Extracting Independence from given medical dataset and build a Bayesian Model*

*As a part of Advanced Artificial Course

Madina Kudaibergenova
Nazarbayev University
School of Engineering and Digital Sciences
Nur-Sultan, Kazakhstan
madina.kudaibergenova@nu.edu.kz

***Abstract*—This project is aimed to analyze the given medical dataset consisted of pyradiomics pneumonia features, and modify them to build Bayesian model. The paper is divided into sections: the Introduction describes Bayesian model in more details, demonstrates the problem, proposed solution to this problem, project plan, and project details; the Related work illustrates characteristics of chosen approach; the Implementation shows all the steps done; the Results confirm all the computations; and the Conclusion highlights important information, and concludes the paper.**

***Index Terms*—bayesian model, probability, radiomics, medical dataset, pneumonia detection, naive bayes classification**

I. Introduction

A. Bayesian Model

Bayesian network (BN) are knows as probabilistic models due to the fact that they are constructed using probabilities [1]. Moreover, BN uses probabilities to quantify uncertainty. Hence, instead of estimations and calculations results will be distributions [2].

B. Problem

The problem is to build BN from implicit medical pyradiomics dataset with a lot of features, extract Independence from this dataset, find relationships, and learn structure. The problem is NP-hard.

C. Solution

The most easiest and fast way (due to time limit) is to select best features for future model with Pairwise Correlation, analyze BN with Naive Bayes Classification, and apply Gaussian Inference for model evaluation.

D. Project Plan

1) Related research 1 week
2) Implementation 1 week
3) Report preparation 1 week

Acknowledgements to Professor Martin Lukac

E. Project details

1) Tools: Python programming language and Google Colab notebook

Frameworks and libraries:

- Working with dataset: pandas, numpy;
- Plotting: matplotlib, seaborn, arviz;
- Analysis: sklearn (GaussianNB), pymc3.

2) Project steps:

- Compare all images with labeled images to remove unlabeled
- Select features by applying pairwise correlation to reduce dimensionality
- Merge all modifications with labeled images data
- Apply Naive Bayes Classification
- Show Bayesian analysis

3) Dataset:

In this project there are 26685 patients' radiomic features, and 2500 labeled data.

- Patient IDs
- First order statistics (18 features)
- Gray Level Cooccurence Matrix (22 features)
- Gray Level Run Length Matrix (16 features)
- Gray Level Size Zone Matrix (16 features)
- Gray Level Dependence Matrix (14 features)

II. Implementation

A. Removing unlabeled data

First step of data preparation is to remove unlabeled data from 26 thousands dataset. It was done with the help of pandas.DataFrame.isin() function, which finds whether each element in data frame is contained in values. The aim is to use these data for further merging process. In the result there are 2500 patients with 86 features.

B. Choosing uncorrelated data

It is somewhat similar to independence extraction if no extra information is provided. It was done by using pandas.DataFrame.corr() function, and the correlation matrix was drawn by seaborn.heatmap() (see "Fig.1"). The target is to

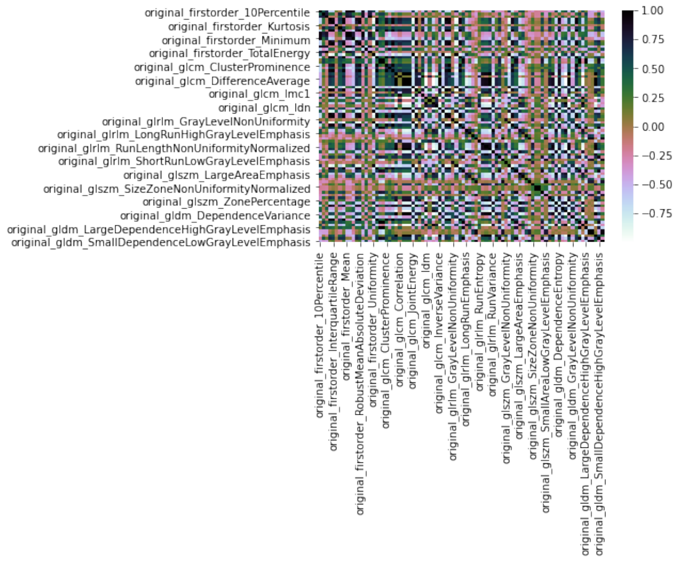


Fig. 1. Correlation matrix

use remove highly correlated data and make the dataset independent.

