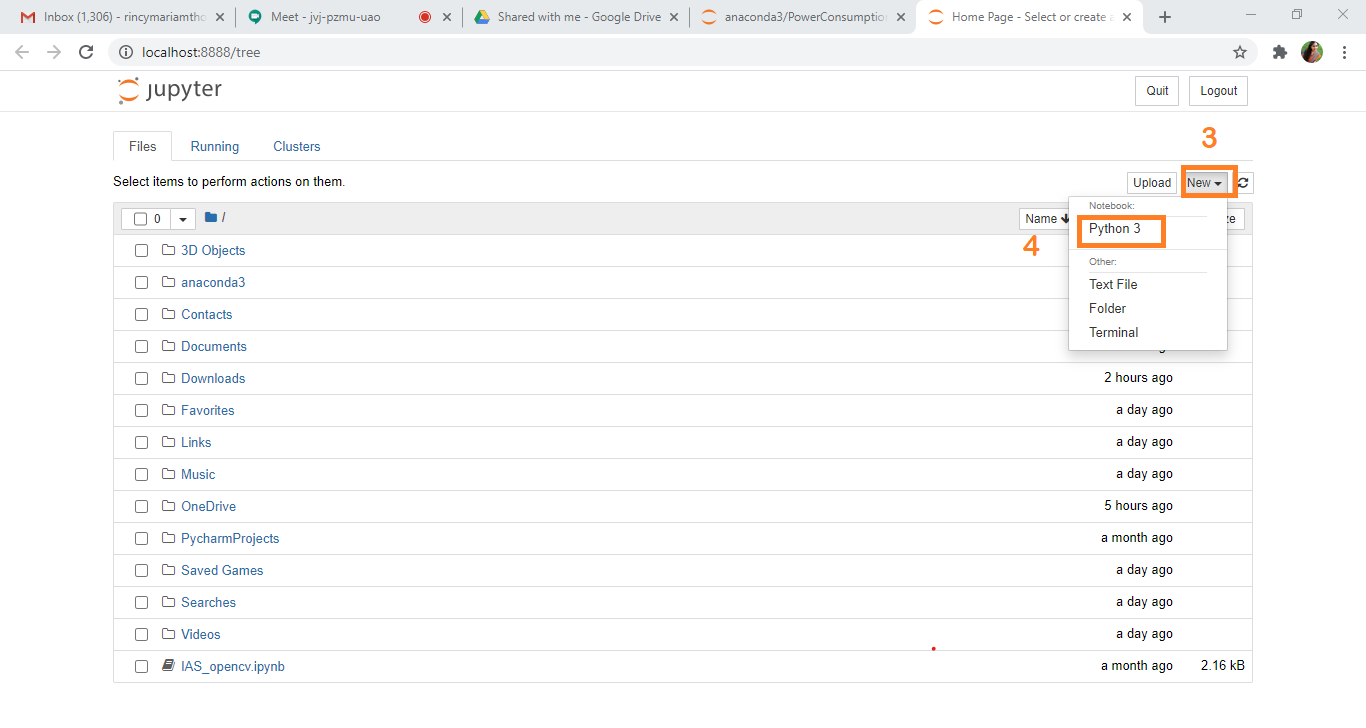
the above steps allow you to install keras and tensorflow in the anaconda environment

* **Deep Learning Concepts** 
  + **CNN:**<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
  + **Resnet:**<https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>
  + **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>
* **Launch Jupyter**
  + Search for Anaconda Navigator and open the Launch Jupyter notebook.





* Then you will be able to see that the jupyter notebook runs on localhost:8888.
* To Create a new file Go to New 🡪Python3. The file in the jupyter notebook is saved with .ipynb extension.



**Project Objectives:**

By the end of this project you will:

* Know fundamental concepts and techniques of CNN
* Know the concept and implementation of Resnet
* Know fundamental concepts of python
* Know how to install necessary packages and set up the environment.
* Know how to build a web application using Flask framework.

