Darai (sujad96@zedat.fu-berlin.de), Mammadli (matanam94@zedat.fu-berlin.de), Hamidovic (samrah96@zedat.fu-berlin.de), Ibraim (ibraimii@hu-berlin.de)

RESEARCH

Project-2

Sujan Darai (sujad96@zedat.fu-berlin.de), Matanat Mammadli (matanam94@zedat.fu-berlin.de), Samra Hamidovic (samrah96@zedat.fu-berlin.de), Ibraimi Ibraim (ibraimii@hu-berlin.de)

Full list of author information is available at the end of the article

Abstract

Goal of the project: We aim to perform the analysis on original, dirty, imputed, and cleaned data, compare the model performance, and explore feature extraction.

Main results of the project: Performed analysis and compared the model evaluation of imputed data and top 3 metabolites were extracted

Personal key learning:

1 Learned to make data dirty,

2 data imputation using mean and median

Estimated working hours: 5-7 for every member

Project evaluation: .....

Keywords: Metabolomics, Imputation, Support vector machine, Principle component regression

1 Introduction

It turns out that data quality is much more influential in building a reliable and reproducible model than the use of ultra suffisticated machine learning tools and models. Missing data is a common issue encountered in real-world datasets, and how it's handled can significantly impact model outcomes.

In this project, we are interested in understanding the effect of inducing uncleaned data on the model performance. In particular, we focused on imputation strategies following the mean and median approach.

Our simulation involves deliberately introducing missing values in a random manner into the original dataset at varying percentages ($5\%, 10\%, \text{and} 20\%$). Subsequently, we train and evaluate the performance of two distinct models: Principal Component Regression (PCR) and Linear Kernel Support Vector Machine (SVM). By intentionally manipulating the data, we aim to simulate real-world scenarios where data quality might be compromised.

To address missing values, we employ two common imputation strategies: mean and median. These approaches are chosen for their simplicity and widespread use in handling missing data. We apply these strategies to impute the missing values in the altered datasets and rerun the models for evaluation.

This projects builds on the previous one which is based on metabolomics data used in training machine learning models with the goal of binary classification. The overarching goal of this study is to observe the impact of uncleaned data on model performance and compare the efficacy of mean and median imputation strategies.

By examining the performance metrics of the models on both original and altered datasets, we would like to understand the role of data quality and the effectiveness of imputation methods in fixing the issue of missing data. Ultimately, this analysis aims to provide valuable guidance on selecting the most suitable imputation strategy for handling missing data in real-world applications.

2 Goal of the project

On this project we focused on making metabolomics data dirty and afterwards we imputated the dirty data by using two different imputation methods. In the next step we performed an analysis on the original data, the dirty data and the cleaned data and compared the overall performance of the models that we used in the first project (PCR and SVM-lin). Lastly we selected the top three features of the datasets from PCR and SVM-lin.

3 Data and preprocessings

We have chosen the same dataset as in Project One. The MTBLS136 dataset which consists of a serum LC-MS dataset. These datasets consist of 949 named metabolites associated with postmenopausal hormone use. Multiple metabolites were compared in the group of women including 332 estrogen users and 337 estrogen plus progestin users versus 667 non-users. In this research, metabolite levels were compared between estrogen-only and estrogen-plus progestin users. This means it consists of the metabolite concentration of users that is supposed to be due to estrogen-only versus estrogen-plus-progestin hormones. [1]
Like in project one, all the required packages were imported and additionally a new package called seaborn with a random seed of 18. The dataset was loaded in the jupyter notebook. The data is then divided into two tables, namely: DataTables and PeakTables using the function `cb.utils.load_dataXL`, like in project one. To get a better understanding of the given dataset, we observed the first 5 rows of the DataTable dataset and the PeakTable dataset. In the next step we checked how many null values are in each column in DataTable with `DataTable.isnull().sum()` and then printed the total sum of null values with `print(DataTable.isnull().sum().sum())`. The DataTable consists of 310432 null values in total. After that we provided a summary statistics of the data stored in a DataTable.
Now for the main step, we made the DataTable dirty by removing 5%, 10% and 15% of the values and imputed them with mean and median imputation. This part will is described in more detail in the methods part.

