Machine Learning Engineer Nanodegree

Ademola-Aliu Oluwatobi

Aug 31 2020

Capstone Project

Project Overview

Today facial recognition has become a security feature of choice for phones, laptops, passports, and payment apps. It promises to revolutionize the business of targeted advertising and speed up the diagnosis of certain illnesses.

In this project I created a web app that can recognize five sports celebrities. The application uses a classifier trained from custom image dataset if got from the web and preprocessed.

Problem Statement

Given a set of five sport celebrity images, we want to correctly classify them. A simple web app would be set up with a drop zone to drag and drop image files and then our classifier would output which of the five celebrities is in the image

Evaluation Metrics

The best_score_ method of the sklearn's GridsearchCV would be used and the algorithm with the highest score will be picked (best estimator). After this, the algorithm would be applied on the test set. After the estimator has been fit on the test set the score method will be called to evaluate its performance. Also, the confusion matrix would be plotted to see how well the algorithm could classify the different categories of celebrities.

Analysis

Data Exploration

I got my data using google search to search for the names of the sports celebrities I had in mind and then clicked on the images panel to see the images. I used the Fatkun downloader to download them in batches. In all I had about 2,477 images in total and were in jpg format. They were colored images and their sizes varied.

Fig 1. Sample of images from the dataset

Exploratory Visualization

I had to visualize a sample of the dataset using python's openCV to read the image and matplotlib's pyplot to visualize the image. Below is a sample of Maria sharapova's image

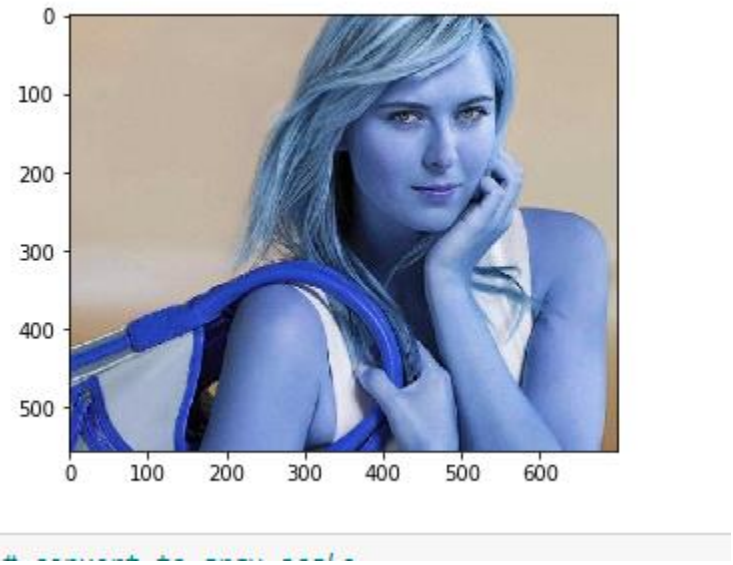

Fig 2 plotted Color Image

Since feature detection algorithms only work on gray images I had to convert the image from color to gray using python Cv2's cvtColor method.

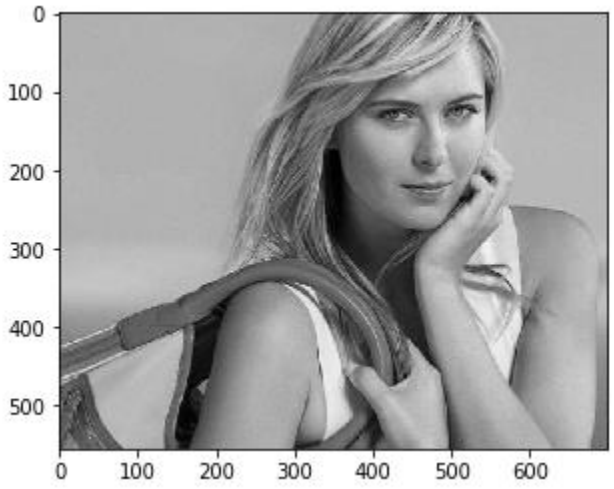

Fig 3 plotted Gray Image

Opencv has some feature detectors all stored as an xml files in a directory known as haarcascade. I used the face_cascade to detect the faces in my image datasets. Then drawing a bounding box/ rectangle around the region of detected face.

