HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

PROJECT 1 REPORT

Footballers Classification

NGUYEN VIET MINH
20214917

Course: Project 1

Supervisor: Ph.D. Nguyen Huu Duc _____

Signature

Department: Computer Science

School: Information and Communication Technology

HANOI, 12/2023

TABLE OF CONTENTS

1 Introduction

In the realm of sports analytics and player performance evaluation, one intriguing challenge is the classification of footballers based on facial images, particularly when faced with the constraint of having images of uniform length. The task involves discerning various attributes, skills, or characteristics of football players solely from their facial features, offering a unique perspective on player profiling and identification.

Given the vast diversity among footballers and the intricate nature of facial recognition, the development of intelligent computer vision algorithms becomes imperative for efficient player classification. The goal of this classification system is to accurately categorize footballers into different classes or groups based on facial attributes. This task assumes significance not only for sports enthusiasts and analysts but also for team managers and scouts seeking to evaluate players in a non-intrusive and rapid manner.

In our approach, we address the challenge of facial classification with a focus on uniform image length. This constraint necessitates a tailored solution that can effectively capture key facial features, despite the variability in individual player appearances. Leveraging deep learning techniques, our proposed algorithm aims to extract meaningful patterns and representations from facial images, enabling the classification of footballers into distinct categories.

A critical aspect of our work involves handling the inherent variability in facial expressions, lighting conditions, and pose variations commonly encountered in footballer images. We recognize the significance of creating a robust model that can generalize well across diverse facial characteristics while maintaining consistency in image length. This ensures the feasibility of implementing the system across various football datasets and scenarios.

In addition to the primary classification task, our work also considers the challenge of addressing facial recognition problems that may arise, such as handling occlusions, variations in facial hair, or changes in player appearance over time. Through innovative techniques and advanced deep learning architectures, we strive to enhance the accuracy and reliability of the facial classification system.

The implications of successfully classifying footballers based on facial images extend beyond the realm of sports analysis. Such a system could be employed in player tracking, team management, and even fan engagement, contributing to a more comprehensive understanding of individual players within the dynamic world of football.

2 Dataset

Our problem is inspired by the 10 famous athletes in football field which are alisson becker, david beckham, cristiano ronaldo, erling halland, n'golo kante, kevin de bruyne, kylian mbappe, lionel messi, neymar junior, virgil vandijk

2.1 Dataset Collected

The dataset consists of 10 famous footballers, each one having 100 images crawled from Google image. So our full dataset is nearly 1000 images, because of some error images that we need to get rid of will be discussed later

The labels of videos are described in detail as:

- **alisson:** 0, full name: Alisson Becker
- **cristiano:** 1, full name: Cristiano Ronaldo
- **beckham:** 2, full name: David Beckham
- **halland:** 3, full name: Erling Halland
- **kevin:** 4, full name: Kevin De Bruyne
- **mbappe:** 5, full name: Kylian Mbappe
- **messi:** 6, full name: Lionel Messi
- **kante:** 7, full name: N'Golo Kante
- **neymar:** 8, full name: Neymar Junior
- **vandijk:** 9, full name: Virgil Vandijk

