Models Comparison for Liver Cirrhosis Survival Prediction

Cleto Pellegrino

August 27, 2024

Contents

1 Introduction

The liver is the **second largest organ** in the human body and **one of the most important for human health**. Cirrhosis is a progressive condition that puts both a person's liver and life at risk. [1]

Different models can be compared to determine **which one best predicts the likelihood of a patient to survive against the disease**.

2 Dataset

This study utilizes the Cirrhosis Prediction Dataset available on **Kaggle**[2], comprising data collected from the **Mayo Clinic trial in primary biliary cirrhosis** (*PBC*) of the liver conducted between 1974 and 1984.

The dataset includes various information such as Cholesterol, Triglycerides, and, crucially, the patient's **Status**. In total, there are 418 observations with 20 attributes.

3 Data Visualization

3.1 Preliminary Adjustments

Before commencing with a complete dataset visualization, several **adjustments** were made for some features:

- All features of type "object" were converted to "**category**" (in order to plot barplots and use the *.describe* function provided in pandas with specific characteristics for this type of features).
- The "ID" column was **dropped** since it's useless for the analysis, solely serving as an **identifier**.
- The "Stage" feature, originally represented as *float64*, was converted to **categorical**, since it can be considered the **name** of the current liver condition.
- Age, initially reported in **days**, was converted to years using the formula: $\text{round}(\frac{x}{365}, 0)$.

  This was done solely for visualization purposes.
- Status **C** was converted to "High chance", Status **CL** was converted to "Transplant needed" and Status **D** was converted to "Low chance" (again, still just for visualization reasons).

3.2 Plot Analysis

By looking at the **heatmap** created by calculating the **mutual information** between the features (Figure 1),

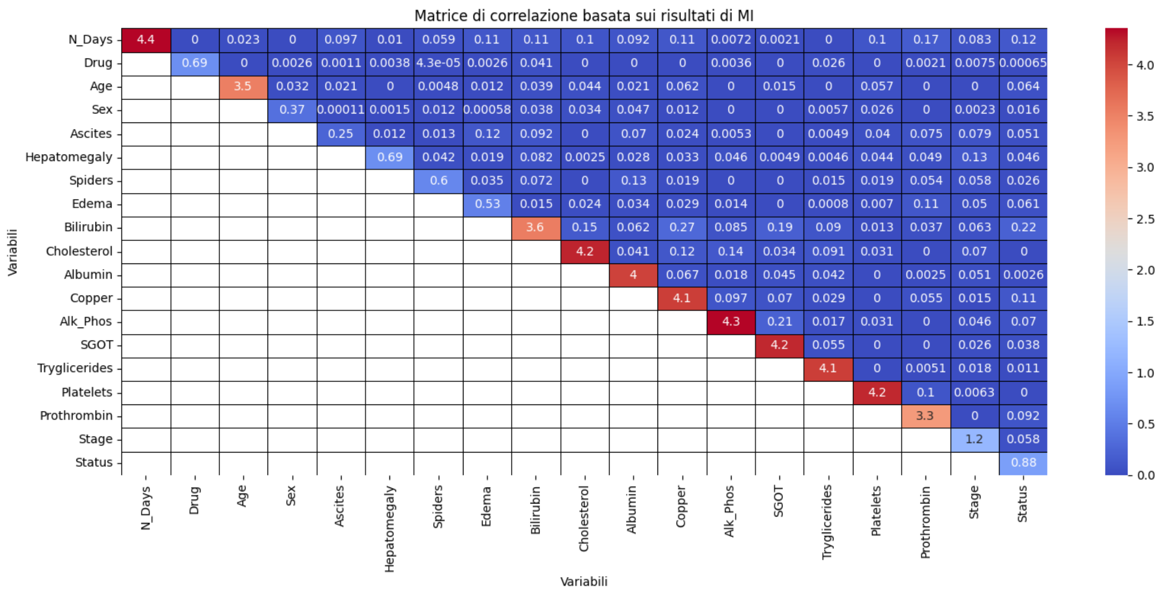


Figure 1: Heatmap of the mutual information between features.

one can see that there is **no strong positive/negative correlation** between them. For this reason, we cannot drop features based on their correlation, as no pairs contain the same type of information.

Figure 2 demonstrates the **imbalanced distribution** of the dataset labels.

