Body Language/Pose Detection

Submitted to,

FeyNN Labs

REPORT BY :

1. Aaryan Palit
2. Aadarsh C R
3. Syed Mohsin
4. Premonvitha Sai

Date of Submission: 11th Dec,2022

STEP-1: Prototype Selection

1. Problem Statement:

Body pose detection works best when the subject's entire body is visible in the frame, but it also detects a partial body pose. Pose estimation is a popular task in Computer Vision. As a field of artificial intelligence (AI), computer vision enables machines to perform image processing tasks with the aim of imitating human vision.In that case the landmarks that are not recognized are assigned coordinates outside of the image. This idea can be used to implement in FITNESS INDUSTRY. It is used to identify human joints and provide the user with guidance on how to exercise the right way. For example, head pose estimation is essential when the user is doing a plank. The app estimates the position of the head to avoid injuries while exercising.

2. Market/Customer/Business Need Assessment:

Pose detection is an active research area in the field of computer vision. The reason so many machine learning enthusiasts are drawn to pose estimation is the wide variety of applications and utilities. In today's world human body language/pose detection can be found from areas of surveillance to AI driven physiotherapy treatments, from yoga/fitness apps to animatronics. This is also a reason that there exists a number of research in this field so that they can make the life of people a little easier and simpler by automating machine for certain types of jobs. Thus, thereis a considerable place with a wide range of applications of the pose detection modelin the market.

3. Target Specifications and Characterization:

Body language detection are often thought of a quite mature technology since thereare groundworks within the areas of applications like behavioral analysis, fitness, rehabilitation, animation, robotics, and even surveillance. Some of the popular use cases in market are:

- SURVEILLANCE- Cameras can be used to automate everyday processes like shopping at a grocery store or to analyze suspicious activities of people in airport.

- AI SELF-COACHING in FITNESS- Fitness applications are another usecase for body pose estimation. The model enforced within the phone appwill use the hardware camera as a sensing element to record somebody doing an exercise and perform its analyses.

4. Benchmarking Alternate Products:

Today a number AI empowered pose/body language detection is used in quiet a few fields. A Japanese telecom company, Nippon Telegraph and Telephone (NTT),has partnered with a startup to build an AI that can be used to detect shoplifters in any store. This AI, being called 'AI Guardman', is built into the CCTV cameras that you see in every store these days.

If take the fitness industry in consideration then we can see that it also uses the power of machine learning and AI to help people with their fitness. Some popular examples of applying pose estimation in fitness are Kaia, VAI Fitness Coach, Allyapps, or the Millie Fit device. Every year the revenue of the fitness industry growsby 8.7%. Millennials use fitness apps more than other age groups, with women using them twice as much as men. 46% want as much quantifiable data about theirhealth as possible, and 54% are likely to buy a body-analyzing device.

An immensely popular Deep Learning app "Homecourt" uses Pose Estimation to analyses Basketball player movements. ITCL Technology Center Simulation and Intelligent Perception Technologies team is leading the European Working Age Project that isdeveloping a computer vision system to detect the joints of people reliably and accurately in their workplace.

5. Applicable Patents Tech/Software/Framework:

The list of applicable patents of tech/software/framework are:

1. Pose Detection: Pose detection is an open-source real-time pose detectionlibrary that can detect human poses in images or videos. It is a pose estimator architecture built on tensorflow.js and allows you to detect bodyparts such as elbows, hips, wrists, knees, ankles, and others for either a single pose or multiple poses.
2. OpenPose: OpenPose is a free human joints detection library that works in real-time. It detects key points for the body, face, hands, and foot estimation.It is the first multi-person system to jointly detect 135 key points in total on a single input image. It is one of the most popular multi-person human pose estimation libraries that uses a bottom-up approach.
3. Dense Pose: Dense human pose estimation is a free, open-source library thatcan

map all human pixels of 2D RGB images to a 3D surface-based model of the body in real-time. This library is implemented in the detectron framework, powered by caffe2, and can also be used for single and multiplepose estimation problems.

4. Alpha Pose: Alphapose also provides an efficient online pose tracker to associate poses that indicate the same person across frames. It is the first open-sourced online pose tracker and is called PoseFlow. This library candetect accurate real-time multi-person and single-person keypoints in images, videos, and image lists.
5. COCO Key points: a large-scale image dataset containing 328,000 images ofeveryday objects and humans. The dataset contains annotations you can use to train machine learning models to recognize, label, and describe objects
6. Human Eva: The Human Eva-I dataset contains 7 calibrated video sequences (4 grayscale and 3 colour) that are synchronized with 3D body poses obtainedfrom a motion capture system. The database contains 4 subjects performing a 6 common actions (e.g. walking, jogging, gesturing, etc.). The dataset contains training, validation and testing (with withheld ground truth) sets.

6. Applicable Regulations:

- Data Security: Faces are becoming easier to capture from remote distancesand cheaper to collect and store. Unlike many other forms of data, faces cannot be encrypted
- Individual privacy rights: "Faces ... are central to our identity." a government that's capable of tracking your face wherever you are/is capableof tracking your location wherever you are, which means it's capable of tracking every association you have.
- Tracking: Facial recognition is unlike other tracking methods—such as carrying a mobile phone or wearing a Fitbit—because consumers cannot easily avoid unwanted tracking of their face. And while most consumers findit unacceptable to use it for commercial purposes.
- Data privacy laws: The Niti Aayog has come up with a list of seven principles for responsible AI that includes principles of safety and reliability,equality, inclusivity and non-discrimination, privacy and security, transparency, accountability and protection and reinforcement of positive human values. Thus, taking these into data must be protected.

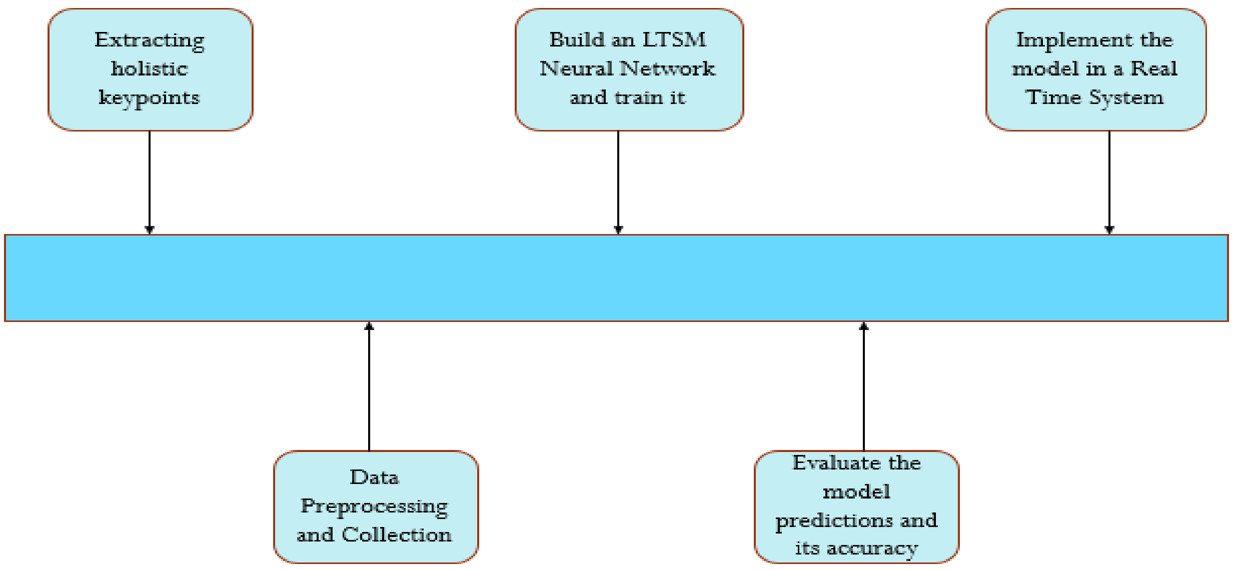
7. Applicable Constraints:

The applicable constraints are:

- Model Training and Testing must be done properly to provide a viablemodel with a good accuracy.
- Implementing the model in real world and in real time would require aproper data base to deal with
- Data processing is required and is a very hectic task
- Approaching a relevant surveillance, fitness, animatronics etc. must be doneto implement this model within real life.
- Regular testing and maintenance of model necessary

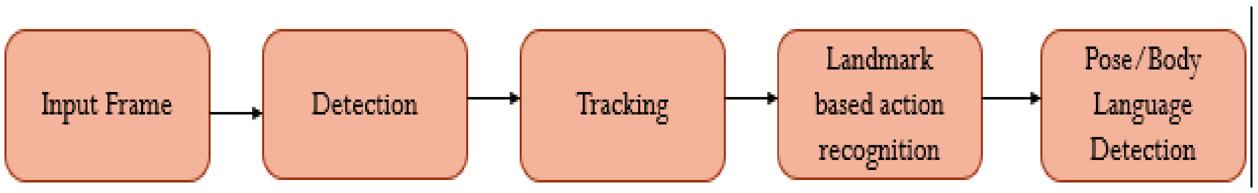
8. Final Product Prototype with Schematic Diagram:

- **Plan Structure:**



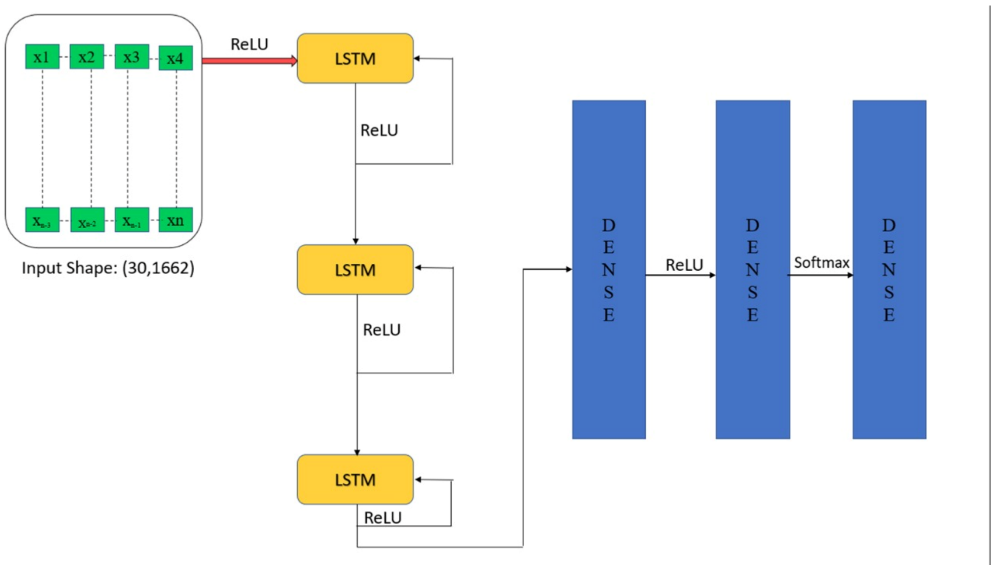
Extracting holistic keypoints
Build an LTSM Neural Network and train it
Implement the model in a Real Time System
Data Preprocessing and Collection
Evaluate the model predictions and its accuracy

- **Working mechanism of our Model:**



Input Frame
Detection
Tracking
Landmark based action recognition
Pose/Body Language Detection

- **Neural Network Architecture:**



9.Product Details:

9.1 How does it work?

The first thing we need to do is extract the key points of the human body with the help of media pipe. We take the input frames from a real time OpenCV window to collect the data. The next thing we do is to preprocess the data and provide them with labels of specifics action we want the action to recognize. Then we need to build our LSTM Neural Network Architecture and train it with the data we collected. After this we need to save the model and the training in a h5 file, so thatwe can use it in the future. The next we evaluate our model using confusion matrixand check for the accuracy. Lastly, we combine all in a real time OpenCV system to recognize a body language/pose.

9.2 Data Source

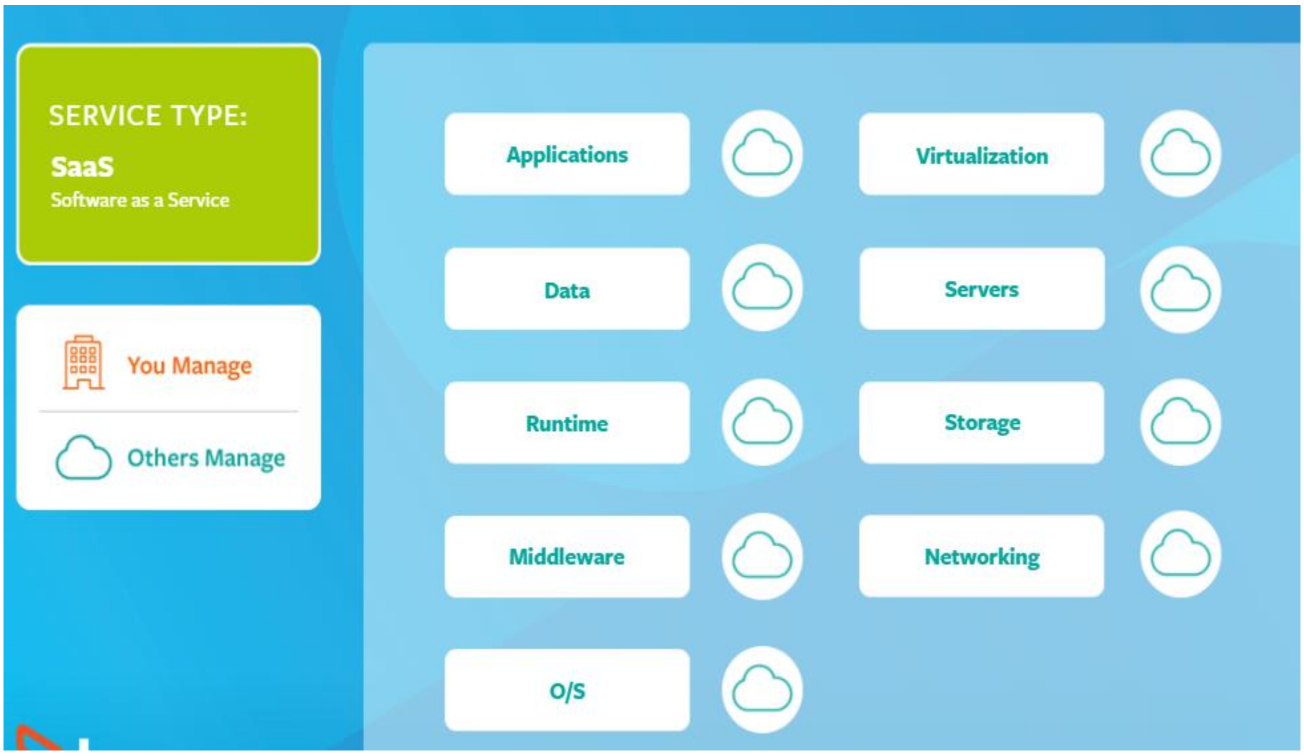
The data is needed to be collected in the for the first time we use the model for the training purposes through OpenCV capture as demonstrated in the code implementation.

