

PROJECT DOCUMENTATION

Classification Of Arrhythmia By Using Deep Learning With 2-D ECG Spectral Image Representation

TeamID - PNT2022MID31063

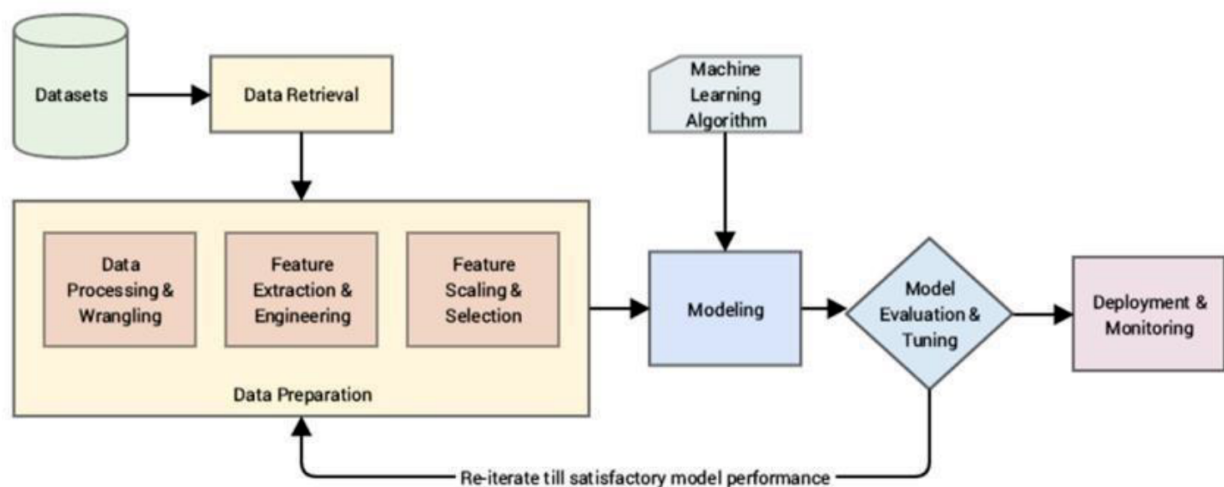
Team Leader-Abishek Govinthan K B

Team Member- Mukeshkumar V

Team Member- Fazil Ahamed J

Team Member- Muneeswaran I

Work Flow Diagram:



Project Objectives

By the end of this project you will:

- Know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
- Gain a broad understanding of image data.
- Work with Sequential type of modeling
- Work with Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

Project Flow

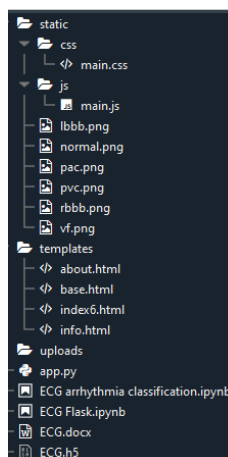
- User interacts with User interface to upload image
- Uploaded image is analyzed by the model which is integrated
- Once model analyses the uploaded image, the prediction is showcased on the UI

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 Testset
- Model Building
 - Import the model building Libraries
 - Initializing the model
 - Adding Input Layer
 - Adding Hidden Layer
 - Adding Output Layer
 - Configure the Learning Process
 - Training and testing the model
 - Optimize the Model
 - Save the Model
- Application Building
 - Create an HTML file
 - Build Python Code

Project Structure

Create a project folder which contains files given below:



We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for server side scripting.

we need the model which is saved and the saved model in this content is ECG.h5.

The static folder will contain js and CSS files.

Whenever we upload an image to predict, those images are saved in the uploads folder.

Prerequisites

To complete this project, you should have the following software and packages

Anaconda Naigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning

related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and Spyder

To build Deep learning models you must require the following packages

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras: Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Flask: Web frame work used for building Web applications

Prior Knowledge

One should have knowledge on the following Concepts:

- Supervised and unsupervised learning
- Regression Classification and Clustering
- Artificial Neural Networks
- Convolution Neural Networks
- Flask

Dataset Collection & Image Preprocessing

TeamID - PNT2022MID31063

Tasks

There are two tasks:

1. Dataset Collection
2. Image Preprocessing

Dataset Collection:

The dataset contains six classes:

1. Left Bundle Branch Block
2. Normal
3. Premature Atrial Contraction
4. Premature Ventricular Contractions
5. Right Bundle Branch Block
6. Ventricular Fibrillation

Image Preprocessing:

Image Pre-processing includes the following main tasks

1. Import ImageDataGenerator Library
2. Configure ImageDataGenerator Class
3. Apply ImageDataGenerator functionality to the trainset and testset

Import ImageDataGenerator Library:

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.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Configure ImageDataGenerator Class:

There are five main types of data augmentation techniques for image data, specifically:

1. Image shifts via the width_shift_range and height_shift_range arguments.
2. Image flips via the horizontal_flip and vertical_flip arguments.
3. Image rotates via the rotation_range argument.
4. Image brightness via the brightness_range argument.
5. Image zooms via the zoom_range argument.

```
|  
from keras.preprocessing.image import ImageDataGenerator  
from keras.preprocessing import image  
  
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)  
test_datagen=ImageDataGenerator(rescale=1./255)
```

An instance of the ImageDataGenerator class can be constructed for train and test.

Apply ImageDataGenerator functionality to the trainset and testset:

We will apply ImageDataGenerator functionality to Trainset and Testset by using the following code.

This function will return batches of images from the subdirectories Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contractions, Right Bundle Branch Block and Ventricular Fibrillation, together with labels 0 to 5

{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5}

We can see that for training there are 15341 images belonging to 6 classes and for testing there are 6825 images belonging to 6 classes.

```
x_train=train_datagen.flow_from_directory(directory=r'D:\Python\train I test\data\train',target_size=(64,64),batch_size=32,class_mode='categorical')
x_test=test_datagen.flow_from_directory(directory=r'D:\Python\train I test\data\test',target_size=(64,64),batch_size=32,class_mode='categorical')
```

Model Building

TeamID - PNT2022MID31063

Task

1. Model Building

We are ready with the augmented and pre-processed image data, we will begin our build our model by following the below steps:

Import The Libraries:

```
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
```

Initializing The Model:

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model.

In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, will initialize our model

Adding CNN Layer:

We are adding a convolution layer with an activation function as “relu” and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer.