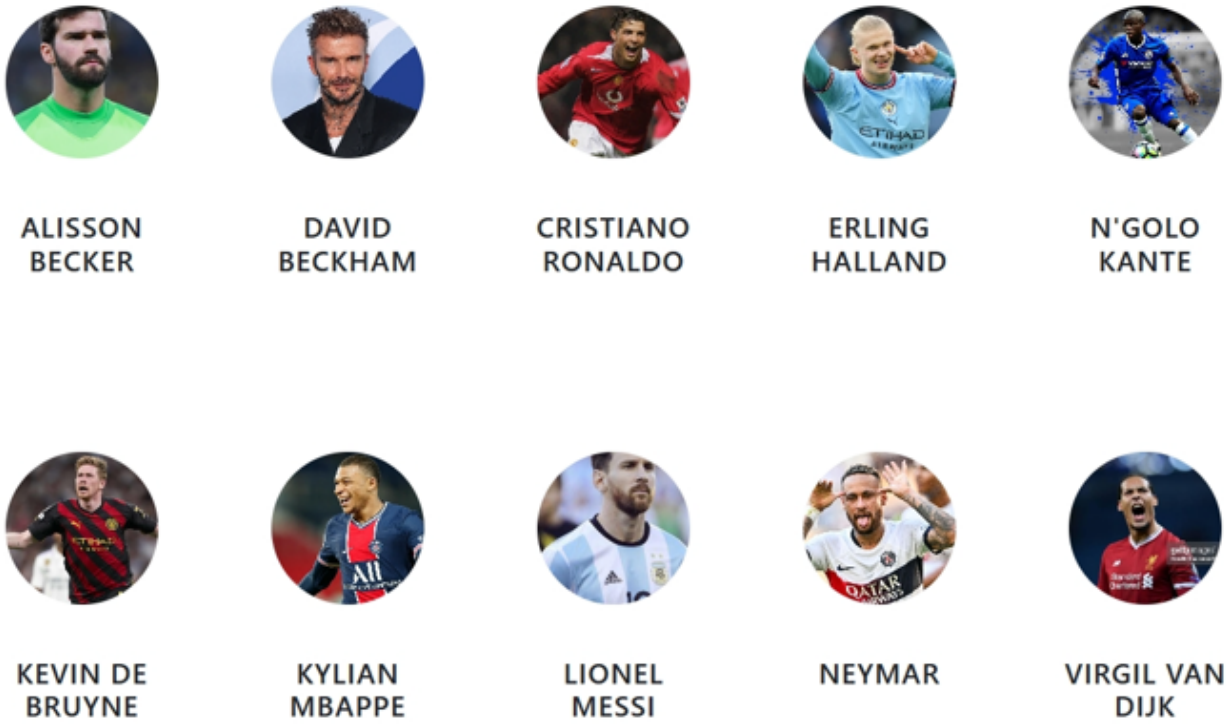

Figure 2.1: Footballers Image

Each image labeled in the dataset captures a specific footballer.

Due to the massive size of the error when collecting the dataset, we will discuss in 2.3.

2.2 Data Crawling

Data crawling, also known as web scraping, is the process of extracting information from websites. It involves fetching and parsing web pages to collect data for various purposes, such as research, analysis, or building datasets.

In the first step identify the target website, we choose Google images as target website for crawling, as google only need us to put our thing we want to look up by put the name in the search bar. After understand the structure, analyze the structure of the website, including the HTML structure, the location of the data I want, and any patterns in the URLs. Then we choose a Crawling Tool which is Selenium (Python), we create and execute the code, make it most suitable with many cases that we do not need to implement many times. Just 1 execute then everything will be running

2.3 Error Dataset

I give this error dataset 1 subsection because there are quite a lot of errors each time crawl from website's Google image. After delete all kind of error image which can be witnessed by eyes, no need putting into haarcascade model yet, below is the table how many remain images each label have.

Label	Number of images
0	75
1	77
2	72
3	81
4	48
5	68
6	79
7	38
8	59
9	61

Table 2.1: Number of images in each folder

Some players' images may have lots of errors compared to others because of something like duplicates, more than 1 person in an image, tattoo or paint, or video game graphics but not the real image of the player, the player turns back into the camera not showing his face. Now let's see some of them, and how the error can be looked like.

Figure 2.2: Some Error Image

Some other error with program

Failed cannot write mode P as JPEG

Failed cannot identify image file <io.BytesIO object at 0x000001FF54A4B1F0>

3 Data Preprocessing

3.1 Haarcascade OpenCV

Before going to this model, we have already deleted some images that we know it is not going to fit with our model in the data cleaning step above, which is in the error dataset subsection. Now if our images still somehow do not work probably with the model like can not detect face, 2 eyes or both

will also be eliminated.

The Haarcascade classifier is a machine learning object detection method used to identify objects in images or video. OpenCV, an open-source computer vision library, provides pre-trained Haar Cascade models for various objects. These models are XML files that contain information about the features of the object they are trained to detect.

We use this classifier model for detecting the face and 2 eyes of an image. If an image can be detected with 2 eyes and face by haarcascade model, we will keep that image for the next preprocessing step.

We not going to go in detail how haarcascade can detect human feature here, just keep in mind that it a pre-trained model doing the job detect and crop the feature of humans by image that we want.

3.2 Wavelet transform

In this project, we crop the image by haarcascade model, crop only the face which image can detect both face and 2 eyes, and save to cropped folder. Then we continue with our second step of preprocessing is applying wavelet transform to the cropped image.

