ASSIGNMENT 2

Group Details:

Group - 002

18CS10069 - Siba Smarak Panigrahi

18CS30051 - Debajyoti Dasgupta

November 1, 2020

A. Procedure:

1. We have used the standard Naive Bayes algorithm for designing the classifier in this assignment. The data used for training was provided in .csv format. We have shuffled the data. We considered the split of 80:20 for the Train and Test set. We reported the results with respect to five-fold cross-validation. The data had the following attributes: case_id, Hospital_code, Hospital_type_code, City_Code_Hospital, Hospital_region_code, Available Extra Rooms in Hospital, Department, Ward_Type, Ward_Facility_Code, Bed Grade, patientid, City_Code_Patient, Type of Admission, Severity of Illness, Visitors with Patient, Age, Admission_Deposit.

   Target Variable: Stay

   (**Note**: All the Train Accuracy labeled values (for instance in five-fold cross-validation) are Validation set accuracy)

2. **Major functions:**

   `naive_bayes`: This is the main function for Naive-Bayes Algorithm. It calculates the posterior probability for each input sample with the help of already stored class-wise mean and standard deviation. The posterior probability requires prior probability and likelihood. The denominator is the same for each input sample and thus can be ignored. The predicted class is the one for which the posterior probability is maximum. The required formulae used are:

$$P(C_i|x) = \frac{P(C_i).P(x|C_i)}{P(x)}$$

Likelihood: $P(x|C_i)$

Posterior Probability: $P(C_i|x)$

Prior Probability: $P(x|C_i)$

For a Gaussian, with mean μ and standard deviation σ, we consider the likelihood to be given by:

$$P(x|C_i) = \frac{1}{\sqrt{2\pi}.\ \sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

`evaluate_algorithm:` This function is the evaluation function which collects together all the steps that need to be performed in the naive bayes classifier. It performs five-fold cross-validation and also calls naive-bayes function mentioned above.

3. Some important Helper functions:

`train_test_split:` This function splits the data into two sets train and test

`load_csv:` This function reads a .csv file and stores the data in a row-format

`cross_validation_split:` splits the input data and returns the indices of the validation set for each iteration of cross-validation

`accuracy_metric:` evaluates the accuracy of predicted values with respect to the actual values. Accuracy is defined as the number of correct predictions divided by total length of predictions

`summarize_dataset:` it returns the mean, standard deviation, number of entries, threshold value for outlier for each column of the dataset

`calculate_probability:` it returns the probability density of the data point on a Gaussian with given input mean and standard deviation
handle_missing_data: this function imputes the missing value(s) of a feature with the most_frequent value of the feature

`encoded_dataset`: this function encodes the categorical columns in the dataset with LabelEncoder from scikit-learn

`sequential_backward_selection:` this function uses sequential backward selection method to remove features

`remove_outliers:` this function removes those samples from dataset which contain the number of outlier features more than the parameter 'outlier'

prepare: this functions preprocesses the dataset and prepares for splitting and training purposes. It imputes the missing data, encodes the categorical values, converts each entry into float and normalizes each feature.

`separate_by_class:` this function is used to split the input dataset as per the classes defined by the target class. The target class will be the last column of the dataset.

`calculate_class_probabilities:` Helper function to calculate the probabilities of each class given the current data value. This function helps in calculating the a posteriori for each class.

`predict:` make predictions for the given vector of data (row) and return the class label that should be assigned to this data element

4. Principal Components Analysis (PCA)

a. **Principal component:** component along the direction **w** such that its variance is maximum among all possible projections. Additionally, the subsequent principal components are perpendicular to the prior principal components.

b. **X** is the random variable whose instance is **x** with mean **m** and covariance matrix **Σ**. Then the first component is evaluated by the following optimization problem (where l is Lagrangian coefficient):

$$w_1 = argmax_w\{w^T\Sigma w - l.(w^T w - 1)\}$$

$$\Sigma.w_1 = l.w_1$$

c. The first component is evaluated by the following optimization problem (where l_1 and l_2 are Lagrangian coefficients):

$$w_2 = argmax_w\{w^T\Sigma w - l_1.(w^T w - 1) - l_2.({w_2}^T w_1 - 0)\}$$

$$\Sigma.w_2 = l_1.w_2$$

5. Five-fold cross-validation

There are a total of five iterations. Each iteration considers one-fifth of the training set as validation set and rest as the train data. The model is trained on the training set and tested on the validation set. This approach helps to provide a generalization accuracy better, i.e., taking the average of the five accuracies above.