The Max pool layer is used to downsample the input. The flatten layer flattens the input.

```
model=Sequential()  
model.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Flatten())
```

Adding Dense & Output Layer:

Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

```
model.add(Dense(32))  
model.add(Dense(units=128, kernel_initializer="random_uniform"))
```


Understanding the model is very important phase to properly use it for training and prediction purposes . keras provides a simple method, summary to get the full information about the model and its layers

```
model.summary()
```

```
>>> Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

===== RESTART: D:/Python/train I test/train.py =====
Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
Model: "sequential"

Layer (type)                 Output Shape              Param #
-----
conv2d (Conv2D)              (None, 62, 62, 32)       896
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)       0
flatten (Flatten)            (None, 30752)            0
dense (Dense)                (None, 32)               984096
dense_1 (Dense)              (None, 128)              4224
dense_2 (Dense)              (None, 6)                774
-----
Total params: 989,990
Trainable params: 989,990
Non-trainable params: 0
```

Configure The Learning Process:

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation processes

Optimization is an important process that optimizes 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 the training process.

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

Train The Model:

We will train our model with our image dataset. `fit_generator` functions used to train a deep learning neural network.

```
model.fit(x_train, steps_per_epoch=len(x_train), epochs=10, validation_data=x_test, validation_steps=len(x_test))
```

```
Epoch 1/10
1/480 [.....] - ETA: 45:06 - loss: 1.8828 - accuracy: 0.1562
..... 2/480 [.....] - ETA: 14:08 - loss: 5.2590 - accuracy: 0.3438
..... 3/480 [.....] - ETA: 16:11 - loss: 6.3511 - accuracy: 0.3954
..... 4/480 [.....] - ETA: 15:41 - loss: 6.1634 - accuracy: 0.3359
..... 5/480 [.....] - ETA: 15:30 - loss: 5.4935 - accuracy: 0.2937
..... 6/480 [.....] - ETA: 15:44 - loss: 4.8936 - accuracy: 0.2708
..... 7/480 [.....] - ETA: 15:51 - loss: 4.6100 - accuracy: 0.2634
..... 8/480 [.....] - ETA: 16:13 - loss: 4.4702 - accuracy: 0.2539
..... 9/480 [.....] - ETA: 16:44 - loss: 4.1128 - accuracy: 0.2986
..... 10/480 [.....] - ETA: 16:38 - loss: 3.9533 - accuracy: 0.3063
..... 11/480 [.....] - ETA: 16:41 - loss: 3.8012 - accuracy: 0.3182
..... 12/480 [.....] - ETA: 16:44 - loss: 3.6622 - accuracy: 0.3229
..... 13/480 [.....] - ETA: 16:41 - loss: 3.5167 - accuracy: 0.3365
..... 14/480 [.....] - ETA: 16:41 - loss: 3.5167 - accuracy: 0.3365
```

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.

```
model.save('ECG.h5')
```

Test The Model:

Load necessary libraries and load the saved model using `load_model`. Taking an image as input and checking the results.

The target size should for the image that is should be the same as the target size that you have used for training.

```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
```

```
model=load_model("ECG.h5")
img=image.load_img("D:/Python/train I test/data/test/Normal/fig_2114.png",target_size=(64,64))
```

```
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict_classes(x)
pred
```

```
index=["Left Bundle Branch Block","Normal","Premature Atrial Contraction","Premature Ventricular Contractions","Right Bundle Branch Block","Ventricular Fibrillation"]
result=str(index[pred[0]])
result
```

Application Building

Team ID - PNT2022TMID31063

1.Create HTML Files

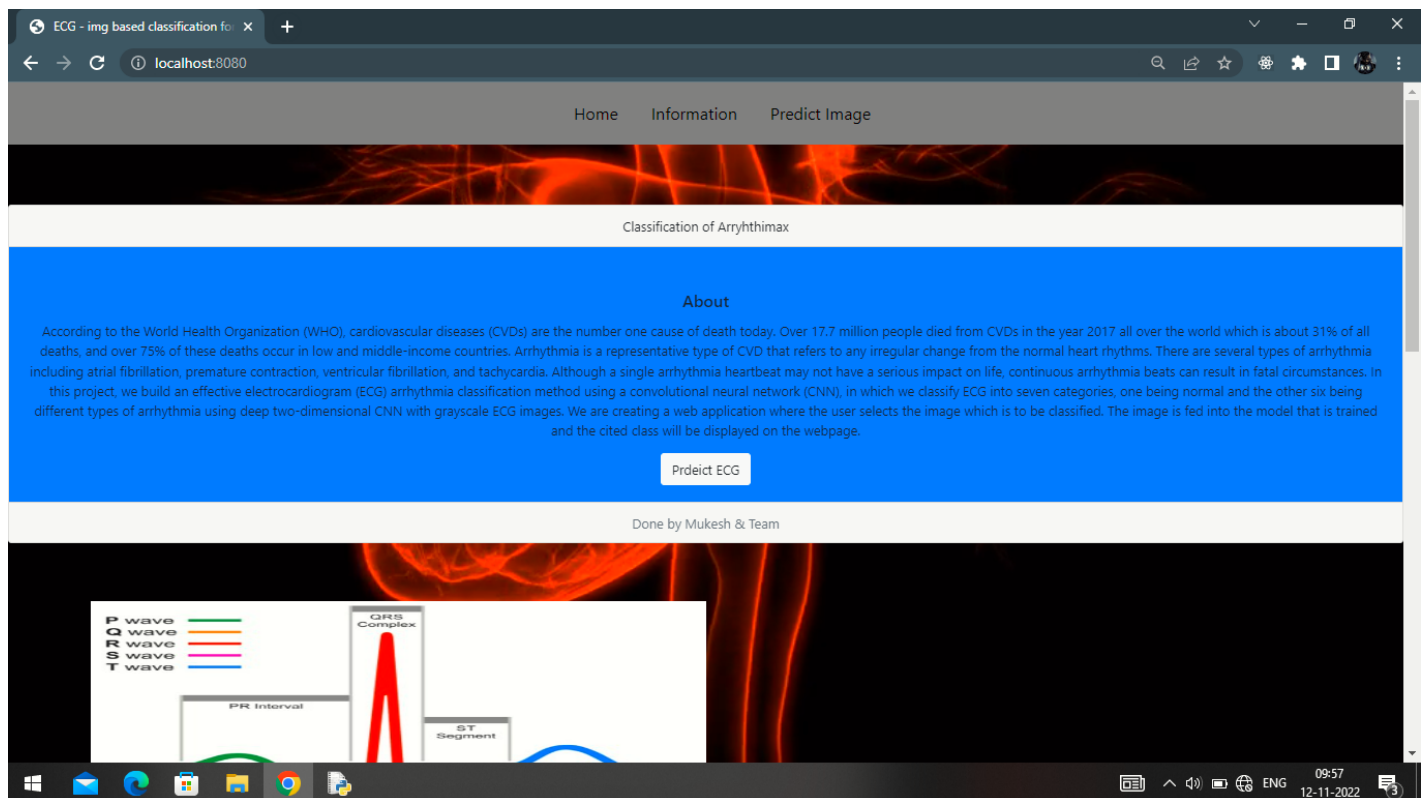
We use HTML to create the front end part of the web page.

