Detection of Face Masks

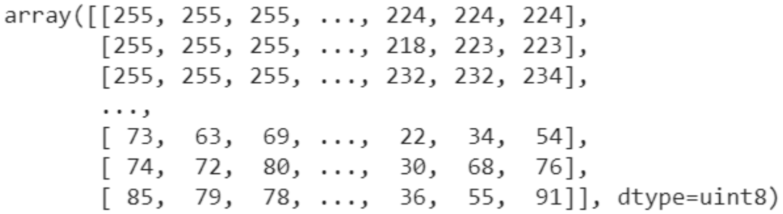
Gautam Kumar and Chirag Jindal

I. Introduction

THE aim of this project is to detect whether a person is wearing face mask or not. Motivation behind this project has come from the situation of covid19 pandemic where it is utmost important for a person to wear face mask while going to a public place. Regulation authority needs a system which can detect whether a person is wearing face mask or not and take the necessary action accordingly.

II. Data Analysis

This Dataset contains around 930 RGB images which includes the images of the person wearing a mask and the person not wearing a mask. It also contains the labels with label 0 for person not wearing a mask and label 1 for person wearing a mask.



```
array([[255, 255, 255, ..., 224, 224, 224],
       [255, 255, 255, ..., 218, 223, 223],
       [255, 255, 255, ..., 232, 232, 234],
       ...,
       [ 73,  63,  69, ...,  22,  34,  54],
       [ 74,  72,  80, ...,  30,  68,  76],
       [ 85,  79,  78, ...,  36,  55,  91]], dtype=uint8)
```

III. Preprocessing

A. Reading the images

Cv2 module is used to read the RGB pixel values of the image.

B. Converting to grayscale image

As colour of the face masks can be anything. Hence they don't add value to the dataset. So images are converted to grayscale images.

C. Resizing of image

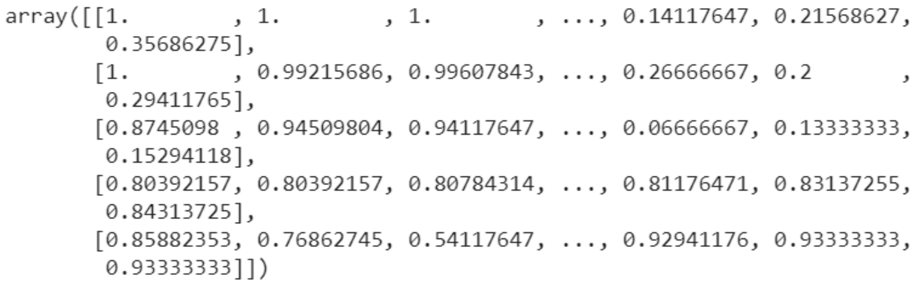
To maintain the uniformity in the dataset, all the images are resized to 100x100 pixels.

D. Rescaling of data

As the size of pixel values doesn't matter. Only their relative strength matters. Hence all the pixel values are rescaled between 0 and 1.

E. Reshaping of data

To work with classifiers which only take 2D data, the images are converted from 2D matrix to 1D array.



```
array([[1.        , 1.        , 1.        , ..., 0.14117647, 0.21568627,
        0.35686275],
       [1.        , 0.99215686, 0.99607843, ..., 0.26666667, 0.2       ,
        0.29411765],
       [0.8745098 , 0.94509804, 0.94117647, ..., 0.06666667, 0.13333333,
        0.15294118],
       [0.80392157, 0.80392157, 0.80784314, ..., 0.81176471, 0.83137255,
        0.84313725],
       [0.85882353, 0.76862745, 0.54117647, ..., 0.92941176, 0.93333333,
        0.93333333]])
```

IV. Data Selection and Extraction

A. Data Selection

As after linearization of data, the number of features become 10000, so reduce the feature, SelectKbest with chi2 method is used.

B. Data Extraction

To reduce the dimensionality of data, PCA is used. It extract useful information from data and project them in a new independent feature space.

V. ML Models

Models used in this project:

- MLP classifier
- SVM Classifier
- Gaussian Naive Bayes Classifier
- Perceptron
- Random Forest Classifier
- Multinomial Naive Bayes
- MLP classifier with PCA
- SVM Classifier with PCA
- Gaussian Naive Bayes Classifier with PCA

A. MLP Classifier

- Alpha: 0.001
- Hidden_layers: 1 layer with 100 neurons
- Learning Rate: 0.001

B. SVM Classifier

- C: 0.8
- Kernel: rbf
- Gamma: auto

C. Gaussian Naive Bayes Classifier

- Default

D. Perceptron

- max_iter=1000
- eta0 = 0.01

E. Random Forest Classifier

- n_estimators=100
- criteria=gini

F. Multinomial Naive Bayes

- default

G. MLP Classifier with PCA

- Alpha: 0.001
- Hidden_layers: 1 layer with 100 neurons
- Learning Rate: 0.001
- Data: transformed using PCA