Wavelet transform is a mathematical technique used for signal and image analysis. It decomposes a signal into components with different frequency bands, allowing for both time and frequency localization. This technique will help us in feature extraction the face of players, make detail and special of it for classify and recognize between 10 footballers

Fourier transform is a mathematical operation that transforms a function of time (or space) into its frequency representation. It decomposes a signal into its constituent frequencies, providing a global view of the signal in the frequency domain.

3.2.1 Mathematical Representation of Fourier transform

The continuous Fourier transform of a function $f(t)$ is given by:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt$$

3.2.2 Characteristics

1. Provides a global frequency representation.
2. Does not provide information about when a particular frequency occurs.

3.2.3 Mathematical Representation

The continuous wavelet transform of a function $f(t)$ is given by:

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \psi^* \left(\frac{t-b}{a} \right) dt$$

Here, a and b control the scale and translation of the wavelet, and $\psi(t)$ is the analyzing wavelet.

3.2.4 Characteristics

1. Provides localized information in both time and frequency domains.
2. Useful for analyzing signals with non-stationary characteristics.
3. Allows a multi-resolution analysis, capturing details at different scales.

3.2.5 Relationship between Wavelet and Fourier Transforms

1. **Time-Frequency Localization:** Fourier Transform provides global frequency information but lacks time localization. Wavelet Transform allows both time and frequency localization, offering a more detailed analysis of signal variations over time.
2. **Multi-Resolution Analysis:** Wavelet Transform supports multi-resolution analysis, capturing details at different scales, which Fourier Transform does not inherently provide.
3. **Complementary Techniques:** Fourier Transform is often suitable for stationary signals with constant frequencies over time. Wavelet Transform is effective for signals with non-stationary characteristics, capturing transient events and localized changes.
4. **Efficiency in Compression:** Wavelet Transform is often used in signal and image compression due to its ability to capture and represent important details at different scales more efficiently.

Using the two preprocessing steps described above and cleaning data for better data quality is instrumental in maintaining a standardized and consistent input format across all stages of a deep learning project. By seamlessly integrating these transformations, we ensure that input data, whether sourced from training or used for inference, adheres to the required format and data types expected by the model. Use wavelet transform as a feature for training our model, we use 1 image of this and 3 raw image stack for training. This careful preprocessing pipeline not only streamlines the data input process but also establishes a foundation for consistent and reliable model training and inference outcomes.

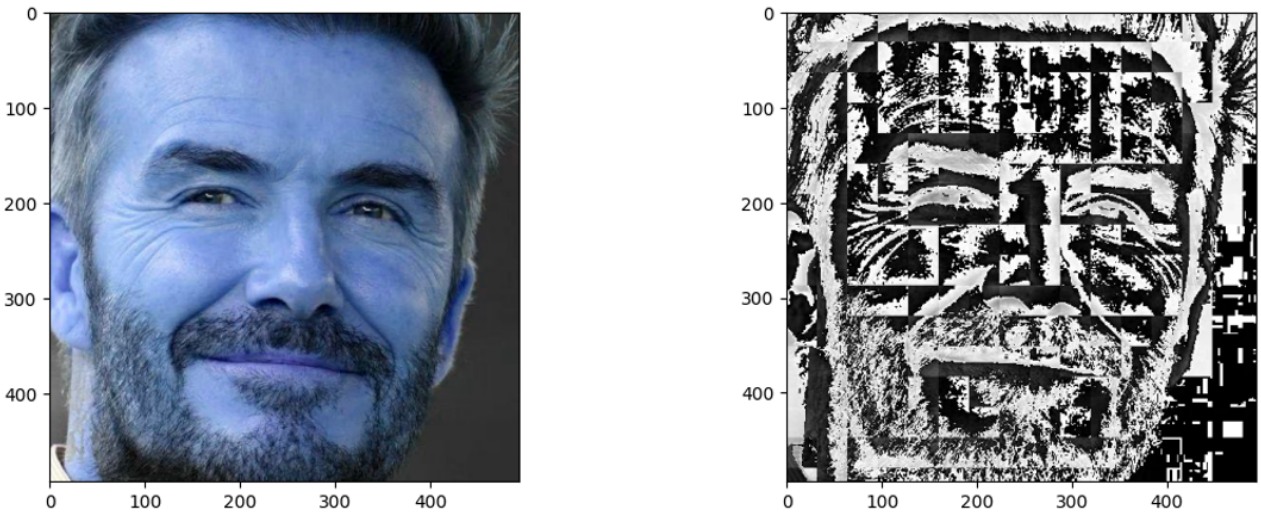

Figure 3.1: Wavelet transform image

In wavelet transformed image, you can see edges clearly and that can give us clues on various facial features such as eyes, nose, lips.

