Minor Project - Face Mask Detection

Pattern Recognition and Machine Learning

CSL2050

Pawandeep Suryavanshi (B19EE063), Raunak Gandhi (B19CSE117)

Pawandeep Suryavanshi (Department of Electrical Engineering, Indian Institute of Technology, Jodhpur).
Raunak Gandhi (Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur),

***Abstract* - This document is an analytic report for the minor course project for the Pattern Recognition and Machine Learning course.**

I. Introduction

This is a supervised binary classification project involving the use of Machine Learning Techniques to classify images of people wearing and not wearing masks. For this project, we have used the self-built-masked-face-recognition-dataset.zip data set.

We have 2 classes with 4300 of total samples (2150 samples per class) and among them, 50% samples are used for testing and comparison between different models.

II. Preprocessing

The dataset mentioned above is in the form of raw images with irregular sizes in separate folders with no labels. So, first we need to collect all the images from each folder and resize them into a regular and uniform size so that it can be trained properly.

If the sizes of all the samples do not match then a sample's feature space's size will vary, which will hinder the process of classification. And the size of image should be small otherwise we would end up with a large feature space which would be difficult to handle. Thus, all the images were resized to 32 x 32.

There were around 90,000 images without masks and 2150 images with masks. So, 2150 images at random were selected from the 90,000 images without a mask.

Then all the images were flattened to convert them into a 1 dimensional vector and then their labels were assigned.

Then the data was converted into a pickle file to upload.

* Code for pre processing is provided in the submission.

III. Data Uploading

The data file created in the above step was uploaded to google drive to make it easily accessible.

Then, google drive was mounted in the google colab notebook file. Required dependencies were imported.

The data was then imported in the colab file. The pickle file was then unpickled in the notebook.

Pandas dataframe was created from the extracted data for ease of use.

Each sample contains a feature space of size 3072 and a label, 0 for person without mask and 1 for person wearing mask.

IV. Feature Selection

For feature selection, SelectKBest model from the sklearn.feature_selection library.

SelectKBest method selects the k best features from the entire feature variable set.

Thus, SelectKBest removes all but k highest scoring features. Thus, the best 1000 features among all other features were selected for creating a new data set.

The SelectKBest model was fitted over the training data . Then, the training, testing feature variables were transformed using the trained model. And then we calculated the score for all the three classifiers and we found that the accuracy without feature selection was more , so we rejected the method of feature selection.{ SVM : 0.9488372093023256 , KNN : 0.8837209302325582 , MLP : 0.9204651162790698}

V. Classifier Models

Support Vector Machine , K Nearest Neighbours and MultiLayer Perceptron Classifier were used for training over the training data and testing over the testing data and then hyperparameter tuning was done with the help of grid search giving us the best parameters for the model.

Performance of these models are compared below.

A. Support Vector Classifier

Support Vector Machine is one of the supervised machine learning algorithms.It creates a decision boundary which segregates the space into classes.It choses some extreme points that help in creating the decision boundary and those points are called support vectors.

SVM works well with higher features space and this dataset contains a feature space of size 3072, and it also works better when number of features are more than the number of samples, here we have 2150 samples and 3072 features in the training set. That's Why this classifier was chosen as a candidate classifier to classify our data.

The SVC model from sklearn.svm library was used for training and testing.

Then we did a grid search for SVC with the below parameters and then we found the best parameters and trained our training data on the model having best parameters and then tested its accuracy which can be seen in the below table.

The following table summarises the results for our SVC model :

	SVC Model
Accuracy on the default SVC model	0.9618604651162791
Parameters chosen for Hyperparameter tuning	{'kernel':['linear', 'poly', 'rbf'], 'C': [0.01, 0.1, 1, 3, 5, 10, 15, 20,30],}
Best Parameters after Hyperparameter tuning	{'C': 10, 'kernel': 'rbf'}
Accuracy On Best Parameters	0.967906976744186

B. K Nearest Neighbors Classifier

K Nearest Neighbors Classifier is also one of the supervised machine learning algorithms. In this algorithm case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

KNN is simple to train and computation time is less that's why this classification technique was chosen.

The KNeighborsClassifier model from sklearn.neighbors library was used for training and testing.

