CHAPTER 1

INTRODUCTION

Facial recognition is a category of biometric software that maps an individual's facial features mathematically and stores the data as a face print. The software uses deep learning algorithms to compare a live capture or digital image to the stored face print in order to verify an individual's identity.

Every Machine Learning algorithm takes a dataset as input and learns from this data. The algorithm goes through the data and identifies patterns in the data. For instance, suppose we wish to identify whose face is present in a given image, there are multiple things we can look at as a pattern:

- Height/width of the face.
- Height and width may not be reliable since the image could be rescaled to a smaller face. However, even after rescaling, what remains unchanged are the ratios – the ratio of height of the face to the width of the face won't change.
- Color of the face.
- Width of other parts of the face like lips, nose, etc.

Clearly, there is a pattern here – different faces have different dimensions like the ones above. Similar faces have similar dimensions. The challenging part is to convert a particular face into numbers – Machine Learning algorithms only understand numbers. This numerical representation of a "face" is termed as a *feature vector*.

1.1 ABOUT THE PROJECT

The project is entitled as **"IMAGE RECOGNITION USING PYTHON"** classify the images based on the features of Olivetti faces. The Olivetti faces dataset consist of ten different images of each of 40 distinct subjects

The images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright position. (with tolerance for some side movement). The images have size of 64 x 64.

CHAPTER 2

SURVEY OF THE TECHNOLOGIES

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a popular platform used for research and development of production systems. It is a vast language with number of modules, packages and libraries that provides multiple ways of achieving a task.

Python and its libraries like NumPy, SciPy, Scikit-Learn, Matplotlib are used in data science and data analysis. They are also extensively used for creating scalable machine learning algorithms. Python implements popular machine learning techniques such as Classification, Regression, Recommendation, and Clustering.

Python offers ready-made framework for performing data mining tasks on large volumes of data effectively in lesser time. It includes several implementations achieved through algorithms such as linear regression, logistic regression, Naïve Bayes, k-means, K nearest neighbour, Support vector Machine and Random Forest.

PYTHON IN MACHINE LEARNING

Python has libraries that enable developers to use optimized algorithms. It implements popular machine learning techniques such as recommendation, classification, and clustering. Therefore, it is necessary to have a brief introduction to machine learning before we move further.

Machine Learning

Data science, machine learning and artificial intelligence are some of the top trending topics in the tech world today. Data mining and Bayesian analysis are trending and this is adding the demand for machine learning.

Machine learning is a discipline that deals with programming the systems so as to make them automatically learn and improve with experience. Here, learning implies recognizing and understanding the input data and taking informed decisions based on the supplied data. It is very difficult to consider all the decisions based on all possible inputs. To solve this problem, algorithms are developed that build knowledge from a specific data and past experience by applying the principles of statistical science, probability, logic, mathematical optimization, reinforcement learning, and control theory.

LIBRARIES AND PACKAGES

To understand machine learning, you need to have basic knowledge of Python programming. In addition, there are a number of libraries and packages generally used in performing various machine learning tasks as listed below −

- **numpy** − is used for its N-dimensional array objects
- **pandas** − is a data analysis library that includes dataframes
- **matplotlib** − is 2D plotting library for creating graphs and plots
- **scikit-learn** − the algorithms used for data analysis and data mining tasks
- **seaborn** − a data visualization library based on matplotlib

Learning can be broadly classified into three categories, as mentioned below, based on the nature of the learning data and interaction between the learner and the environment.

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning

SUPERVISED LEARNING

Supervised learning is the Data mining task of inferring a function from **labeled training data**. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A **supervised learning algorithm** analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for **unseen instances**.

Classification attempts to find the appropriate class label, such as analyzing positive/negative sentiment, male and female persons, benign and malignant tumors, secure and unsecure loans etc.

FEATURES OF PYTHON

Python has libraries that enable developers to use optimized algorithms. Let us mention some of the features:

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

CHAPTER 3

REQUIREMENT AND ANALYSIS

3.1 HARDWARE AND SOFTWARE REQUIREMENTS

PROCESSOR : Intel core i3

RAM : 2 GB

HARD DISK : 320 GB

VIDEO : 1204 X 1366 RESOLUTION

SOFTWARE SPECIFICATION

OPERATING SYSTEM : WINDOWS 7

ANALYTICAL TOOL : PYTHON (ANACONDA)

CHAPTER 4

MODEL ANALYSIS

4.1 SYSTEM ANALYSIS

Model analysis is the analysis of the model developed and identification of the requirements that it should meet. The system flow diagram and various analysis such as data preprocessing are discussed in this chapter.