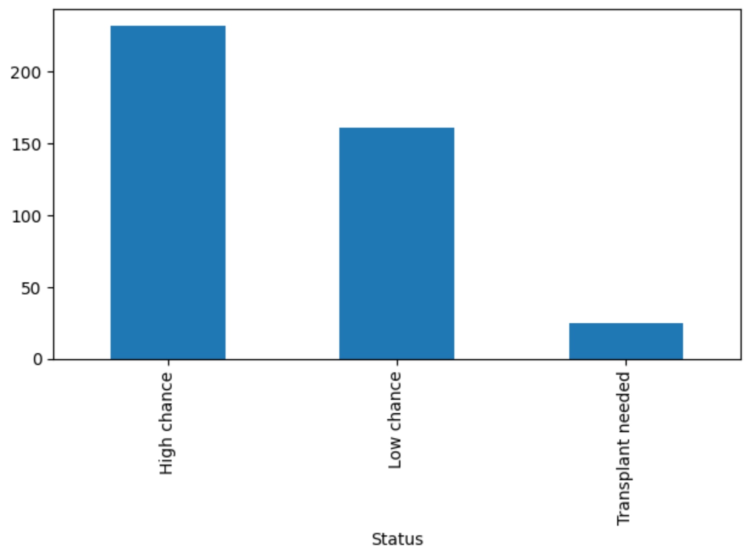


Figure 2: Distribution of patient statuses.

This is an important information not only for the **correct implementation of the pipelines** (imputing missing values, re-balancing), but also to decide **how we will evaluate their performances**, in order to select the best at predicting the survival chance of the patient.

4 Preprocessing

The Train set showed lots of ***NaN* values**, but due to the low total number of observations, removing all of them was not a possible option in order to **maintain a substantial number of information**.

To replace *NaN* values in numerical features with plausible values, a **CustomKNNImputer** with **different values of *k_neighbours*** was used. The imputing was done also by dividing patients **by label**, in order to create new values only considering **neighbours of the same class** (this can lead to more separate classes and better classification. Indeed by doing this we assure that imputed values are **drawn from the same distribution of the other instances in the class**, reducing the risk of **bias** introduced by imputing based on the **majority** class).

For categorical features, in order to replace *NaN* values with plausible values, a **CustomSimpleImputer** with "mode" (reported in python as *most_frequent*) was used. Also in this case, patients where divided by label.

Before deciding for the best **scaler**, it's important to analyze the **boxplots** (and so the distribution) of our numerical features one by one:

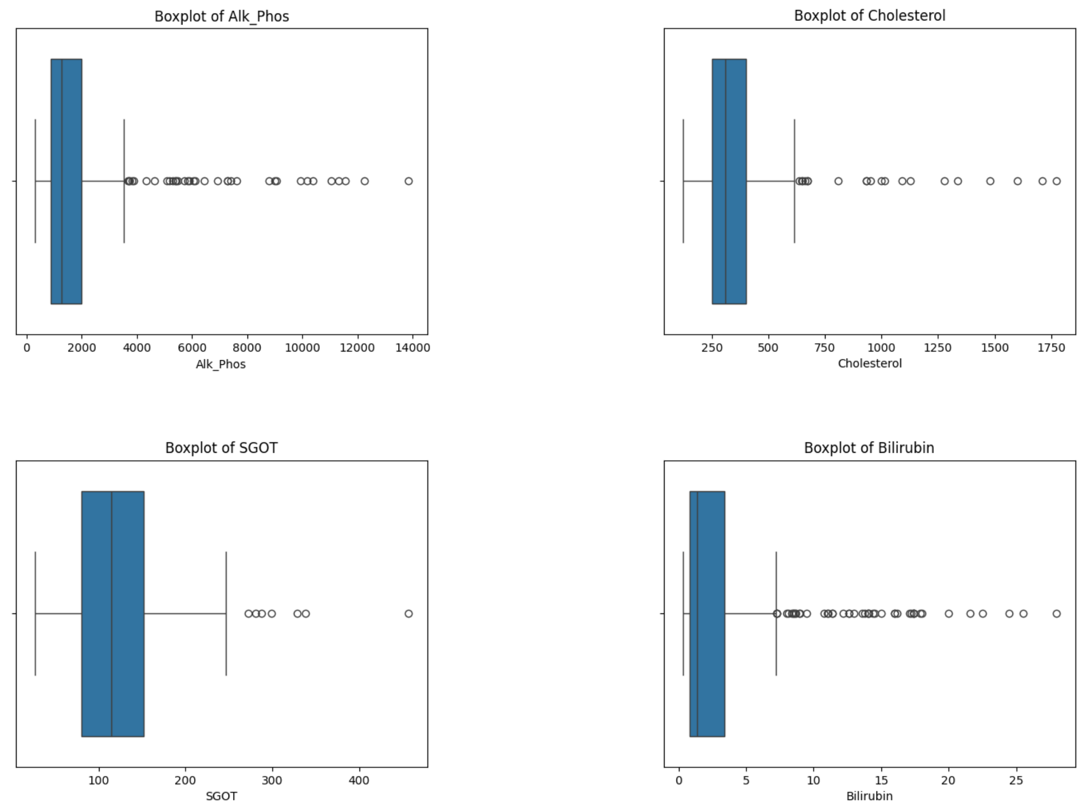


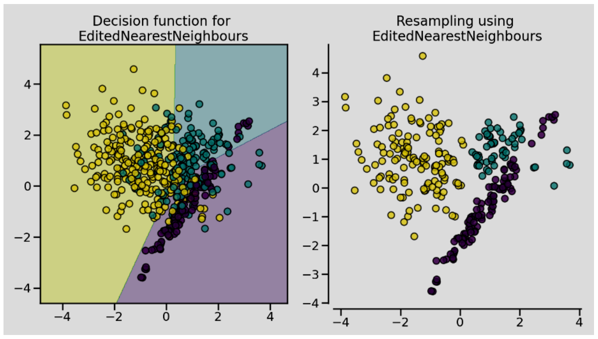
Figure 3: Boxplots of some numerical features as examples.

Given the relevant presence of outliers, the best choice is to apply a **RobustScaler**.

For the feature reduction, **normalized mutual information** was used, applied to a **SelectKBest** strategy with various values of K.

To balance instances, a double balancing technique was used. The undersampling technique was the **EditedNearestNeighbour**, with sampling strategy to decrease the number of instances in the "High chance" class.

This method was used since it can not only decrease the number of instances, but it's also capable of **separating** classes in the process, by deleting instances that may be close while being in different classes.

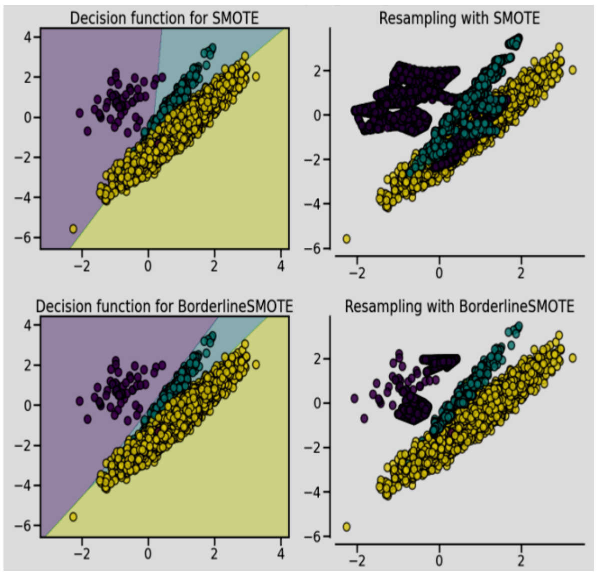


This was done with the goal of deleting instances of the High chance class that may make more difficult to classify the others.

The only problem with this approach is that there is no way to control **how many instances are deleted** (due to how it's implemented), differently from the other approach presented for the oversampling.

Oversampling was done in order to increase the number of instances in the Transplant needed class. The technique utilized was **SMOTE-ENC** [3](SMOTE for numerical and categorical), where after the encoding there is the option to over sample with SMOTE or **BorderlineSMOTE**, that can increase the chance of classifying instances which are considered hard to classify due to their position (close to the border of other classes).

SMOTE-ENC, differently from SMOTEN and SMOTE-NC, can handle continuous and categorical **simultaneously**, while SMOTEN and SMOTE-NC cannot.



All the methods were also used with different values of *n_neighbours*.

