Machine learning project report
Glass vs No Glass classifier

Ronak Singhvi, Shantanu Bakshi, and Man Rajkotiya

Abstract

Here in this report we have presented and modelled three approaches for building a classifier that classifies an image of a person as wearing a glass or not by the use of three methods namely , support vector machine(SVM) , K-Nearest-Neighbors(KNN) , MultiLayer-Perceptron(Nnet) and report the performance of the respective models

I. Introduction

The structure of this report will be as follows

1) Data pre-processing and information/insights
2) Support Vector Machine
   a) Motivation, working and parameters
   b) Performance evaluation
3) KNN
   a) Motivation, working and parameters
   b) Performance evaluation
4) Neural Network
   a) Motivation, working and parameters
   b) Performance evaluation
5) Performance on real(GAN) images

A. Data pre-processing and information/insights

1) The Numerical data consists of data of images split into 1*512 vectors
2) the dimensions are (4500,514)
3) the distribution of the data is as follows

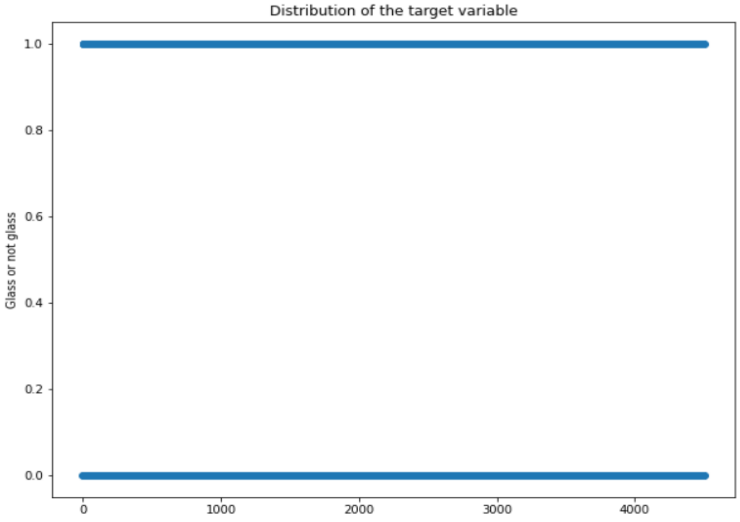

Fig. 1. the distribution of the target variable

4) For preprocessing we have applied Standardscaler and PCA(0.95) explained variance

B. SVM Classifier

1) Support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.
2) Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
3) In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
   SVMs can be used to solve various real-world problems:
   a) SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.[7] Some methods for shallow semantic parsing are based on support vector machines.[8]
   b) Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.[9][10]
   c) Classification of satellite data like SAR data using supervised SVM.[11]
   d) Hand-written characters can be recognized using SVM.[12][13]
   e) The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90 percent of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models.[14][15]
      Support-vector machine weights have also been used to interpret SVM models in the past.[16] Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

   The SVM implementation used over here is a part of the sklearn library with the hyperparameters as follows

   a)
```
{   'C': 1.0,
    'break_ties': False,
    'cache_size': 200,
    'class_weight': None,
    'coef0': 0.0,
    'decision_function_shape': 'ovr',
    'degree': 3,
    'gamma': 'scale',
    'kernel': 'rbf',
    'max_iter': -1,
    'probability': False,
    'random_state': 0,
    'shrinking': True,
    'tol': 0.001,
    'verbose': False}
```

   b) here is a list of evaluation metrics for the classifier
      i) 5-fold CV score : [1,1,1,1,1]
      ii) confusion matrix [[323, 0], [0, 577]]

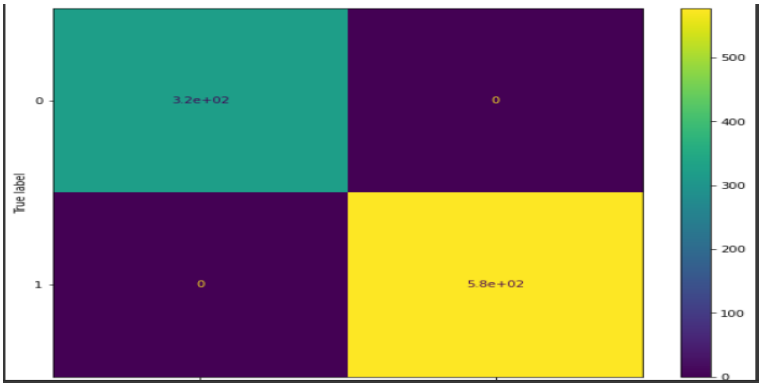

Fig. 2.