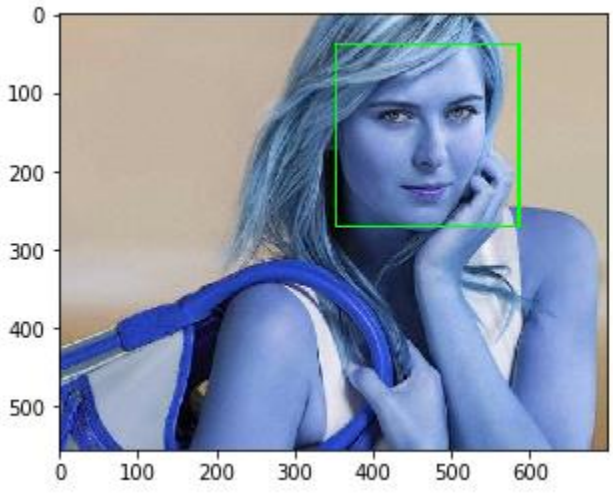

Fig 4 plotted face detected Image

I then used the eye_cascade to detect the eyes in faces of my image datasets. Then drawing a bounding box/ rectangle around the region of detected eyes.

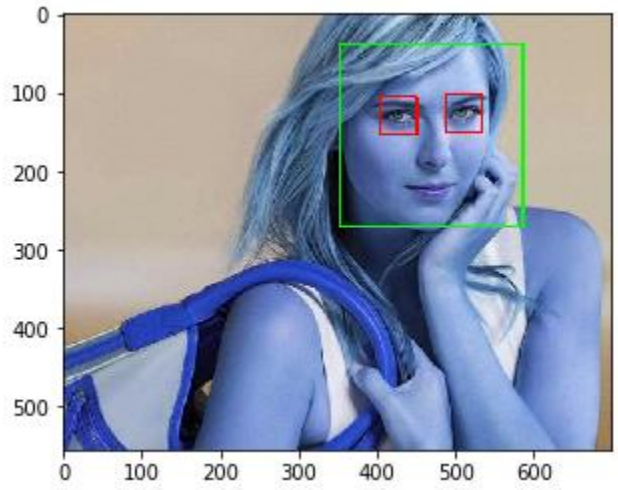

Fig 4 face and eyes detected Image

I then wrote a function to crop the images to just the detected faces, since that bears sufficient features to classify the different images.

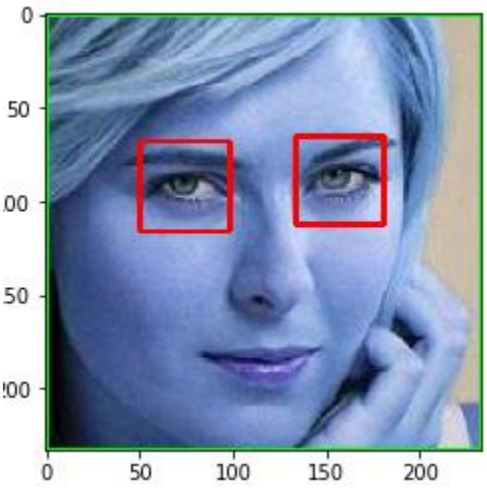

Fig 5 cropped face Image

Algorithms and Techniques

The classifier tried were logistic regression, support vector machine's support vector classifier, and random forest. They were all trained with the processed data set and then hyperparameter tuning was carried out on the three algorithms to dtermine the best. This was done using sklearn's GridsearchCV. The best_score method was used to determine the best estimator among the three algorithms. After this, the algorithm was applied on the test set. After the estimator has been fit on the test set the score method was called to evaluate its performance. Also, the confusion matrix would be plotted to see how well the algorithm could classify the different categories of celebrities. Logistic regression seemed to outperform the other two. And so logistic regression was selected as our classifier model.

