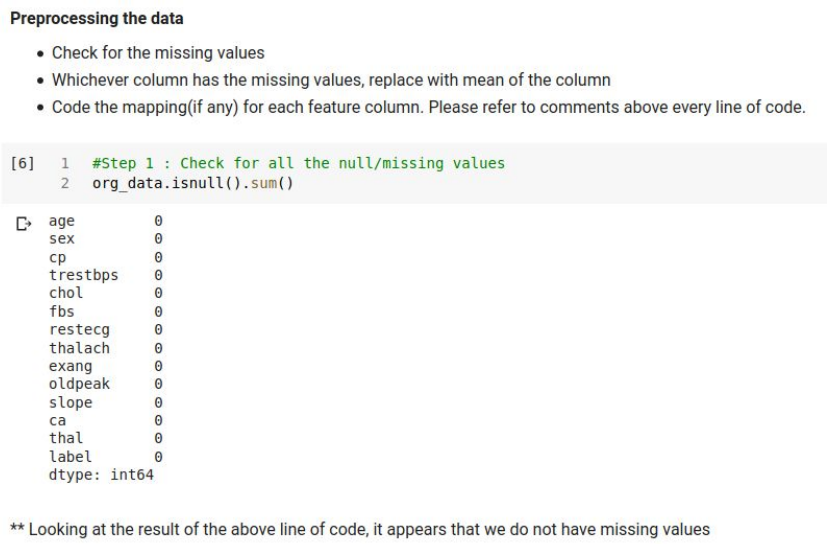
Statistical Computation | CS 609
End-Sem Assignment | Divyanshu Kumar Singh (2017048)

Dataset: We were given a dataset named Heart Disease[1] from UCI's repository, the data set had 14 variables including the target variable which was the ground truth. The variables were as follows: Age, Sex, Chest Pain, Resting Blood Pressure, Serum Cholesterol, Fasting Blood Sugar, Resting ECG, Max heart rate achieved, Exercise-induced chest pain, ST depression induced by exercise, peak exercise ST segment, Number of major vessels, Thal: Thalassemia, Ground truth/label/target.

Data Pre-Processing: For data preprocessing, Please refer to my filename 'data_preprocessing.py'. The steps followed in that are as follows

1. The original data file does not have column names, so we add those manually.
2. Then we check for the missing values in the data by using the command data_file.isnull().sum()
    a. This returns the result as shown below



Preprocessing the data

- Check for the missing values
- Whichever column has the missing values, replace with mean of the column
- Code the mapping(if any) for each feature column. Please refer to comments above every line of code.

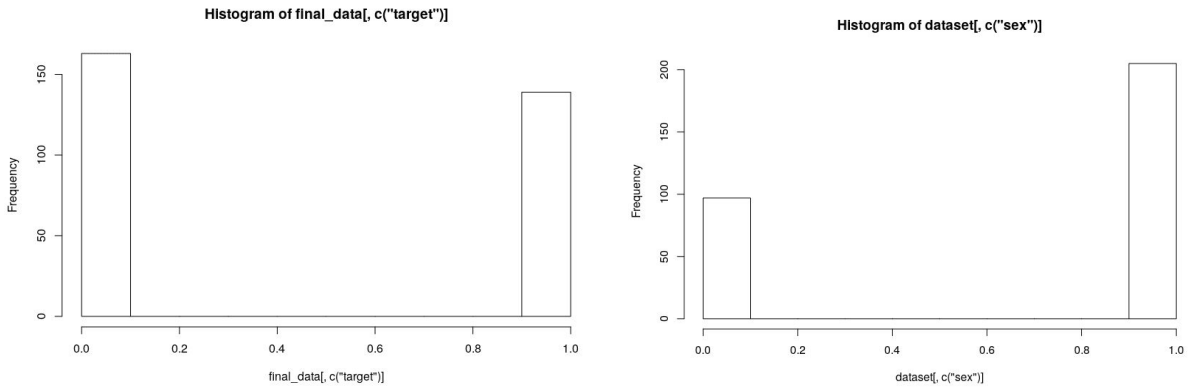
```
[6]  1  #Step 1 : Check for all the null/missing values
     2  org_data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
label       0
dtype: int64
```