Here, we created 4 html pages- upload.html, predict.html, home.html, info.html. home.html displays the home page.

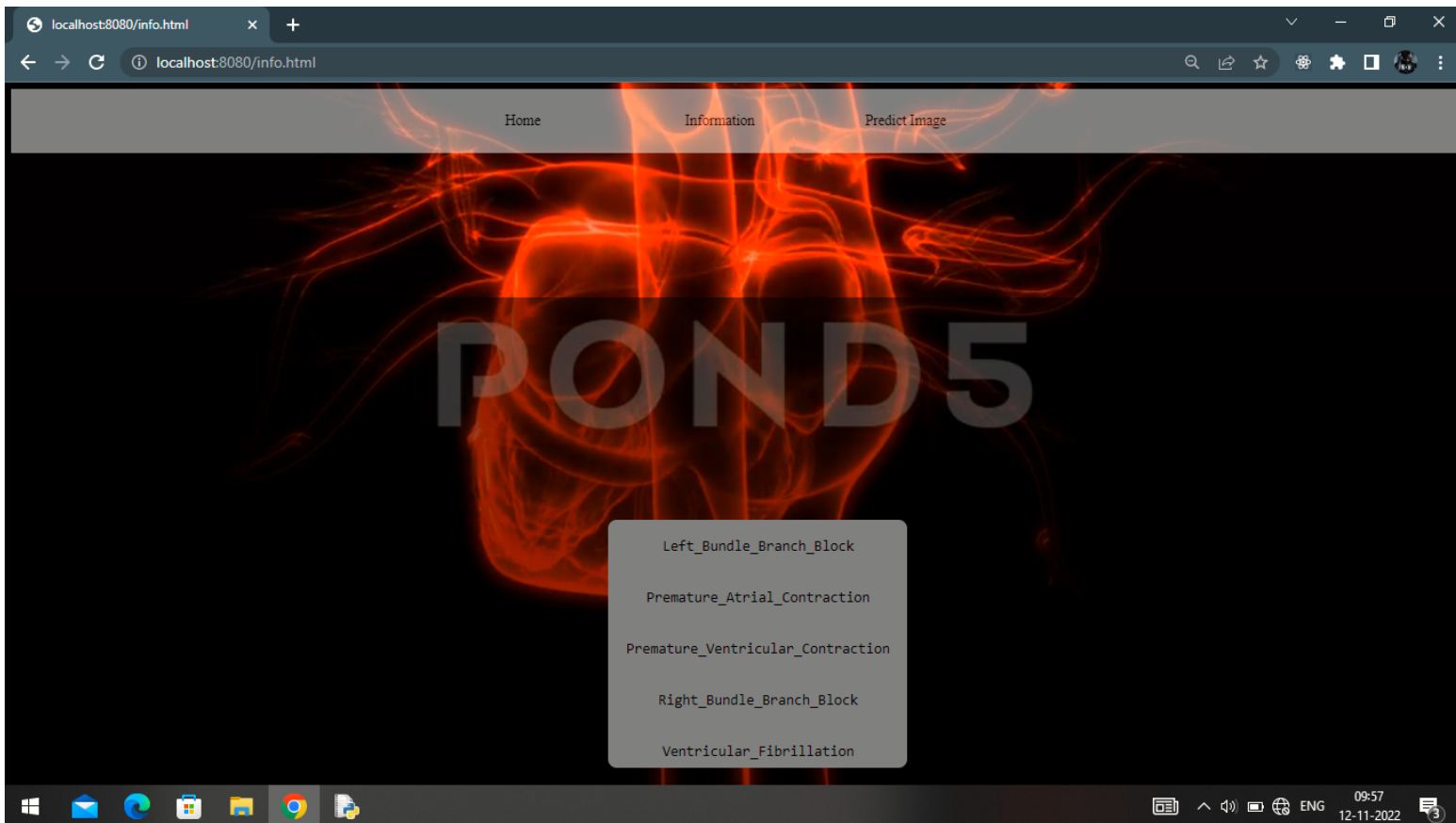
Info.html displays all important details to be known about ECG.

upload.html and predict.html accept input from the user and predicts the values.