4.2 SYSTEM FLOW DIAGRAM

System flow diagram is a graphical representation of flow of data in the analysis, represents the work process of analysis and used in designing and documenting complex process or programs. It helps to visualise what is going on and thereby help the people to understand the process.

The entire flow of the analysis is shown. The flow shows the analysis to be carried out during analysis of any data provided. The data is to be collected through primary or secondary sources. Pre-processing the dataset and based on the type of data the model is to be choosen for analysing. The model should be evaluated using the evaluation metrics.

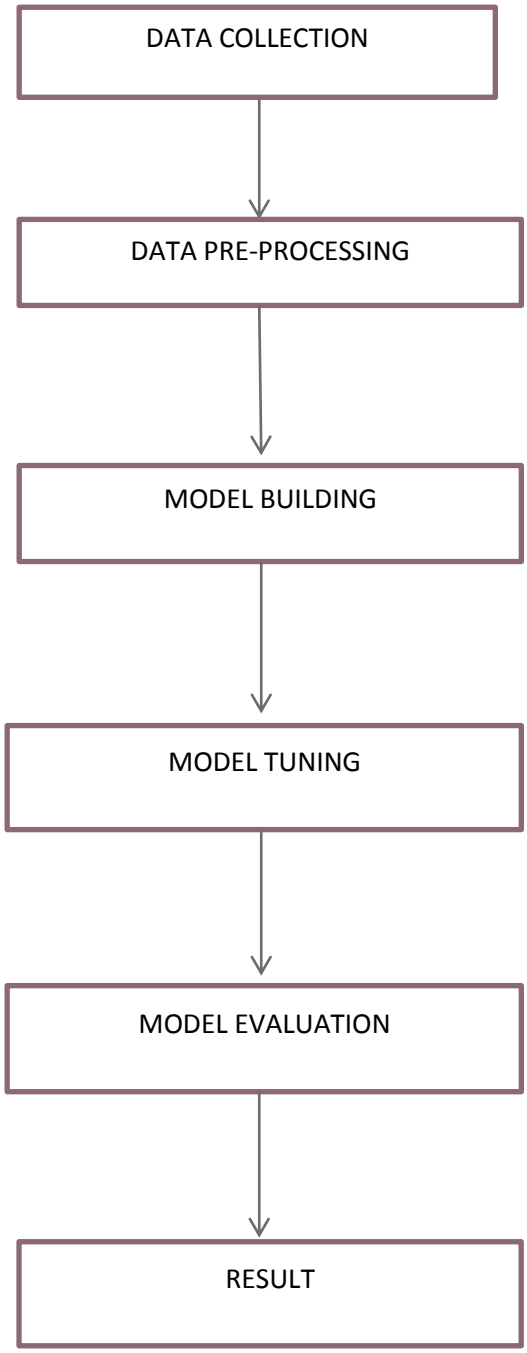

Figure 4.2 Flow Chart

4.3 DATA PREPROCESSING

Import the libraries numpy , matplot, sklearn. The numpy package is the core library for scientific computing , the matplot can be used to create visualisation plots and sklearn is machine learning library for data analysis and data mining. Using sklearn.datasets fetch the Olivetti faces.

DESCR is used to see the description about the Olivetti dataset which consist of 400 images of ten different persons. KEYS shows that the dataset consist of data, images, target.shape is used to view the dimension of the images.

Flatten the images

It is used to merge the layers of the image into a single layer which flattered the Olivetti images to 64 X 64 array. Displayed the distinct iamges of the 40 persons.

