Contents

Abstract
This project illustrates data processing and applying different Machine Learning Algorithms on a dataset containing occurrences of hepatitis in people. These algorithms determine the accuracy of the model in predicting the chance of being alive or dead. For data processing, we applied missing values solutions, such as: 1. Dropping rows with unknown values, 2. Dropping the features with the most unknown values, 3. Imputation: (a). Filling with the mean for numerical value, (b). Filling the categorical value with the same type of missing values. Then, we applied six ML methods such as: Gradient Boosting Classifier, Random Forest Classifier, Decision Tree Classifier, K Neighbors Classifier, Support Vector Machine classifier, MLP Classifier. In each algorithm, the confusion matrix was calculated and the accuracy of the model was compared before and after data processing in Table 2. For data processing, we illustrated how to deal with missing values by using Python pandas with scikit-learn. At the end we used a heatmap, which illustrated the correlation and impact of each features to other features and the class. Due to the restriction on the numbers of the pages in this report, only the complete procedure of before data processing and one applied method after data processing (dropping rows) is represented. The result of other methods is presented in Table.2. For more information about the detailed procedure of all the applied methods, please refer to the attached code.

I. Dataset, Tools, and Utilities

This dataset contains occurrences of hepatitis in people. The dataset has been downloaded from [1], and the code was programmed in Google Colab. The .ipynb file used is attached. The file is explained step-by-step in the report. We set Colab to GPU Runtime for a faster running time.

II. Data Preparation

To be able to program this project in Python, we had to import multiple appropriate libraries. After importing libraries, we read the .csv file from the computer by uploading it to the session storage in Colab. The dataset has several unknown values such as "NaN" or empty values. Several things should be done to the input data to get accurate output data. In this section, we state the procedure for preparing the dataset.

Table 1. Some parts of the dataset.

age	sex	steroid	antivirals	anorexia	…	protime	histology	class
30	male	FALSE	FALSE	FALSE	…		FALSE	live
50	female	FALSE	FALSE	FALSE	…		FALSE	live
78	female	TRUE	FALSE	FALSE	…		FALSE	live
31	female		TRUE	FALSE	…	80	FALSE	live
34	female	TRUE	FALSE	FALSE	…		FALSE	live
34	female	TRUE	FALSE	FALSE	…	75	FALSE	live
51	female	FALSE	FALSE	TRUE	…		FALSE	die
23	female	TRUE	FALSE	FALSE	…		FALSE	live
39	female	TRUE	FALSE	FALSE	…		FALSE	live
30	female	TRUE	FALSE	FALSE	…		FALSE	live
39	female	FALSE	TRUE	FALSE	…	85	FALSE	live

32	female	TRUE	TRUE	FALSE	…	54	FALSE	live
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

First, we obtained the unique values of each feature in the dataset using "set". As shown in Table. 1, the dataset has ordinal attributes, which are qualitative values. Now, we have to prepare the dataset and get rid of unknown values. We found the unique values of each features, then counted the unknown values and found the index of each unknown value. We then compare different data preparation approaches to determine their efficiency in using different Machine Learning algorithms. After that, The LabelEncoder has been used to change the qualitative values into numerical values. Finally, the dataset should only include values such as 0, 1, and 2. To train the model, we should separate the labels/classes from the dataset. In what follows, the unique values of some of the features is shown:

```
Age
{7, 20, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 59, 60, 61, 62, 64, 65, 66, 67, 69,
70, 72, 78}
sex
{'male', 'female'}
:
:
:
steroid
{False, True, nan}
class
{'live', 'die'}
```

Number of unknown values in each features/columns:

```
age                0
sex                0
steroid            1
antivirals         0
fatigue            1
malaise            1
anorexia           1
liver_big         10
liver_firm        11
spleen_palpable    5
spiders            5
ascites            5
varices            5
bilirubin          6
alk phosphate     29
sgot               4
albumin           16
protime           67
histology          0
class              0
```

We now split the project into two parts.