6. Sequential Backward Selection Method

Steps:

i. Train the Naive-Bayes classifier with all the features and store the accuracy in current_accuracy.
ii. Find the feature, dropping which increases the current_accuracy the most.
iii. Drop that feature. Compute the new accuracy with the dropped feature and store it in current_accuracy.
iv. Repeat step (ii) and (iii) until there is no more increase in accuracy.

There is another approach for feature selection, called **Sequential Forward Selection Method**. It starts with zero features and adds the feature which leads to a maximum increase of accuracy.
Both the methods mentioned above are **local search method** and there complexity is $O(d^2)$. But the **Sequential Forward Selection Method** is less computationally expensive than the **Sequential Backward Selection Method**. This is because Forward selection starts with a null set of features and hence training is less expensive.

B. Results

(**Note:** Instead of using the entire dataset, **--frac <fracVal>** in the command-line argument considers **fracVal * total_data** as the training set, **fracVal** should be a value between 0 and 1. Here the results are reported below for fracVal = 1.)

1. Handling missing values, encoding the categorical variables and results of five-fold cross-validation:

a. Handling the missing values:

i. The missing values are handled by replacing the **most_frequent value** for categorical attributes and **mean value** for the continuous attributes.
ii. In our dataset, only categorical attributes (i.e., Bed Grade and City_Code_Patient) had missing value, we used **most_frequent value** to impute a particular feature.

b. Encoding the categorical variables:

i. **LabelEncoder** class of **scikit-learn** was used for encoding the categorical variables. It was done for the following reasons:
   1. It does not increase the size of the dataset. If we use One-hot encoding, then the resultant dataset consumes more memory and also becomes sparse, i.e., contains a very large number of zeros.
   2. Label encoding later helps in fitting Gaussian to categorical values (which improves the accuracy to a large extent), while fitting Gaussian on one-hot encoded data doesn't seem a better approach.

c. Results of five-fold cross-validation:

i. Handling of missing values and encoding of categorical variables are done prior to splitting the data into train and test. This is to maintain uniformity in replacing the missing values and encoding the feature values.

ii. The accuracies obtained on the **validation set** on different iterations of five-fold cross-validation are (in %): 36.19, 35.79, 35.87, 35.90, and 36.21.

iii. The average **validation set accuracy** and **test accuracy** reported (with the model that gave the best validation accuracy above) is **35.99%** and **36.11%** respectively.

iv. The results are comparable to the results of **GaussianNB** class from **sklearn.naive_bayes**. After normalization we have fitted Gaussian to each of the features before calling `evaluate_algorithm`.

2. Application of PCA and results of five-fold cross-validation:

a. Application of PCA:

i. PCA is applied to the dataset and the required graphs were plotted. We observed that at least 95% of the variance is captured with 13 features. The required plot is attached below, which justifies our statement:

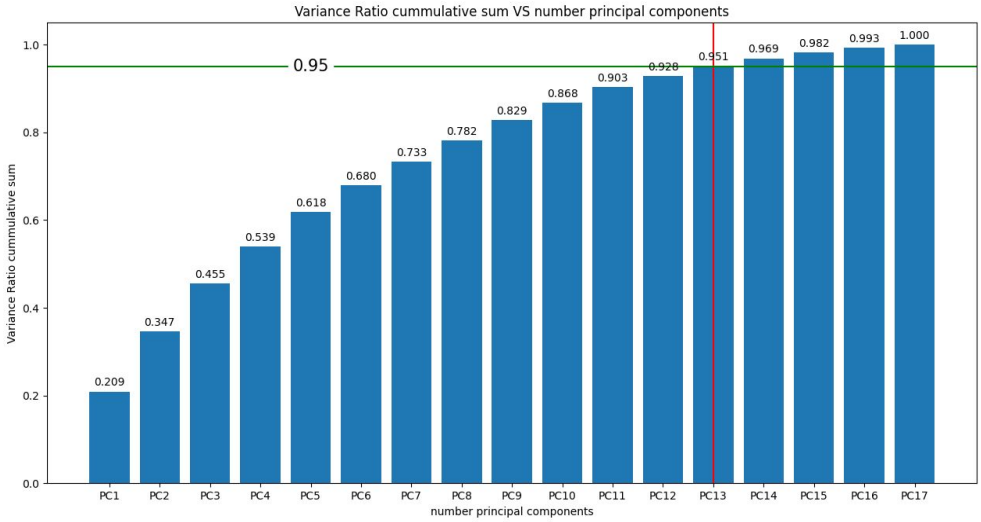

FIG 1: Variance ratio cumulative sum vs the number of principal components

ii. The plot of variance captured by each of the feature generated by PCA is shown below:

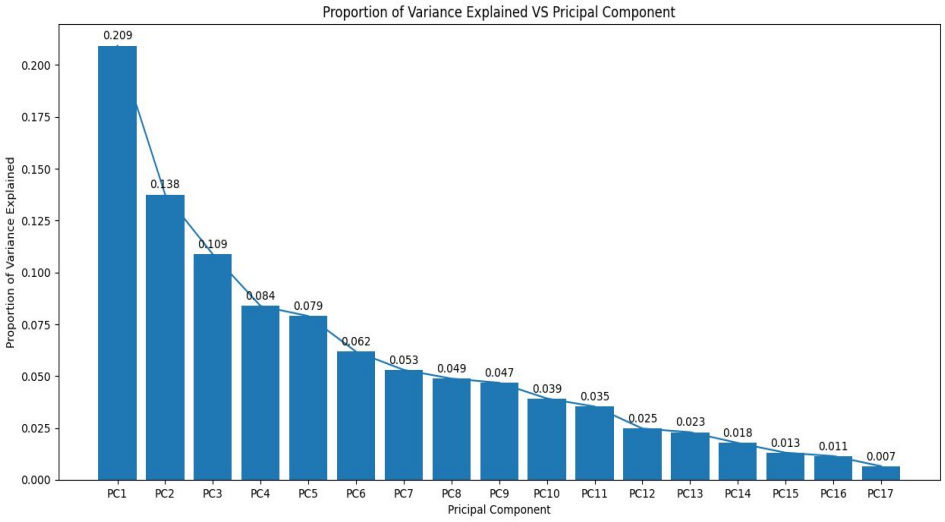

FIG 2: Proportion of variance explained vs Principal Component

b. Results of five-fold cross-validation:

i. The accuracies obtained on the **validation set** on different iterations of five-fold cross-validation are (in %): 35.76, 38.70, 33.20, 31.43, and 36.74.

ii. The average **validation set accuracy** and **test accuracy** reported (with the model that gave the best validation accuracy above) is **35.17%** and **29.67%** respectively.

3. Outlier Removal, Sequential Backward Selection Method, and results of five-fold cross-validation:

a. Outlier Removal:

i. A feature value is considered as an outlier if its value is greater than **mean + 3 * standard deviation**.

ii. The threshold is a value, such that, if a sample has more than or equal to threshold outliers, then it can be dropped from the dataset.

iii. We have a total of 18 features. We checked that a sample could have a maximum of three feature values which are outliers, thus we have three as the default threshold for dropping a sample. It can be changed with the **--outlier <threshold>** command-line argument, where the **threshold** is either 1, 2, or 3.

b. Sequential Backward Selection Method:

i. Our implementation of the sequential backward selection method on the entire dataset drops the following features in order and the accuracy after dropping it are mentioned in the table below:

(The initial accuracy after dropping samples having outliers greater than or equal to the threshold (here 3) with no feature removal: **35.99%**)

The removed features are City_Code_Patient, Department, Hospital_region_code, Hospital_type_code, Hospital_type_code, Severity of Illness, case_id, Age, and Ward_Facility_Code.

Table 1: Feature removal order and improved accuracy

Order	Feature Removed	Improved accuracy (in %)
1	City_Code_Patient	36.83
2	Department	37.33
3	Hospital_region_code	37.84
4	Hospital_type_code	38.01
5	Severity of Illness	38.47
6	case_id	38.80
7	Age	39.02
8	Ward_Facility_Code	39.23

No more features were removed after feature 8 i.e., Ward_Facility_Code.

c. Results of five-fold cross-validation:

i. The accuracies obtained on the **validation set** on different iterations of five-fold cross-validation are (in %): 38.12, 43.57, 37.47, 39.16, and 39.58.

ii. The average **validation set accuracy** and **test accuracy** reported (with the model that gave the best validation accuracy above) is **39.58%** and **30.30%** respectively.