C. *Removing highly correlated data*

To remove highly correlated, first, we need to remove constant variables with zero variance; however, I haven't found such features. Next, we should drop duplicates, and in the result the duplicate was 'originalfirstorderTotalEnergy' feature. And finally, we remove correlated features which are higher than '0.8' correlation coefficient. In the result, we have 19 uncorrelated features of 2500 patients.

D. *Merging labels with features*

Additional steps are finding maximum accuracy, or i.e. correctly and wrong classified pneumonia, and which algorithm done its job better, then delete unnecessary columns. In the result, we have 2500 patients with 20 features. It was achieved with numpy.max(), pandas.DataFrame.pop(), and by creating new columns in dataframe and copying them. The main aim is to build the model on this data.

E. *Naive Bayes Classification*

In addition, I have split dataset into train and test to avoid overfitting when applying classification. Then I converted categorical data to numeric, i.e. correctly classified equals to 1, and wrong - 0, and called these variable 'algocleaned'.

Further steps:

1) initiate classifier with given features;

2) train classifier with 'algocleaned';

3) find probability distribution (e.g. P(originalfisrtorder10Percentile — algo=0 and algo=1)), and assume the distribution is Gaussian because we applying Gaussian Naive Bayes.

Results:

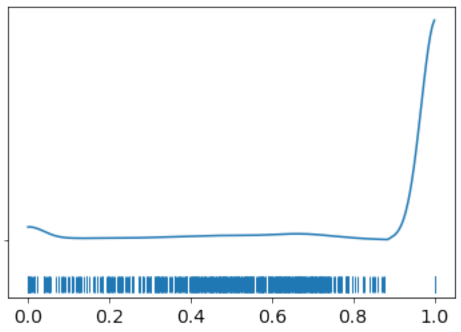
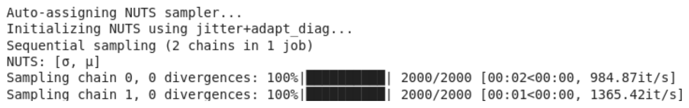


Fig. 2. KDE plot for algorithm accuracy



```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Sequential sampling (2 chains in 1 job)
NUTS: [σ, μ]
Sampling chain 0, 0 divergences: 100%|██████████| 2000/2000 [00:02<00:00, 984.87it/s]
Sampling chain 1, 0 divergences: 100%|██████████| 2000/2000 [00:01<00:00, 1365.42it/s]
```

Fig. 3. PyMC3 sampling results

Number of mislabeled points out of a total 1250 points : 351, performance 71.92

Std originalfirstorder10Percentile notalgo: 26.16

Std originalfirstorder10Percentile algo: 24.86

Mean originalfirstorder10Percentile notalgo 71.14

Mean originalfirstorder10Percentile algo: 53.96

F. *Bayesian analysis*

The main aim of this step is to understand and evaluate the model, and apply further steps for improvement of the model. It was done with pymc3 (for building model) and arviz(for plotting results). The method is to perform Gaussian inference on features.

Step1: check if there is any missing data

Step2: draw KDE (kernel density estimate) of Gaussian-like distribution. The calculated mean value of 'maxA' column is 0.82, and from the "Fig.2" we can see that a lot of data is far away from it, and it doesn't look Gaussian. However, if we don't know mean or standard deviation values, we should set priors, and then apply Gaussian inference on them.

Step3: initiate model in PyMC3 by sampling 1000 features of 'maxA'. Importantly, likelihood 'y' function conditions for the unknown on the known data (see "Fig.3").

Step4: plot Gaussian model trace. On the left of "Fig.4" - KDE plot with probabilities, on the right - sampled values at each stage. Posterior is bi-dimensional, therefore, we assume that it is Marginal distribution of each parameter.

Step5: plot joint distribution. From the "Fig.5" we can see that there is no correlation, and hence, no collinearity in the model which is desirable.