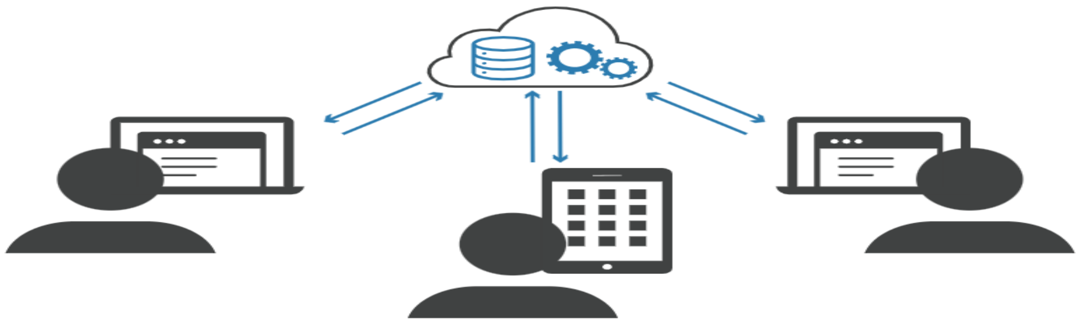
9.3 Algorithm

- LSTM Neural Network Algorithm: Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Recurrent neural networks contain cycles that feed the network activations from a previous time step asinputs to the network to influence predictions at the current time step. We need this system to remember the stated action and according to that predictthe current pose state.
- Multilabel Confusion Matrix Algorithm: A confusion matrix is a table that isused to define the performance of a classification algorithm. We use multilabel confusion matrix for the multi-class classification task, where each instance can only be labeled as one class, the confusion matrix is a powerful tool for performance assessment by quantifying the classification overlap. In our case different classes are different actions that we want our model to recognize.

a) Feasibility:

✓ This project can be developed and deployed within a few years as SaaS (Software as a Service) for anyone to use.





Users can access SaaS applications on any device

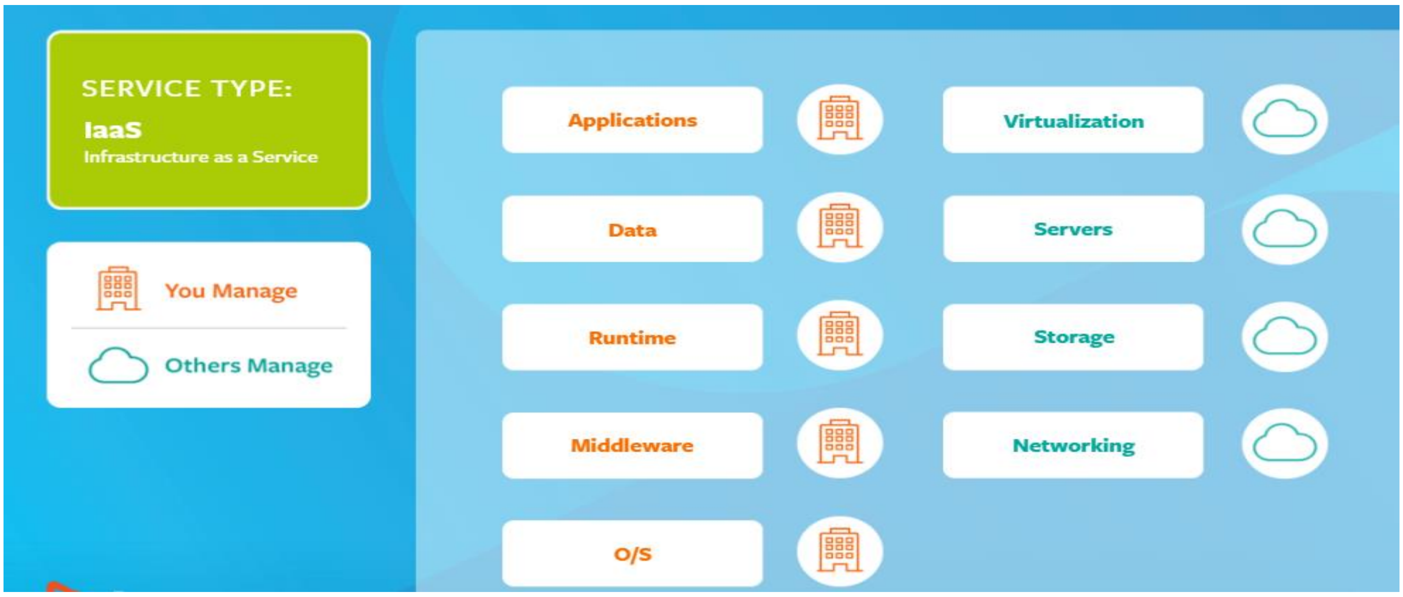
SaaS Characteristics

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users not responsible for hardware or software updates

SaaS may be the most beneficial option in several situations, including:

- Start-ups or small companies that need to launch ecommerce quickly and don't have time for server issues or software
- Short-term projects that require quick, easy, and affordable collaboration
- Applications that aren't needed too often, such as tax software
- Applications that need both web and mobile access
- ✓ **This project can also be developed and deployed within a few years as IaaS**

(Infrastructure as a Service) for anyone to use.



IaaS clients complete control over the entire infrastructure. IaaS provides the same technologies and capabilities as a traditional data centre without having to physically maintain or manage all of it. IaaS clients can still access their servers and storage directly, but it is all outsourced through a "virtual data centre" in the cloud.

IaaS Characteristics

- Resources are available as a service
- Cost varies depending on consumption
- Services are highly scalable
- Multiple users on a single piece of hardware
- Organization retain complete control of the infrastructure
- Dynamic and flexible

Start-ups and small companies may prefer IaaS to avoid spending time and money on purchasing and creating hardware and software.

b)Viability :

As the retail industry grows in India and the world, there will always be small businesses existing that can use this service to improvise their sales and data warehousing techniques.

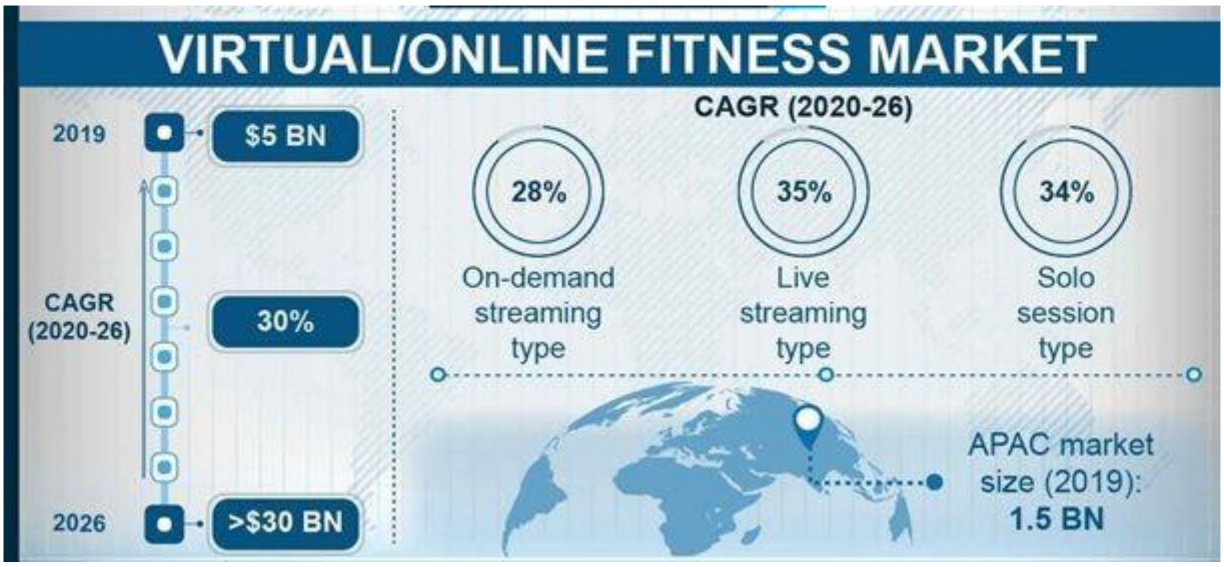
We can use body pose detection in AI fitness coach apps, the common flow looks as follows:

- Capture the user's movements while doing an exercise
- Analyze the correctness of an exercise performance
- Display mistakes in the user interface

After the pandemic, most people are unable to reach their fitness goals as there is no direct in-person coaching available to them. So we cannot surely assure that situation that is faced during covid doesn't repeat once again in the future.

Even though using the suggestions from Youtube Fitness courses and google suggestions may only work for an overview. But what about correcting the user's positions if they are wrong?

Fitness maintenance is not as easy as seen. People may get injured if they understand the positions wrong. By using Body pose detection we can warn the users if they are going wrong that they may get injured. So, it is viable to survive in the long-term future as well but improvements are necessary as new technologies emerge.



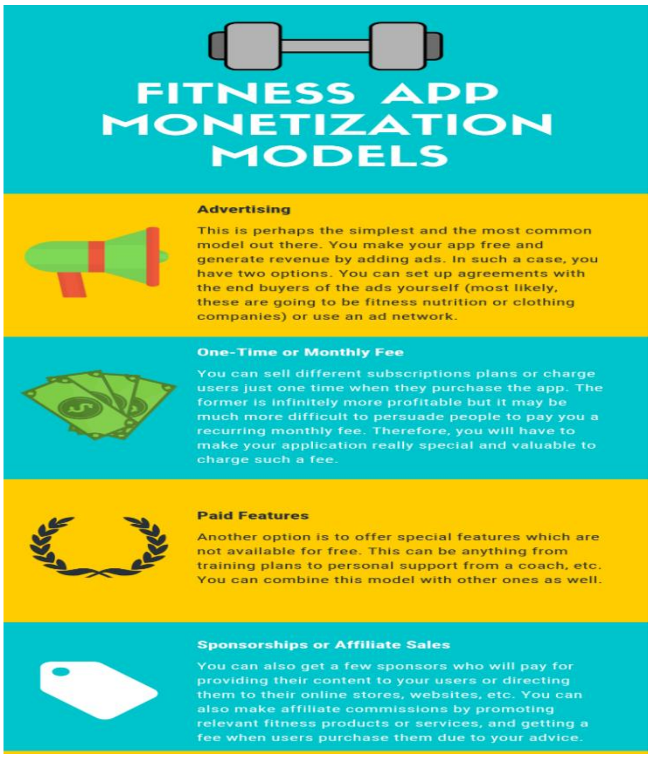
c) Monetization:

This service is directly monetizable as it can be directly released as a service on completion which can be used by Fitness Industry. We can develop a web app and deploy its versions. We can also monetize by availing this service to those already developed apps as a warning feature.

We can monetize using

1. Subscriptions. The subscription economy has grown 435% over the past decade.
2. Freemium. Most mobile apps are now monetized through the freemium model as it enables the product to drive marketing.
3. Affiliate marketing.
4. Print on Demand.
5. Advertising.
6. in-app purchases.
7. Sponsorship

WAYS TO MONETIZE YOUR APP
Freemium Model
In-App Purchases
One-time Paid Apps
Become an Affiliate
Ads



FITNESS APP MONETIZATION MODELS

Advertising

This is perhaps the simplest and the most common model out there. You make your app free and generate revenue by adding ads. In such a case, you have two options. You can set up agreements with the end buyers of the ads yourself (most likely, these are going to be fitness nutrition or clothing companies) or use an ad network.

One-Time or Monthly Fee

You can sell different subscriptions plans or charge users just one time when they purchase the app. The former is infinitely more profitable but it may be much more difficult to persuade people to pay you a recurring monthly fee. Therefore, you will have to make your application really special and valuable to charge such a fee.

Paid Features

Another option is to offer special features which are not available for free. This can be anything from training plans to personal support from a coach, etc. You can combine this model with other ones as well.

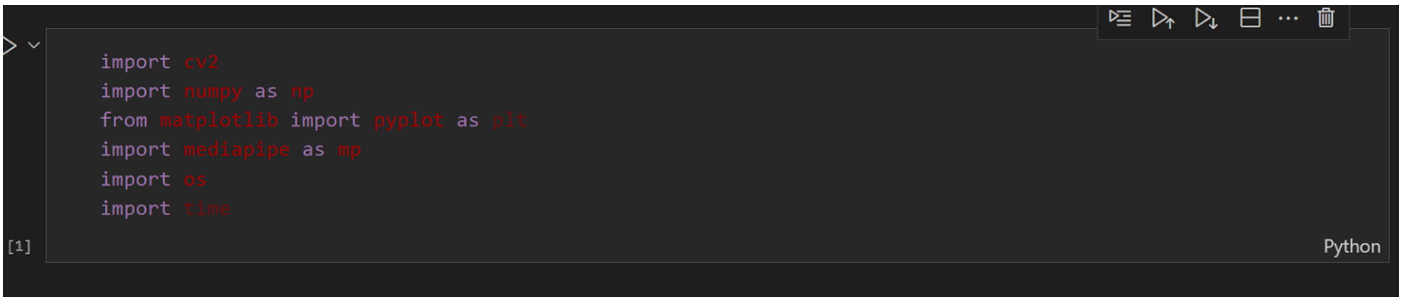
Sponsorships or Affiliate Sales

You can also get a few sponsors who will pay for providing their content to your users or directing them to their online stores, websites, etc. You can also make affiliate commissions by promoting relevant fitness products or services, and getting a fee when users purchase them due to your advice.