**Project Flow:**

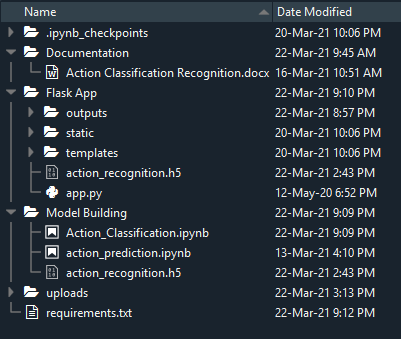
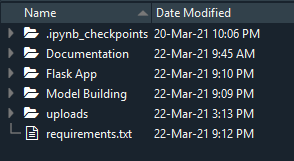
* User interacts with the UI (User Interface) to upload the image and predict the results
* Entered data is analyzed by the model which is integrated
* Once the model analyses the input the prediction of the next day is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
  + Collect the dataset or Create the dataset
* Data Preprocessing.
  + Import the ImageDataGenerator library
  + Configure ImageDataGenerator class
  + Apply ImageDataGenerator functionality to Trainset and Test Set
* Model Building
  + Import the model building Libraries
  + Initializing the model
  + Adding CNN layer
  + Adding Max pooling layer
  + Adding Hidden layer
  + Adding Output Layer
  + Compile the model
  + Generate the model
  + Save the Model
* Application Building
  + Create an HTML file
  + Build Python Code

**Project Structure:**

Create a Project folder which contains files as shown below



* We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for server side scripting
* app.py - contains the actual python code that will import the app and start the development server.
* Action\_Classification.ipynb - This is where you define models for your application.
* action\_recognition.h5 - This is our model weights file
* static - contains static files i.e. CSS, Javascript, images.
* templates - This is where you store your html templates i.e. base.html
* requirements.txt - This is where you store your package dependencies.

**Milestone 1: Data Collection**

ML depends heavily on data, without data, a machine can't learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

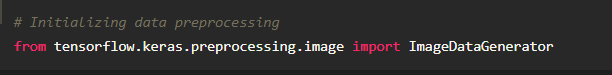
**Milestone 2: Data Preprocessing**

Data Pre-processing includes the following main tasks

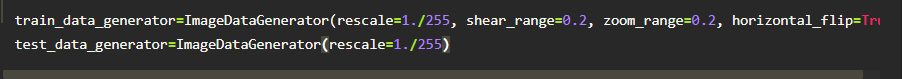
* + Import the ImageDataGenerator library
  + Configure ImageDataGenerator class
  + Apply ImageDataGenerator functionality to Trainset and Test Set

**Activity 1**: **Import the ImageDataGenerator library**

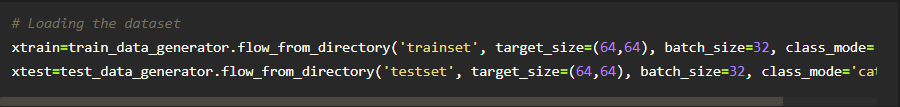
Image data augmentation is used to expand the training dataset in order to improve the performance and ability of the model to generalize. Image data augmentation is supported in the Keras deep learning library via the ImageDataGenerator class. How to use shift, flip, brightness, and zoom image data augmentation



**Activity 2: Configure ImageDataGenerator class**

****

**Activity 3: Apply ImageDataGenerator functionality to Trainset and Test Set**

****

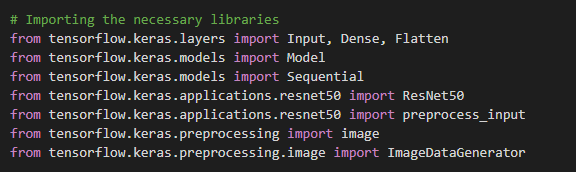
**Milestone 3: Model Building**

Model Building Includes:

* + Import the model building Libraries
  + Adding Resnet Model
  + Adding CNN Layers
  + Adding Flatten Layer
  + Adding Dense Layers
  + Configure the Learning Process
  + Training the model
  + Save the Model

**Activity 1 : Importing the Model Building Libraries**

Importing the necessary libraries



**Sequential:** Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer that follows it. We use the 'add()' function to add layers to our model. We will add two layers and an output layer.

**ResNet50:** ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of the ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

**Preprocess\_Input:** Keras works with batches of images. So, the first dimension is used for the number of samples (or images) you have. When you load a single image, you get the shape of one image, which is (size1,size2,channels) . The preprocess\_input function is meant to adequate your image to the format the model requires.

**ImageDataGenerator:** Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Image data augmentation is supported in the Keras deep learning library via the ImageDataGenerator class.

**Dense:** A densely-connected neural network layer. Dense implements the operation activation(matmul(input, weight) + bias) , where weight is a weight matrix, bias is a bias vector, and activation is an element-wise activation function. This layer also supports 3-D weight tensors with 2-D bias matrices.

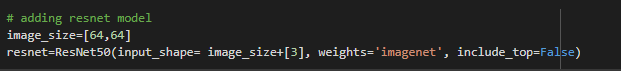
**Convolution:** A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image

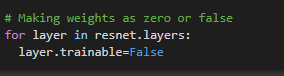
**Max pooling:** Max Pooling is a convolution process where the Kernel extracts the maximum value of the area it convolves. Max Pooling simply says to the Convolutional Neural Network that we will carry forward only that information, if that is the largest information available amplitude wise.