iii) roc-curve

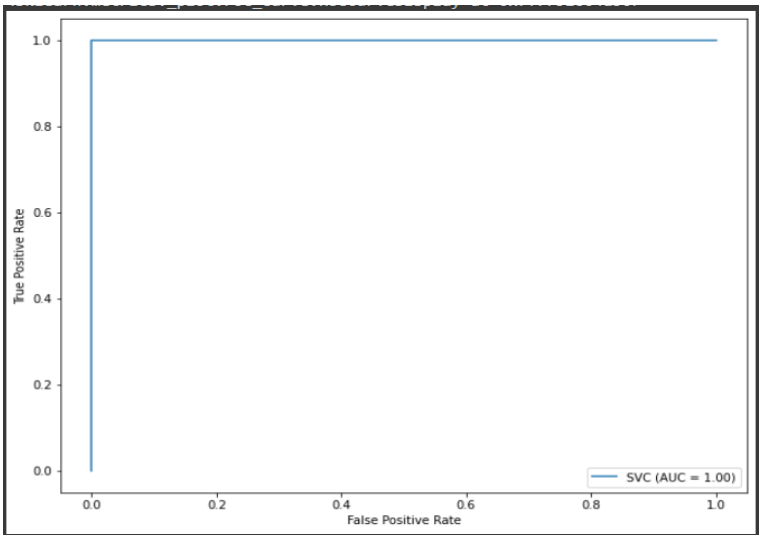

Fig. 3. Caption

iv) (array([1., 1.]), array([1., 1.]), array([1., 1.]), array([323, 577]))

v) accuracy-score : 1.0

C. KNN Classifier

1) the k-nearest neighbors algorithm (k-NN) is a non-parametric classification method. It is used for classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression:
2) In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
3) In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.
4) The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.
5) In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.
6) A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance).

a)
```
{   'algorithm': 'auto',
    'leaf_size': 30,
```

```
'metric': 'minkowski',
'metric_params': None,
'n_jobs': None,
'n_neighbors': 5,
'p': 2,
'weights': 'uniform'}
```

b) here is a list of evaluation metrics for the classifier

i) 5-fold CV score : array([1., 1., 0.99848024, 1., 1.])

ii) confusion matrix: array([[317, 0], [6, 577]])

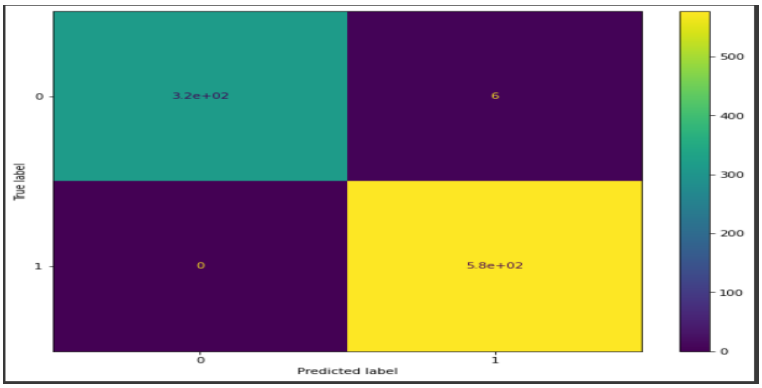

Fig. 4.

iii) roc-curve

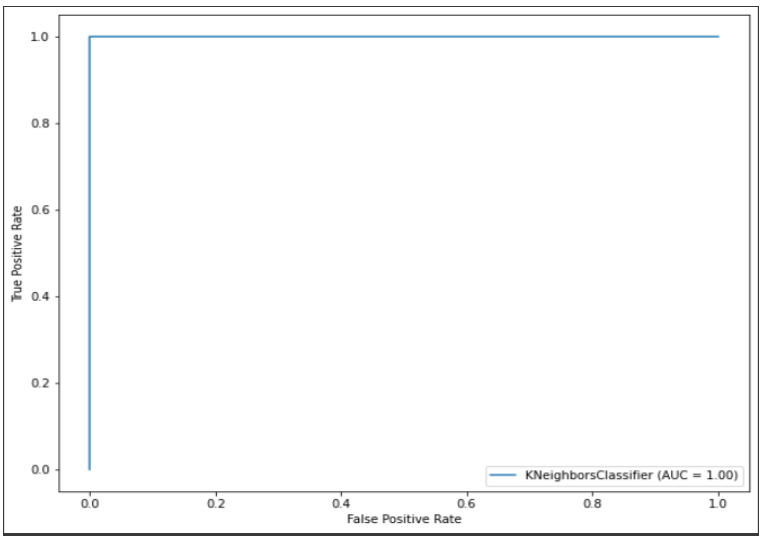

Fig. 5. Caption

iv)

```
(array([1.        , 0.9897084]),
 array([0.98142415, 1.        ]),
 array([0.990625  , 0.99482759]),
 array([323, 577]))
```

v) accuracy-score : 0.9933333333333333

D. Neural Network

1) A perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line. Input is typically a feature vector x multiplied by weights w and added to a bias b: y = w * x + b. A

perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights (and sometimes passing the output through a nonlinear activation function).