A. Before data preparation.
B. After data preparation [2], [3]. In this part we used RepeatedStratifiedKFold function to repeat Stratified K-Fold n times with different randomization in each repetition. This allows for model accuracy to be more reliable [4]. In this step, we applied the following methods for data preparation and data processing and missing values:

1. Dropping rows with unknown values (NaN)
2. Dropping the features with the most unknown values (NaN)
3. Imputation [5]

- Filling with the mean or median value if its a numerical value.
- Filling with mode if its categorical value.
- Filling the numerical value with 0 or -999, or some other number which will not occur in the data.
- Filling the categorical value with type new for the missing values.

III. Applied Machine Learning Strategies

In this project we applied the following methods and compared the results before and after data preparation [6]**.**

- Gradient Boosting Classifier
- Random Forest Classifier
- Decision Tree Classifier
- K Neighbors Classifier
- Support Vector Machine classifier
- MLP Classifier

To evaluate the performance of each approach we calculate the confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. The most basic terms:

- *True Negative (TN)*
- *False Positive (FP)*
- *False Negative (FN)*
- *Accuracy*
- *Recall*

1. Gradient Boosting Classifier

Gradient Boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. [7].

A. Before data preparation:

```
Accuracy --> 0.7941666666666667
```

B. After data preparation:

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values

```
Accuracy --> 0.8333333333333334
[[ 2  1]
 [ 3 10]]
              precision    recall  f1-score   support

           0       0.40      0.67      0.50         3
           1       0.91      0.77      0.83        13
```

```
    accuracy                           0.75        16
   macro avg       0.65      0.72      0.67        16
weighted avg       0.81      0.75      0.77        16

Mean Absolute Error: 0.25
Mean Squared Error: 0.25
Root Mean Squared Error: 0.5
```

2. *Random Forest Classifier*

Random Forests and Random Decision Forests are an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time.

A. Before data preparation:

```
Accuracy --> 0.8451612903225806
```

B. After data preparation:

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values

```
Accuracy --> 0.8958333333333334
[[ 2  1]
 [ 1 12]]
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.92      0.92      0.92        13

    accuracy                           0.88        16
   macro avg       0.79      0.79      0.79        16


weighted avg       0.88      0.88      0.88        16

Mean Absolute Error: 0.125
Mean Squared Error: 0.125
Root Mean Squared Error: 0.3535533905932738
```

3. *Decision Tree Classifier*

Decision Tree learning is one of the predictive modelling approaches used in Statistics, Data Mining, and Machine Learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves) [8].

A. Before data preparation:

```
Accuracy --> 0.7806451612903226
```

B. After data preparation:

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values

```
Accuracy --> 0.8375
[[2 1]
 [4 9]]
              precision    recall  f1-score   support
```

```
           0       0.33      0.67      0.44         3
           1       0.90      0.69      0.78        13

    accuracy                           0.69        16
   macro avg       0.62      0.68      0.61        16


weighted avg       0.79      0.69      0.72        16
Mean Absolute Error: 0.3125
Mean Squared Error: 0.3125
Root Mean Squared Error: 0.5590169943749475
```

4. K Neighbors Classifier

In Statistics, the K-Nearest Neighbors algorithm (k-NN) is a non-parametric classification method and later expanded for classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression [9]:

A. Before data preparation:

```
At k=1 accuracy-->0.7290322580645161
At k=2 accuracy-->0.6903225806451612
At k=3 accuracy-->0.7806451612903226
At k=4 accuracy-->0.7548387096774192
At k=5 accuracy-->0.767741935483871
At k=6 accuracy-->0.735483870967742
At k=7 accuracy-->0.7612903225806452
At k=8 accuracy-->0.7419354838709677
At k=9 accuracy-->0.7806451612903226
At k=10 accuracy-->0.7806451612903226
At k=11 accuracy-->0.7935483870967742
At k=12 accuracy-->0.7806451612903226
At k=13 accuracy-->0.7935483870967742
At k=14 accuracy-->0.7935483870967742
At k=15 accuracy-->0.7935483870967742
At k=16 accuracy-->0.7935483870967742
At k=17 accuracy-->0.7935483870967742
At k=18 accuracy-->0.7935483870967742
At k=19 accuracy-->0.7935483870967742
```