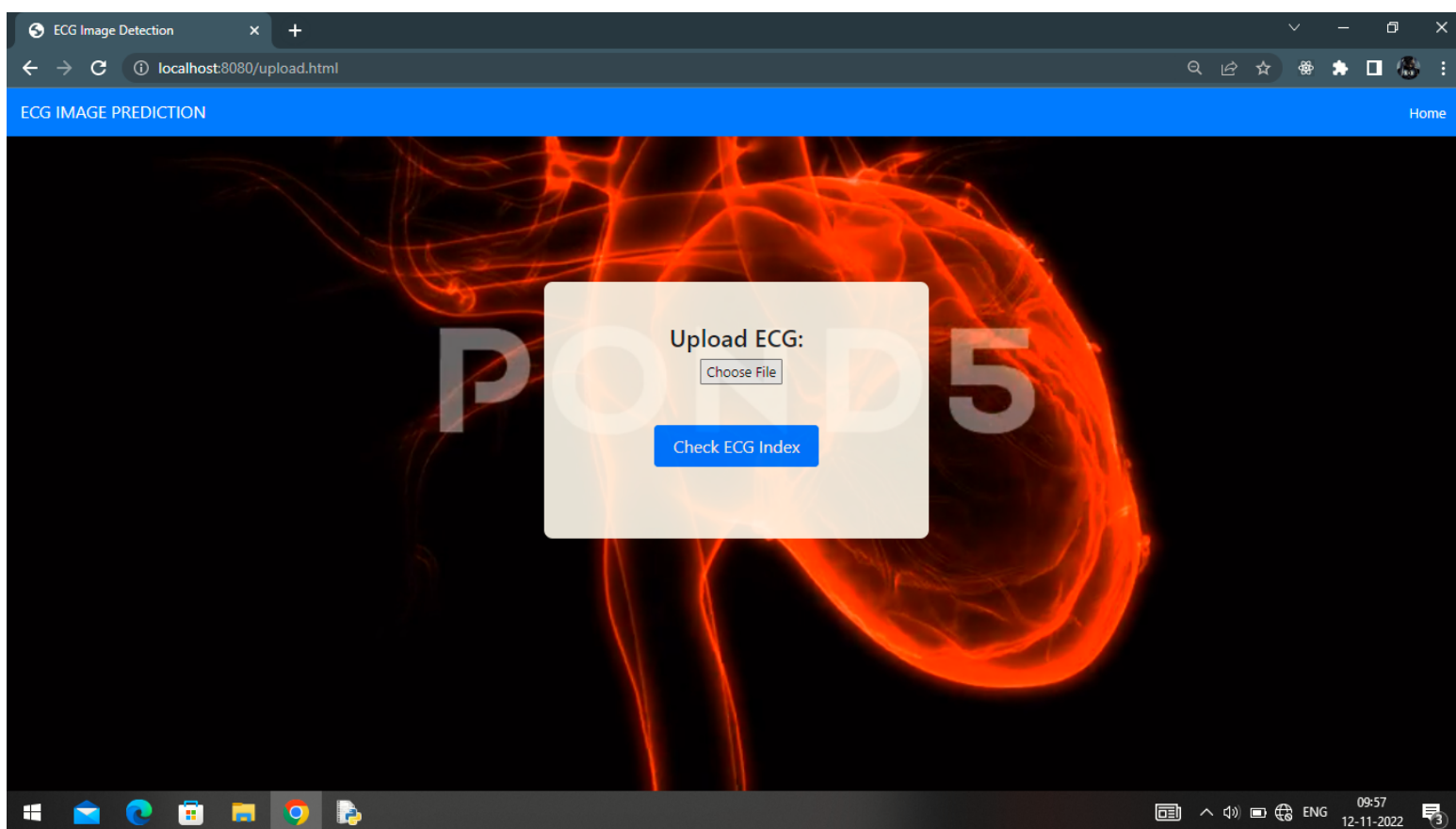
Home.html looks Like this



Info.html looks like this



upload.html look like this



2.Build Python Code

Let us build the flask file 'app.py' which is a web framework written in python for server-side scripting.

Let's see step by step procedure for building the backend application.

1 . The app starts running when the “__name__” constructor is called in main.

2. `render_template` is used to return HTML file.
3. "GET" method is used to take input from the user.
4. "POST" method is used to display the output to the user.

App.py

Import the libraries

```
app.py - D:\Python\Project Development\Sprint-4\app.py (3.10.7)
File Edit Format Run Options Window Help

import os
import numpy as np
import tensorflow as tf
from flask import Flask, request, render_template, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

Routing to the HTML Page

```
app=Flask(__name__)
model=load_model('ECG.h5')

@app.route('/')
@app.route('/home.html')
def about():
    return render_template("home.html")
@app.route('/info.html')
def info():
    return render_template('info.html')
@app.route("/upload.html")
def test():
    return render_template("upload.html")
@app.route("/predict.html/<result>/<filepath>")
def test1(result,filepath):
    return render_template("predict.html",result=result,filepath=filepath)
@app.route("/Left_Bundle_Branch_Block.html")
def Left_Bundle_Branch_Block():
    return render_template("Left_Bundle_Branch_Block.html")
@app.route("/Premature_Atrial_Contraction.html")
def Premature_Atrial_Contraction():
    return render_template("Premature_Atrial_Contraction.html")
@app.route("/Premature_Ventricular_Contractions.html")
def Premature_Ventricular_Contractions():
    return render_template("Premature_Ventricular_Contractions.html")
@app.route("/Right_Bundle_Branch_Block.html")
def Right_Bundle_Branch_Block():
    return render_template("Right_Bundle_Branch_Block.html")
@app.route("/Ventricular_Fibrillation.html")
def Ventricular_Fibrillation():
    return render_template("Ventricular_Fibrillation.html")
```

Showcasing prediction on UI

When the image is uploaded, it predicts the category of uploaded the image is either 'Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation'. If the image predicts value as 0, then it is displayed as "Left Bundle Branch". Similarly, if the predicted value is 1, it displays "Normal" as output and so on.

```
@app.route('/upload.html', methods=['GET', 'POST'])
def upload():
    if request.method=='POST':
        f=request.files['file']
        if f.filename=='':
            flash('No file selected')
        else:
            print("Analysing...")
            basepath=os.path.dirname('__file__')
            filepath=os.path.join(basepath,"static\\uploads",f.filename)
            f.save(filepath)

            img=tf.keras.utils.load_img(filepath,target_size=(64,64))
            x=tf.keras.utils.img_to_array(img)
            x=np.expand_dims(x,axis=0)
            print("Predicting...")
            pred=model.predict(x)
            classes_x=np.argmax(pred,axis=1)
            print(classes_x)

            index=["Left Bundle Branch Block","Normal","Premature Atrial Contraction","Premature Ventricular Contractions","Right Bundle Branch Block","Ventricular Fibrillation"]
            result=str(index[classes_x[0]])
            print("Prediction Done...")