4 Methods

4.1 Process to make data dirty

First, we attempted to make data dirty by removing 5 % of data, using `np.random.choice()` function. We randomly generated null values from 5% of the data. We did those same steps with 10 % and 15% of the data as well, created randomly generated null values.
We also made data dirty by changing values randomly with the help of `np.random.randint()` function, generating random integers in the range between 1 and 900000000. First we did it with 5 % of data, then with 10 % of data and finally, with 15 % of data.

4.2 Data imputation (mean and median technique)

Later on, we proceeded to clean our impure data by imputing the missing or changed values.
First, we cleaned our data using mean imputation technique. For that we separated the dataframe `Random_5_missing_DataTable` into two subsets: one containing only numerical columns (`df_numerical`) and the other containing only non-numerical columns (`df_non_numerical`).
For each numerical column in `df_numerical`, we filled missing values with the mean of that column, using `fillna()` function.
For each non-numerical column in `df_non_numerical`, it fills missing values with the mode (most frequent value) of that column.
We then concatenated `df_numerical` and `df_non_numerical` back together along the columns axis, using `pd.concat()` function, resulting in the dataframe `imputed_5_miss`.
We did these steps for 5 %, 10 % and 15 % dirty data, that we created earlier.
We did same steps mentioned above, separated our dataframe into two subsets, but this time used `fillna(df_numerical[col].median())` function for our numerical column, basically we filled missing values with the median of that column.
For non-numerical column we still used mode to fill the missing values.

4.3 Data cleaning

We later on cleaned our data by filtering out rows with a percentage of missing values less than 20%. For example, in the code snippet `PeakTableClean = PeakTable[(PercMiss < 20)], Perc_missing` represents the percentage of missing values for each row in the DataFrame. This line ensures that only rows with a relatively low percentage of missing data in the `Perc_missing` column are retained in the DataFrame PeakTableClean

4.4 Hyperparameter optimization

We began by splitting our data into training and testing sets using the `train_test_split` function from scikit-learn. The training set ($DataTrain$) contains 2/3 of the data, while the test set (`DataTest`) contains 1/3 of the data. We also split the target variable Y accordingly. Following the split, we extracted the training and testing datasets separately. For each dataset, we applied a logarithmic transformation and then imputed missing values using k-nearest neighbors with k=3.
Then we did hyperparameter optimization for our linear SVM model using k-fold cross-validation and evaluated the model's performance using AUC and R^2Q^2 metric. For cross-validation we used `cb.cross_val.KFold(model=cb.model.SVM, X=XTrainKnn, Y=YTrain, param_dict=param_dict, folds=5, n_mc=10)` function.
Our hyperparameter here being optimized through cross-validation is C, which represents the regularization parameter for the Support Vector Machine (SVM) model.
Later we plotted our model with AUC and R^2Q^2 metrics.

4.5 Bootstrap evaluation

At last we performed bootstrap resampling on the data, trained multiple models on the bootstrapped samples, and evaluated the ensemble model's performance using separate evaluation datasets for training and testing. For that we used

`cb.bootstrap.Per(model, bootnum=100)` and `bootmodel.evaluate(trainset=EvalTrain, testset=EvalTest)` functions.
This approach helps assess the stability and generalization ability of the model by estimating its performance across multiple resampled datasets.

