Assignment 2
Bayesian Learning

Group-8 -> Aryaman Jain (18CS30007) | Archit Agarwal(18CS10006)

Objective:

Given Dataset:

Hospital _code	City_Code _Hospital	Department	Ward_ Type	Bed Grade	patientid	Type of Admission	Severity of Illness	Visitors with Patient	Age	Admission_ Deposit	Stay
8	3	radiotherapy	R	2	31397	Emergency	Extreme	2	51-60	4911	0-10
2	5	radiotherapy	S	2	31397	Trauma	Extreme	2	51-60	5954	41-50
10	1	anesthesia	S	2	31397	Trauma	Extreme	2	51-60	4745	31-40

(Table shown above contains some features only, whereas total features are 17)

- It contains the record of various patients and the hospitals they were admitted to, when they suffered from a particular disease, under different conditions.

- There are 17 features provided in total including target feature. They are:
  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 and Stay.
  Target feature is Stay.

- We have to predict the label of stay(in days) of a patient in a hospital using Naive Bayes Classifier.
  The different labels associated with stay are: 0-10, 11-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90, 91-100, more than 100.

Implementation:

We read the data set(.csv file) using pandas library.

Part 1

- We fill the missing categorical values in the feature with the most frequent value(MODE) of the corresponding feature.

- Then we splitted the data into 80:20 where 80 parts are for training and the remaining 20 parts are for testing.

- Then we encode the categorical values using *label encoder*(inbuilt). If there are n classes then the label encoder will label the values from 0 to n-1.

Stay (Given)	Stay (Encoded)
0-10	0
11-20	1
..	..
more than 100	10

- Then we feed the above training dataset to the *fit_naive_bayes* function which stores the frequency of each categorical for all different classes of the target variable.

- After the above step we find the frequency of each class of the target variable.

- To predict the testing data we used naive bayes where we find the predicted class is the class which gives the maximum a posteriori probability

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

  We took log of the above formula to reduce the time taken by the algorithm to give predictions.

- We also used the Laplacian Corrector to make predictions more accurate.

- To get 5-fold cross validation we passed the training data set with k = 5 as our arguments in *KFoldCrossValidation* function. This partitions training dataset into 5 different datasets, 4 parts of which are used for training and 1 part is used for validation. This process is repeated 5 times with different validation data each time.

Part 2

- We took the encoded data from the first part and applied an inbuilt *PCA* function to obtain eigen-vectors with decreasing eigen-values.
- Then we take eigen-vectors till we get cumulative eigenvalues >= 0.95.
- We used matplotlib library to plot the graphs between eigen-values and number of features.
- We took the eigen-vectors obtained from the above steps and computed the new dataset with the reduced features by doing matrix multiplication of the encoded dataset and taking transpose of the eigenvector matrix.
- After getting the new dataset we applied the method mentioned in part 1 to get the final test accuracy.
- Finally using *KFoldCrossValidation* function, 5-fold cross validation is computed by passing training dataset and k = 5 as arguments.

Part 3

- We find the mean and standard deviation for all the features in the encoded dataset from part 1.
- Then we find the number of outliers in each row and delete those rows which have the maximum number of outliers in the dataset.
- We split training dataset into 80:20 parts where 80 parts are the new training dataset and other 20 parts is the validation dataset
- We then feed our training and validation dataset into the *backward_selector* function, it recursively eliminates those features which are responsible for decrease in validation accuracy.
- After getting the new dataset we applied the method mentioned in part 1 to get the final test accuracy.
- Finally using *KFoldCrossValidation* function, 5-fold cross validation is computed by passing training dataset and k = 5 as arguments.

