1. Introduction

**Student Number: 215905 & Kaggle Team Name: HS462**

The problem statement is classification of Memorable versus non. Memorable pictures. Each data instance is represented as a 4608-dimensional feature vector. This vector is a concatenation of 4096 dimensional deep Convolutional Neural Networks (CNNs) features extracted from the fc7 activation layer of CaffeNet and 512-dimensional GIST features.

2. Data Loading

The data to be loaded is in csv files. The problem has four csv files available. The training.csv is the main data file. The additional\_training\_data.csv is additional data with some feature values missing. We load both. Also, we make use of the annotation confidence data by reading annotation\_confidence.csv.

3. Data preprocessing

After loading all the data from csv files to pandas’ objects, we apply data preprocessing. We combine both the training data and additional training data. Since some values are missing in the additional data, we fill the blanks with the median of that entire column. We also read the annotation confidence data which can be feed as sample weights to the classifier.

We use StandardScalar method of the sklearn library to Standardize the features by removing the mean and scaling to unit variance.

4. Data visualization and Model selection

Each data row has 4096 CNN features and 512 Gist features. So, to visualize the data, we first apply dimensionality reduction where we reduce the features to two columns. We use Principal Component Analysis (PCA) to reduce the dimension of the dataset. After dimensionality reduction, the scatter plot of the data can be seen.

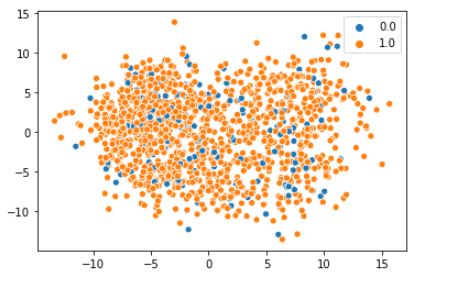


Figure 1: Scatter Plot of Reduced Data

From the above figure, we can infer that the data points are not linearly separable. So, linear models cannot be used for the classification problem. Also, we know that the CNN features and GIST features are independent of each other. Therefore, Probability theory works best for the classification. Naïve Bayes classifier is the selected classifier.

## 5. Naïve Bayes Classifier.

A Naïve Bayes classifier corresponds to a Bayesian network, as in the Eq (1). Here, a single class variable C and m attribute variables Xi. Let c denote a class label and xi denote a value of an attribute Xi. A naïve Bayes induces a distribution:

…Eq (1)

Where we have a class prior Pr(C) and conditional distributions Pr(Xi|C). We can estimate these parameters from data, using maximum likelihood or MAP estimation. Once we have learned a naïve Bayes classifier from data, we can label new instances by selecting the class label c\* that has maximum posterior probability given observation sx1, …., sxm.

…Eq (2)

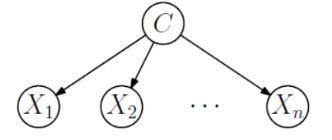
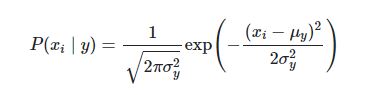


Figure 2: Naive Bayes Classifier

6. Gaussian Naïve Bayes

We are using Guassian naïve bayes out of all available naïve bayes algorithm like Complement Naïve Bayes, Multinomial Naïve Bayes, and Bernouli Naïve Bayes. because this algorithm is well suited for continuous data.

GuassianNB class from sklearn implements the Gaussian Naïve Bayes algorithm for classification. The features likelihood is assumed to be Gaussian:



7. Building and Training the model

**Student Number: 215905 & Kaggle Team Name: HS462**

We use totally 50 GuassianNB models. In each of 50 iterations, first we take a sample data of size 1500 out of which 1250 are positive predictions and 250 are negative predictions. We build a GaussianNB model with prior probability of 0.6152 for zeros and 0.3848 for ones. The predicted values are added to last predicted values.

At the end of all the iterations we will have for each data sample, the number of chances it being classified as 1 out of 50 since we added the predictions from each model. Now we will sort the added predictions based on the arguments using argsort method of the numpy library. After sorting the values, the higher weights are chosen to be predicted as 1. Since we know the testing proportions, we can now classify these weighted predictions as 1 and 0 to match the testing proportions.

Using this method, we can reduce the error rate that can be caused by the single model since we are making use of multiple models and each model uses a sample of data from the entire dataset.

The hyper parameters considered when building the model are shown in the below table:

|  |  |  |
| --- | --- | --- |
| Sl No | Parameter | Value |
| 1 | Prior Probability | [0.6152, 0.3848] |
| 2 | Sample weight | 1 or 0.66 based on annotation confidence value |

8. Results and Future Work

With the validation set taken from the same sample and averaging it to over 50 model’s accuracy score, we got an accuracy score of 79.245 in the best run.

The model can further be improved by playing with the value of number of models. With the increase in the number of the models’ number, the accuracy seems to increase at the end.

# References:

1. P. Kaviani, S. Dhotre, ‘Short Survey on Naïve Bayes Algorithm’, 2017. [Online]. Available: <https://www.researchgate.net/publication/323946641_Short_Survey_on_Naive_Bayes_Algorithm>. [Accessed: 01-June-2020].