5 Results and Discussion

We have used PCR and SVM-lin methods only due to word limitation.
The following figures illustrate the ROC curves. Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the performance of a binary classification model across different threshold values. The ROC curve plots the true positive rate (TPR), also known as sensitivity, against the false positive rate (FPR), which is the complement of specificity. Sensitivity measures the proportion of actual positive samples that are correctly identified as positive by the classifier, while specificity measures the proportion of actual negative samples that are correctly identified as negative.
There are three ROC curves demonstrated in Figure 1, all of them describe ROC curves of PCR data, imputed with mean values. First subfigure (see 1a) demonstrates ROC curve for 5% imputed PCR data, the closer to the top left the curve is, the higher is sensitivity the more correctly identified positive samples are and the better performance our model (or classifier) has. The second and third subfigures (see 1b and 1c), which are ROC curves for 10% and 15% imputed data, demonstrate less sensitivity and performance compared to first subfigure. Next figure (see Figure 2) shows three subfigures of ROC curves for PCR data imputed with median values. Here we can also observe same tendencies, subfigure 2a (for 5% imputed data) has highest sensitivity (True positive rate) and higher AUC, therefore higher performance of the classifier. The subfigure 2b (for 10% imputed data) and 2c (15% imputed data) showcase less sensitivity and performance abilities. The third figure (Figure 3) consists of three subfigures as well, demonstrates ROC curves for SVM-lin data imputed with mean values. The first subfigure (3a) shows ROC curve for 5% imputed data, has lower sensitivity and performance than other subfigures. The subfigure 3b shows ROC curve for 10% imputed data and displays higher sensitivity and performance (and higher AUC) than other subfigures. The last subfigure (3c) for 15% imputed data also has less sensitivity and performance than the second one. The last and fourth figure (Figure 4) illustrates once again three ROC curves for SVM-lin data imputed with median values. The first subfigure (4a) for 5% imputed data, has quite high sensitivity and performance, just a little less than second one but higher than third subfigure. The second subfigure (4b) for 10% imputed data has the highest sensitivity and performance and AUC than all other subfigures in Figure 4 and the third subfigure (4c) for 15% imputed data has less sensitivity and performance than the others.

After building and tuning the model, evaluating it with bootstrap, we conducted feature importance analysis to identify and extract the most important features, in our case, metabolites.
As we can see from table 1, most important features aka metabolites from 5% imputed data with mean imputation are, number 1 is cystine for PCR data and 2-Deoxyuridine for SVM-lin data.

Cystine is a non-essential amino acid that plays several important roles in the body, it has antioxidant activity being a precursor to glutathione, one of the body's most important antioxidants. Glutathione helps neutralize harmful free radicals and protects cells from oxidative stress. Cystine is also important for detoxification (Glutathione is involved in detoxifying harmful substances, such as heavy metals, pollutants, and carcinogens). Other than that, cystine plays a key role in supporting immune function, because glutathione helps regulate the immune response by modulating the activity of immune cells, such as T cells and macrophages. Cystine is also essential for protein structure and stability (being a component of cysteine), collagen synthesis and hair as well skin health.

2-Deoxyuridine is a modified nucleoside that serves several important functions in the body, it is valueable for DNA Synthesis and Repair. 2-Deoxyuridine is one of the building blocks (nucleotides) used in the synthesis and repair of DNA. This nucleoside is also important for cellular signaling, it can act as a signaling molecule in various cellular processes. It has been implicated in signaling pathways related to cell proliferation, differentiation, and apoptosis (programmed cell death). There are also some studies suggesting that 2-deoxyuridine may exhibit antiviral activity against certain viruses.

As we can notice from table 1, both PCR and SVM-lin datas have 2-Aminoheptaonate as second most important feature/metabolite.

2-Aminoheptanoate is an amino acid derivative with various biological roles, such as intermediate in lysine biosynthesis, precursor in the biosynthesis of several antibiotics, such as cephalosporins and penicillins. 2-Aminoheptanoate can also be generated as a metabolite during the degradation of fatty acids, particularly unsaturated fatty acids. It is also potentially a neurotransmitter in the brain.

The most important features from 10% imputed data (with mean imputation) are, number 1 S-methylcysteine for PCR data and 1-linoleoyl-GPA (18:2)* for SVM-lin data (see table 2).

S-methylcysteine is a naturally occurring amino acid derivative found in various foods and plants. It plays diverse roles in biological systems, including detoxification, antioxidant defense, liver health, anti-inflammatory effects, cardiovascular protection, and antimicrobial activity.