            return render_template('predict.html',result=result,filepath="static/uploads/"+f.filename)
    return None
if __name__=="__main__":
    from waitress import serve
    serve(app, host="0.0.0.0", port=8080)
```

Run The APP

Open Python IDLE from the start menu

Navigate to the folder where your python script is.

Now type app.py" command

Navigate to the localhost where you can view your web page

Then it will run on localhost:8080

Navigate to the localhost:8080 where you can view your web page.

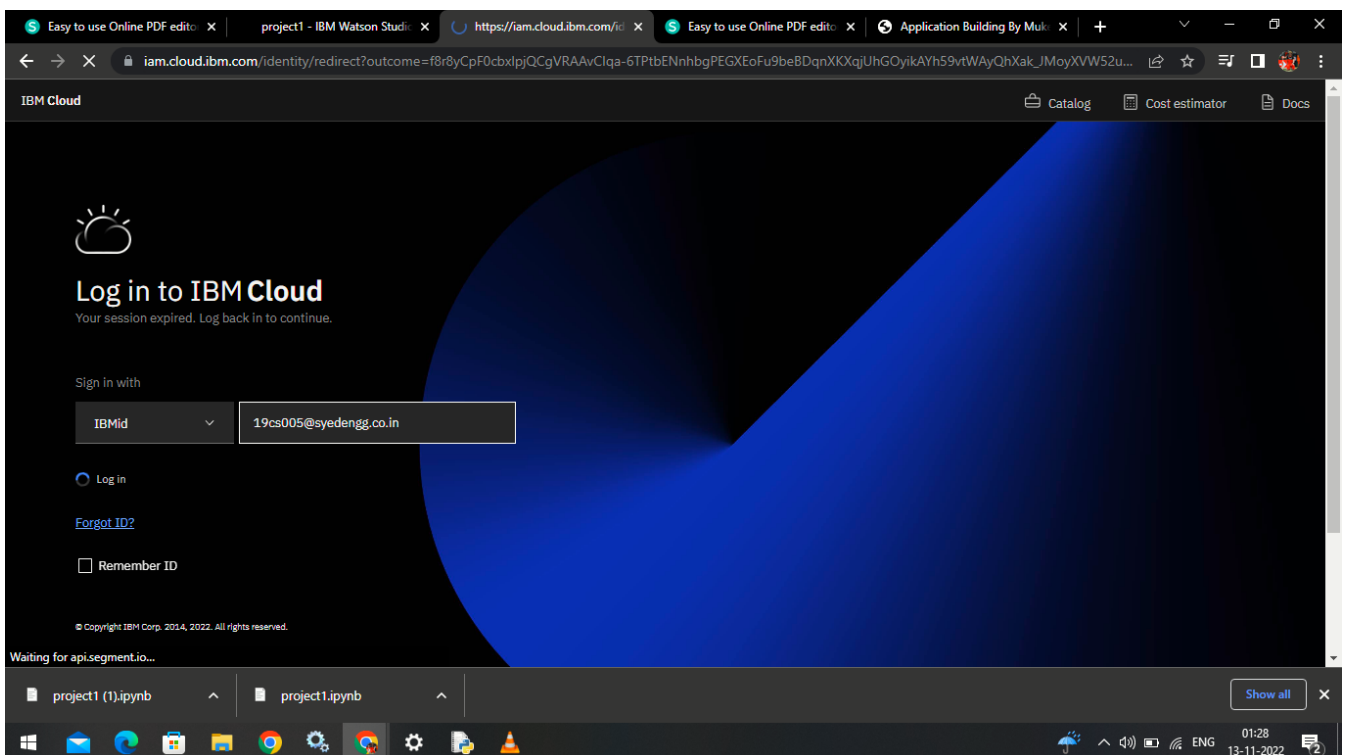
```
===== RESTART: D:\Python\SI-GuidedProject-89222-1657968896-main\app.py =====
serving on http://0.0.0.0:8080
Analysing...
Predicting...
1/1 [=====] - ETA: 0s [=====]1/1 [=====] - 5s 5s/step
[4]
Prediction Done...
Your have been Detected as Right Bundle Branch Block
Analysing...
Predicting...
1/1 [=====] - ETA: 0s [=====]1/1 [=====] - 0s 152ms/step
[3]
Prediction Done...
Your have been Detected as Premature Ventricular Contractions
Analysing...
Predicting...
1/1 [=====] - ETA: 0s [=====]1/1 [=====] - 0s 120ms/step
[0]
Prediction Done...