Results:

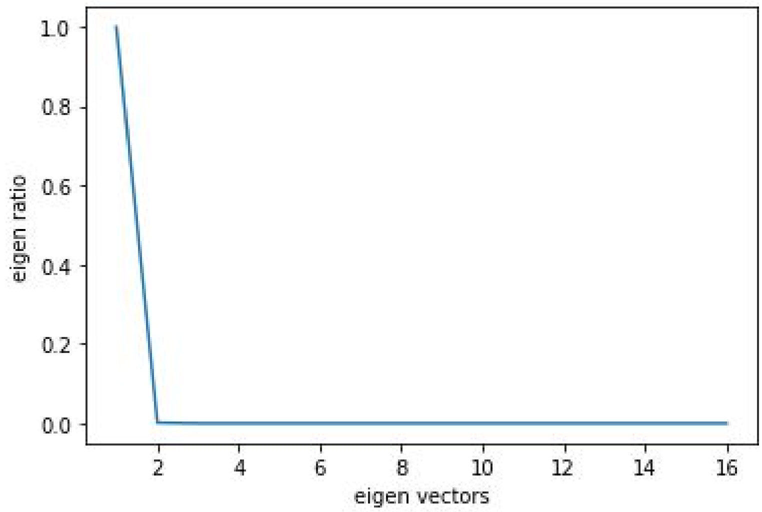
Part 1:

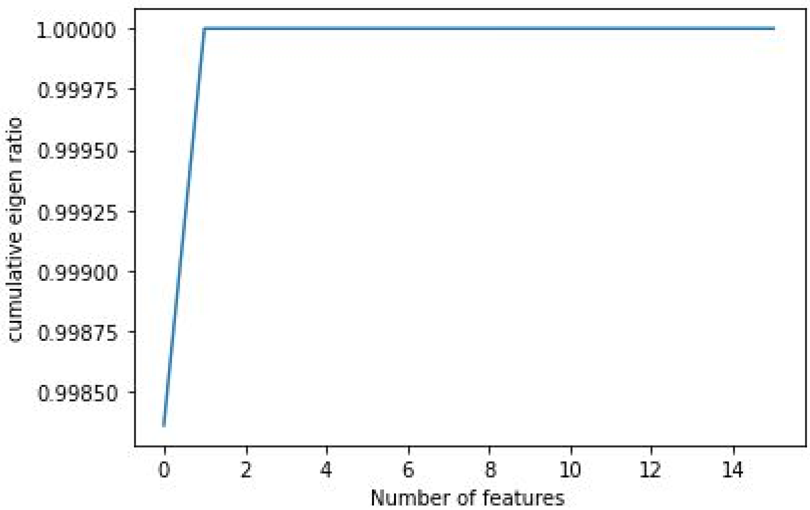
5-Fold Cross Validation Accuracies = [0.29695780176643766 , 0.2929342492639843, 0.29328753680078506 , 0.29501472031403336 , 0.2911874386653582]

Final Test Accuracy : 0.2975756814470544

Part 2:

First graph: Shows that on selecting a number of features how much variance ratio is retained.
Second graph: Shows the eigen-values ratio corresponding to their eigen-vectors(Scree graph).





5-Fold Cross Validation Accuracies= [0.27636898920510305 , 0.2745829244357213 , 0.2740529931305201 , 0.2746221786064769 , 0.2722080471050049]

Final Test Accuracy : 0.2764728049240045

Part 3:

Final Set of Features formed are : Total Final Features = 8
['Hospital_code', 'Hospital_type_code', 'City_Code_Hospital', 'Hospital_region_code', 'Ward_Type', 'Ward_Facility_Code', 'Bed Grade', 'Visitors with Patient']

5-Fold Cross Validation Accuracies = [0.29404490853419174, 0.2941823035251629 ,0.2893145952736123 , 0.2932205385883646, 0.2912381251472089]

Final Test Accuracy : 0.2932669660521936

Note:

1. Running the complete Notebook will take **1hr to 1:30hr**.

2. Used Decision trees to compute validation accuracy in backward_selection.If using the implemented naive_bayes instead of decision trees it will take a lot of time to finish