1-Linoleoyl-GPA (18:2)* is a specific molecular species of lysophosphatidic acid (LPA), which is a bioactive lipid molecule that plays various roles in biological systems. It plays diverse and essential roles in cellular signaling, smooth muscle regulation, wound healing, angiogenesis, neuronal function, inflammation, and immune response.

The most important features from 15% imputed data (with mean imputation) are, number 1 2-Aminoheptaonate for PCR data and 1-Linoleoyl-GPA (18:2)* for SVM-lin data (see table 3.

We have already described biological roles and functions of these metabolites previously.

The most important features from 5% imputed data (with median imputation) are, number 1 cystine for PCR data and 2-Deoxyuridine for SVM-lin data (see table 4.

The functions and biological roles of these metabolites were already described previously.

The most important features from 10% imputed data (with median imputation) are, number 1 S-methylcysteine for PCR data and 1-Linoleoyl-GPA (18:2)* for SVM-lin data (see table 5).
The most important features from 15% imputed data (with median imputation) are, number 1 gluconate for PCR data and 1-Linoleoyl-GPA (18:2)* for SVM-lin data (see table 6).
Gluconate, or gluconic acid, is a naturally occurring organic compound that serves several important biological roles and functions. It plays diverse roles in metabolism (gluconate participates in various metabolic pathways in the body, such as pentose phosphate pathway (PPP) and nicotinamide adenine dinucleotide phosphate (NADPH)), energy production (gluconate can be metabolized to produce energy in the form of adenosine triphosphate (ATP)), calcium regulation, heavy metal chelation, skin health, and food preservation.
We have already talked about biological roles and functions of 1-Linoleoyl-GPA (18:2)* previously.

Number	PCR	SVM-lin
1	cystine	2'-deoxyuridine
2	2-aminoheptanoate	2-aminoheptanoate
3	S-methylcysteine	1-linoleoyl-GPA (18:2)*

Table 1: Important features i.e. metabolites on 5% imputed data with mean imputation

Number	PCR	SVM-lin
1	S-methylcysteine	1-linoleoyl-GPA (18:2)*
2	S-methylcysteine sulfoxide	lactosyl-N-behenoyl-sphingosine (d18:1/22:0)*
3	cystine	succinimide

Table 2: Important features i.e. metabolites on 10% imputed data with mean imputation

Number	PCR	SVM-lin
1	2-aminoheptanoate	1-linoleoyl-GPA (18:2)*
2	cystine	androstenediol (3beta,17beta) disulfate (2)
3	eugenol sulfate	2-aminoheptanoate

Table 3: Important features i.e. metabolites on 15% imputed data with mean imputation

Number	PCR	SVM-lin
1	cystine	2'-deoxyuridine
2	S-methylcysteine	2-aminoheptanoate
3	carotene diol (3)	1-linoleoyl-GPA (18:2)*

Table 4: Important features i.e. metabolites on 5% imputed data with median imputation

Number	PCR	SVM-lin
1	S-methylcysteine	1-linoleoyl-GPA (18:2)*
2	cystine	lactosyl-N-behenoyl-sphingosine (d18:1/22:0)*
3	S-methylcysteine sulfoxide	succinimide

Table 5: Important features i.e. metabolites on 10% imputed data with median imputation

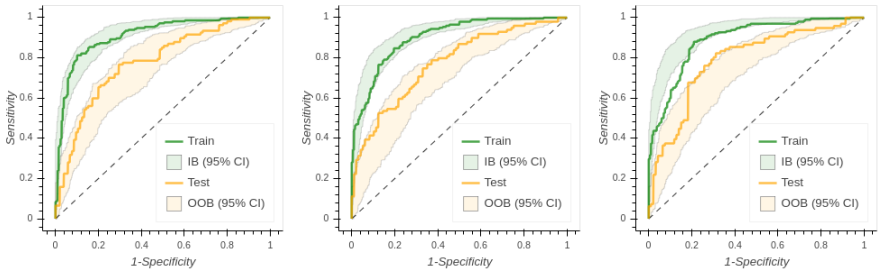
Number	PCR	SVM-lin
1	gluconate	1-linoleoyl-GPA (18:2)*
2	perfluorooctanesulfonic acid (PFOS)	2-aminoheptanoate
3	2-aminoheptanoate	androstenediol (3beta,17beta) disulfate (1)