**Flatten:** Flattening a tensor means to remove all of the dimensions except for one. A Flatten layer in Keras reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor.

**Activity 2 : Adding Resnet Model**

ResNet, short for Residual Networks, is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of the ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks were difficult due to the problem of vanishing gradients.

****

****

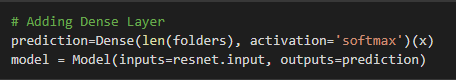
**Activity 3 : Adding Flatten Layer**

Flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer. Dense adds the fully connected layer to the neural network.

****

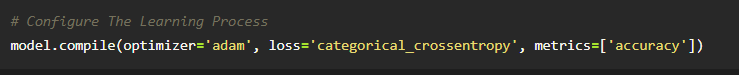
**Activity 4: Adding Dense Layers**

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.



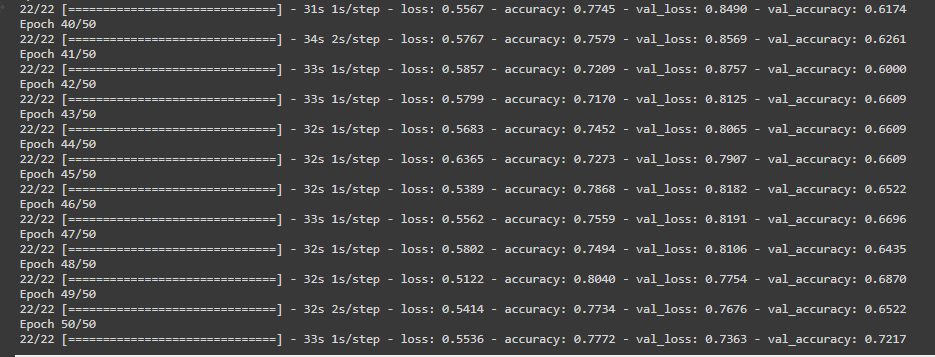
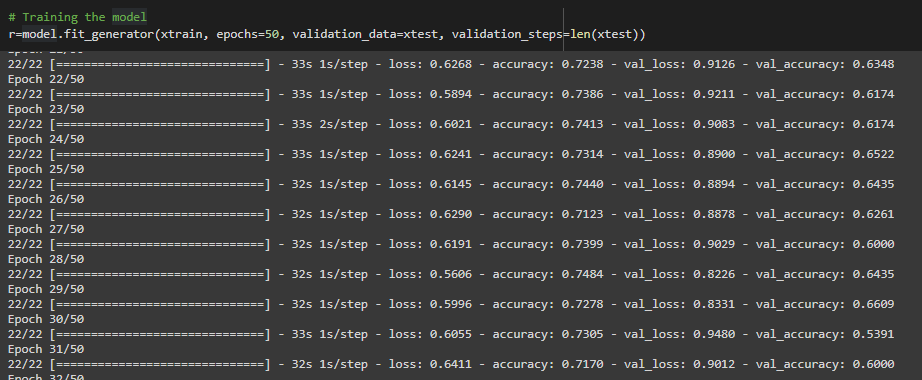
**Activity 5 : Configure The Learning Process**

* The compilation is the final step in creating a model. Once the compilation is done, we can move on to training phase.Loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process.
* Optimization is an important process which optimize the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
* Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process



**Activity 6: Train The model**

Now ,let us train our model



**Arguments:**

* Epochs : an integer and number of epochs we want to train our model for.
* validation\_data can be either:

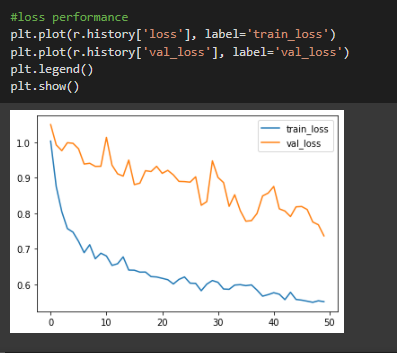
- an inputs and targets list

- a generator

- an inputs, targets, and sample\_weights list which can be used to evaluate

the loss and metrics for any model after any epoch has ended.