Step 2: Prototype Development

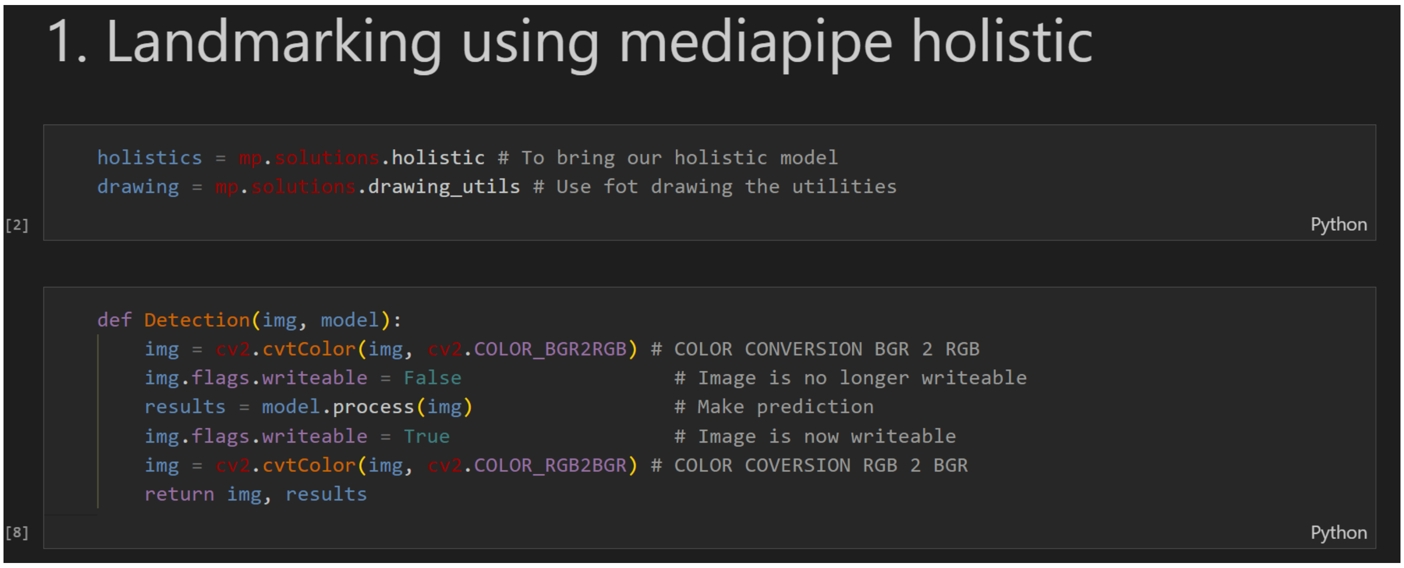
GitHub Link: https://github.com/Aaryan-palit/Virtual_Excercise

Implementation:



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt
import mediapipe as mp
import os
import time
```

[1] Python



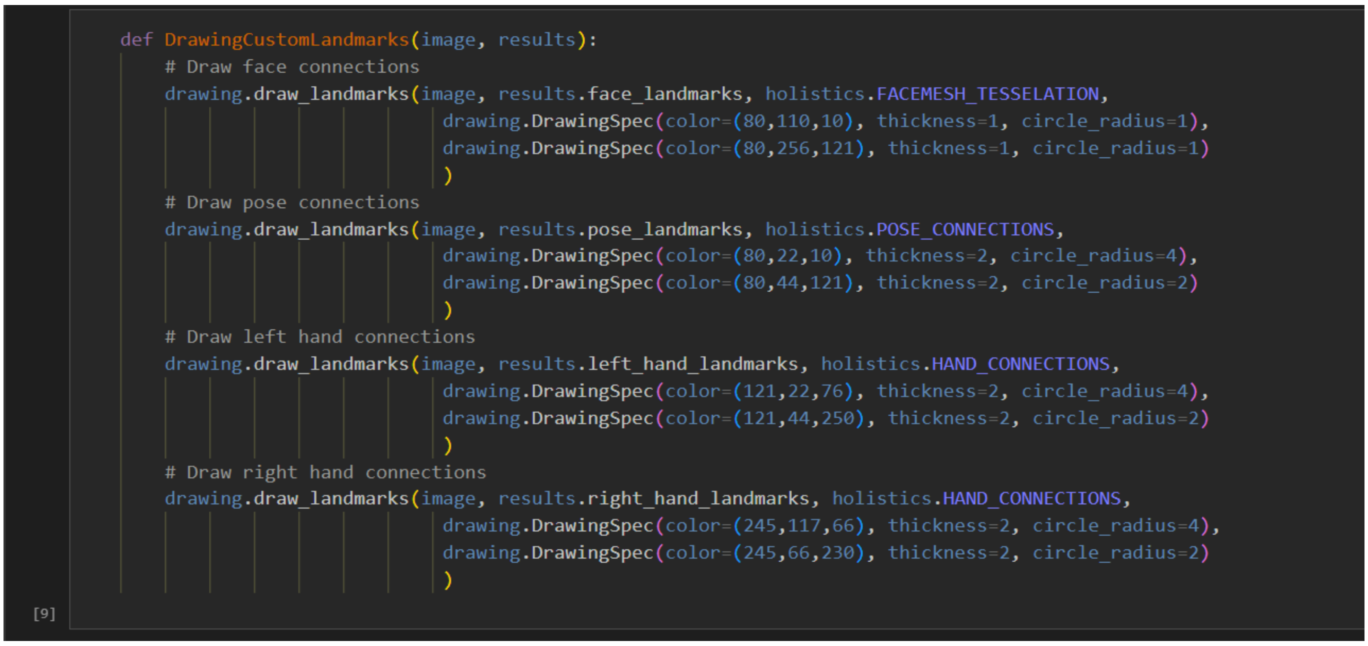
1. Landmarking using mediapipe holistic

```python
holistics = mp.solutions.holistic # To bring our holistic model
drawing = mp.solutions.drawing_utils # Use fot drawing the utilities
```

[2] Python

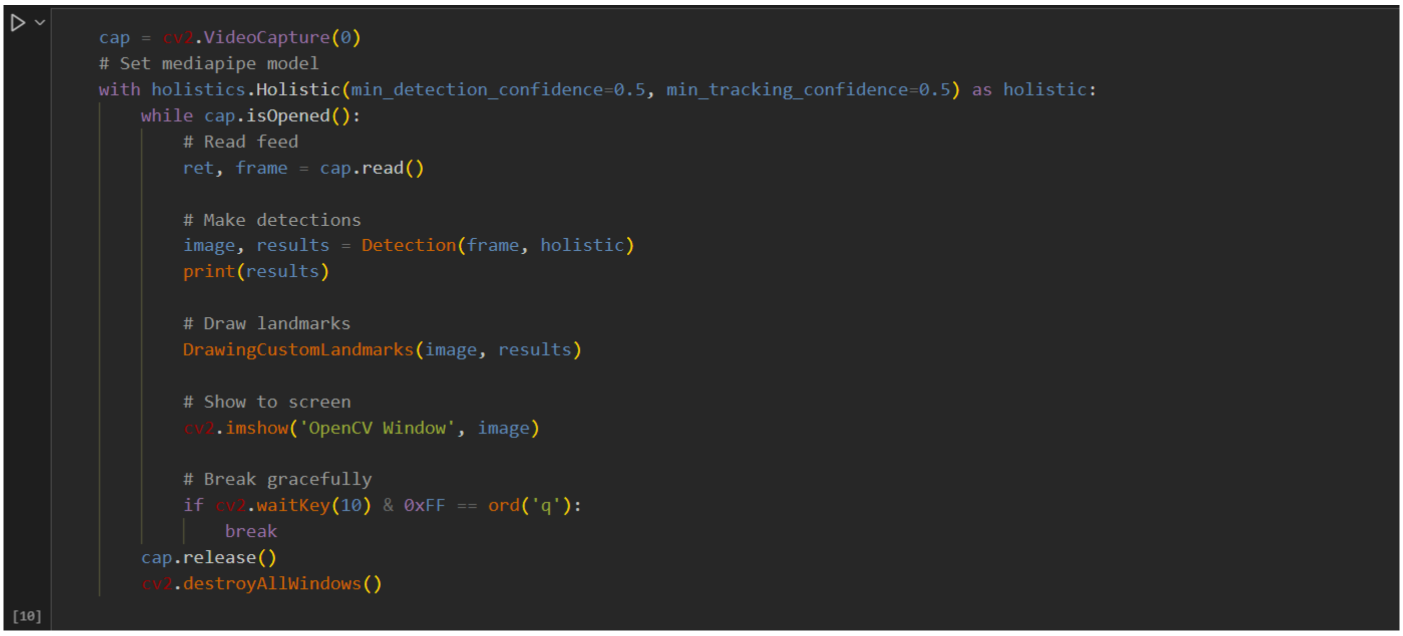
```python
def Detection(img, model):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    img.flags.writeable = False                  # Image is no longer writeable
    results = model.process(img)                 # Make prediction
    img.flags.writeable = True                   # Image is now writeable
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return img, results
```

[8] Python



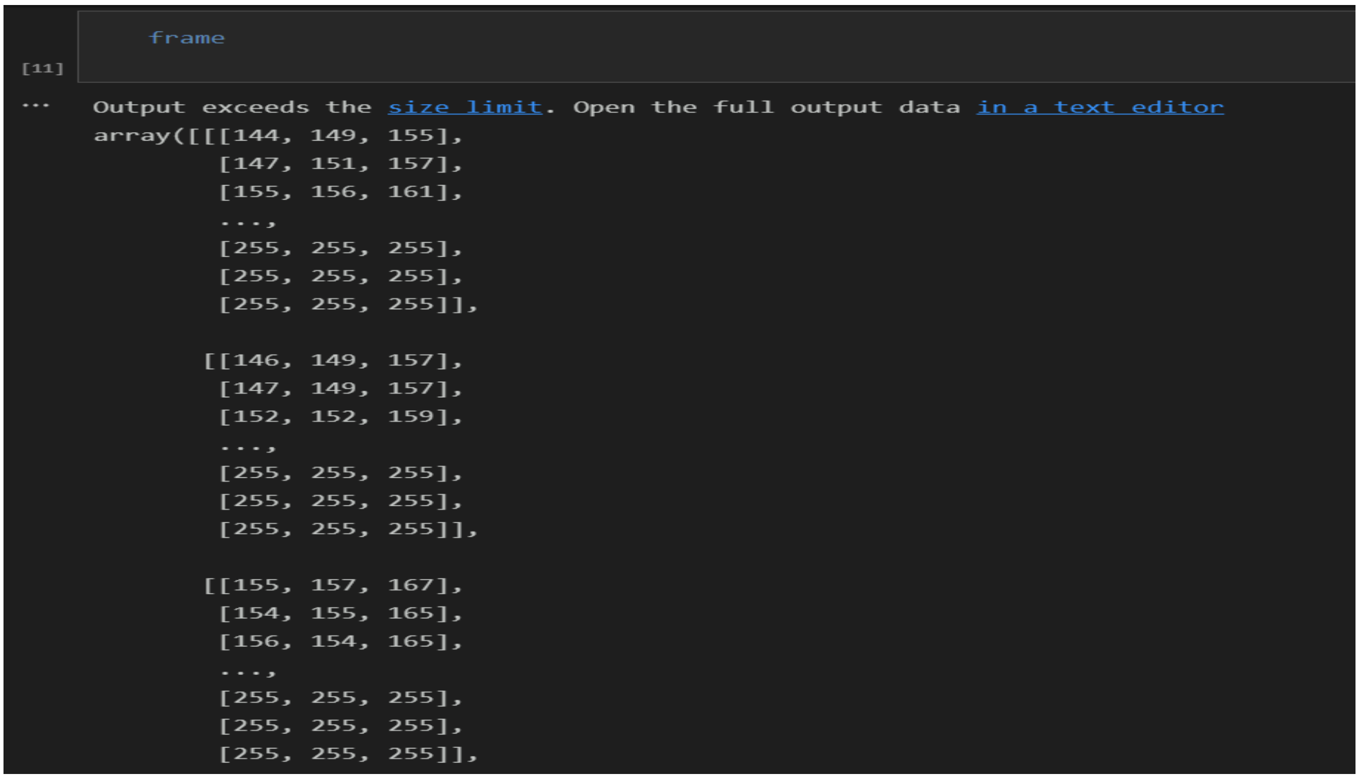
```python
def DrawingCustomLandmarks(image, results):
    # Draw face connections
    drawing.draw_landmarks(image, results.face_landmarks, holistics.FACEMESH_TESSELATION,
                           drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                           drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                           )
    # Draw pose connections
    drawing.draw_landmarks(image, results.pose_landmarks, holistics.POSE_CONNECTIONS,
                           drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                           drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                           )
    # Draw left hand connections
    drawing.draw_landmarks(image, results.left_hand_landmarks, holistics.HAND_CONNECTIONS,
                           drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                           drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                           )
    # Draw right hand connections
    drawing.draw_landmarks(image, results.right_hand_landmarks, holistics.HAND_CONNECTIONS,
                           drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                           drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                           )
```

[9]



```
cap = cv2.VideoCapture(0)
# Set mediapipe model
with holistics.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = Detection(frame, holistic)
        print(results)

        # Draw landmarks
        DrawingCustomLandmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Window', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