Table 6: Important features i.e. metabolites on 15% imputed data with median imputation

6 Contributions

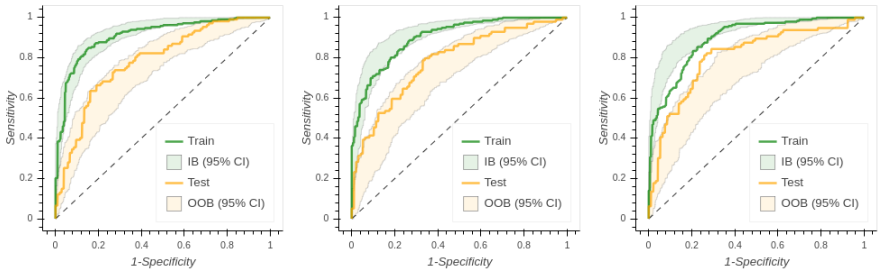
Sujan Darai: running and implementing code
Matanat Mammadli: writing report (Methods, Results and Discussion)
Samra Hamidovic: running code and writing report (Goal, Data & preprocessing)
Ibraim Ibraimi: writing report (Abstract, Introduction), double check the report

7 Appendix



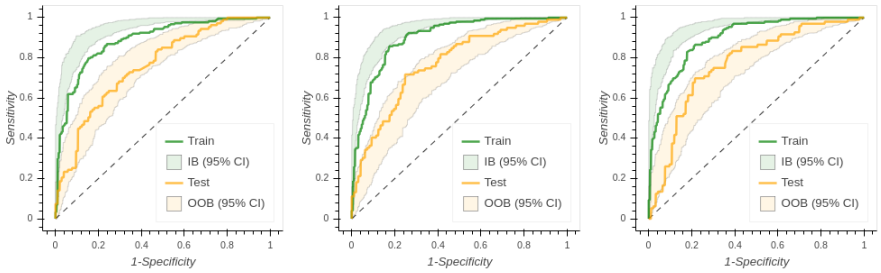
(a) 5% imputed (b) 10% imputed (c) 15% imputed

Figure 1: PCR method, imputed with mean values



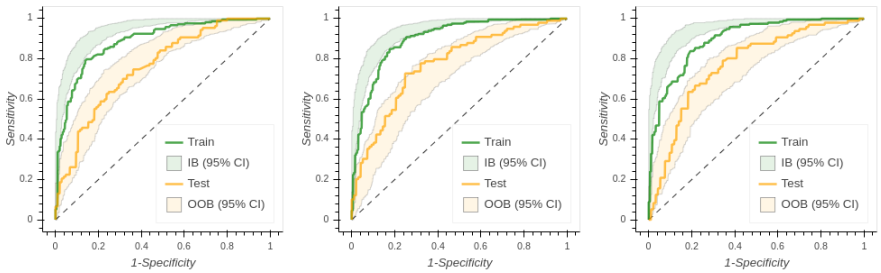
(a) 5% imputed (b) 10% imputed (c) 15% imputed

Figure 2: PCR method, imputed with median values



(a) 5% imputed (b) 10% imputed (c) 15% imputed

Figure 3: SVM-lin method, imputed with mean values



(a) 5% imputed (b) 10% imputed (c) 15% imputed

Figure 4: SVM-lin method, imputed with median values

References

1. Victoria L Stevens, Ying Wang, Brian D Carter, Mia M Gaudet, and Susan M Gapstur. Serum metabolomic profiles associated with postmenopausal hormone use. *Metabolomics*, 14:1–14, 2018.