Your have been Detected as Left Bundle Branch Block
Analysing...
Predicting...
1/1 [=====] - ETA: 0s [=====]1/1 [=====] - 0s 64ms/step
[1]
Prediction Done...
Your have been Detected as Normal
```

Training & Testing Model on IBM cloud

Team ID - PNT2022TMID31063

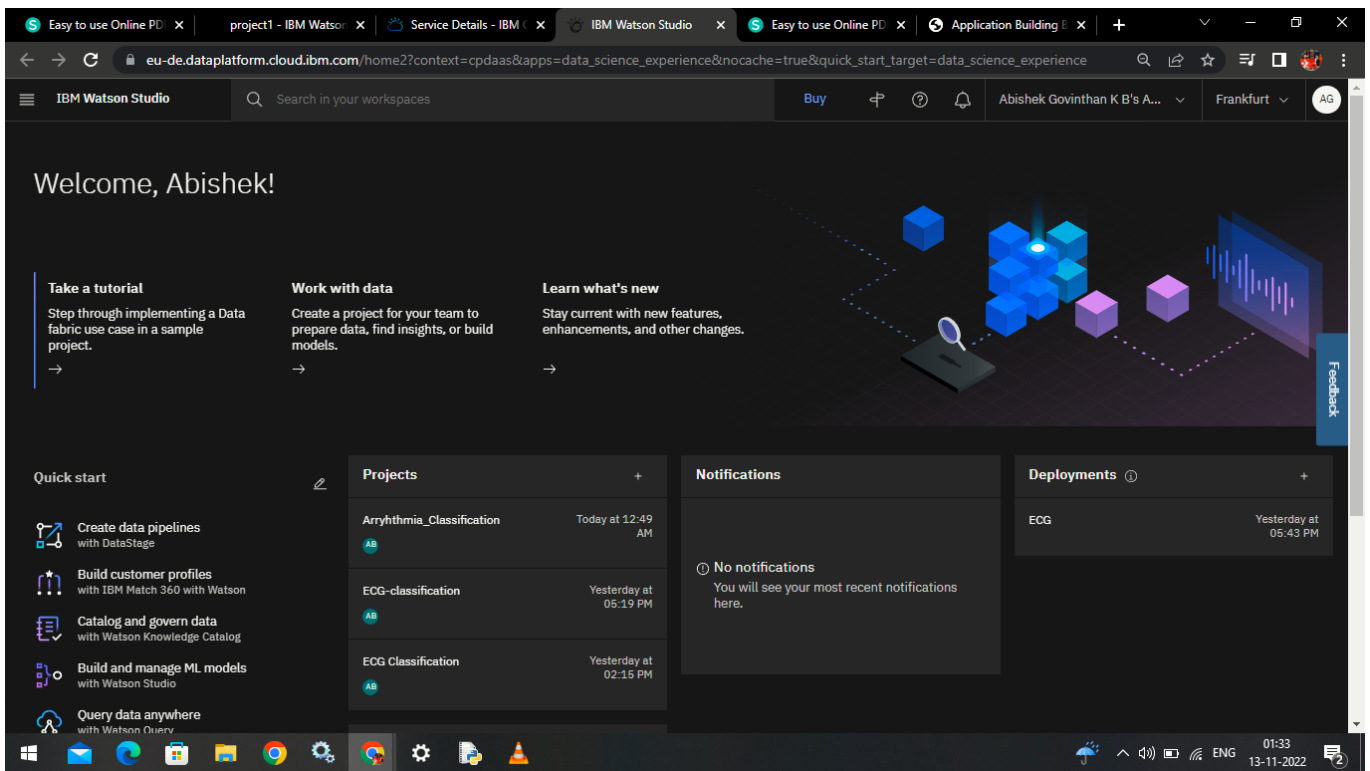
1.Register & Login IBM cloud

From given link we can Register and afterwards login the IBM cloud using credentials.



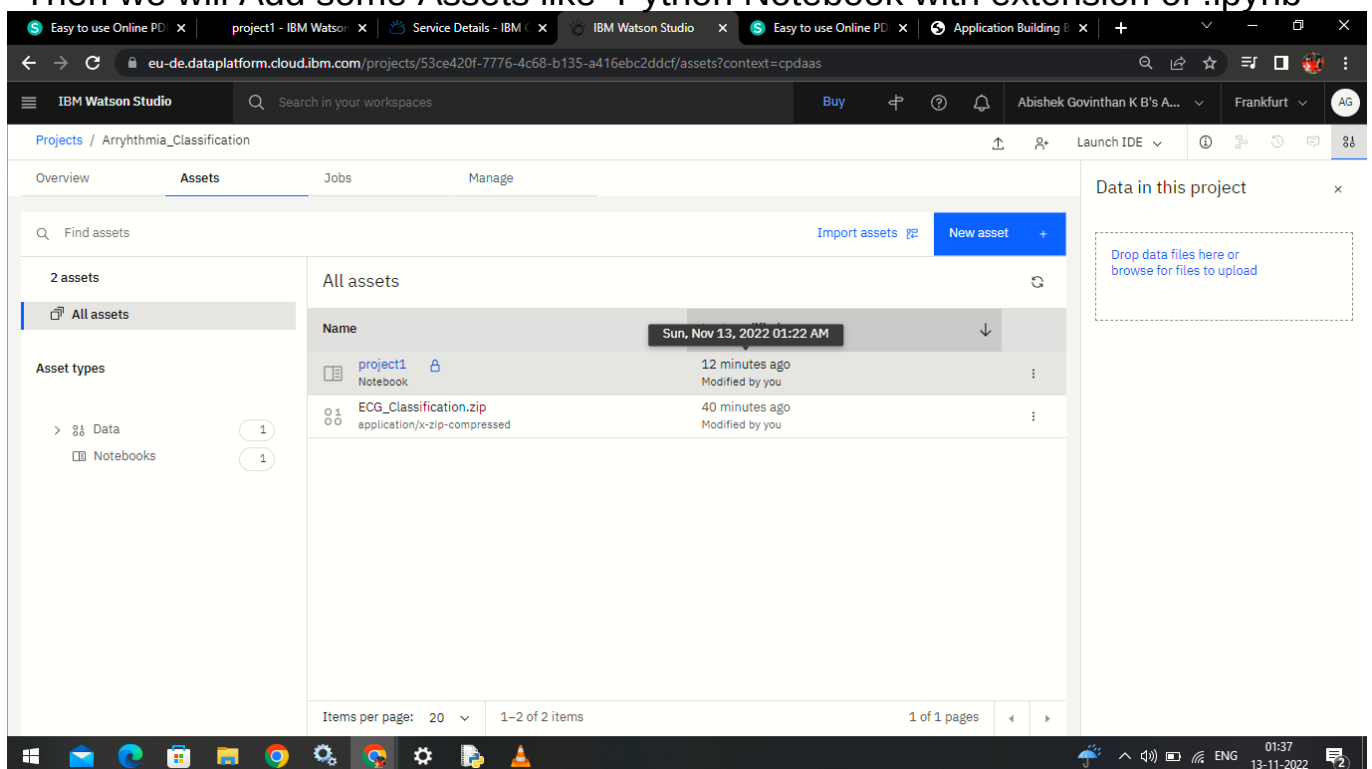
2.Opening IBM Watson Studio

To Run our model we use IBM watson studio inside we will create our own new project.



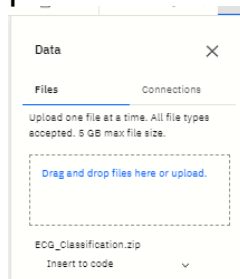
3.Adding Assests

Once we create a new project named Arryhythmia_Classification
Then we will Add some Assets like Python Notebook with extension of .ipynb



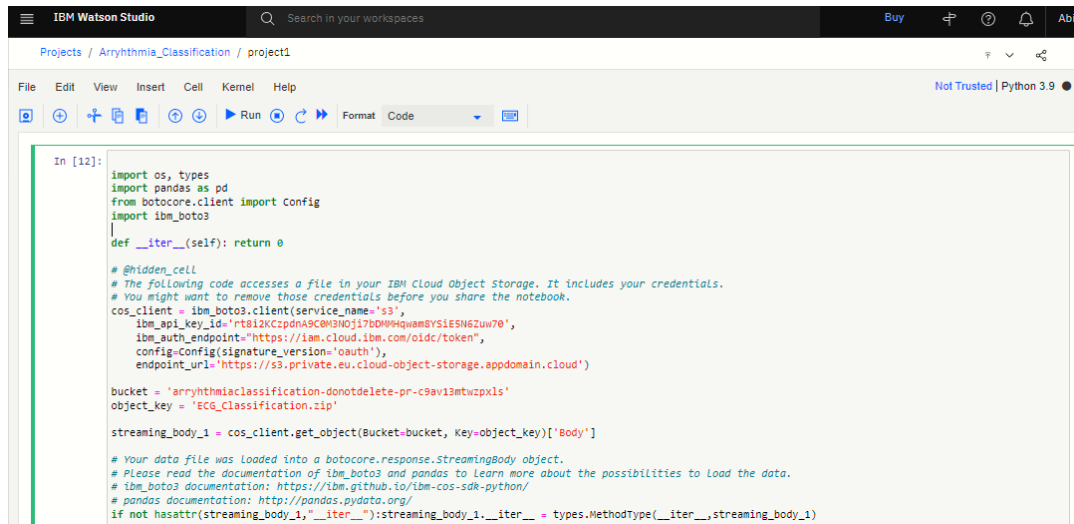
4 .Opening notebook file

At the end we will open the note book file and after we download the dataset
once we download it into zip file we will add the file into data assests.



5. Insert File code

Then we will create a new cell and insert the code into it.