Benchmark Model

The solution model i.e. the Logistic regression model, came from the hyperparameter tuning process. After trying out different classification algorithms namely; logistic regression, SVC, and Random forest I picked Logistic regression. This was used on the test set and saved as the model the server would use for the image classification task

Methodology

Data Preprocessing

This was done using the jupyter note book in the following steps

- Used cv2 to crop the images to only the face region
- Created a cropped directory to house al the cropped images of each of the five sport celebrities
- Created a dictionary to map the names of the celebrities to numbers which would be our labels for training
- Used a wavelet function to transform the images to ehance feature detection
- Scaled the images since they all had different sizes
- Stacked the wavelet transformed images and the cropped scaled images
- Created arrays to house the feature images and the labels
- Converted the values of the feature array to floats to enhance compartibility with sklearn

Implementation

The impementation can be split into two stages:

- The classifier Training stage
- The web app development stage

The classifier Training Stage

- Necessary libraries were imported such as the sklearn pipeline, metrics, model_selection, preprocessing
- The train_test_split from the model selection was used to split the data to training and test data.
- The pipeline contained the standard scaler for feature scaling, as well as the algorithm of choice
- The fit method was then called on the pipeline instance to train the model on the training data
- The score method was then used to evaluate the model's performance.
- The classification report method was used to further see other evaluation metrics such as precision, recall, f1-score.