**Activity 7: Visualization for Loss and Accuracy**

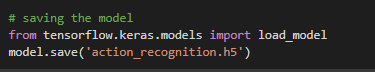
****

****

**Activity 8: Save the Model**

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



**Milestone 4: Application Building**

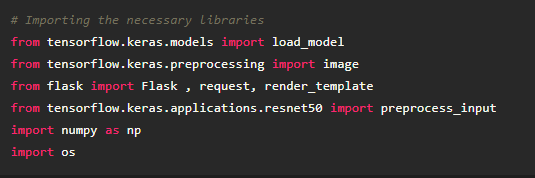
* + **Link**: <https://thesmartbridge.com/documents/spsaimldocs/FlaskML.pdf>

**Activity 1 : Create an HTML File**

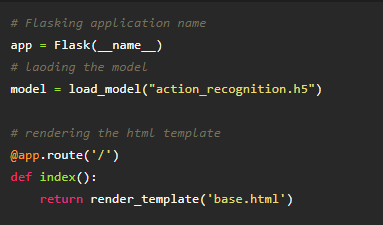
* We use HTML to create the front end part of the web page.
* Here, we created one html page- base.html.
* base.html displays the home page.
* For more information regarding HTML refer to the link below.
  + **Link:** [**https://www.w3schools.com/bootstrap/bootstrap\_forms\_inputs.asp**](https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp)
  + **Link:**<https://www.w3schools.com/css/>

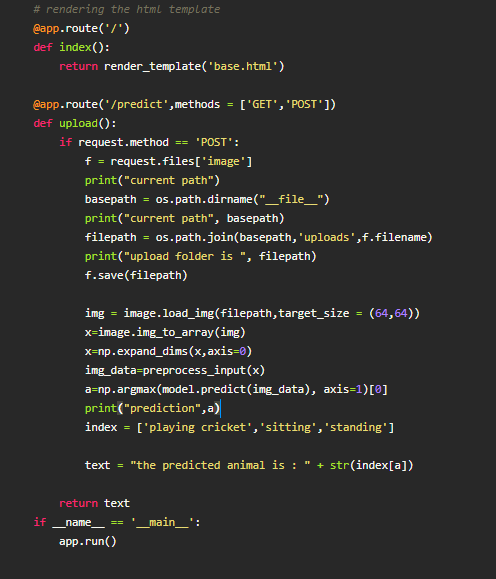
**Activity 2 : Build python code**

* Let us build a flask file ‘app.ipynb’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
* App starts running when the “\_\_name\_\_” constructor is called in main.
* render\_template is used to return html files.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.
* Importing Libraries

****

* + Routing to the html Page

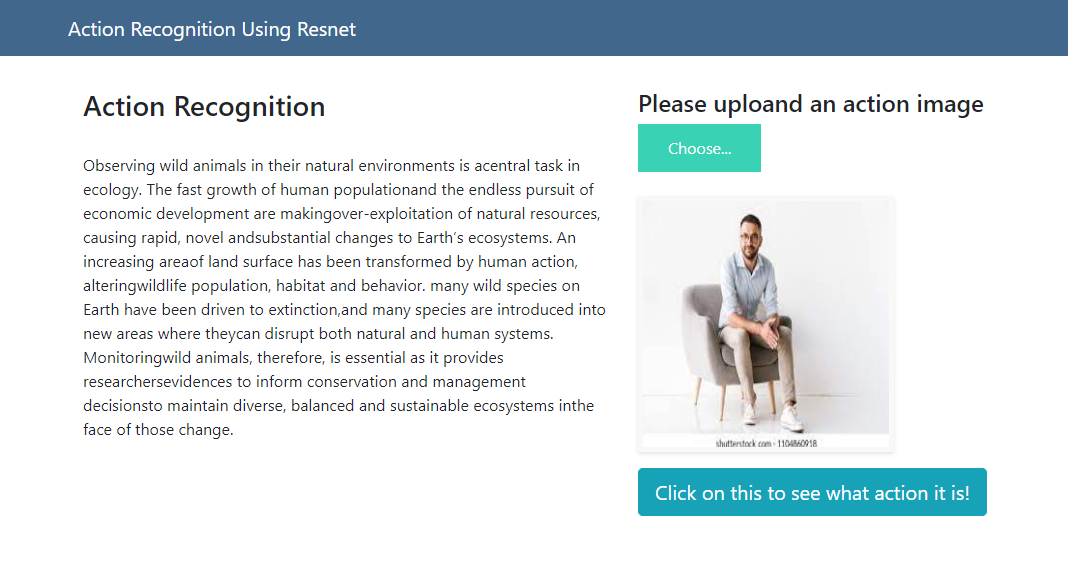




**Activity 3: Run The app in local browser**

* + Open Jupyter notebook from the start menu
  + Navigate to the folder where your python script is.
  + Now Press Ctrl+Enter
  + Navigate to the localhost where you can view your web page

**Activity 4: Showcasing prediction on UI**

****