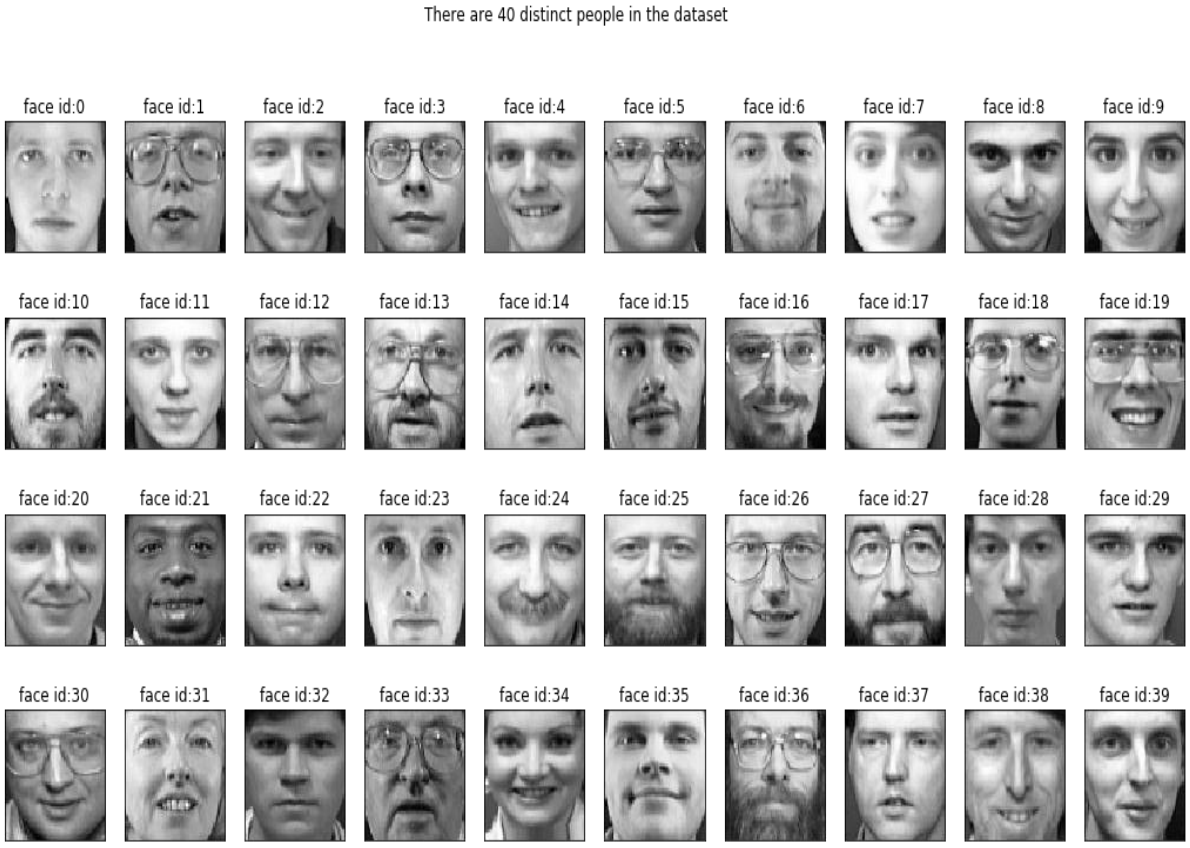

Figure 4.3 Distinct images

Displayed the first image in the Olivetti faces dataset using the index as o and imshow() of the Matplot library.

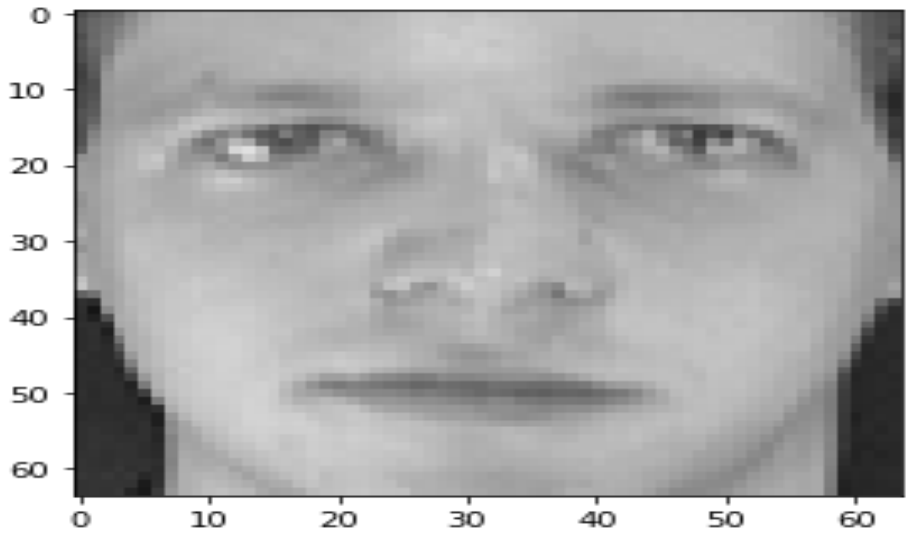

Figure 4.3 First image

Displayed the last image in the Olivetti faces dataset using the index as 39 and imshow() of the Matplot library.