SVC was first tried and the results are as follow:

```
[5]: #Import necessary libraries
     from sklearn.svm import SVC
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.pipeline import Pipeline
     from sklearn.metrics import classification_report

[7]: X_train, X_test, y_train, y_test = train_test_split(x,y, random_state=0)

[8]: pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'rbf', C =10))])
     pipe.fit(X_train, y_train)
     pipe.score(X_test, y_test)

[8]: 0.6666666666666666

[9]: print(classification_report(y_test, pipe.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.50	0.75	0.60	8
1	0.73	0.80	0.76	10
2	0.67	0.67	0.67	6
3	0.73	0.80	0.76	10
4	1.00	0.25	0.40	8
accuracy			0.67	42
macro avg	0.72	0.65	0.64	42
weighted avg	0.73	0.67	0.65	42

Figure 6 SVC

This was not good enough and so I performed hyperparameter tuning using different algorithms and hyperparameters such as C, kernel type and got the following results.

	model	best_score	best_params
0	svm	0.741667	{'svc__C': 10, 'svc__kernel': 'rbf'}
1	random_forest	0.516667	{'randomforestclassifier__n_estimators': 10}
2	logistic_regression	0.782667	{'logisticregression__C': 1}

Fig 7 Different Models

Therefore, I chose logistic regression as the final model.

The web app development stage

This involved creating a server directory and a UI (User interface) directory

The server directory :

- This housed a server.py file which will serve our web app
- A util.py file which would contain several functions that would help process the images into the format the algorithm can accept and also the image classifier function which the server.py file would call to classifier the image.

The UI directory:

- This housed the html web app the JavaScript app and the CSS app.
- This also contained images that would be displayed on the webapp.
