Glass or No-Glass Classification

Harish Kumar *(B19CSE035)*, Harsh Kumar *(B19CSE036)*, Himanshu Raj *(B19EE038)*

Abstract

The problem of this project consists of Glass vs No-glass classification. For this project, the aim is to determine if a person is wearing glasses or not by using three different Classification Models. A Generative Adversarial Neural Network (GAN) created all of the people that you see in this competition. The GAN network creates these images using a 512 number latent vector. Both the latent vectors and the faces produced by those vectors are given.

I. Introduction

This Project Report contains the analysis and the comparison of the three different Machine Learning Classification models employed for this classification task. The three Classifications models used are **Multi-Layer Perceptron (MLP) Classifier**, **Gaussian Naive Bayes Classifier** and **Random Forest Classifier**.

Each model is evaluated by 10-fold Cross Validation using *StratifiedKFold* from Scikit Learn. Also the Accuracy scores, Confusion Matrices and Classification Reports are also used for evaluating Model Performance.

A. Data Visualization

1) About the Images or Latent Vectors: The Dataset consists of 512 number latent vectors created by Generative Adversarial Neural Network (GAN). The faces produced by some of these vectors is shown below in Figure 1. Some images have people with glasses and some without glasses as shown.

2) About the Datasets: There are two datasets which are given namely *train.csv* and *test.csv*. *train.csv* has 4500 rows and 514 columns including *id* and *glasses* (target) columns. The *test.csv* has 150 rows and 513 columns with an extra *id* column. It is unlabelled. One of the objectives is to predict the output of the test dataset using the three models used.