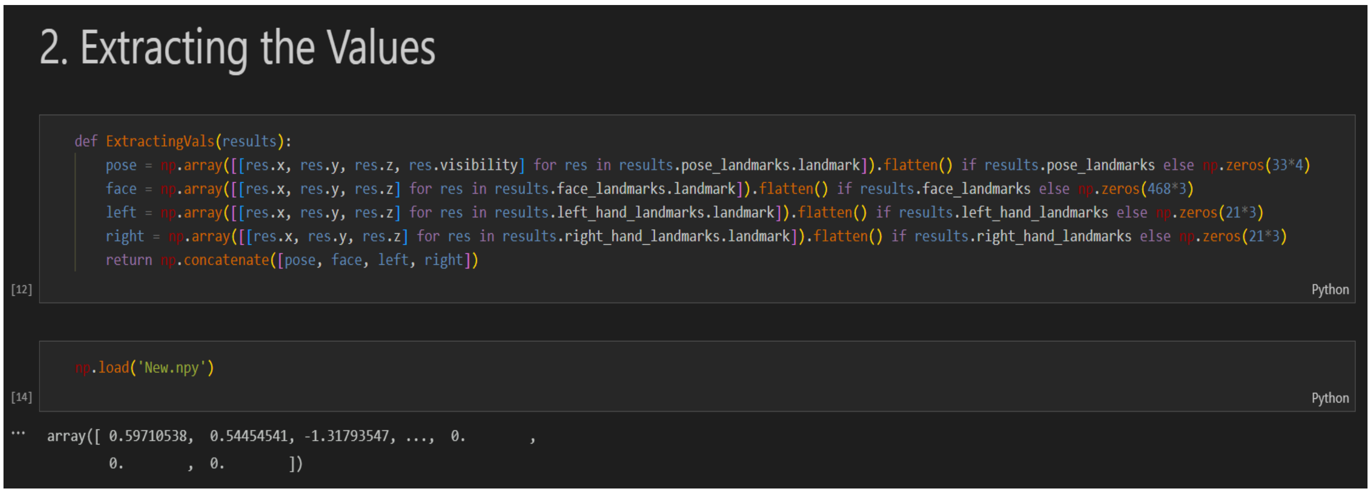
[10]



```
frame
```

[11]

```
Output exceeds the size limit. Open the full output data in a text editor
array([[[144, 149, 155],
        [147, 151, 157],
        [155, 156, 161],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[146, 149, 157],
        [147, 149, 157],
        [152, 152, 159],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[155, 157, 167],
        [154, 155, 165],
        [156, 154, 165],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],
```



2. Extracting the Values

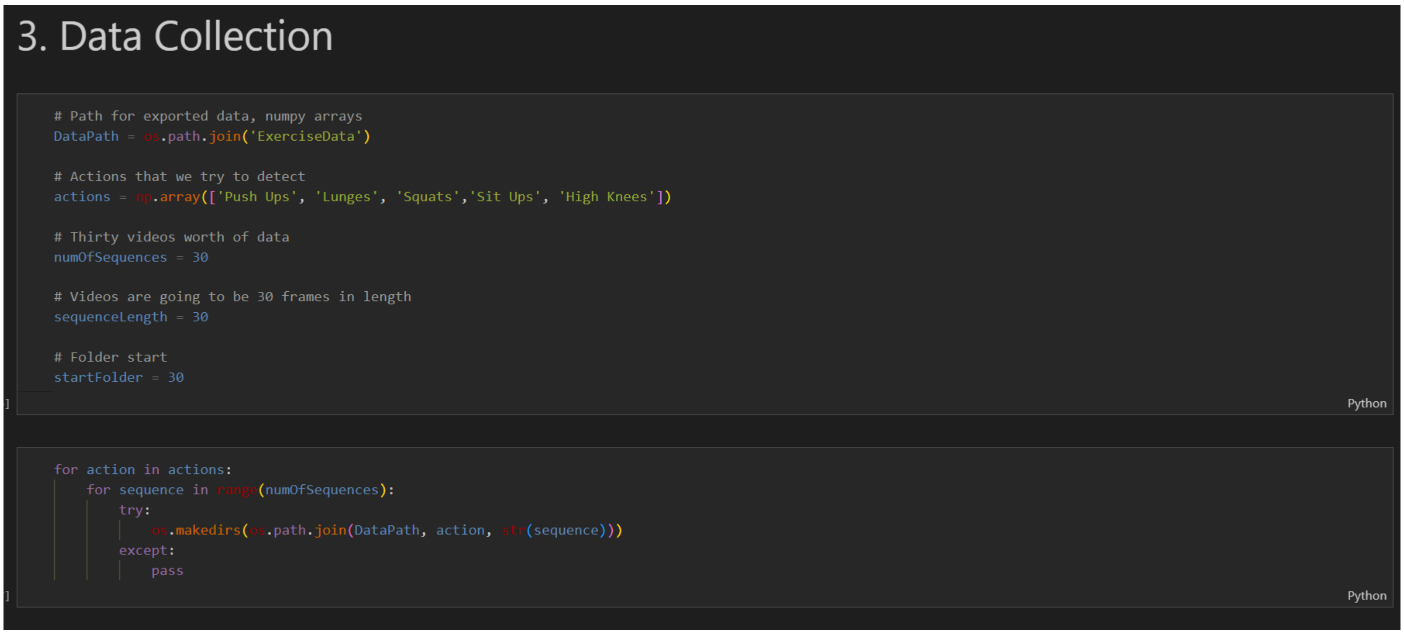
```
def ExtractingVals(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros(33*4)
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark]).flatten() if results.face_landmarks else np.zeros(468*3)
    left = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark]).flatten() if results.left_hand_landmarks else np.zeros(21*3)
    right = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark]).flatten() if results.right_hand_landmarks else np.zeros(21*3)
    return np.concatenate([pose, face, left, right])
```

[12] Python

```
np.load('New.npy')
```

[14] Python

```
array([ 0.59710538,  0.54454541, -1.31793547, ...,  0.        ,
        0.        ,  0.        ])
```



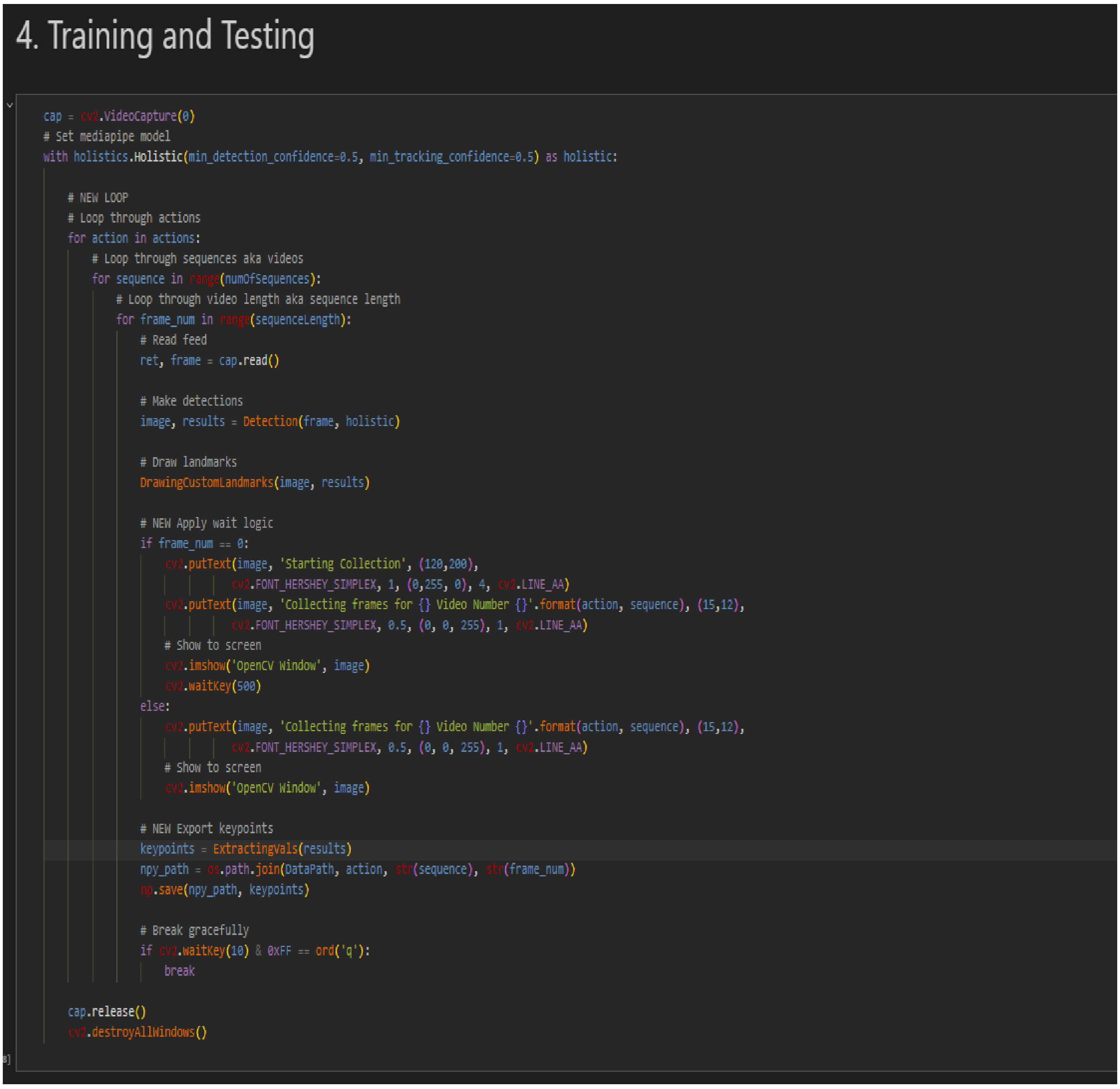
3. Data Collection

```python
# Path for exported data, numpy arrays
DataPath = os.path.join('ExerciseData')

# Actions that we try to detect
actions = np.array(['Push Ups', 'Lunges', 'Squats','Sit Ups', 'High Knees'])

# Thirty videos worth of data
numOfSequences = 30

# Videos are going to be 30 frames in length
sequenceLength = 30

# Folder start
startFolder = 30
```

Python

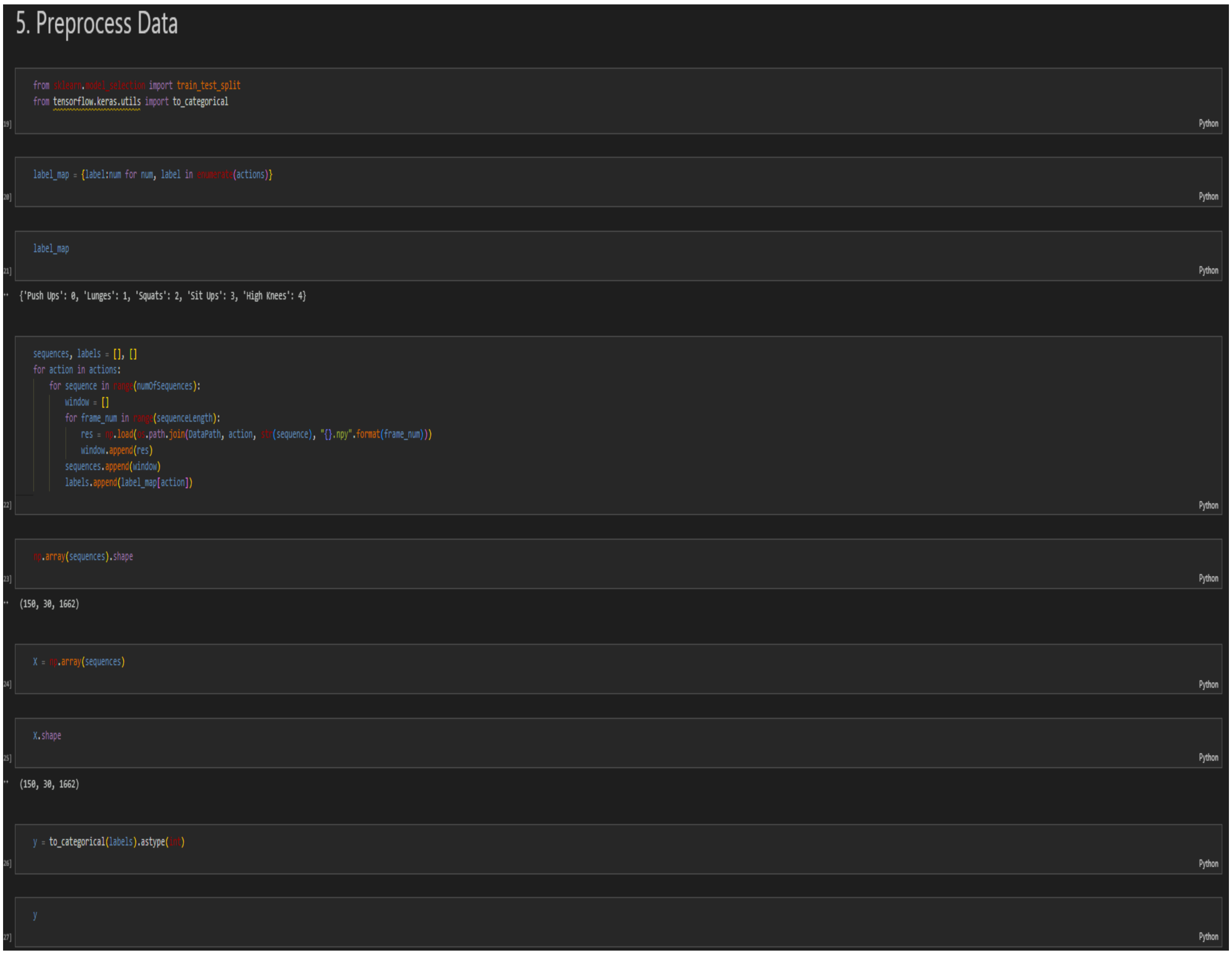
```python
for action in actions:
    for sequence in range(numOfSequences):
        try:
            os.makedirs(os.path.join(DataPath, action, str(sequence)))
        except:
            pass
```