```
In [12]:
import os, types
import pandas as pd
from botocore.client import Config
import boto3

def __iter__(self): return 0

#@hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = boto3.client(service_name='s3',
    iam_api_key_id='rt812K2CpdnA5CM3NOj17BDMMHqWamsYSIESN6Zuw70',
    iam_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

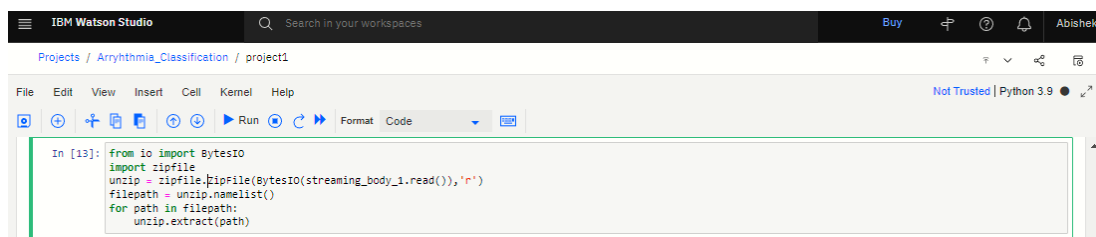
bucket = 'arrythmiaclassification-donotdelete-pr-c9av13mtwzpx1s'
object_key = 'ECG_Classification.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of boto3 and pandas to learn more about the possibilities to load the data.
# boto3 documentation: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html#python
# pandas documentation: http://pandas.pydata.org/
if not hasattr(streaming_body_1, '__iter__'): streaming_body_1.__iter__ = types.MethodType(__iter__, streaming_body_1)
```

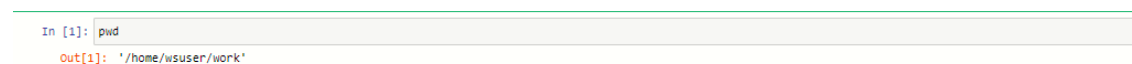
6. Unzip the file

Once we store the file we will deprecate or unzip the file from the beginning.



```
In [13]:
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
filepath = unzip.namelist()
for path in filepath:
    unzip.extract(path)
```

Know About the directory by using pwd command.



```
In [1]: pwd
Out[1]: '/home/wsuser/work'
```

7. Train the Model in IBM Watson Studio

After Completion of these stuffs we will move forward to start Train the model.



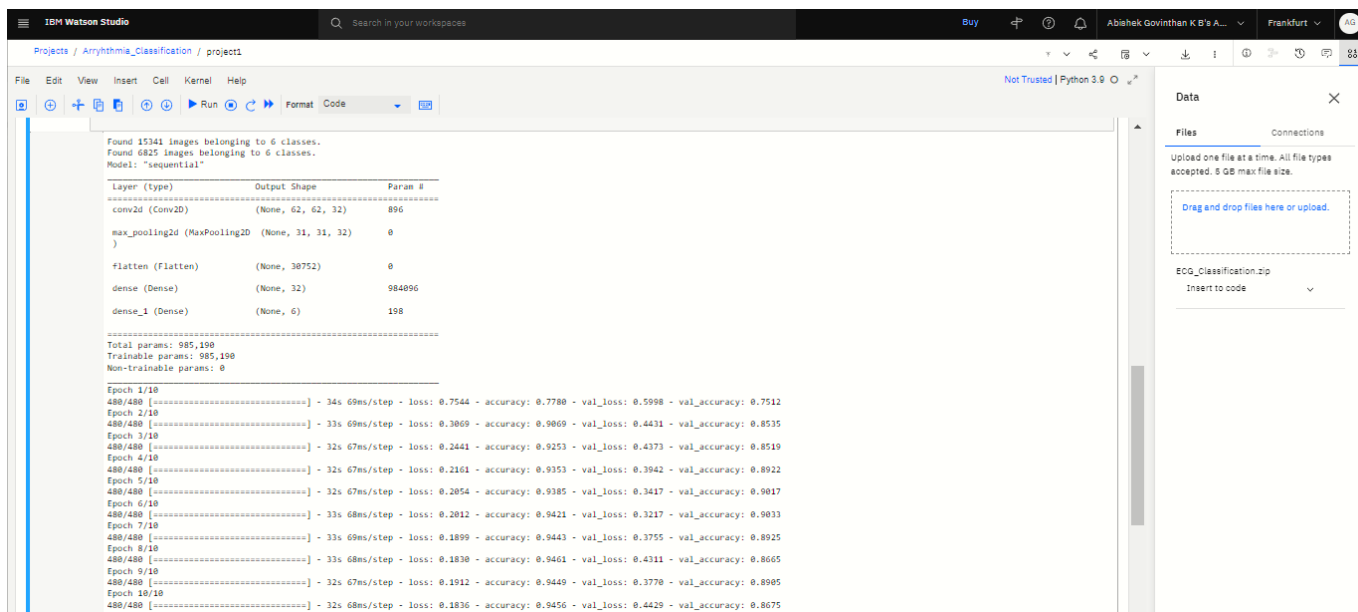
```
In [*]:
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
x_train=train_datagen.flow_from_directory(directory=r'/home/wsuser/work/data/train', target_size=(64,64), batch_size=32, class_mode='categorical')
x_test=test_datagen.flow_from_directory(directory=r'/home/wsuser/work/data/test', target_size=(64,64), batch_size=32, class_mode='categorical')

import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
model=Sequential()
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(32))
model.add(Dense(6,activation='softmax'))
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(x_train, steps_per_epoch=len(x_train), epochs=10, validation_data=x_test, validation_steps=len(x_test))

model.save('ECG.h5')
```

Outcome of the Trained Model will be shown after compiling and summarizing it.



```
Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
Model: "sequential"

Layer (type)                 Output Shape              Param #
-----
conv2d (Conv2D)              (None, 62, 62, 32)       896
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)       0
flatten (Flatten)            (None, 30752)             0
dense (Dense)                (None, 32)               984896
dense_1 (Dense)              (None, 6)                198