4 Model

We utilize 3 different basic ML models

4.1 SVM Model

A support vector machine (SVM) is a machine learning algorithm that uses supervised learning models to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs. SVMs are widely adopted across disciplines such as healthcare, natural language processing, signal processing applications, and speech image recognition fields.

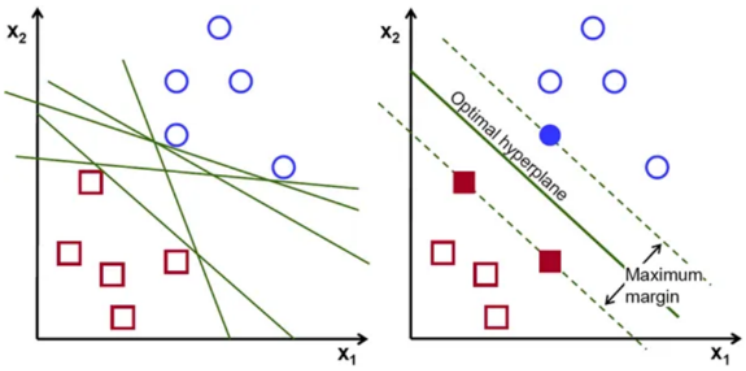

Figure 4.1: SVM model

Technically, the primary objective of the SVM algorithm is to identify a hyperplane that distinguishably segregates the data points of different classes. The hyperplane is localized in such a manner

that the largest margin separates the classes under consideration.

4.1.1 SVM Model for multiclass classification

Support Vector Machines (SVMs) are initially designed for binary classification problems, but they can be extended for multiclass classification through various techniques. One common approach is to use a combination of binary classifiers in a one-vs-one or one-vs-all (also known as one-vs-the-rest) strategy.

Here's an explanation of both strategies:

One-vs-One (OvO): For N classes, create N * (N-1) / 2 binary classifiers, where each classifier is trained to distinguish between two classes. During prediction, each classifier provides a class label, and the class with the most "votes" is the final predicted class.

One-vs-All (OvA or OvR - One-vs-Rest): For N classes, create N binary classifiers, each trained to distinguish one class from the rest. During prediction, each classifier provides a score for its designated class, and the class with the highest score is the final predicted class.

The choice between OvO and OvA depends on factors such as the number of classes and the efficiency of the classifier.

When applying SVMs for multiclass classification, the typical SVM optimization problem is modified to handle multiple classes. Some popular methods for extending SVMs to multiclass problems include:

One-vs-One (OvO) Extension: Train multiple binary SVM classifiers, each distinguishing between two classes. During prediction, use a voting scheme to decide the final class based on the outputs of all pairwise classifiers.

One-vs-All (OvA) Extension Train N binary SVM classifiers, each distinguishing one class from the rest. During prediction, choose the class with the highest decision function output among all classifiers.