- There is also a drop zone were images will be dropped for classification.

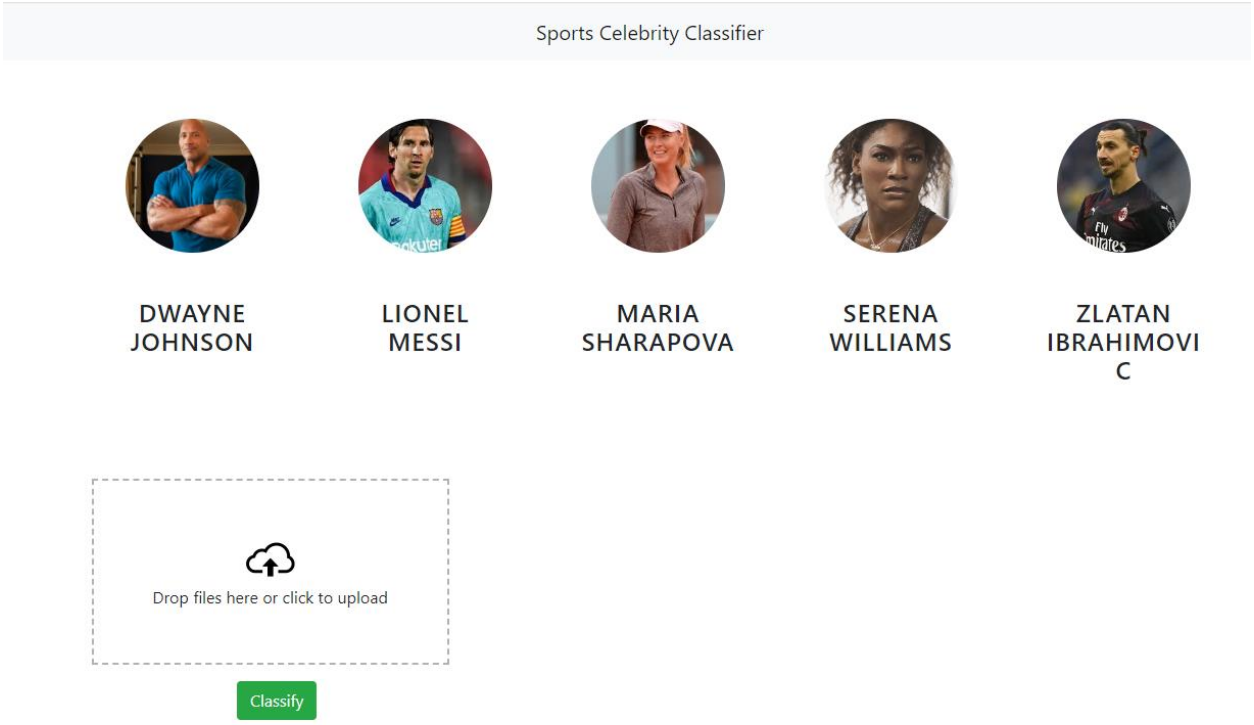

Fig 8 UI web app

Results

- The logistic regression perfomed fairly okay on the test set with about 73% accuracy.
- Also the confusion matrix was plotted with seaborn to visualize the correct and wrong classifications among the five sport celebrity categories. The result is in the fig below.

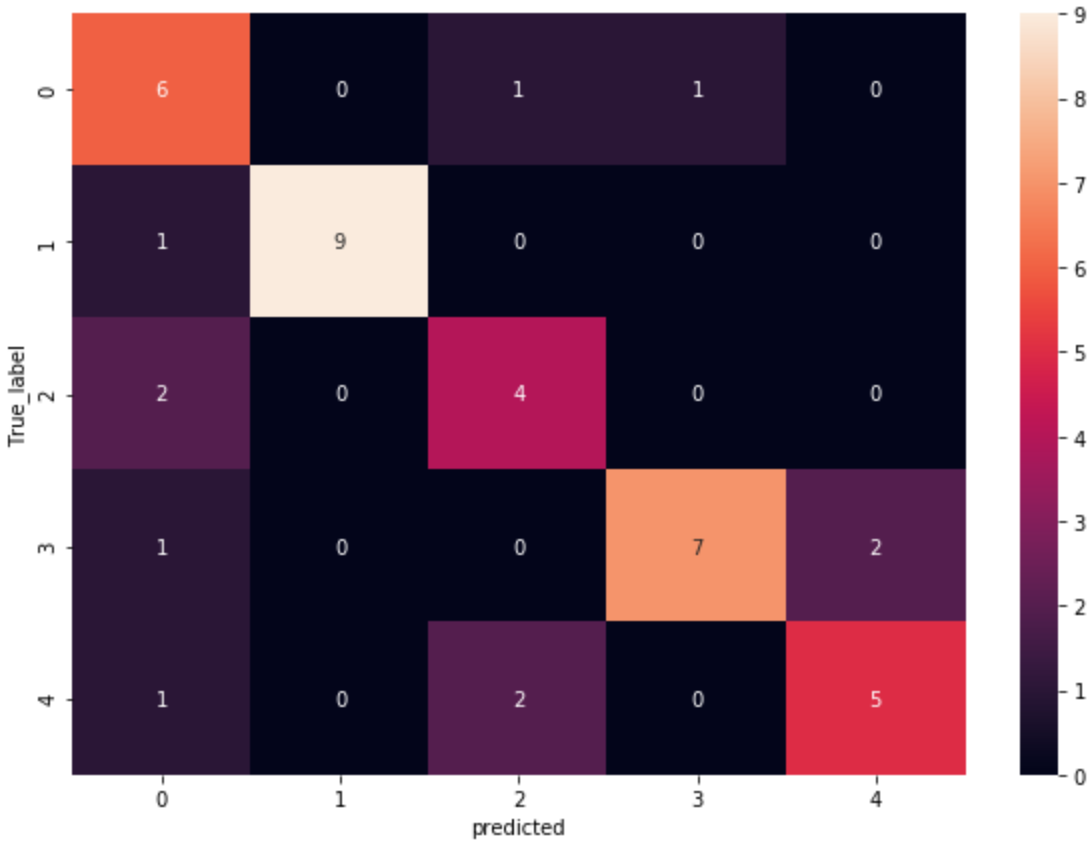

Fig 9 Confusion Matrix

Some test images were used with the web app and the result was as follows:

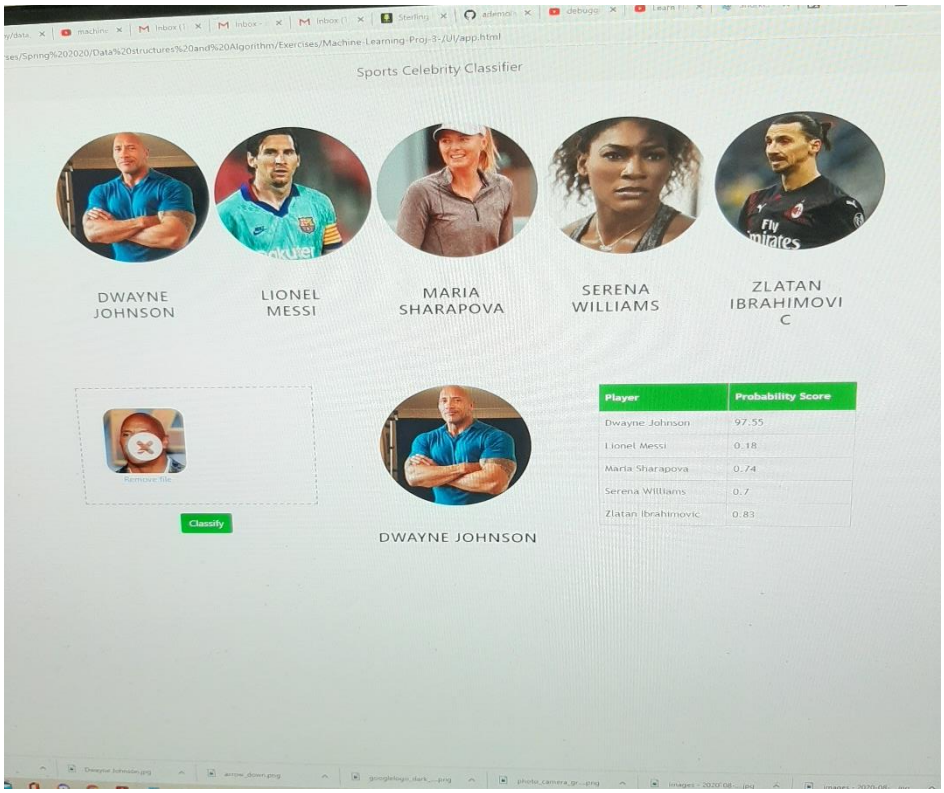

Fig 10 Dwaye Johnson

An image of Dwayne Johnson was put in the drop zone and the classifier classified it correctly with a probability score table of the probability of the image being that of all the five sport celebrities. Here the probability score was 97.55%

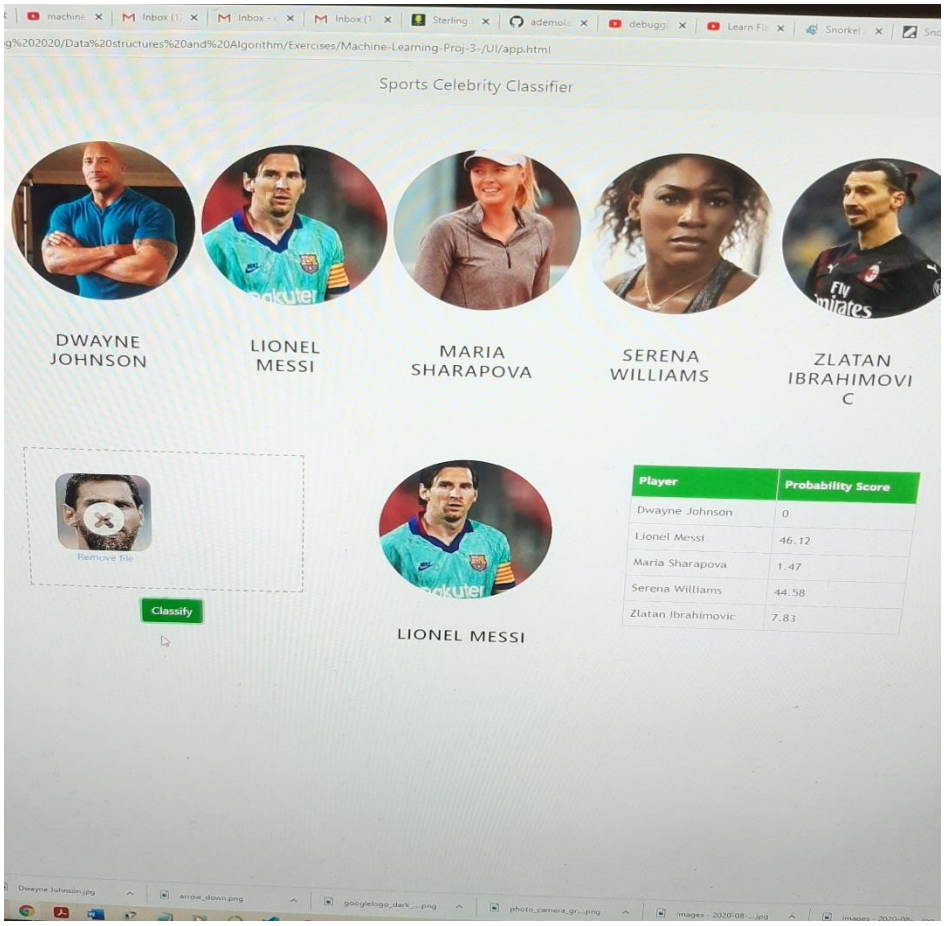