2) A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function.
3) In the forward pass, the signal flow moves from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels.In the backward pass, using backpropagation 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.
4) The MLP implementation used over here is a part of the sklearn library with the hyperparameters as follows

a)
```
{'activation': 'relu',
 'alpha': 0.0001,
 'batchsize': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_fun': 15000,
 'max_iter': 300,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 1,
 'shuffle': True,
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

b) here is a list of evaluation metrics for the classifier
   i) 5-fold CV score : [1,1,1,1,1]
   ii) confusion matrix :array([[321, 0],[2, 577]])

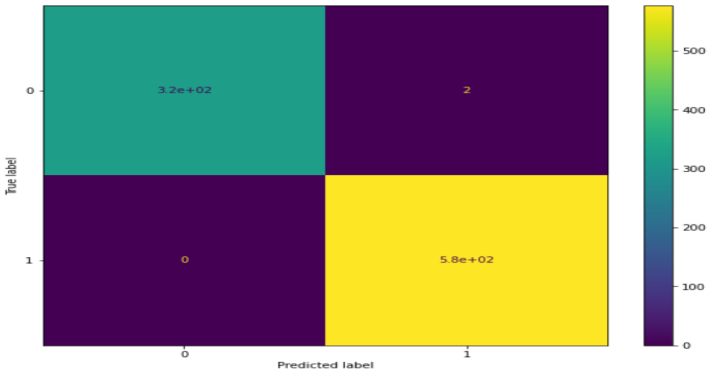

Fig. 6.

   iii) roc-curve

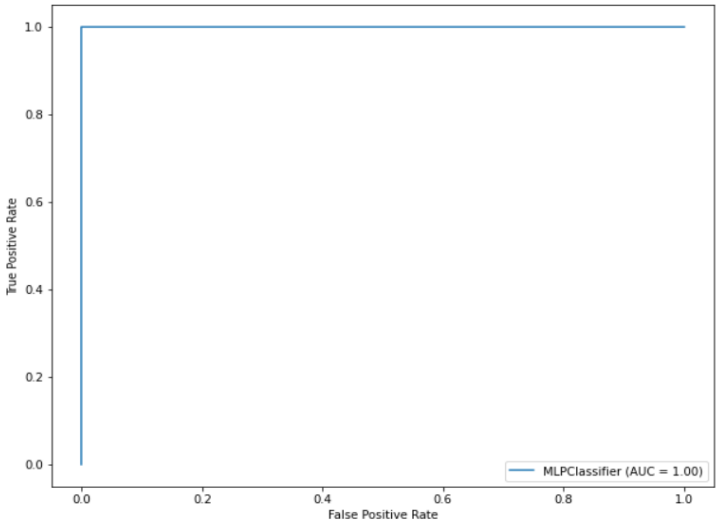

Fig. 7. Caption

iv)
```
array([0.99380805, 1.0]),
array([0.99689441, 0.9982699 ]),
array([323, 577]))
```

v) accuracy-score : 0.9977777777777778

E. Comparison

Based on all evaluation metrics for all classifiers we have chosen accuracy score as a metric to determine which classifier is the most useful

classifier	accuracy
Neural Network	0.9977777777777778
SVM(radial)	1.0
KNN	0.9933333333333333

Clearly the support Vector classifier performs the best . Performance of an SVM is substantially higher compared to NN. For NN, prediction requires successive multiplication of an input vector by two 2D matrices (the weight matrices). For SVM, classification involves determining on which side of the decision boundary a given point lies, in other words a cosine product. Training an SVM, by contrast, means an explicit determination of the decision boundaries directly from the training data. This is of course required as the predicate step to the optimization problem required to build an SVM model: minimizing the aggregate distance between the maximum-margin hyperplane and the support vectors.

In practice though it is harder to configure the algorithm to train an SVM. The reason is due to the large (compared to NN) number of parameters required for configuration:

choice of kernel

selection of kernel parameters

selection of the value of the margin parameter

Fig. 8. NO-Glass

Fig. 9. Glass

The SVM is now capable of classifying such type of images having glasses or no glasses

F. Contribution by each member

1) Ronak Singhvi : SVM Classifier and Report
2) Shantanu Bakshi : MLP classifier and Report
3) Mann Rajkotiya : KNN classifier and Report