H. SVM Classifier with PCA

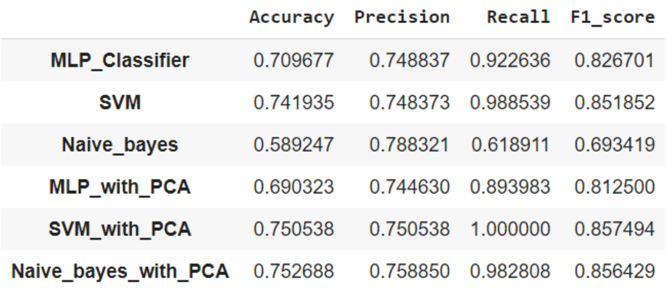
- C: 1
- Kernel: rbf
- Gamma: scale
- Data: transformed using PCA

I. Gaussian Naive Bayes Classifier

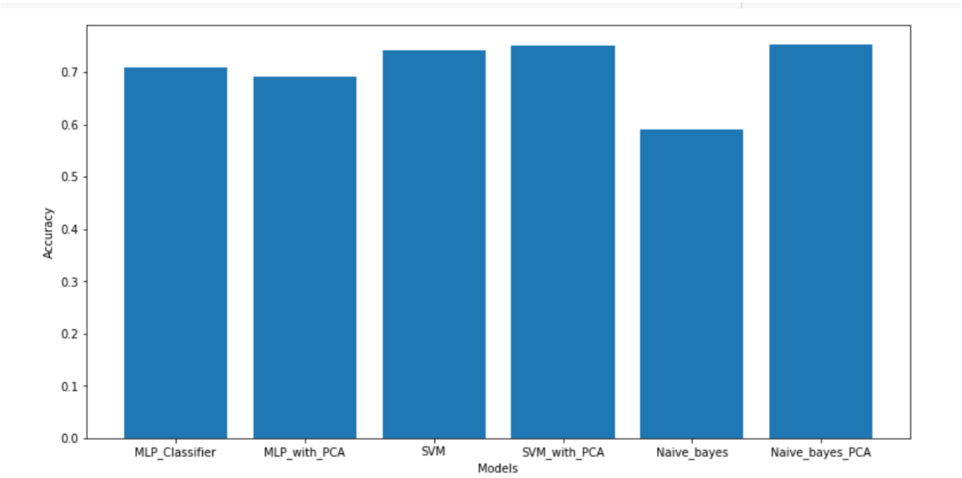
- Default
- Data: transformed using PCA

VI. Results

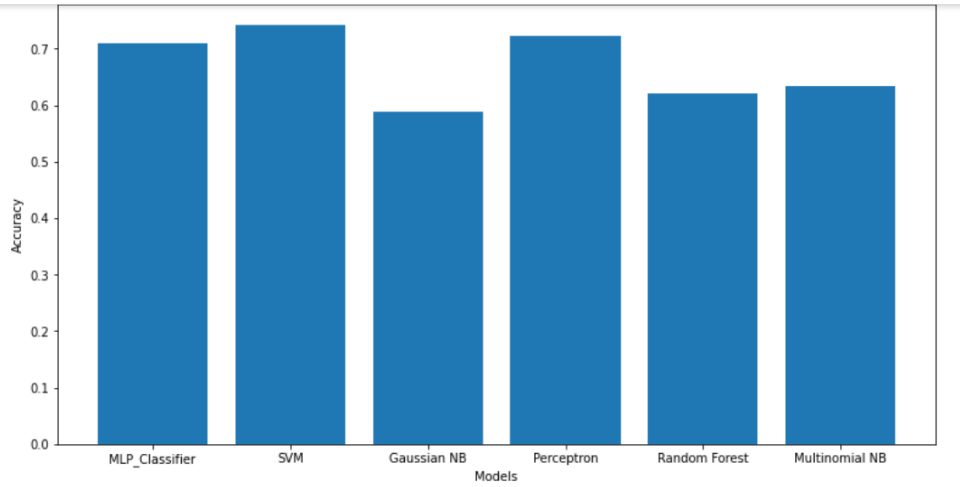
Metrics of MLP, SVM and Gaussian Naive Bayes classifier with and without PCA.



	Accuracy	Precision	Recall	F1_score
MLP_Classifier	0.709677	0.748837	0.922636	0.826701
SVM	0.741935	0.748373	0.988539	0.851852
Naive_bayes	0.589247	0.788321	0.618911	0.693419
MLP_with_PCA	0.690323	0.744630	0.893983	0.812500
SVM_with_PCA	0.750538	0.750538	1.000000	0.857494
Naive_bayes_with_PCA	0.752688	0.758850	0.982808	0.856429



Accuracy of different models.



VII. Inference

For this dataset, SVM gives the best accuracy among all the models. After applying PCA, SVM and Naive Bayes have comparable accuracies of around 75%.

VIII. Acknowledgment

We would like to express our gratitude to our teacher Dr. Richa Singh who gave us this wonderful opportunity to work on a project based on real life problem. This project helped us to implement many of the concept of ML, we learned in the course.

IX. Reference

- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html