Python



4. Training and Testing

```python
cap = cv2.VideoCapture(0)
# Set mediapipe model
with holistics.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:

    # NEW LOOP
    # Loop through actions
    for action in actions:
        # Loop through sequences aka videos
        for sequence in range(numOfSequences):
            # Loop through video length aka sequence length
            for frame_num in range(sequenceLength):
                # Read feed
                ret, frame = cap.read()

                # Make detections
                image, results = Detection(frame, holistic)

                # Draw landmarks
                DrawingCustomLandmarks(image, results)

                # NEW Apply wait logic
                if frame_num == 0:
                    cv2.putText(image, 'Starting Collection', (120,200),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Window', image)
                    cv2.waitKey(500)
                else:
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Window', image)

                # NEW Export keypoints
                keypoints = ExtractingVals(results)
                npy_path = os.path.join(DataPath, action, str(sequence), str(frame_num))
                np.save(npy_path, keypoints)

                # Break gracefully
                if cv2.waitKey(10) & 0xFF == ord('q'):
                    break

    cap.release()
    cv2.destroyAllWindows()
```



5. Preprocess Data

```python
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```python
label_map = {label:num for num, label in enumerate(actions)}
```

```python
label_map
```

```
{'Push Ups': 0, 'Lunges': 1, 'Squats': 2, 'Sit Ups': 3, 'High Knees': 4}
```

```python
sequences, labels = [], []
for action in actions:
    for sequence in range(numOfSequences):
        window = []
        for frame_num in range(sequenceLength):
            res = np.load(os.path.join(DataPath, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

```python
np.array(sequences).shape
```

```
(150, 30, 1662)
```

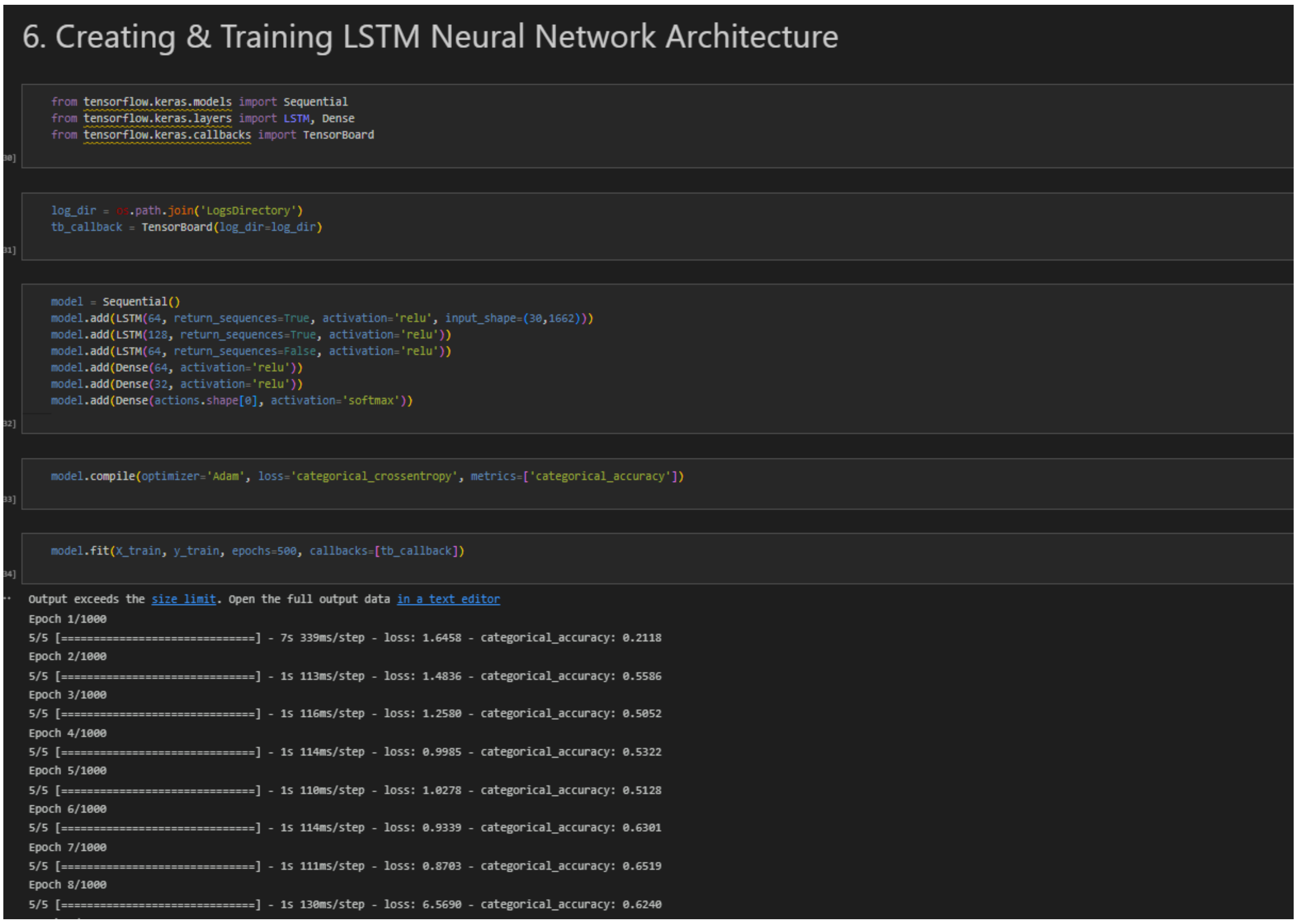
```python
X = np.array(sequences)
```

```python
X.shape
```

```
(150, 30, 1662)
```

```python
y = to_categorical(labels).astype(int)
```

```python
y
```



6. Creating & Training LSTM Neural Network Architecture

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
```

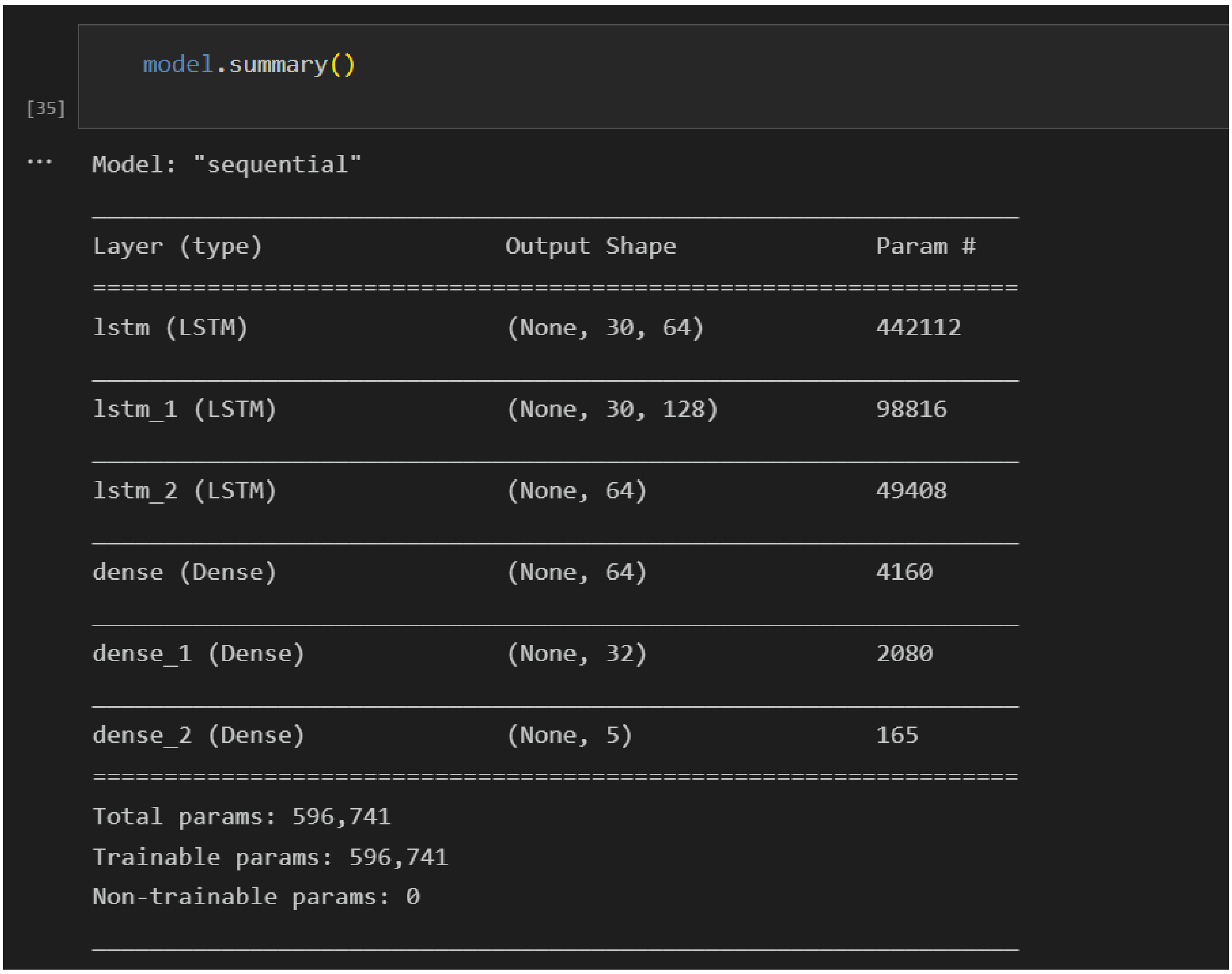
```python
log_dir = os.path.join('LogsDirectory')
tb_callback = TensorBoard(log_dir=log_dir)
```

```python
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

```python
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
```

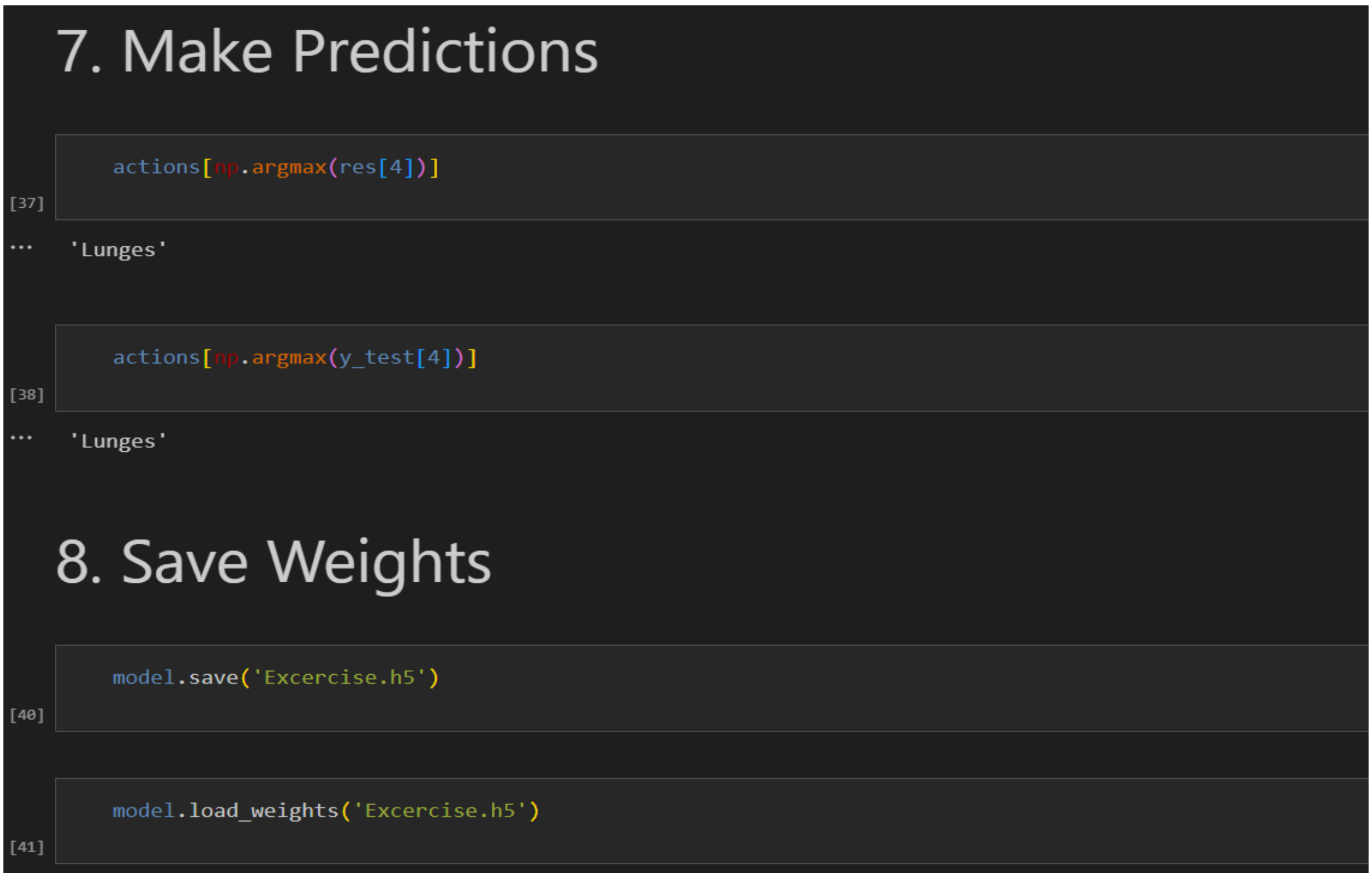
```python
model.fit(X_train, y_train, epochs=500, callbacks=[tb_callback])
```

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/1000
5/5 [==============================] - 7s 339ms/step - loss: 1.6458 - categorical_accuracy: 0.2118
Epoch 2/1000
5/5 [==============================] - 1s 113ms/step - loss: 1.4836 - categorical_accuracy: 0.5586
Epoch 3/1000
5/5 [==============================] - 1s 116ms/step - loss: 1.2580 - categorical_accuracy: 0.5052
Epoch 4/1000
5/5 [==============================] - 1s 114ms/step - loss: 0.9985 - categorical_accuracy: 0.5322
Epoch 5/1000
5/5 [==============================] - 1s 110ms/step - loss: 1.0278 - categorical_accuracy: 0.5128
Epoch 6/1000
5/5 [==============================] - 1s 114ms/step - loss: 0.9339 - categorical_accuracy: 0.6301
Epoch 7/1000
5/5 [==============================] - 1s 111ms/step - loss: 0.8703 - categorical_accuracy: 0.6519
Epoch 8/1000
5/5 [==============================] - 1s 130ms/step - loss: 6.5690 - categorical_accuracy: 0.6240
```



```
[35]
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
lstm (LSTM)                 (None, 30, 64)            442112
_________________________________________________________________
lstm_1 (LSTM)               (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)               (None, 64)                49408
_________________________________________________________________
dense (Dense)               (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)             (None, 5)                 165
=================================================================
Total params: 596,741
Trainable params: 596,741
Non-trainable params: 0
_________________________________________________________________
```