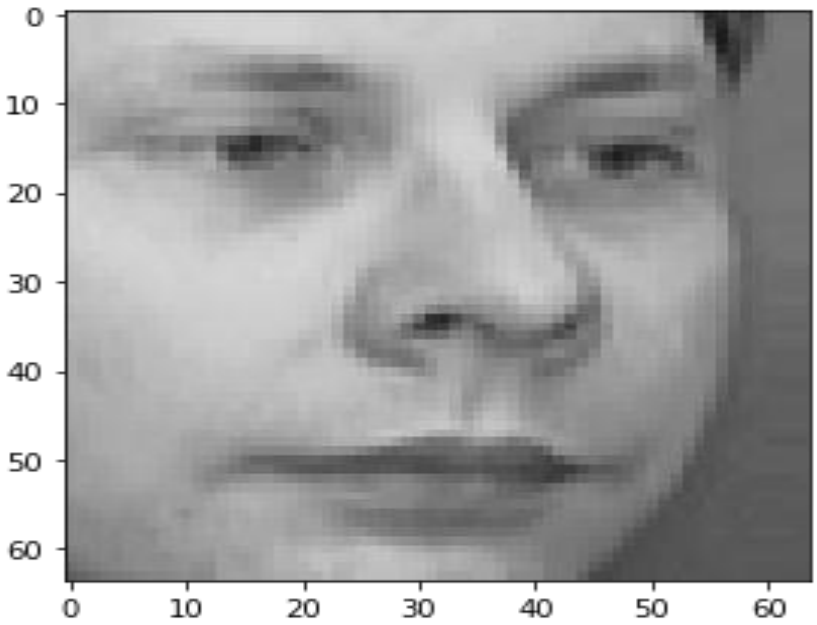

Figure 4.3 Last image

CHAPTER 5

MODEL IMPLEMENTATION

Support Vetctor Machine has become one of the state-of-the-art machine learning models for many tasks with excellent results in many practical applications. One of the greatest advantages of Support Vector Machines is that they are very effective when working on high-dimensional spaces, that is, on problems which have a lot of features to learn from. They are also very effective when the data is sparse. Besides, they are very efficient in terms of memory storage, since only a subset of the points in the learning space is used to represent the decision surfaces.

Support Vector Machines (SVM) are supervised learning methods that try to obtain these hyperplanes in an optimal way, by selecting the ones that pass through the widest possible gaps between instances of different classes. New instances will be classified as belonging to a certain category based on which side of the surfaces they fall on.

We will apply SVM to image recognition, a classic problem with a very large dimensional space.

The SVC implementation has different important parameters like kernel, alpha value and c value probably the most relevant is kernel, which defines the kernel function to be used in our classifier.

The SVM algorithm is implemented in practice using a kernel.

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra, which is out of the scope of this introduction to SVM.

A powerful insight is that the linear SVM can be rephrased using the inner product of any two given observations, rather than the observations themselves. The inner product between two vectors is the sum of the multiplication of each pair of input values.

The equation for making a prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B0 + sum(a i * (x, xi))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and a i (for each input) must be estimated from the training data by the learning algorithm.

Linear Kernel SVM

The dot-product is called the kernel and can be re-written as:

$$K(x, xi) = sum(x * xi)$$

The kernel defines the similarity or a distance measure between new data and the support vectors. The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs.

Other kernels can be used that transform the input space into higher dimensions such as a Polynomial Kernel and a Radial Kernel. This is called the Kernel Trick.

It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. This in turn can lead to more accurate classifiers.

Polynomial Kernel SVM

Instead of the dot-product, we can use a polynomial kernel, for example:

$$K(x, xi) = 1 + sum(x * xi)^d$$

Where the degree of the polynomial must be specified by hand to the learning algorithm. When d=1 this is the same as the linear kernel. The polynomial kernel allows for curved lines in the input space.

Radial Kernel SVM

Finally, we can also have a more complex radial kernel. For example:

$$K(x, xi) = exp(-gamma * sum((x – xi^2))$$

Where gamma is a parameter that must be specified to the learning algorithm. A good default value for gamma is 0.1, where gamma is often 0 < gamma < 1. The radial kernel is very local and can create complex regions within the feature space, like closed polygons in two-dimensional space.