B. After data preparation:

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values.

```
At k=1 accuracy-->0.825
At k=2 accuracy-->0.8375
At k=3 accuracy-->0.8416666666666667
At k=4 accuracy-->0.8583333333333333
At k=5 accuracy-->0.85
At k=6 accuracy-->0.8291666666666667
At k=7 accuracy-->0.8583333333333333
At k=8 accuracy-->0.8416666666666667
At k=9 accuracy-->0.8416666666666667
At k=10 accuracy-->0.8416666666666667
At k=11 accuracy-->0.8333333333333334
At k=12 accuracy-->0.8208333333333333
At k=13 accuracy-->0.8208333333333333
At k=14 accuracy-->0.8125
At k=15 accuracy-->0.8333333333333334
At k=16 accuracy-->0.8291666666666667
At k=17 accuracy-->0.8375
At k=18 accuracy-->0.8416666666666667
At k=19 accuracy-->0.8375
[[ 0  3]
 [ 0 13]]
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.81      1.00      0.90        13

    accuracy                           0.81        16
   macro avg       0.41      0.50      0.45        16

weighted avg       0.66      0.81      0.73        16
Mean Absolute Error: 0.1875
Mean Squared Error: 0.1875
Root Mean Squared Error: 0.4330127018922193
```

5. *Support Vector Machine classifier*

In Machine Learning, Support-Vector Machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

A. Before data preparation:

kernel = poly

```
Accuracy --> 0.7741935483870966
```

kernel = Linear

```
Accuracy --> 0.7870967741935484
```

kernel = rbf

```
Accuracy --> 0.7870967741935483
```

B. After data preparation:

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values. Examine kernel: rbf

```
Accuracy --> 0.875
[[ 2  1]
 [ 0 13]]
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.93      1.00      0.96        13

    accuracy                           0.94        16
   macro avg       0.96      0.83      0.88        16

weighted avg       0.94      0.94      0.93        16
Mean Absolute Error: 0.0625
Mean Squared Error: 0.0625
Root Mean Squared Error: 0.25
```

6. *MLP Classifier*

A Multilayer Perceptron (MLP) is a class of feedforward Artificial Neural Network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN or strictly to refer to networks composed of multiple layers of perceptrons [10].

A. Before Data preparation

```
4/4 [==============================] - 0s 17ms/step - loss: 0.4887 - accuracy: 0.7977 -
val loss: 0.4656 - val accuracy: 0.8333
Epoch 97/100
4/4 [==============================] - 0s 15ms/step - loss: 0.4363 - accuracy: 0.8341 -
val_loss: 0.4654 - val_accuracy: 0.8333
Epoch 98/100
4/4 [==============================] - 0s 20ms/step - loss: 0.4548 - accuracy: 0.8248 -
val loss: 0.4653 - val accuracy: 0.8333
Epoch 99/100
4/4 [==============================] - 0s 18ms/step - loss: 0.4828 - accuracy: 0.7998 -
val_loss: 0.4651 - val_accuracy: 0.8333
Epoch 100/100
4/4 [==============================] - 0s 16ms/step - loss: 0.4968 - accuracy: 0.7904 -
val_loss: 0.4650 - val_accuracy: 0.8333
```

B. After data preparation

In order to reduce the number of the pages only one case is investigated: Dropping rows with unknown values.

```
Epoch 98/100
2/2 [==============================] - 0s 49ms/step - loss: 0.3859 - accuracy: 0.8438 -
val_loss: 0.5038 - val_accuracy: 0.8333
Epoch 99/100
2/2 [==============================] - 0s 50ms/step - loss: 0.3686 - accuracy: 0.8542 -
val loss: 0.5037 - val accuracy: 0.8333
Epoch 100/100
2/2 [==============================] - 0s 51ms/step - loss: 0.3973 - accuracy: 0.8333 -
val_loss: 0.5036 - val_accuracy: 0.8333
```

In the following table, the model accuracy in different applied ML algorithms have been compared. According to this table, the maximum accuracy was achieved by algorithms (rows) in which unknown values were dropped.