Step6: posterior predictive checks. This step is needed to validate the model, and it was done by generating data from the model using parameters from posterior. On the "Fig.6" we see drawn plot of generated 1000x2500 samples. We can analyze that the inferred mean (0.85) is not equal to actual

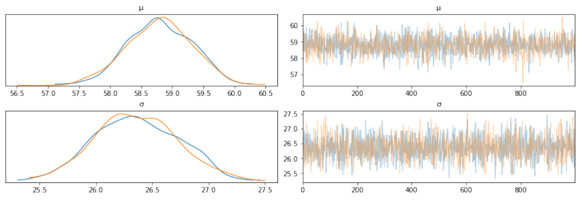


Fig. 4. KDE and sampling

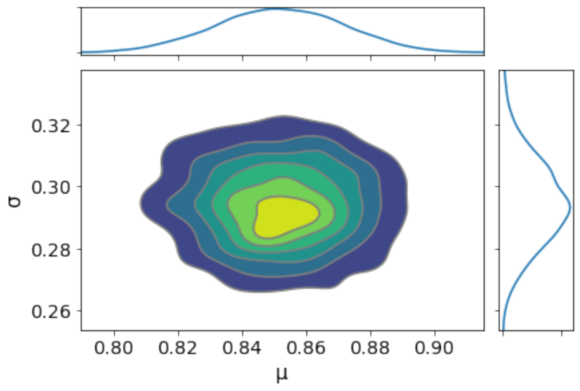


Fig. 5. Joint distribution

mean (0.82). Therefore, here we should take another features, and observe them the similar way.

III. Further steps

1) Apply NBC and BA for all features
2) Find another classifier for big dataset
3) Analyze dependence relations
4) Find powerful laptop to use bayesian structure learning (from bnlearn and pgmpy libraries) and plot BN

Conclusion

Overall, building BN is an NP-hard problem, especially when implicit unlabeled data is given. And the proposed solution for the problem is naive, and needs a lot of improvements. However, each model building requires some updates in dataset, and in this case it was, first of all, to extract Independence we need to find uncorrelated data. To reduce dimensionality we applied constant variables and highly correlated data removal. Then NBC was applied only on train data, and it was done to avoid overfitting. The NBC is fast, simple, and works well with high dimension, but inefficient for big dataset, and for '0 frequency' cases. Bayesian Analysis showed us that as we added more samples, the distribution looked similar to Gaussian. And finally, BN takes into account conditional independence to simplify graph and decrease the number of parameters for joint distribution; however, we need labels for images to do so.

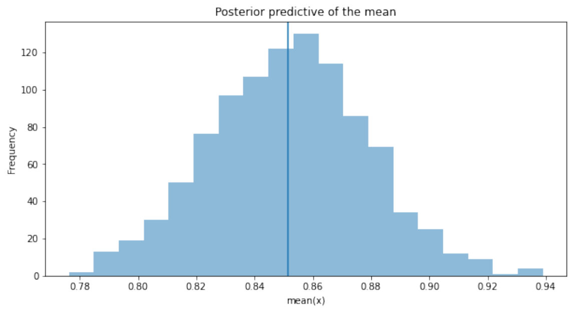


Fig. 6. Inferred mean of samples

References

[1] S. Monti, G.F. Cooper, "Learning Bayesian belief networks with neural network estimators," [Online]. Available: https://www.dbmi.pitt.edu/sites/default/files/Monti

[2] G.A.R. HerediaL, "Bayesian networks for classification, clustering, and high-dimensional data visualisation," Cardiff University, May-2008. [Online]. Available: https://orca.cf.ac.uk/54722/1/U585111.pdf. [Accessed: 26-Apr-2021].

[3] "Risk parity," Wikipedia, 20-Jan-2021. [Online]. Available: https://en.wikipedia.org/wiki/Riskparity. [Accessed: 02-May-2021].

[4] M. Muller, "Naive Bayes Classification with sklearn," Medium, Sicara. 28-Feb-2018. [Online]. Available: https://medium.com/sicara/naive-bayes-classifier-sklearn-python-example-tips-42d100429e44. [Accessed: 29-Apr-2021].

[5] S. Li, "Hands On Bayesian Statistics with Python, PyMC3 ArviZ," Towards Datascience, 17-Jul-2019. [Online]. Available: https://towardsdatascience.com/hands-on-bayesian-statistics-with-python-pymc3-arviz-499db9a59501. [Accessed: 1-May-2021].

[6] W. Koehrsen, "Estimating Probabilities with Bayesian Modeling in Python," Towards Datascience, 28-Nov-2018. [Online]. Available: https://towardsdatascience.com/estimating-probabilities-with-bayesian-modeling-in-python-7144be007815. [Accessed: 03-May-2021].