5.1 TRAINING THE DATA

The training set contains a known output and the model learns on this data in order to be generalized to other data later on.we usually have the train data with 80 % or 70% which should be always high when compare to the test data

We import the train_test_split from the sklearn library and used the faces.data and faces.target and the training set we taken size of 70 and remaining 30 for the testing the data.we have used the 20 images for training the dataset and it is displayed here.

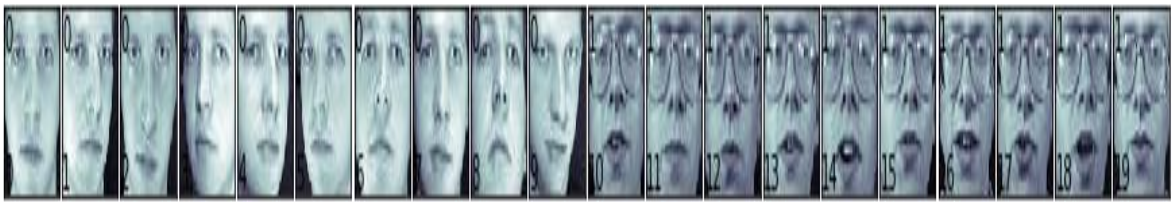

Figure 4.3 First 20 images

METRICS EVALUATION

K-Fold Cross Validation is a common type of cross validation that is widely used in machine learning. In the k-fold cross validation method, all the entries in the original training data set are used for both training as well as validation. Also, each entry is used for validation just once.

Generally, the value of *k* is taken to be 10, but it is not a strict rule, and *k* can take any value and it is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

5.2 TESTING THE DATA

The test dataset (or subset) in order to test our model's prediction on this subset.The testing size used as 30.we used the K-Fold Cross validation for checking the mean value it shows the accuracy score as 0.75 and the classification report is generated with precition and recall value which is used for checking the classifier quality.

We used the kernel poly and is shows the precision and recall value as 0.75 and we classified the dataset based on the feature glasses/no glasses. the index ranges of images of people with glasses are taken for 10 images and we have set the target as glasses.and we have set the condition if the images have the glasses are indexed as 1 and remaining as 0 but the kernel poly does classify well.

The image number 10 in the preceding figure has glasses and was classified as no glasses. If we look at that instance, we can see that it is different from the rest of the images with glasses (the border of the glasses cannot be seen clearly and the person is shown with closed eyes), which could be the reason it has been misclassified.

Figure 5.2 Classified using poly

We used the kernel linear and it shows the precision and recall value as 0.85 and we classified the dataset based on the feature glasses/no glasses. the index ranges of images of people with glasses are taken for 20 images and we have set the target as glasses.and we have set the condition if the images have the glasses are indexed as 1 and remaining as 0 but the kernel linear has classify well and the score is 85 which shows the svm model is good for the ollivetti dataset.

Figure 5.2 Classified using linear

CHAPTER 6

CONCLUSION

The Olivetti dataset is used for doing the classification based on the features using the machine learning model support vector machine. In support vector machine we have tuned the data using the kernel by changing the kernel to poly we get the accuracy score as 75 % and the classification doesn't done well because it shows misclassification, so we tuned kernel to linear it shows the accuracy as 85% and it classifies well so the svc kernel linear is good for this Olivetti dataset Hence SVM shows the best model.

FUTURE WORK:

Face recognition systems used today work very well under constrained conditions, although all systems work much better with frontal mug-shot images and constant lighting. All current face recognition algorithms fail under the vastly varying conditions under which humans need to and are able to identify other people. Next generation person recognition systems will need to recognize people in real-time and in much less constrained situations.

BIBLIOGRAPHY

REFERENCES

- Shai Shalev-Shwartz and Shai Ben-David, " Understanding Machine Learning ",2012.
- Michael Danson, " Python programming for the absolute beginner " 3rd edition ,2010.
- Sandro Tosi, " Matplotlib for Python Developers ", 2014.
- Sridevi pudipeddi, " Image Processing and Acquisition using Python ", 2014.
- https://www.analyticsvidhya.com /blog/2014/12/image-processing-python-basics
- https://docs.python-guide.org/scenarios/imaging
- http://www.scipy-lectures.org/advanced/image_processing/
- https://scikit-learn.org/stable/datasets/olivetti_faces.html
- https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms
- https://pythonprogramming.net/machine-learning-tutorial-python-introduction
- Image Processing and Acquisition using Python By Sridevi pudipeddi