7. Make Predictions

```
[37]
actions[np.argmax(res[4])]
```

```
'Lunges'
```

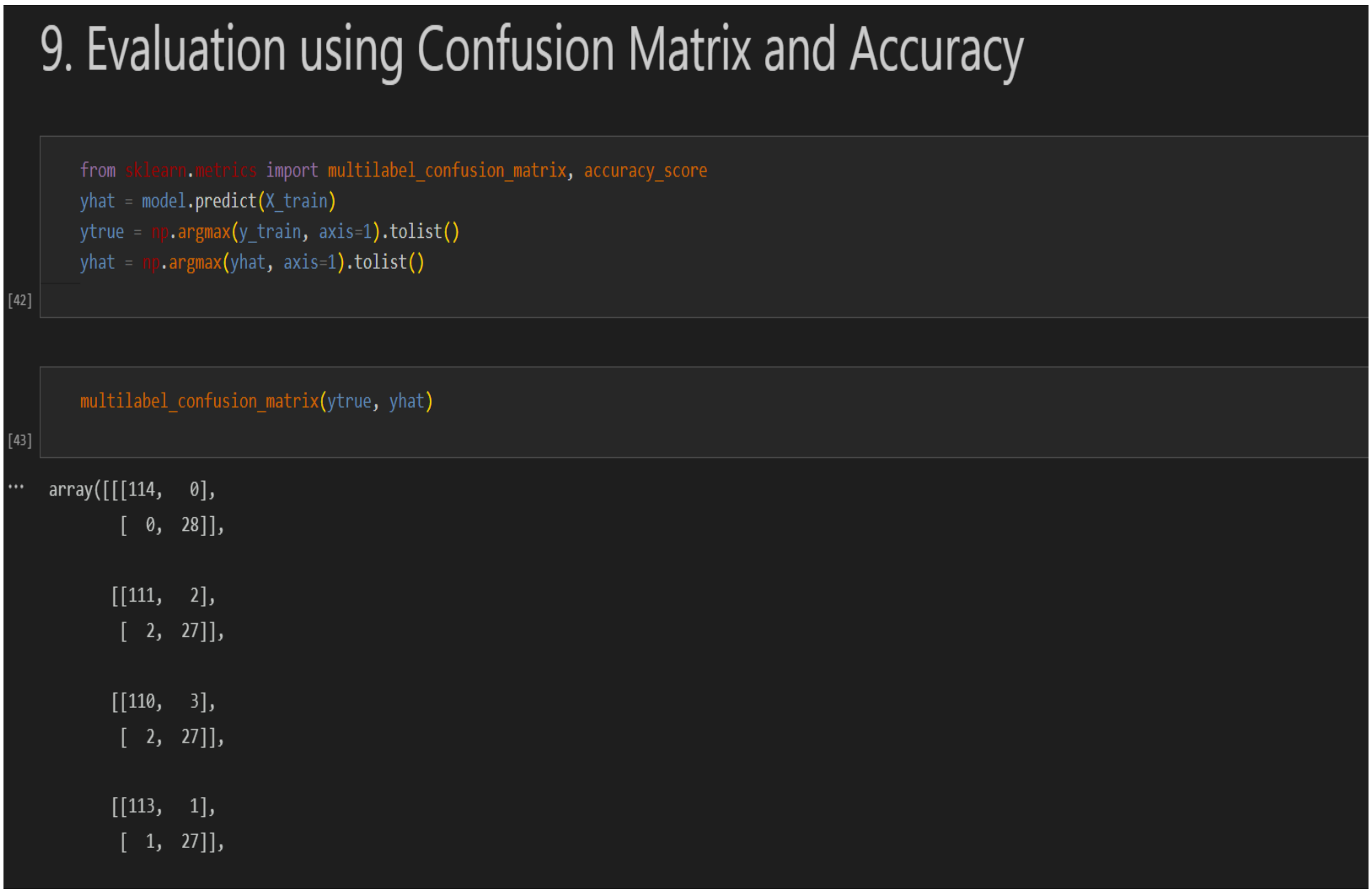
```
[38]
actions[np.argmax(y_test[4])]
```

```
'Lunges'
```

8. Save Weights

```
[40]
model.save('Excercise.h5')
```

```
[41]
model.load_weights('Excercise.h5')
```



9. Evaluation using Confusion Matrix and Accuracy

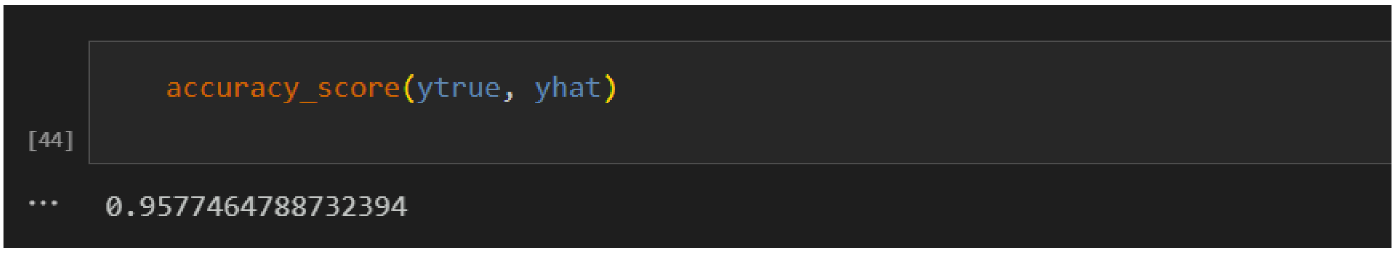
```python
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
yhat = model.predict(X_train)
ytrue = np.argmax(y_train, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

[42]

```python
multilabel_confusion_matrix(ytrue, yhat)
```

[43]

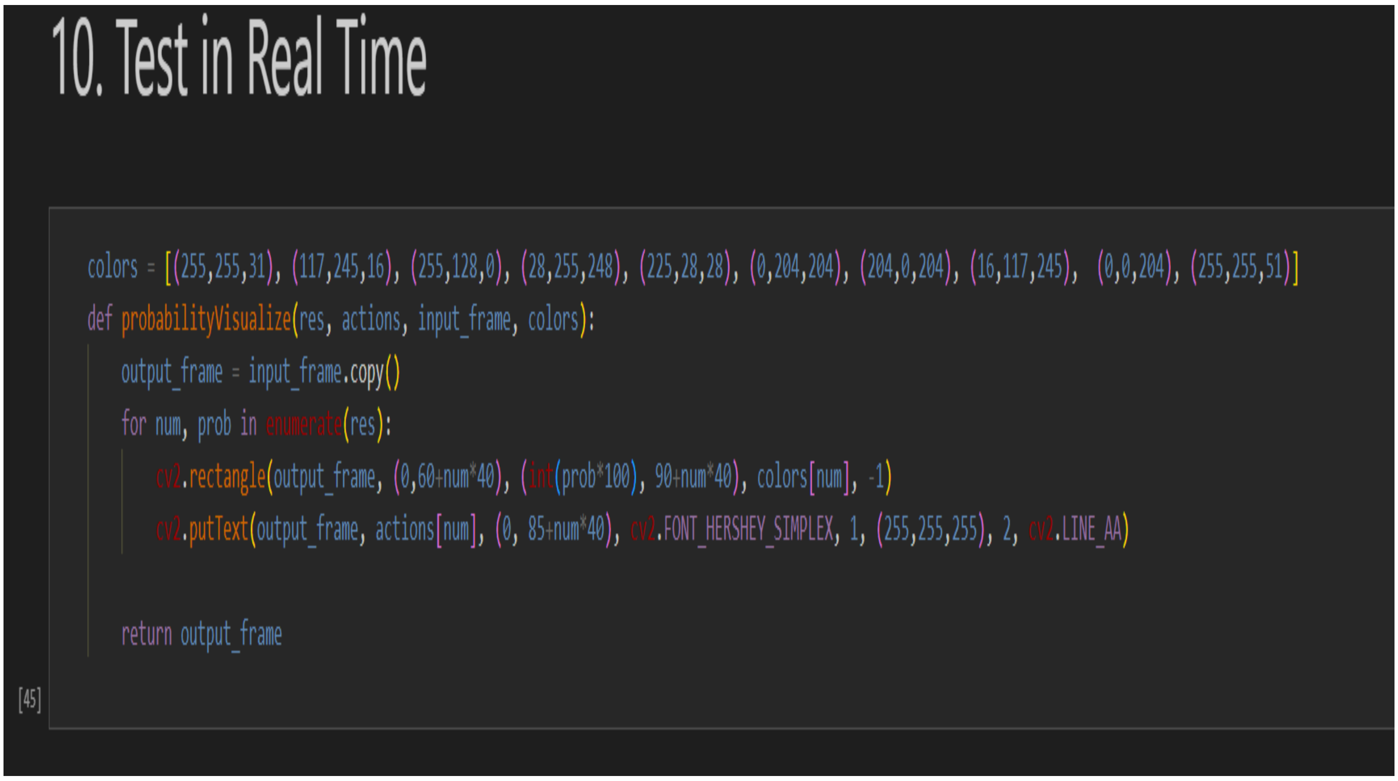
```
array([[[114,   0],
        [  0,  28]],