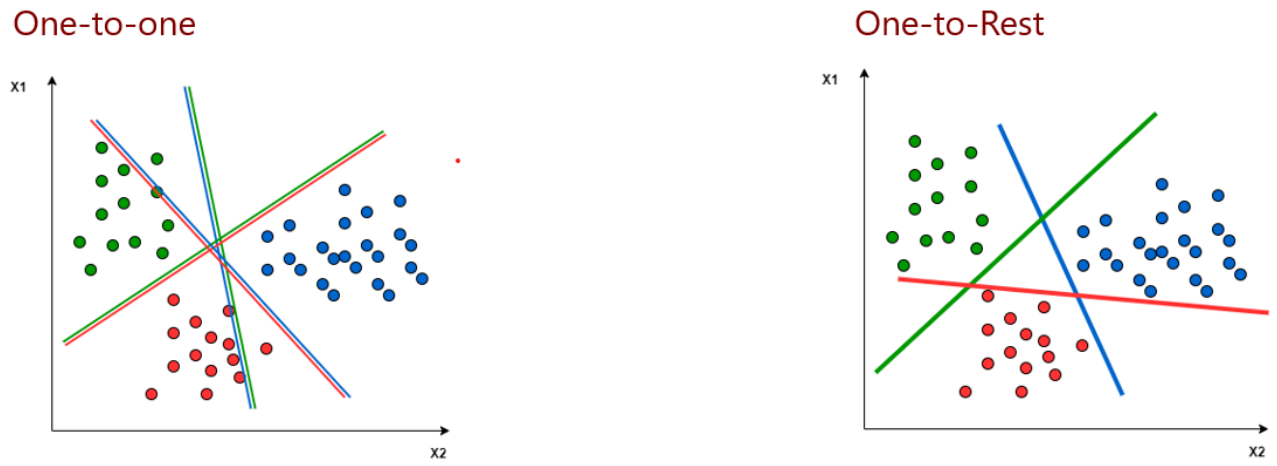

Figure 4.2: SVM multiclass classification model

We will use libraries like Scikit-learn in Python make it easy to implement SVMs for multiclass classification. we will use the SVC (Support Vector Classification) class for this purpose.

4.2 Random Forest Model

Random Forest is a popular ensemble learning method used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

Here are the key features and concepts associated with Random Forest:

Ensemble Learning: Random Forest belongs to the ensemble learning family, where multiple models are combined to create a more robust and accurate model.

Decision Trees: The basic building blocks of Random Forest are decision trees. Each tree is constructed using a subset of the training data and a random subset of features at each split.

Bagging (Bootstrap Aggregating): During training, each tree is trained on a bootstrap sample, which is a random sample drawn with replacement from the original dataset. This technique helps introduce diversity among the trees.

Random Feature Selection: At each node of a decision tree, only a random subset of features is considered for splitting. This introduces further diversity and reduces the risk of overfitting.

Voting (Classification) or Averaging (Regression): For classification tasks, each tree "votes" for a class, and the class with the most votes becomes the final prediction. For regression tasks, the individual tree predictions are averaged to obtain the final prediction.

Out-of-Bag (OOB) Error: Since each tree is trained on a bootstrap sample, there will be data points that are not used in the training of each tree. These out-of-bag samples can be used to estimate the performance of the Random Forest without the need for a separate validation set.

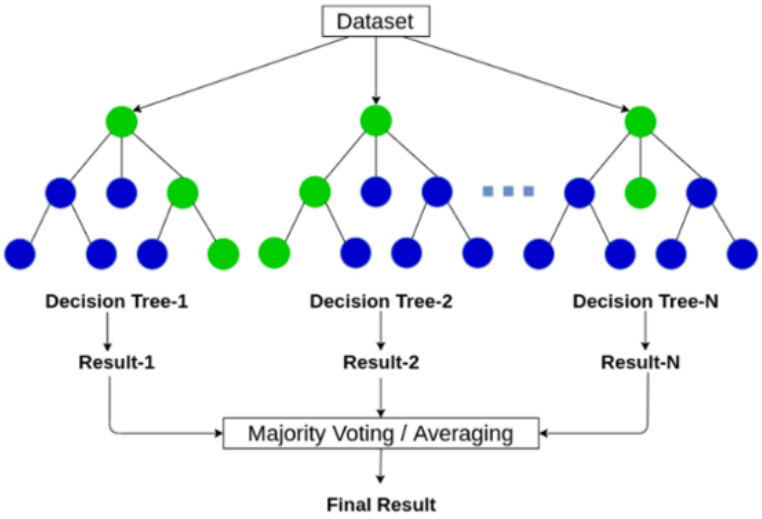

Figure 4.3: Random Forest model

We also use libraries like Scikit-learn in Python make it easy to implement RF for multiclass classification. we will use the RandomForestClassifier class in sklearn.ensemble for this purpose.

4.3 Logistic regression Model

Logistic regression is a statistical method used for binary classification, which means predicting the probability of an instance belonging to one of two classes. Despite its name, logistic regression is a classification algorithm, not a regression algorithm.