Fig 11 Lionel Messi

An image of Lionel Messi was put in the drop zone and the classifier classified it correctly with a probability score table of the probability of the image being that of all the five sport celebrities. Here the probability score was 46.12%

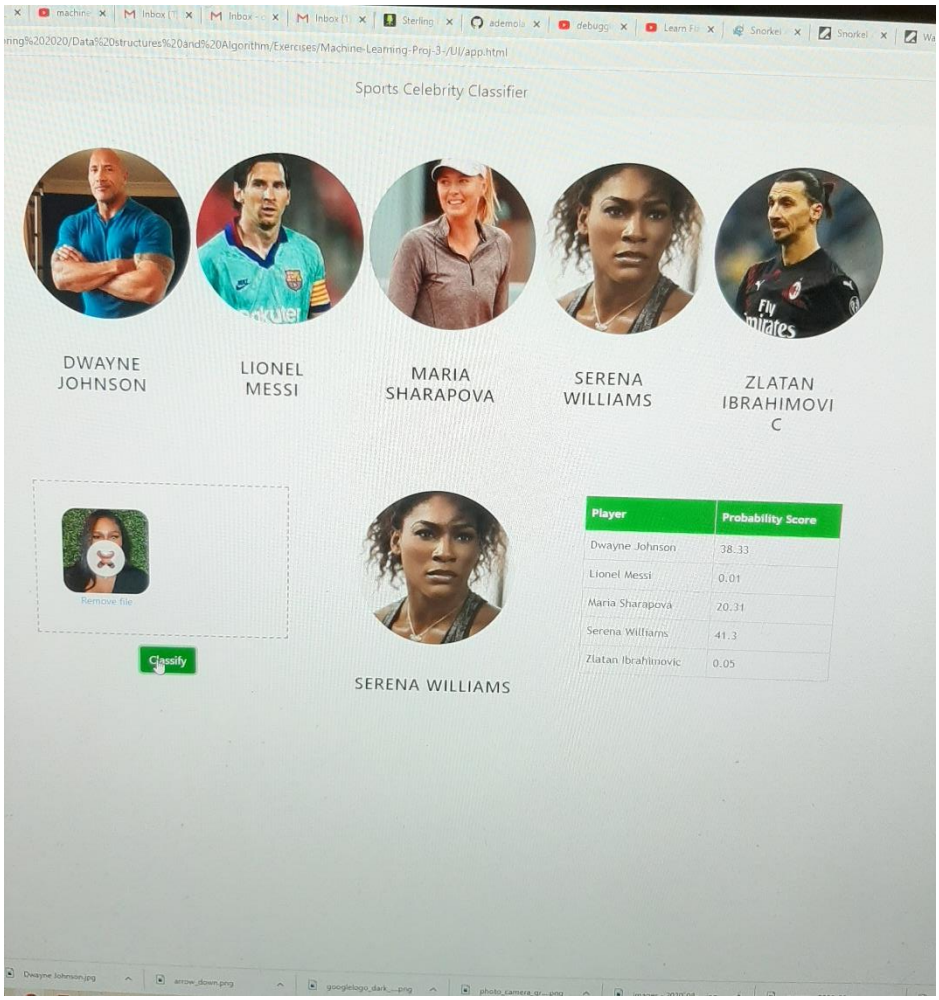

Fig 12 Serena Williams

An image of Serena Williams was put in the drop zone and the classifier classified it correctly with a probability score table of the probability of the image being that of all the five sport celebrities. Here the probability score was 41.3%

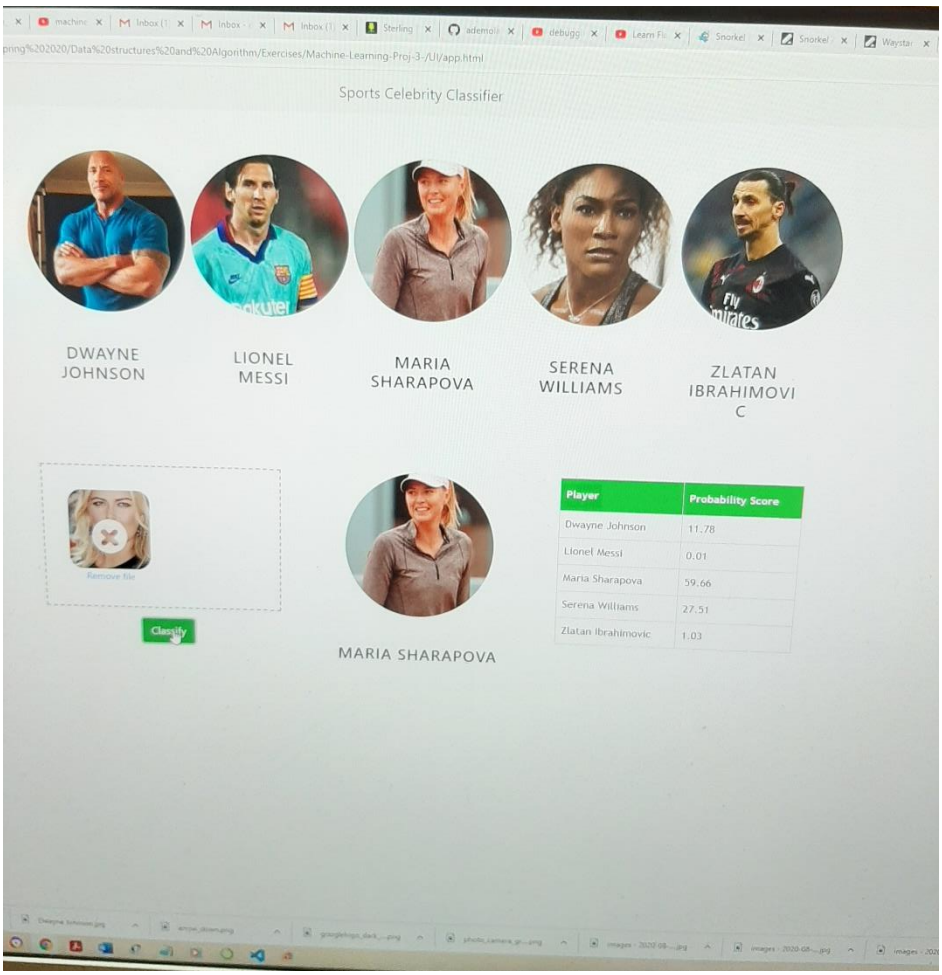

Fig 13 Maria Sharapova

An image of Maria Sharapova was put in the drop zone and the classifier classified it correctly with a probability score table of the probability of the image being that of all the five sport celebrities. Here the probability score was 59.66%

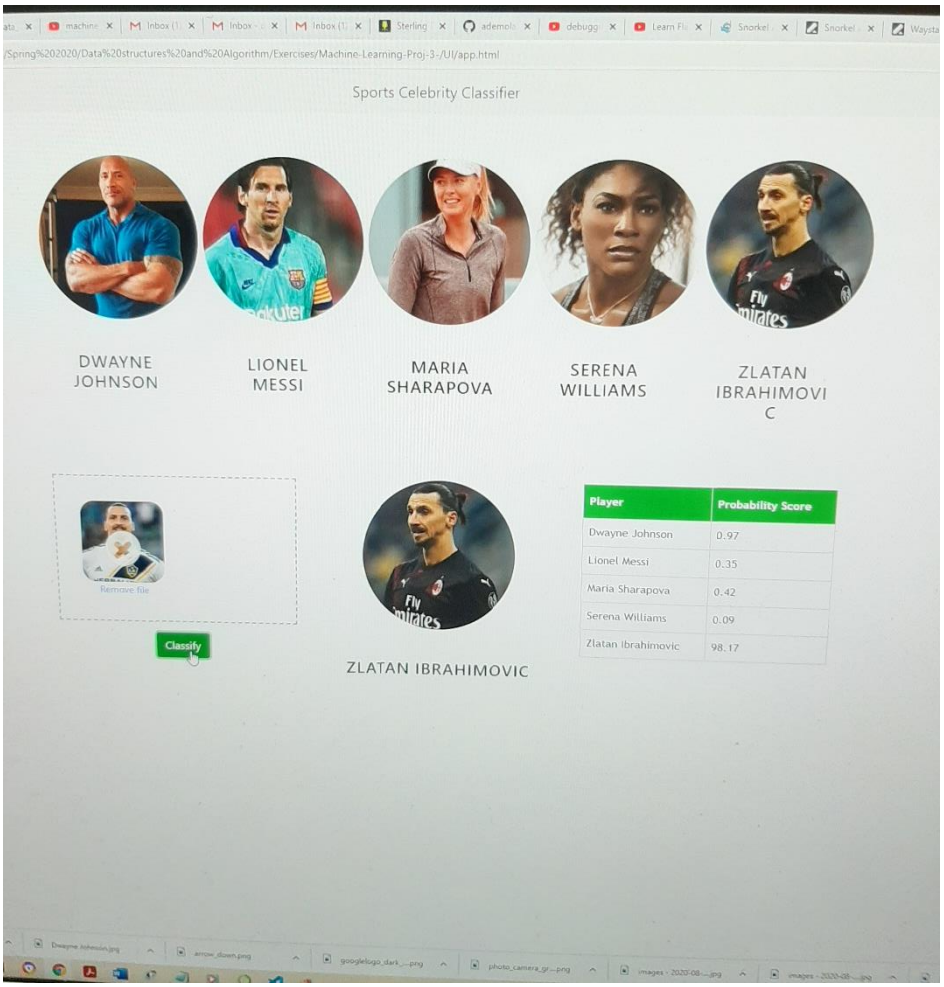

Fig 14 Zlatan Ibrahimovic

An image of Zlatan Ibrahimovic was put in the drop zone and the classifier classified it correctly with a probability score table of the probability of the image being that of all the five sport celebrities. Here the probability score was 98.17%

Conclusion

Our classifier was able to correctly classify the images of the sports celebrity correctly. Some of the probability scores/ confidence level may seem low but it was even lower for all the other categories that were wrong.

Improvement

The model's accuracy can be improved using neural networks. The use of differnt neural network architectures can increase the model's confidence level in predicting the images. Also

the haarcascade could be trained on the dataset so that it will perform better in detecting the facial features in the images and hence we will have much more data to rain the model on.