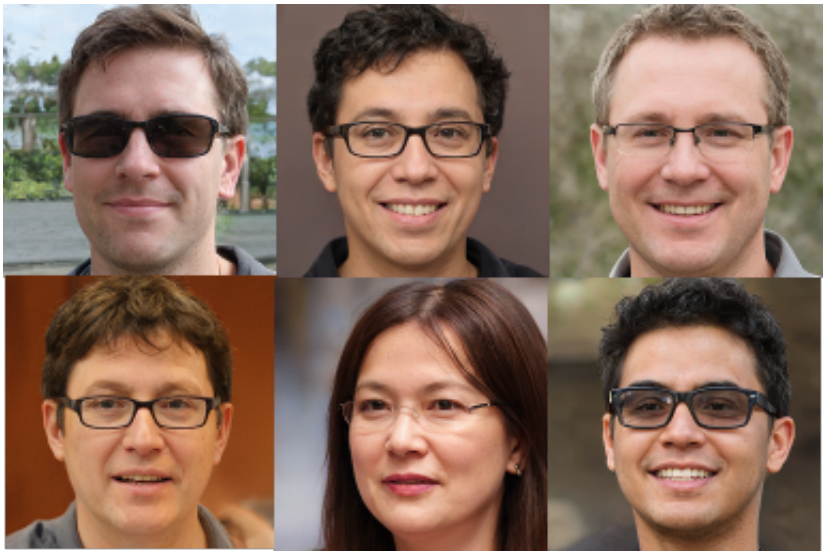

Fig. 1: Some of the images produced by GAN

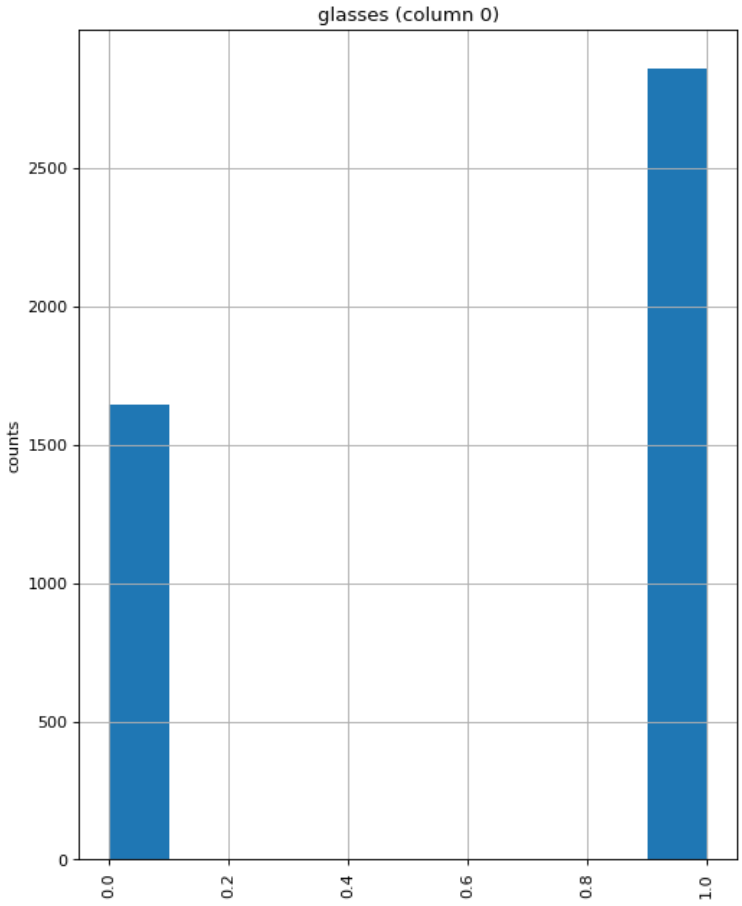

Fig. 2: Count Plot of the two target variables

B. Preprocessing

The Preprocessing involved first normalizing all the feature columns of the dataset using *Standard-Scaler()* from scikit learn .Then the whole dataset was separated into training and testing part using train_test_split with 70% training part and 30% testing part. The training and testing parts were also stored separately by applying Principal Component Analysis (PCA) by reducing the dimensions to first 100 EigenVectors (or Principal Components).

C. Models Trained

For Classification Task, we have used three models namely **Multi-Layer Perceptron (MLP) Classifier**, **Gaussian Naive Bayes Classifier** and **Random Forest Classifier**. For each model trained, first cross-validation score (10-fold) was calculated for judging the model hyperparameters. Then each model was trained on the training split of the dataset and tested accordingly using the various evaluation metrics like Confusion Matrix and Classification Reports.

A short explanation about the models used in this project is given below.

*1) **Multi-layer Perceptron (MLP)**:* A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as ”vanilla” neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

In the backward pass, using back propagation and the chain rule of calculus, partial derivatives of the error function w.r.t. the various weights and biases are back-propagated through the MLP. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters may be adjusted as they move the MLP one step closer to the error minimum. This can be done with any gradient-based optimisation algorithm such as stochastic gradient descent. The network keeps playing that game of tennis until the error can go no lower. This state is known as convergence.

*2) **Gaussian Naive Bayes Classifier**:* Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features (see Bayes classifier). They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of naive Bayes is called **Gaussian Naive Bayes**. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

*3) **Random Forest Classifier**:* Random forest classifiers fall under the broad umbrella of ensemble-based learning methods. They are simple to implement, fast in operation, and have proven to be extremely successful in a variety of domains. The key principle underlying the random forest approach comprises the construction of many "simple" decision trees in the training stage and the majority vote (mode) across them in the classification stage. Among other benefits, this voting strategy has the effect of correcting for the undesirable property of decision trees to overfit training data .

In the training stage, random forests apply the general technique known as bagging to individual trees in the ensemble. Bagging repeatedly selects a random sample with replacement from the training set and fits trees to these samples. Each tree is grown without any pruning. The number of trees in the ensemble is a free parameter which is readily learned automatically using the so-called out-of-bag error.

D. Dimensionality Reduction

1) Principal Component Analysis (PCA): The principal components of a collection of points in a real coordinate space are a sequence of p unit vectors, where the i-th vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors. Here, a best-fitting line is defined as one that minimizes the average squared distance from the points to the line. These directions constitute an orthonormal basis in which different individual dimensions of the data are linearly uncorrelated. Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

II. RESULTS

The Model Comparisons and their metric evaluation results for each model used from their classification reports is given in the tables below :

Multi-layer Perceptron (MLP)		
Evaluation Metrics	Original Dataset with 512 features	Dataset with Dimensionality Reduction (PCA)
Precision	0.996	0.673
Recall	0.993	0.484
Accuracy	0.997	0.558
F1-Score	0.998	0.634

Gaussian Naive Bayes Classifier		
Evaluation Metrics	Original Dataset with 512 features	Dataset with Dimensionality Reduction (PCA)
Precision	0.987	0.967
Recall	0.977	0.940
Accuracy	0.985	0.970
F1-Score	0.969	0.957

Random Forest Classifier		
Evaluation Metrics	Original Dataset with 512 features	Dataset with Dimensionality Reduction (PCA)
Precision	0.955	0.955
Recall	0.917	0.917
Accuracy	0.970	0.970
F1-Score	0.977	0.977

CONTRIBUTION OF EACH TEAM MEMBER

Each Team Member focused on a specific model used for classification for this task. MLP Classifer - **Harish Kumar (B19CSE035)**, Gaussian Naive Bayes Classifier - **Harsh Kumar (B19CSE036)**, Random Forest Classifier - **Himanshu Raj (B19EE038)**.

REFERENCES

Kaggle Link (with Datasets)
Wikipedia - Principal Component Analysis (PCA)
Wikipedia - Random Forest Classifier
Wikipedia - Naive Bayes Classifier