Then we did a grid search for k-NN with the below parameters and then we found the best parameters and trained our training data on the model having best parameters and then tested its accuracy which can be seen in the below table.

The following table summarises the results for our k-NN model :

	k-NN Model
Accuracy on the default k-NN model	0.8916279069767442
Parameters chosen for Hyperparameter tuning	{'n_neighbors':[3, 5, 7, 10, 15 , 20], 'leaf_size':[20,30,45],}
Best Parameters after Hyperparameter tuning	{'leaf_size': 20, 'n_neighbors': 3} {'n_neighbors': 3, 'leaf_size': 20,}
Accuracy On Best Parameters	0.9004651162790698

C. Multi Layer Perceptron

Multi Layer Perceptron Classifier is a class of Neural network which includes an input layer, at least one hidden layer and an output layer. It uses gradient descent to find optimum values of weights and biases that are initialised randomly.

Neural Networks tend to do well with image classification because of their ability to extract features using deep layers of perceptrons that's why it was chosen as a candidate model to classify.

The MLPClassifier model from sklearn.neural_network library was used for training and testing.

Then we did a grid search for MLP with the below parameters and then we found the best parameters and trained our training data on the model having best parameters and then tested its accuracy which can be seen in the below table.

Note : Here we converted the training data in range of (0,1) by dividing 255 to all the values so that no feature becomes dominant over other features.

The following table summarises the results for our MLP model :

	MLP Model
Accuracy on the default MLP model	0.9604651162790697
Parameters chosen for Hyperparameter tuning	{ 'hidden_layer_sizes' : [(512,128,64),(512,256,128),(512,512,256)], 'learning_rate': ['invscaling','adaptive'], 'learning_rate_init': [0.05], }
Best Parameters after Hyperparameter tuning	{ 'hidden_layer_sizes' : (512,128,64), 'learning_rate': 'invscaling', 'learning_rate_init': 0.05, }
Accuracy On Best Parameters	0.8865116279069768

* This suggests that the default parameters were best for MLP.

VI. Best Model

So the best model we got is a SVM model with Parameters C=10 and kernel = ‘rbf’ whose testing accuracy is 0.967906976744186 which is the largest score achieved by us after training a different model with different parameters.

Radial Basis Function (RBF) projects data into infinite dimensions to get the most suitable decision boundary hyperplane.

The feature space's size was too big to visualize a decision boundary so it was not plotted.

We got 352 and 331 support vectors for classes 0 and 1.

There were 3072 features in this dataset so a lot of dimensions were required for SVM to properly draw a decision hyperplane boundary, this may be the reason why this model worked better than other models.

Other evaluation metrics obtained from this model,

F1 Score	Precision	Recall	Accuracy
0.9665211062590975	0.97265625	0.9604628736740598	0.967906976744186

VII. Testing on Real World Examples and Visualization

10 real world images were obtained that were not from the same distribution as the input data and the best model was used to predict whether the person in the image is wearing a mask or not.

Prediction can be cross checked visually, images were plotted using matplotlib library.

- Predictions Model classified 9 out of 10 images correctly.

It confuses if the person has strong facial features (jawline) because it also creates a pattern similar to the shape of the mask.

So, the model works pretty well on data coming from distributions different from the training and testing dataset.

VIII. Conclusion

The data was analysed, uploaded, and preprocessed. Models were trained for different classifiers.

Then hyper parameter tuning was done for all the three models.

The best model we got after that was the SVM model with Parameters C=10 and kernel = ‘rbf’ whose testing accuracy is 0.967906976744186.

RBF kernel works better on this data than linear kernel, this means that this data is not linearly separable.

Then with this model we tested some real world examples and visualized the data that helped us to get a better understanding about how Machine Learning is useful for real world applications.

IX. Contribution

The complete machine learning pipeline was thoroughly discussed by both the team members. Every decision was made unanimously.

Raunak Gandhi (B19CSE117) - Worked on normalised data, feature selection, wrote code for SVM and other classifiers, collected real world examples.

Pawandeep Suryavanshi (B19EE063) - Preprocessed the raw data, wrote code for hyperparameter tuning and visualization of the data.

The work was divided only for writing the code. Discussion about the data and planning was done by all team members at length.