** Looking at the result of the above line of code, it appears that we do not have missing values

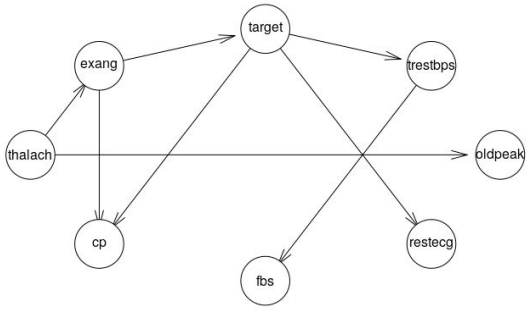
Visualizations: Here before proceeding to any machine learning we should first visualize the data for any possible class imbalance. Therefore, I have drawn histograms in R, for target values those are 0 and 1. To check, if data is biased or not. Secondly, I draw a histogram to check the data imbalance in the "sex" variable. The results are shown below:

[1] https://archive.ics.uci.edu/ml/datasets/Heart+Disease



Clearly from the above histogram, we can see there is no class imbalance in target values but there is a class imbalance in the "sex" variable.

R Code: First import the bnlearn package, then set the working directory.
Read the CSV file and add column names as the data is missing the column names. Now find 7 distinct random numbers between, 1-13, these are basically your column number. For these columns only we do Bayesian learning.
If any of these columns is continuous i.e if the column belongs to the list as following: "Age", "trestbps", "chol", "thalach", "oldpeak" or "thal".
then discretize the column using the command as shown below:
v2_new <- discretize(data.frame(dataset[,c(v2)]), breaks = 5)
Finally, replace the column in your dataset, instead I have made a new dataset with all the relevant features which I had to consider. Before, using structure learning first label the target values in 0 and 1 where 0 mean absent and 1 mean present, therefore, any 'target' column values less than 0 is assigned 0, and all values greater than equal to 1 are assigned 1.
Finally, spilt your dataset for training and testing. Out of 302 rows, I have used the first 290 rows for training and structure learning and rest for predictions.
I have used 'hc' i.e hill-climb function for structure learning. My network plot as shown below:



Then you fit your data using bn.fit(), and finally, make predictions, so my prediction came out to be: **0.301** for not present and **0.7553191** for present

```
           pred
  [1,] 0.3010204 0
  [2,] 0.7553191 1
  [3,] 0.7553191 1
  [4,] 0.7553191 1
  [5,] 0.3010204 0
  [6,] 0.3010204 1
  [7,] 0.7553191 1
  [8,] 0.3010204 1
  [9,] 0.3010204 1
 [10,] 0.7553191 1
 [11,] 0.3010204 1
 [12,] 0.3010204 0
```

The hill-climb algorithm that we are using here in structure learning is a score-based algorithm that uses score function and afterward searches over the directed acyclic graph for structure with the maximal score. Majorly, heuristic-based search algorithms such as simple hill-climbing works, as it examines every neighboring node and finally selects the first neighboring node which would optimize the current state of the node. We start with a graph, then at each step the algorithm tries to change the structure of the graph but by just a single operation such as removing edge, adding edge or, maybe reversing edge. No operation should not change the cyclic properties of the graph. Then we check if there is an increase in the score, then we adopt that change else we try again. The major loophole here could be the hill-climbing algorithm itself because in some it is unable to reach global maxima, and restricts itself to local maxima. Also, it depends on local information so looking at immediate neighboring nodes and, hence it lacks global information.

On the other hand the fit function *bn.fit(),* helps fit the parameters for a Bayesian Network given its structure and dataset. There are two methods in here one is ‘mle’ i.e maximum likelihood estimate and other ‘bayes’ for bayesian learning.

File Organization:

1.) Report_2017048.pdf
2.) End_Sem_Assgn_2017048.R #this is the R code complete implementation
3.) data_preprocessing.py # python file which I used for some basic data exploration
4.) processed_cleveland.csv # data file
5.) data_preprocessing.ipynb # juypter notebook file for same python file above

Reference

[1] Scutari, M. (2009). Learning Bayesian networks with the bnlearn R package. *arXiv preprint arXiv:0908.3817*.