       [[111,   2],
        [  2,  27]],

       [[110,   3],
        [  2,  27]],

       [[113,   1],
        [  1,  27]],
```



```python
accuracy_score(ytrue, yhat)
```

[44]

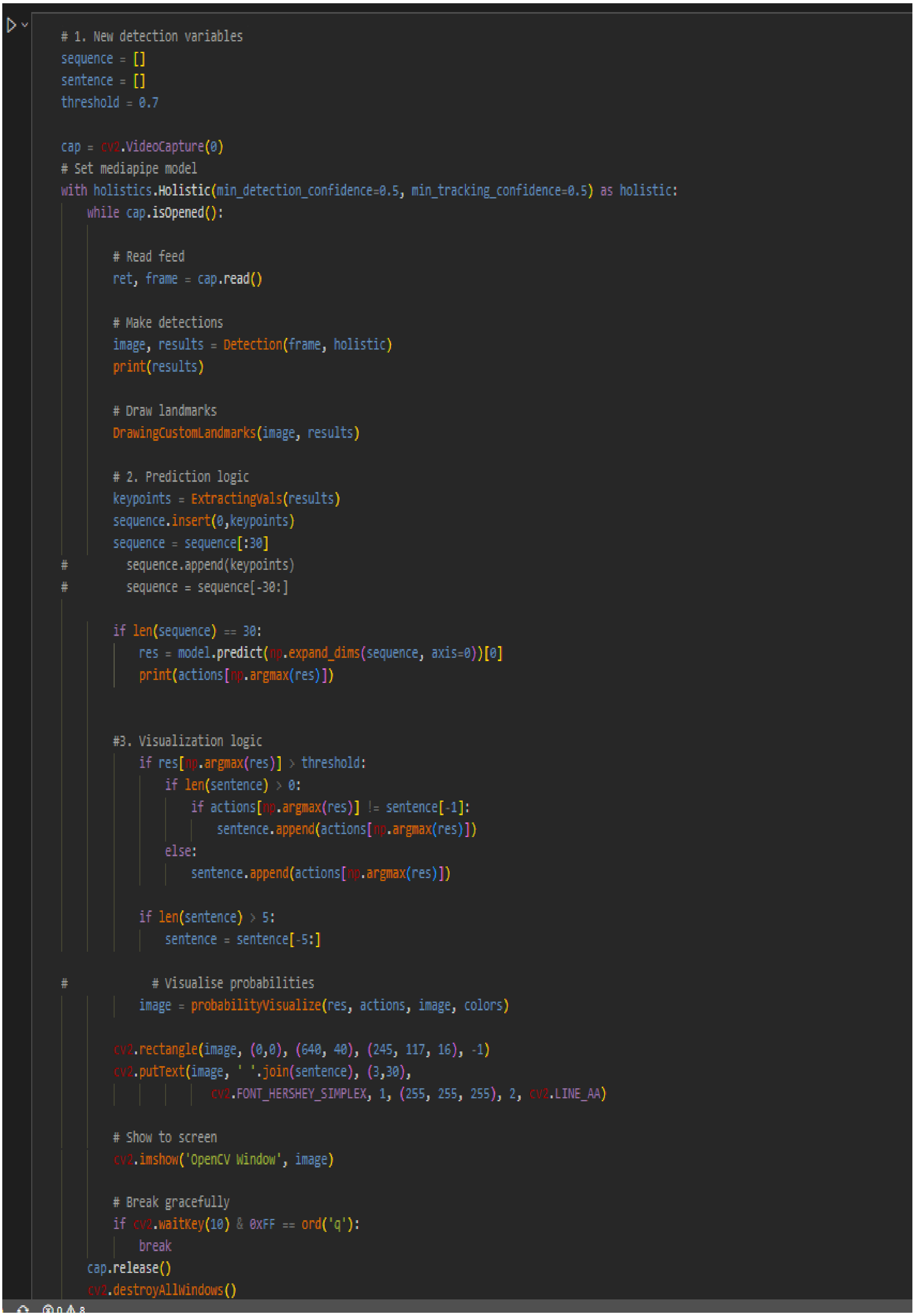
```
0.9577464788732394
```



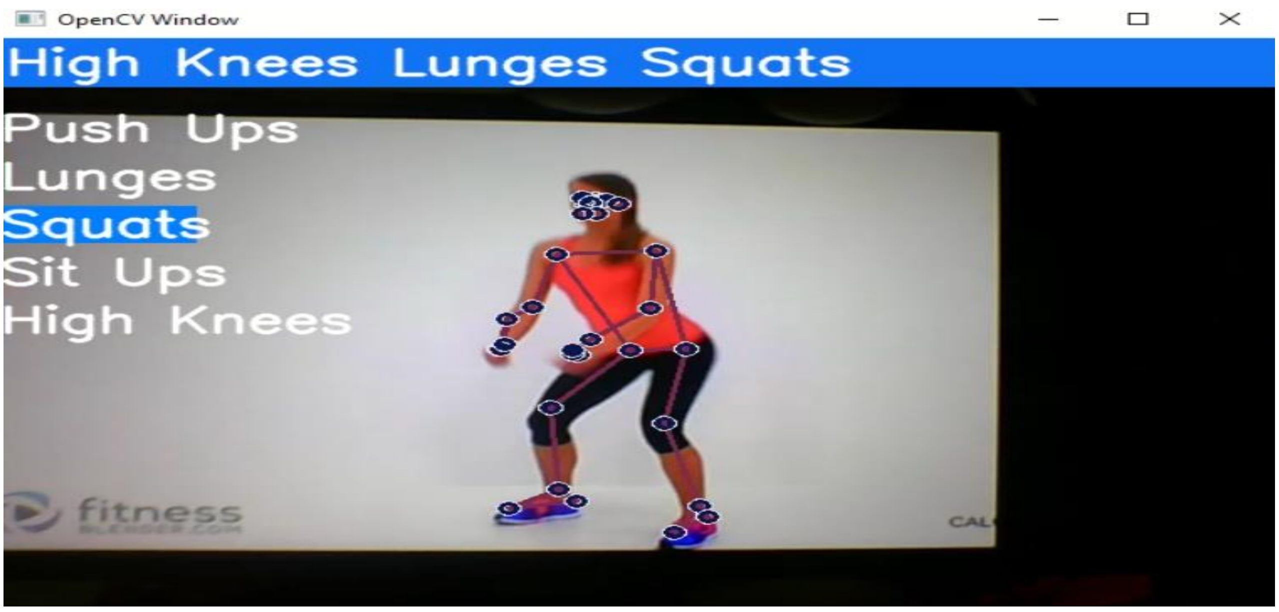
10. Test in Real Time

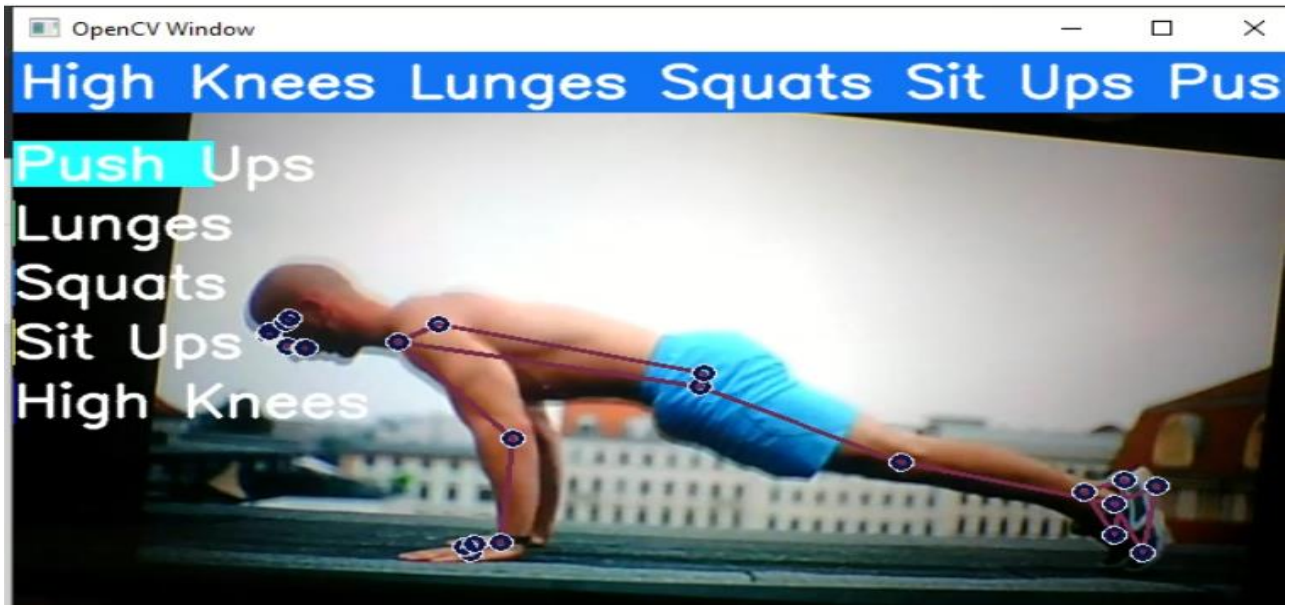
```python
colors = [(255,255,31), (117,245,16), (255,128,0), (28,255,248), (225,28,28), (0,204,204), (204,0,204), (16,117,245),  (0,0,204), (255,255,51)]
def probabilityVisualize(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)

    return output_frame
```

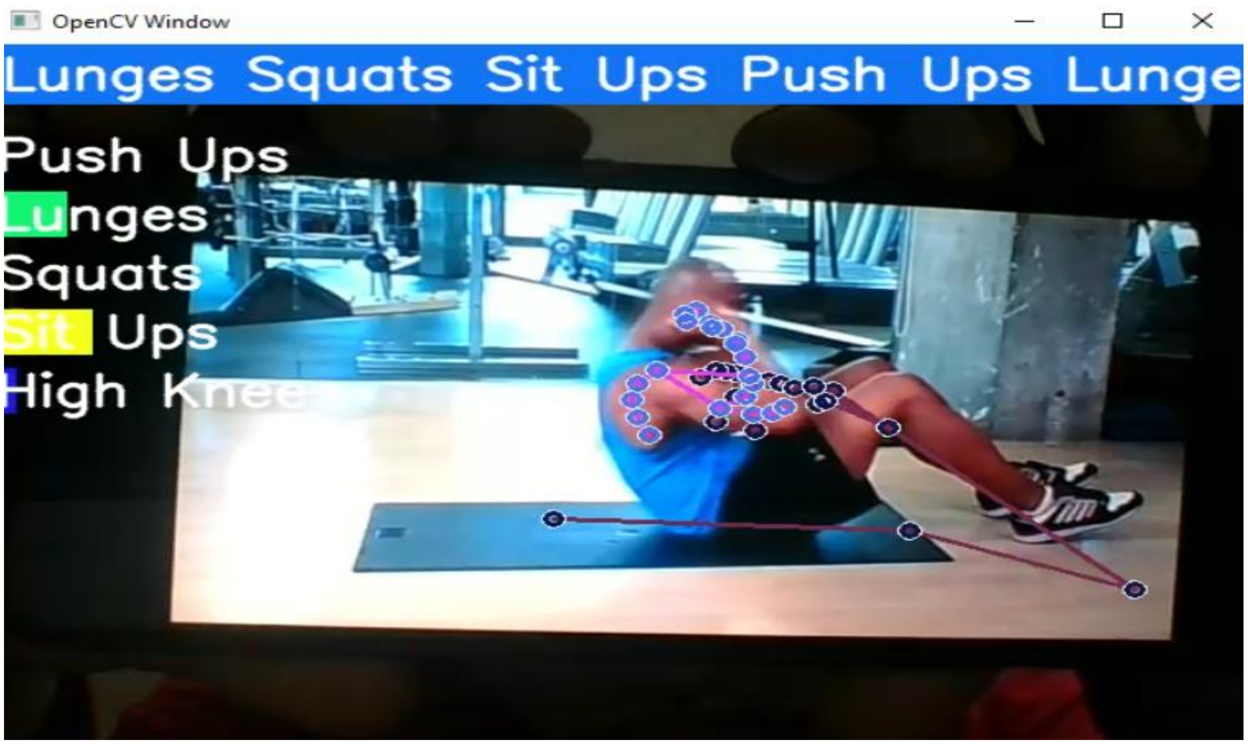
[45]



```python
# 1. New detection variables
sequence = []
sentence = []
threshold = 0.7

cap = cv2.VideoCapture(0)
# Set mediapipe model
with holistics.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = Detection(frame, holistic)
        print(results)

        # Draw landmarks
        DrawingCustomLandmarks(image, results)

        # 2. Prediction logic
        keypoints = ExtractingVals(results)
        sequence.insert(0,keypoints)
        sequence = sequence[:30]
#         sequence.append(keypoints)
#         sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])


        #3. Visualization logic
            if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                    if actions[np.argmax(res)] != sentence[-1]:
                        sentence.append(actions[np.argmax(res)])
                else:
                    sentence.append(actions[np.argmax(res)])

            if len(sentence) > 5:
                sentence = sentence[-5:]

#             # Visualise probabilities
            image = probabilityVisualize(res, actions, image, colors)

        cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3,30),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        # Show to screen
        cv2.imshow('OpenCV Window', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```


OpenCV Window
High Knees Lunges Squats
Push Ups
Lunges
Squats
Sit Ups
High Knees
fitness

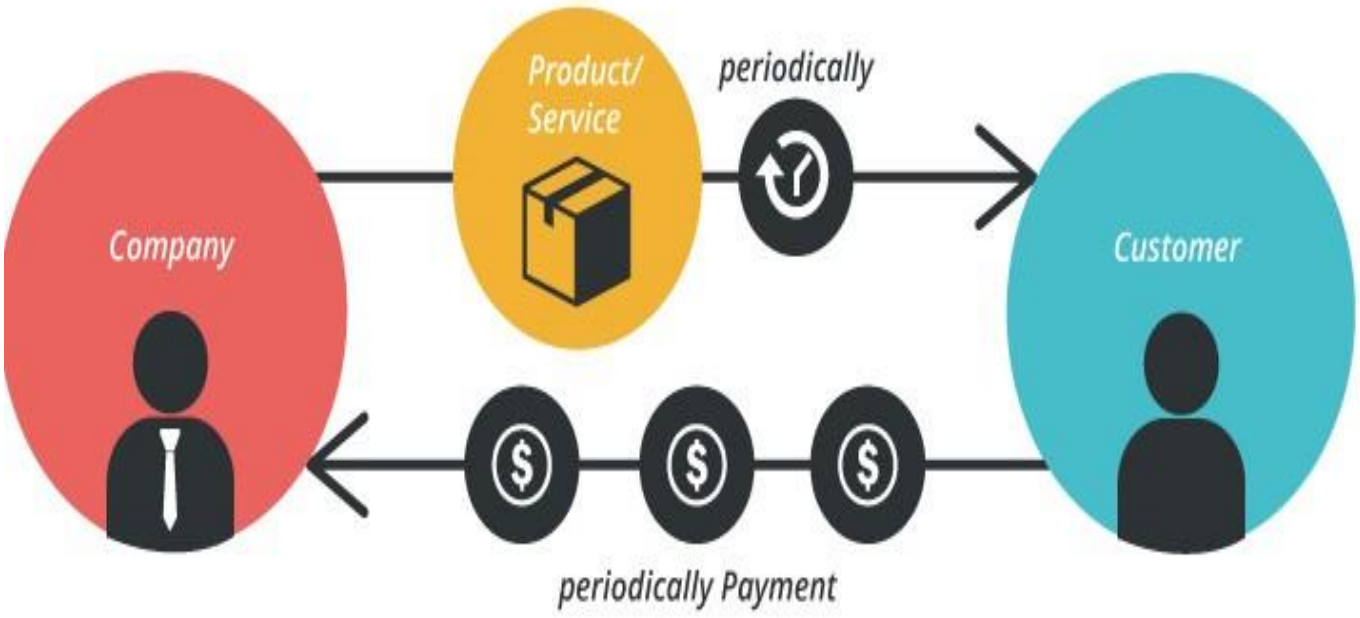

OpenCV Window
High Knees Lunges Squats Sit Ups Pus
Push Ups
Lunges
Squats
Sit Ups
High Knees


OpenCV Window
Lunges Squats Sit Ups Push Ups Lunge
Push Ups
Lunges
Squats
Sit Ups
High Knee

Step 3: Business Modelling

BUSINESS MODEL

The best business model for an AI fitness coach app is a **free trial subscription business model**. The free trial model has become one of the more well-liked customer acquisition techniques for subscription-based businesses among the various subscription business models. Offering a brief free trial period is the simplest way to persuade potential customers to sign up for a subscription service.



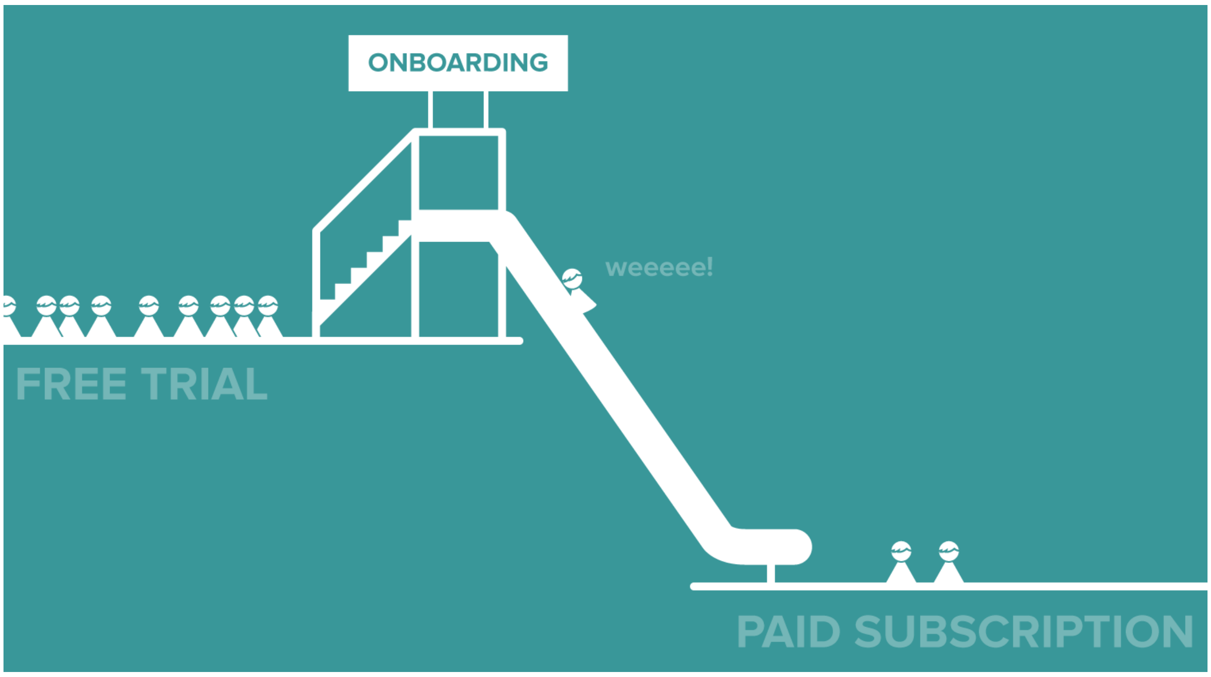
Why provide a free trial?