Logistic regression is widely used in various fields such as medicine, marketing, finance, and more for binary classification tasks. It can be extended to handle multiclass classification through techniques like one-vs-all or one-vs-one. Additionally, logistic regression assumes that the relationship between the independent variables and the log-odds of the dependent variable is linear. If the relationship is more complex, other models like decision trees or neural networks may be more appropriate.

Logistic Regression for Multiclass Classification:

In logistic regression, the hypothesis function is typically the sigmoid function for binary classification. For multiclass problems, the softmax function is commonly used to generalize the sigmoid function. The softmax function takes a vector of raw scores (logits) and converts them into probabilities.

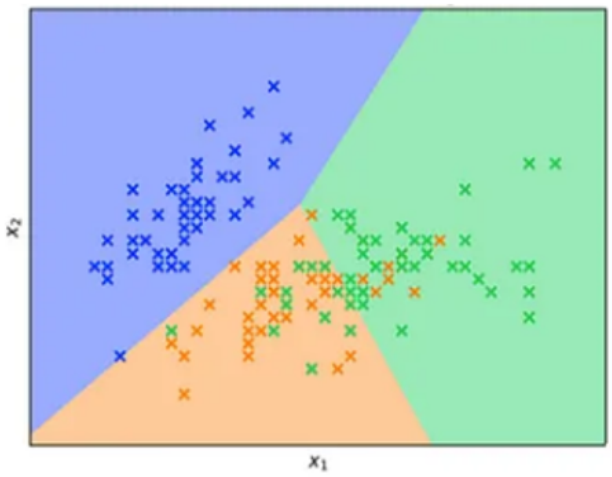

Figure 4.4: Logistic Regression model

4.4 Fine-Tuning Hyperparameter

Fine-tuning hyperparameters is a crucial step in machine learning to improve the performance of your model. Hyperparameters are configuration settings for your model, and finding the right combination can significantly impact the model's accuracy and generalization.

I use grid search with some most basic parameter that could make model better then save the best parameter for each model

```python
model_params = {
    'svm': {
        'model': svm.SVC(gamma='auto',probability=True),
        'params' : {
            'svc__C': [1,10,100,1000],
            'svc__kernel': ['rbf','linear']
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'randomforestclassifier__n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'logisticregression__C': [1,5,10]
        }
    }
}
```

Figure 4.5: Hyperparameter Basic Tuning

5 Experiment Result

After training which is 80% of full dataset, we ran our models on the test set which is 20% full dataset and obtained following result.

Our models return a integer value from 0 to 9 represent for 10 footballers in each number. Also the model can guess the percentage of which classes the image might belong to, for example it guess 25% is messi class 4 and 45% is ronaldo class 3, of course final result will be ronaldo

The figure below show the result prediction of our 3 models when using their best parameter.

Support Vector Machine	Random Forest	Logistic Regression
0.6372549019607843	0.3235294117647059	0.7254901960784313

Figure 5.1: Result of 3 best para model

As you can see, our result showing that logistic regression is best model so far for this problem of classification, it maybe because it can welly divided clearly how region and facial expression also maybe color between 10 players face.

So we will take logistic regression with best parameter as our final model for prediction and take into application

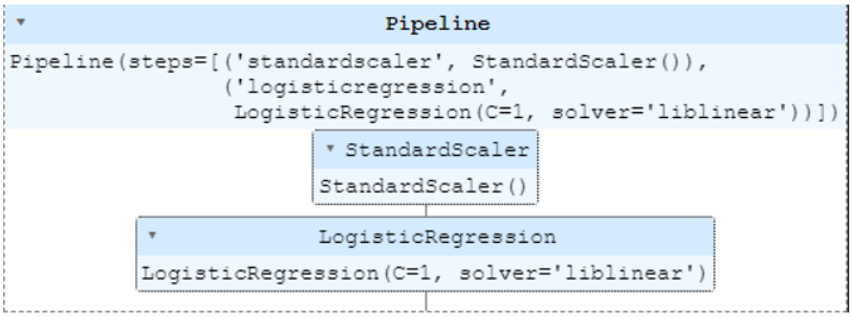

Figure 5.2: Final Model

Next, we draw confusion matrix to see more clearly how well is our model with label 0 to 9 each represent for a player