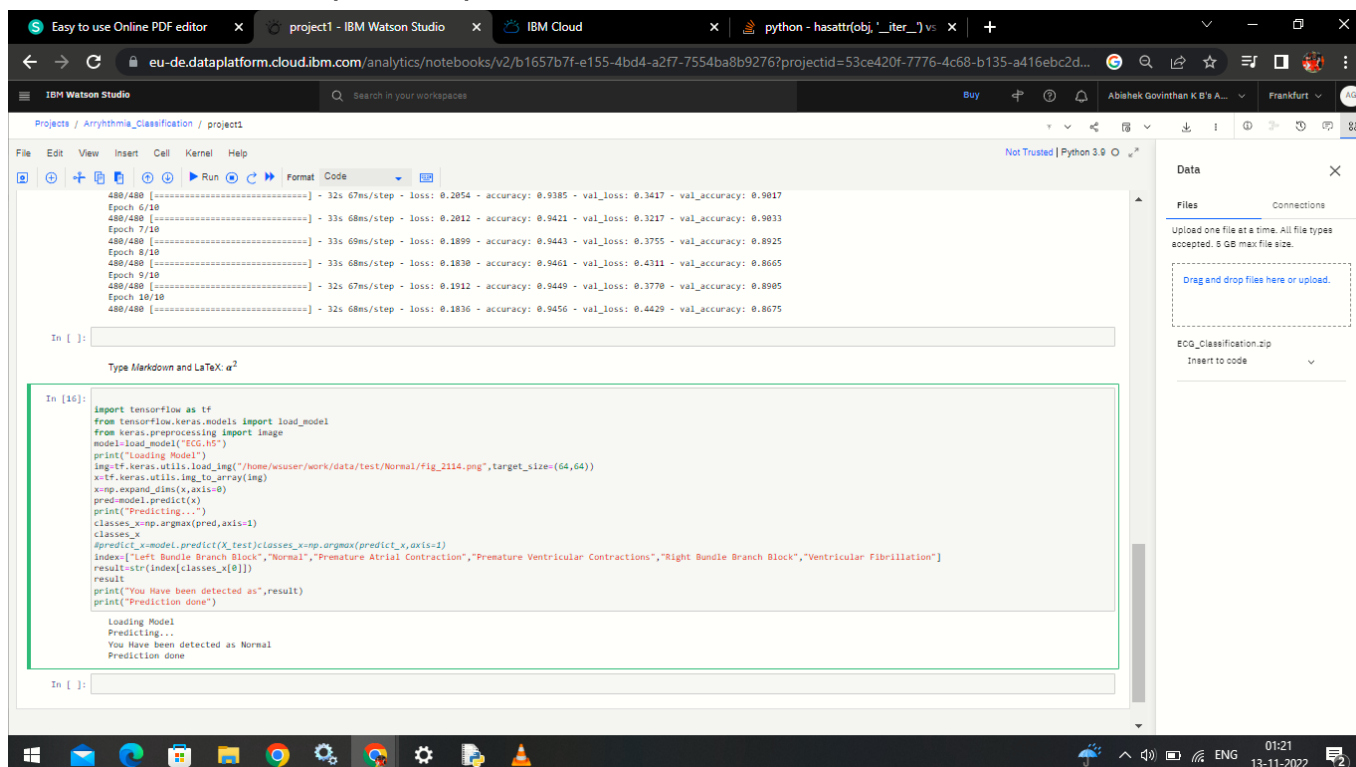
Total params: 985,190
Trainable params: 985,190
Non-trainable params: 0

Epoch 1/10
480/480 [=====] - 34s 69ms/step - loss: 0.7544 - accuracy: 0.7780 - val_loss: 0.5998 - val_accuracy: 0.7512
Epoch 2/10
480/480 [=====] - 33s 69ms/step - loss: 0.3069 - accuracy: 0.9069 - val_loss: 0.4431 - val_accuracy: 0.8535
Epoch 3/10
480/480 [=====] - 32s 67ms/step - loss: 0.2441 - accuracy: 0.9253 - val_loss: 0.4373 - val_accuracy: 0.8519
Epoch 4/10
480/480 [=====] - 32s 67ms/step - loss: 0.2161 - accuracy: 0.9353 - val_loss: 0.3942 - val_accuracy: 0.8922
Epoch 5/10
480/480 [=====] - 32s 67ms/step - loss: 0.2054 - accuracy: 0.9385 - val_loss: 0.3417 - val_accuracy: 0.9017
Epoch 6/10
480/480 [=====] - 33s 68ms/step - loss: 0.2012 - accuracy: 0.9421 - val_loss: 0.3217 - val_accuracy: 0.9033
Epoch 7/10
480/480 [=====] - 33s 69ms/step - loss: 0.1899 - accuracy: 0.9443 - val_loss: 0.3755 - val_accuracy: 0.8925
Epoch 8/10
480/480 [=====] - 33s 68ms/step - loss: 0.1830 - accuracy: 0.9461 - val_loss: 0.4311 - val_accuracy: 0.8665
Epoch 9/10
480/480 [=====] - 32s 67ms/step - loss: 0.1912 - accuracy: 0.9449 - val_loss: 0.3778 - val_accuracy: 0.8985
Epoch 10/10
480/480 [=====] - 32s 68ms/step - loss: 0.1836 - accuracy: 0.9456 - val_loss: 0.4429 - val_accuracy: 0.8675
```

This how the Model was Trained in IBM watson Studio.

8. Testing the Model

At the end of the day we will try to test the model which can give the desired and precise prediction based on the trained model.



```
In [16]:
import tensorflow as tf
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model=load_model('ECG.h5')
print('Loading Model')
img=tf.keras.utils.load_img('/home/wuser/work/data/test/Normal/fig_2114.png',target_size=(64,64))
x=tf.keras.utils.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=model.predict(x)
print("Predicting...")
classes_x=np.argmax(pred,axis=1)
classes_x
#predict_x=model.predict(X_test)classes_x=np.argmax(predict_x,axis=1)
index=[0,1,2,3,4,5]
result=str(index[classes_x[0]])
print("You Have been detected as",result)
print("Prediction done")

Loading Model
Predicting...
You Have been detected as Normal
Prediction done
```

This How the Model was tested in IBM Watson Studio.

Regards
Mukesh & Team

PHASES

Team ID - PNT2022TMID31063

Ideation Phase

In this milestone you are expected to get started with the Ideation process.

Literature Survey on The Selected Project & Information Gathering

In this activity you are expected to gather/collect the relevant information on project use case, refer the existing solutions, technical papers, research publications etc.

Prepare Empathy Map

In this activity you are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.

Ideation

In this activity you are expected to list the ideas (at least 4 per each team member) by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.

Project Design Phase – I

From this milestone you will be starting the project design phase. You are expected to cover the activities given.

Proposed Solution

In this activity you are expected to prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.

Problem Solution Fit

In this activity you are expected to prepare problem - solution fit document and submit for review.

Solution Architecture

In this activity you are expected to prepare solution architecture document and submit for review.

Project Design Phase -II

From this milestone you will be continue working on the project design phase. You are expected to cover the activities given.

Customer Journey

Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).

Functional Requirement

In this activity you are expected to prepare the functional requirement document.

Data Flow Diagrams

In this activity you are expected to prepare the data flow diagrams and submit for review.

Technology Architecture

In this activity you are expected to draw the technology architecture diagram.

Project Planning Phase

In this milestone you are expected to prepare milestones & tasks, sprint schedules.

Prepare Milestone & Activity List

In this activity you are expected to prepare the milestones & activity list of the project.

Sprint Delivery Plan

In this activity you are expected to prepare the sprint delivery plan.

Project Development Phase

In this milestone you will start the project development and expected to perform the coding & solutioning, acceptance testing, performance testing based as per the sprint and submit them.

Project Development - Delivery of Sprint-1

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-2

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-3

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-4

In this activity you are expected to develop & submit the developed code by testing it.

Done by - Abishek Govinthan and Team