Customers may find it challenging to make the first commitment required by a paid subscription. A customer may be hesitant to pay for a full month or more of a service about which they are unsure.

With the free trial model, customers can use a fully functional product for a brief period of time—typically a few weeks or months—without paying anything.

After a statistically significant number of conversions from free offers have been accumulated, subscription businesses should study the source of these clients and alter their ongoing marketing campaigns accordingly.

The problem that many businesses struggle with is how to turn a free user into a paid customer.



Ultimately, businesses must combine the following in order for the premium version to succeed and transition users to more expensive plans:

- Limit the features available to free users to encourage them to upgrade for an improved experience.
- Provide more individualized or customer service related to an account.
- It's crucial to offer a wide range of payment options so that customers can choose the one they prefer.
- Nurture your free users by explaining the benefits of paid features
- Share positive stories and accounts from paying users
- Craft a compelling offer.
- Sprinkle in a sense of urgency.

Step-4: Financial Modelling (equation) with Machine Learning & Data Analysis

- **Identify which Market your product/service will be launched into**

The top 2 Market where our product/service will be very useful is in the field of Gym/Sports Industry and the other is in the field of Surveillance Camera monitor.

With the Gym/Sports person getting injured very frequently. A slight correction in the Postures would benefit them a lot.

Since the **Virtual-Online Fitness market** is growing rapidly, We believe it is important to detect & correct the Posture of Workout Exercises to avoid injuries and get the most out of the exercise.

- **Collect some data /statistics regarding that Market Online.**

Consumer spending on online fitness

- Consumers have increased spending in digital fitness has increased 30% to 35% in comparison to before the pandemic [2021]
- Although 75% plan to return to some form of pre-pandemic exercise routine (including the gym), 30% of consumers expect to continue spending more on online fitness solutions than they did before the pandemic [2021]
- 40% of gym members were paying for online fitness by companies they've never visited before [2020]

Changes in users behavior: live streaming vs on-demand fitness

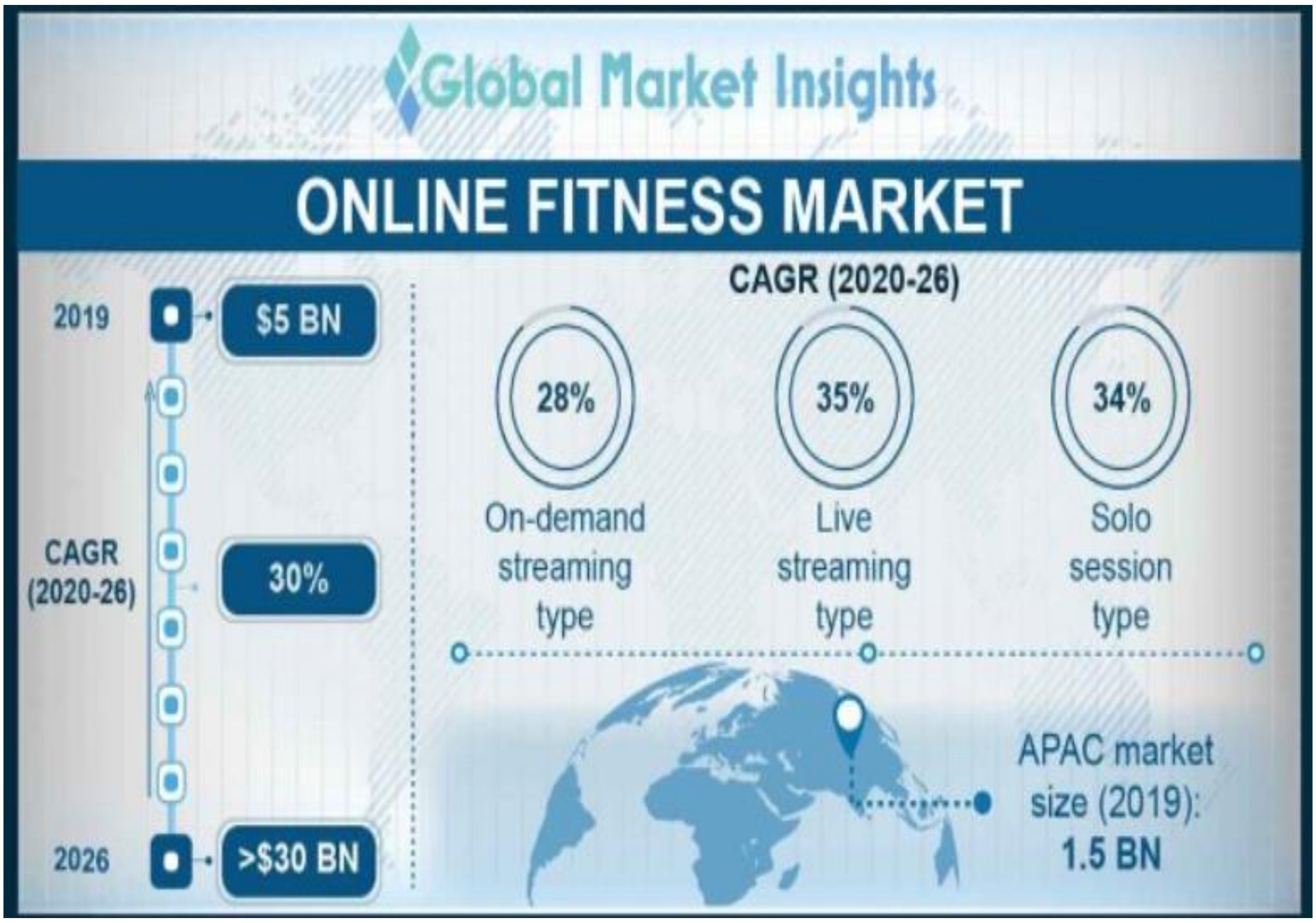
- 72% of consumers surveyed say that they prefer the flexibility of online fitness to be able to join a class whenever they want and get away from the normal "gym membership" method where they had to work around the gym's hours and busiest times [2020]
- U.S. study of 2,000 Americans, 75% said it's easier to stay fit at home [2021]
- 72% of US adults were exercising just as much (56%) or more (16%) through the use of online fitness videos in comparison to pre-pandemic levels [2020]
- Since the pandemic, 75% of active adults have used live streaming workouts and 70% used on-demand videos to support their exercise regiment [2021]
- From March to August of 2020, the minutes spent live streaming health and fitness content increased 1,300%

Mindbody app users using video for their workouts grew by 177% before and after covid (from 26% to 74%) [2020]

When comparing the type of offerings, they found:

- Pre-recorded, on-demand video usage grew 311% (from 17% pre-covid to 70% post-covid)

- Live streaming video usage grew by 971% (from 7% pre-covid to 75% post-covid)



Gyms adoption of online fitness

- Traditional gyms, health clubs, and fitness studios continue to adopt a hybrid model with on-site and online fitness solutions
- 75% of those that use virtual fitness also go to in-person classes
- Gyms and studios that offer virtual workouts increase their in-person class attendance by 12%
- More than half of online fitness users also go to the gym 3x a week and spend more time on average than other members
- 72% of fitness club owners created and started offering an on-demand or live streaming service, an increase of 25% from 2019 [2021]

How do people find their online workouts?

- 63% of people who use live-streamed workouts and 53% who use on-demand workouts found them through the local studio's own website or via social media [2020]
- 26% of people who use live-streamed workouts and 13% who use on-demand workouts found them via recommendation from friends or family [2020]
- 12% of people who use live-streamed workouts and 38% who use on-demand workouts found them by searching on YouTube or Google [2020]

Why is online fitness growing and beneficial?

With 57% of the global population being active internet users as of 2019, the market was primed for the widespread adoption of online fitness solutions ahead of the pandemic. Many gyms and studios provide access to live-streamed classes and on-demand content to better support their members as well.

Two interesting benefits of online fitness are:

- When working out in groups (in-person or streamed live) people cycle 21% further and exercise 10% longer on average [2020]
- Those who use live stream or on-demand workouts exercise 20% more per month on average than they did the previous year [2021]

Most popular class bookings

When analysing the breakdown of virtual bookings of Mindbody users, we found that the following were the most popular classes:

- Yoga (32%)
- HIIT (15.6%)
- Pilates (8.3%)
- Barre (7.9%)

References

https://www.wellnessliving.com/blog/what-does-vaccine-mean-fitness-industry/

https://runrepeat.com/pandemics-impact-fitness-industry

https://www.fitnessmentors.com/personal-trainer-stats/

https://codete.com/blog/wearable-fitness-technology-trends-and-statistics-2020/

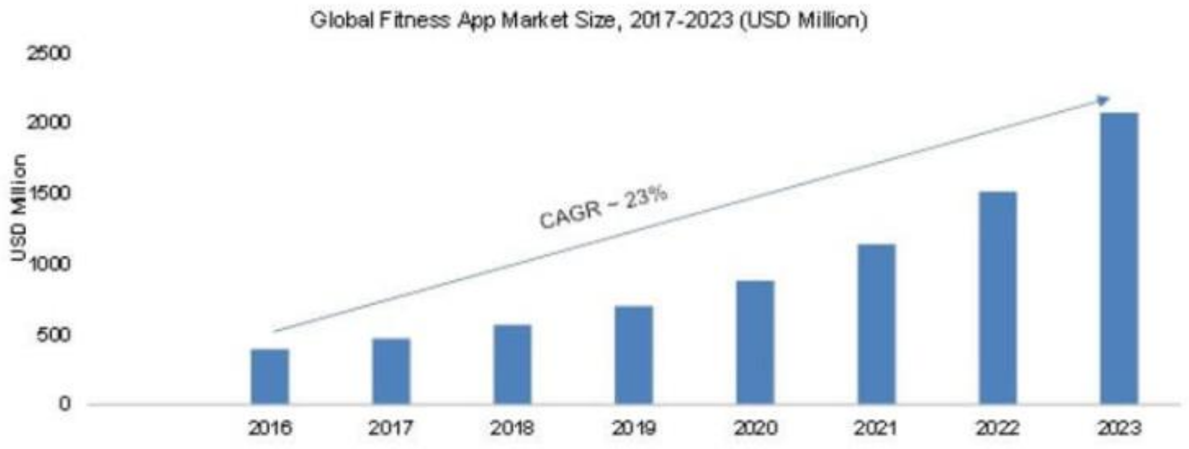
https://cedcommerce.com/blog/7-ecommerce-business-ideas-in-covid-19-pandemic/

https://www.lek.com/insights/ei/covid-19-accelerating-digital-fitness-boom

https://www.weforum.org/agenda/2020/09/fitness-apps-gym-health-downloads/

- **Design Financial Equation corresponding to that Market Trend.**

- In 2021, the global online fitness industry is worth $10.71 billion dollars
- The industry has grown 77.33% since it's estimated value of $6.04 billion in 2019
- The online fitness industry is projected to be worth more than $59 billion by 2027, growing at a rate of 30.0% to 33.1% per year
- Since 2017, on-demand fitness spending is up 128% while traditional gyms grew just 6%.
- Due to the demand for customized and increasingly niche fitness class offerings, the live streaming fitness market is expected to grow 35% per year till 2026
- 72% of active US adults are exercising just as much (56%) or more (16%) through the use of online fitness videos in comparison to pre-pandemic levels



Market is growing linearly,

Financial Equation:

$\boxed{p=wy(d)+n,}$

where
p= total profit,
w=pricing of your product,
y(d)=total sales (market as a function of time)
n=production, maintenance etc costs.

Conclusion:

Thus, we can understand that body language/ pose estimation is a computer visiontechnique that predicts and tracks the location of a person or object. This is done by looking at a combination of the pose and the orientation of a given person/object.

Humans belong to a class of objects that are flexible. The key points of our body will be in different positions relative to others as we bend our arms or legs. The majority of inanimate objects are rigid. For example, regardless of the orientationof the brick, the corners are always the same distance apart. Rigid pose estimationis the process of predicting the position of these objects.

People are only perceived as a frame in traditional object detection (a square). Computers can learn about human body language by performing pose detection and pose tracking. Traditional pose tracking methods, on the other hand, are neither fast nor resistant to occlusions. Some of the most significant trends in computer vision will be driven by high-performance real-time pose detection and tracking. For example, real-time tracking of human poses will allow computers todevelop a finer-grained and more natural understanding of human behavior, Andhere comes our body language/pose estimation model to solve this issue.

This idea can be used to implement in FITNESS INDUSTRY. It is used to identify human joints and provide the user with guidance on how to exercise the right way. For example, head pose estimation is essential when the user is doing a plank. The app estimates the position of the head to avoid injuries while exercising.

Body language is a type of visual communication produced by the motions of the hands, face, and body. A human's facial expression, posture, body language tells alot more than we normally perceive. By analyzing it we can predict the hidden motive and emotion of a person. This is one of the important reasons that several research is going on in the fields of psychology and neuroscience to read and analyze such human body language.