Table 2. Model accuracy comparison in different ML algorithms

Accuracy				
ML Algorithms	Before Data Processing	After Data Processing and Preparation		
		Dropping rows with NaN	Dropping columns with the most NaN	Imputation
Dataset Dimension: X.shape	(155, 19)	(80,19)	(155, 14)	(155, 19)
Gradient Boosting Classifier	0.79	0.83	0.79	0.79
Random Forest Classifier	0.84	0.89	0.79	0.83
Decision Tree Classifier	0.78	0.83	0.75	0.77
K Neighbors Classifier	0.79	0.83	0.78	0.78
SVM classifier	0.77	0.87	0.79	0.79
MLP Classifier	0.79	0.83	0.78	0.83

IV. Feature Selection

Feature Selection is one of the core concepts in Machine Learning which significantly impacts the performance of a model. Irrelevant or partially relevant features can negatively impact model performance. Feature Selection methods include:

- Univariate Selection
- Feature Importance
- Correlation Matrix with Heatmap

In this project, we applied Correlation Matrix with Heatmap to find the most important features for training the mode. Correlation states how the features are related to each other or the target variable. Correlation can be positive (an increase in one value of a feature increases the value of the target variable) or negative (an increase in one value of a feature decreases the value of the target variable). A Heatmap makes it easy to identify which features are the most related to the target variable. We will plot a Heatmap of correlated features using the seaborn library [11]. The last row of Figure 1.(b) shows how the class is correlated with other features. Protime and albumin are the most highly correlated with class, followed by sex and anorexia. Histology and ascites seem to be the least correlated with class.