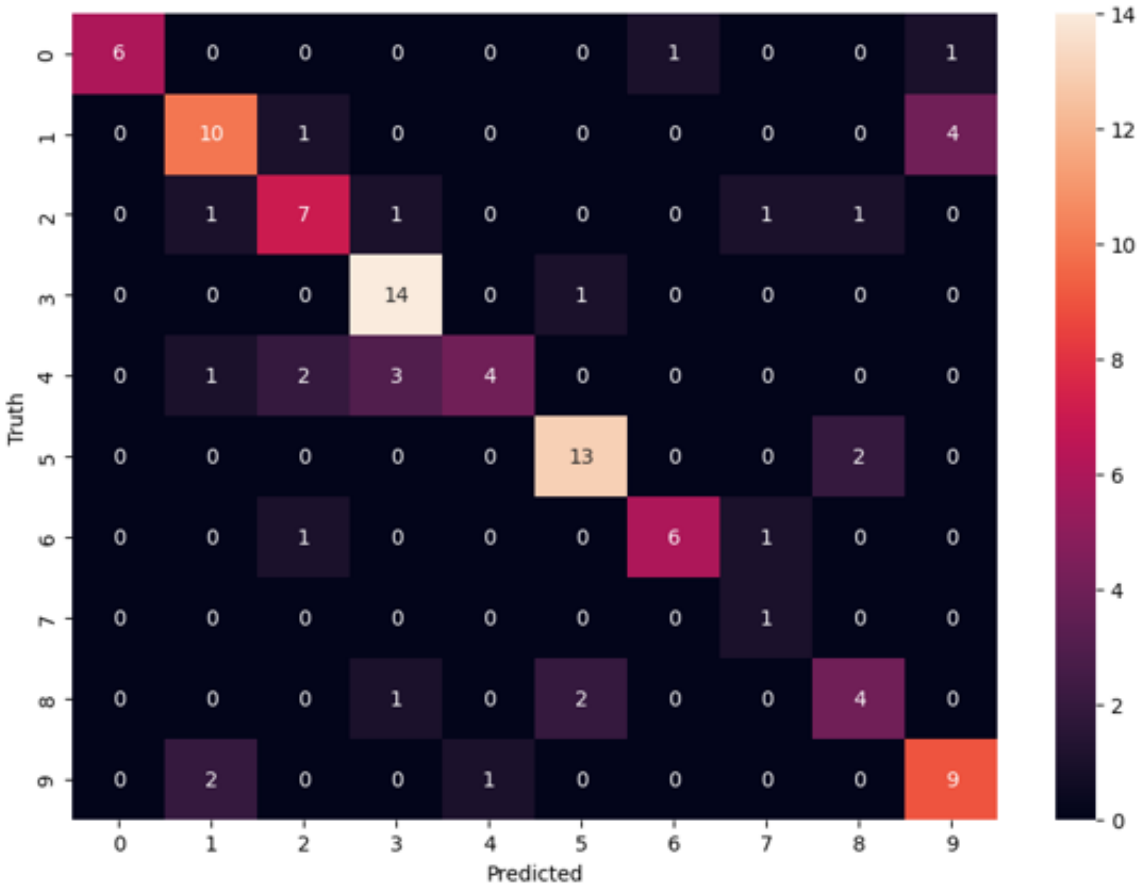

Figure 5.3: Confusion matrix of `Final Model`

We can see clearly our model do quite well with some footballers while other is not, this may because of the data collecting not so precisely

6 Conclusion

With the data crawled from https://www.google.co.uk/ images, search bar which after includes 100 images for each footballer, some error need to be eliminated but others still remain good quality enough for us to built up a good model classification.

We have taken different approaches to our problem of footballers classification included 3 basic machine learning method SVM, Random Forest, Logistic Regression then choose the best model with parameter that give the most accuracy result. After all step, we built a server for host to website and User Interface built by HTML file, supported by css and js file all of this for making application for our model to be applied to real life problem. Some of the list that we can do further for better result model is:

Although our basic ML model do quite well, we can use stronger ML model like XGB or DL model.

We can apply data augmentation or maybe take more data from google like 1000 for each classes, take more mean clean and check more.

Use other model in preprocessing step, though very appreciate to haarcascade to one of the first model in detecting part of human body. Now it do not quite good as expected, we can try another model to detect face and 2 eyes, take full advantages of data collected.