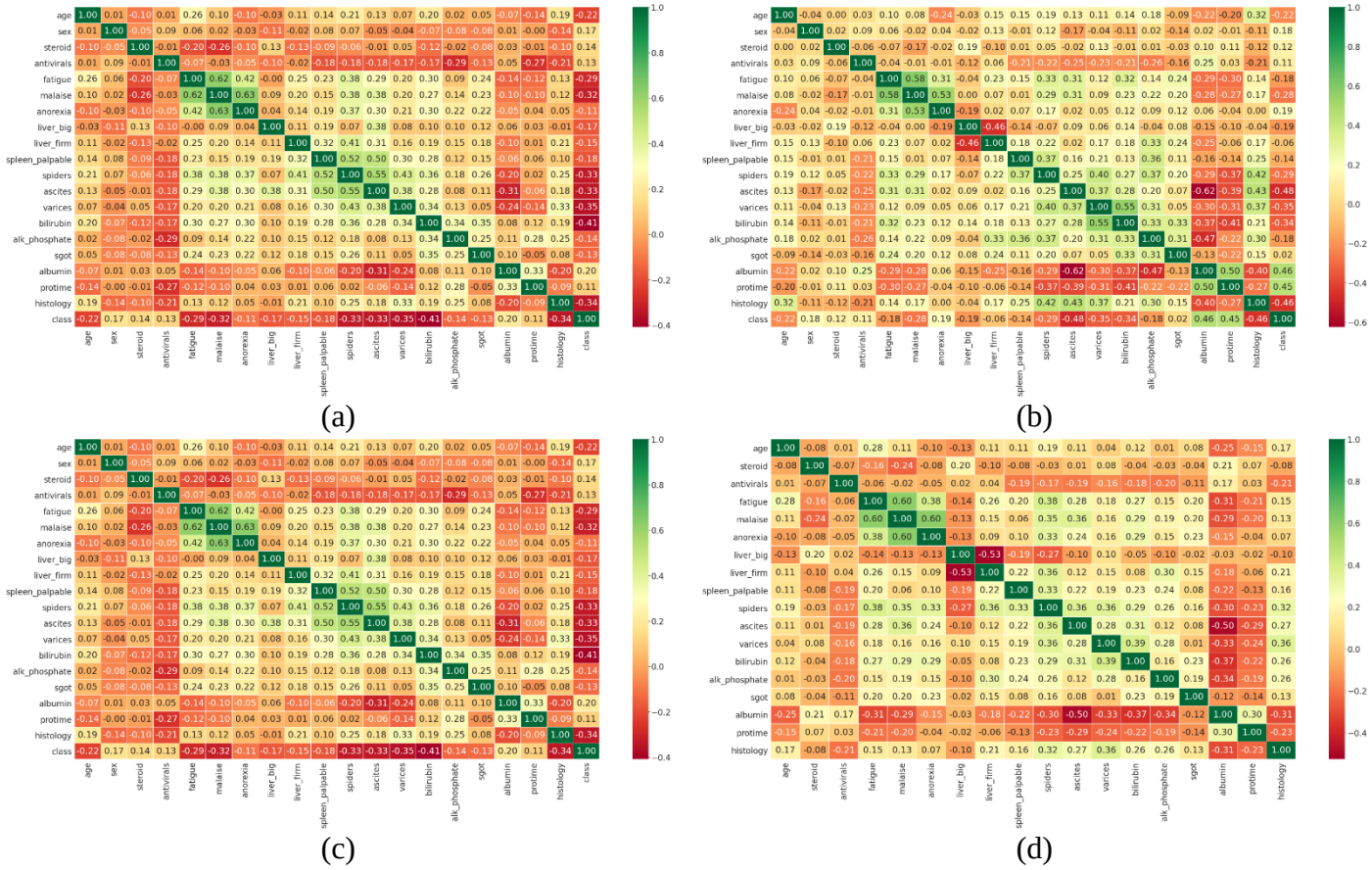

Figure 1. (a): Correlation Matrix with Heatmap before data preparation. (b): Correlation Matrix with Heatmap after data preparation by dropping rows including NaN. (c): Correlation Matrix with Heatmap after data preparation by dropping columns including the most NaN. (d) Correlation Matrix with Heatmap after data preparation using Imputation.

V. Conclusion

Based on comparing the accuracy of different approaches before and after data preparation, we conclude that deleting the entire column is not an appropriate solution. But this is an extreme case and should only be used when there are many (more than 50%) null values in the column. Deleting rows, including 'NaN' values, can be a better solution. In this case, we were able to achieve better accuracy than before. Using Imputation approach caused a decrease in the model accuracy. This is possibly

because the filled columns contained more valuable information than we expected. This will not happen in general. However, in this case, it means that the mean/Boolean has not filled the ‘NaN’ value properly.

References

[1] "hepatitis_csv dataset." https://www.openml.org/d/55

[2] "Working with missing data" https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html.

[3] "Dealing with Missing Values in a Dataset — Data Cleaning in Python." https://python.plainenglish.io/dealing-with-missing-values-in-a-dataset-data-cleaning-in-python-57bf9c8564da.

[4] "RepeatedStratifiedKFold" https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RepeatedStratifiedKFold.html

[5] "How to Handle Missing Data in Pandas DataFrame." https://stackabuse.com/python-how-to-handle-missing-dataframe-values-in-pandas/.

[6] "ML Algorithms." https://github.com/Aakash10399/diabetes-dataset/blob/master/findings.ipynb.

[7] "Gradient boosting." https://en.wikipedia.org/wiki/Gradient_boosting.

[8] "Decision Tree Classifier." https://en.wikipedia.org/wiki/Decision_tree_learning.

[9] "K Neighbors Classifier." https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[10] "MLP Classifier." https://en.wikipedia.org/wiki/Multilayer_perceptron.

[11] "Feature Selection Techniques in Machine Learning" https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e.