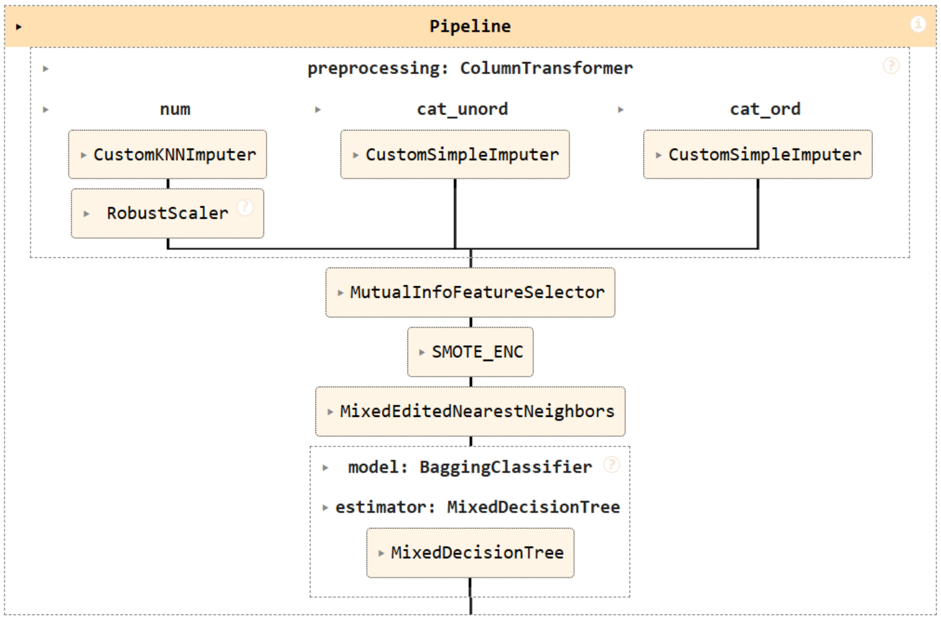
Overall, the goal of the whole sampling procedure was to **decrease the number of instances of the majority class down to the closest class in terms of instances, then increasing the lowest up to the closest**.

e.g.) Suppose:

- Starting labels: Low chance = 100, High chance = 200 and Transplant needed = 20;
- Final result: 100 for all the classes.

This decision was driven by the fact that relying solely on oversampling could have led to **overfitting**, as generating too many synthetic values may cause the model to learn the noise in the data rather than the **underlying patterns**.

The overall pipeline is displayed below.



5 Models Definition and Evaluation

In order to conduct the experiments, four different models where used:

- Decision Tree with Bagging;
- KNNeighbour;
- AdaBoost with naive Bayes classifier.
- Neural Network

All the model parameters where optimized using the **Randomized Cross Validation** method.

It was also important to select the scoring metric to determine the best classifier, and the decision was to choose the classifier that improved the *f1_macro_average* score. This was done to **penalize** classifiers that did not achieve good performances in classifying the Transplant needed class, which is actually the hardest given the very low number of instances, while also being crucial since a liver transplant is a very complex procedure, it is highly regulated, and only performed at designated transplant medical centers by highly trained transplant physicians and supporting medical team.

The most challenging decision was determining how many **folds** were needed to gather a sufficient amount of data to perform statistically valid analysis on the results. While most tests may be inefficient with a small number of values, a confusion matrix with too few instances may be inadequate for comparing performances.

The chosen method was the **Kruskall-Vallis** test (combined with the **Wilcoxon** test if statistically relevant differences were found between groups), which is a **non-parametric** test, that can be used to compare three or more independent distributions, and it has as an assumption, as mentioned in the sklearn page, to have **5 or more values**, together with independence and equal variances (in order to avoid staying "at the border" of the assumption, **10 folds** were selected).

Furthermore, in order to convert categorical features to numerical ones (Neural Networks accept only numbers), **OneHotEncoding** and **OrdinalEncoding** were used.

For the OneHotEncoding strategy, "Drug" was converted in "Drug_D-penicillamine" and "Drug_Placebo", while "Sex" became "Sex_M" and "Sex_F".

For the remaining categorical features, an OrdinalEncoder for each feature is used, with sorting strategy that represents with high values **serious conditions** (the higher the value, the worse the condition).

5.1 Validation of the Assumptions

Both the test mentioned before were used since the Q-Q plot Figure 4 of the scores showed that the distributions are **not normally distributed**, so non-parametric tests were selected.

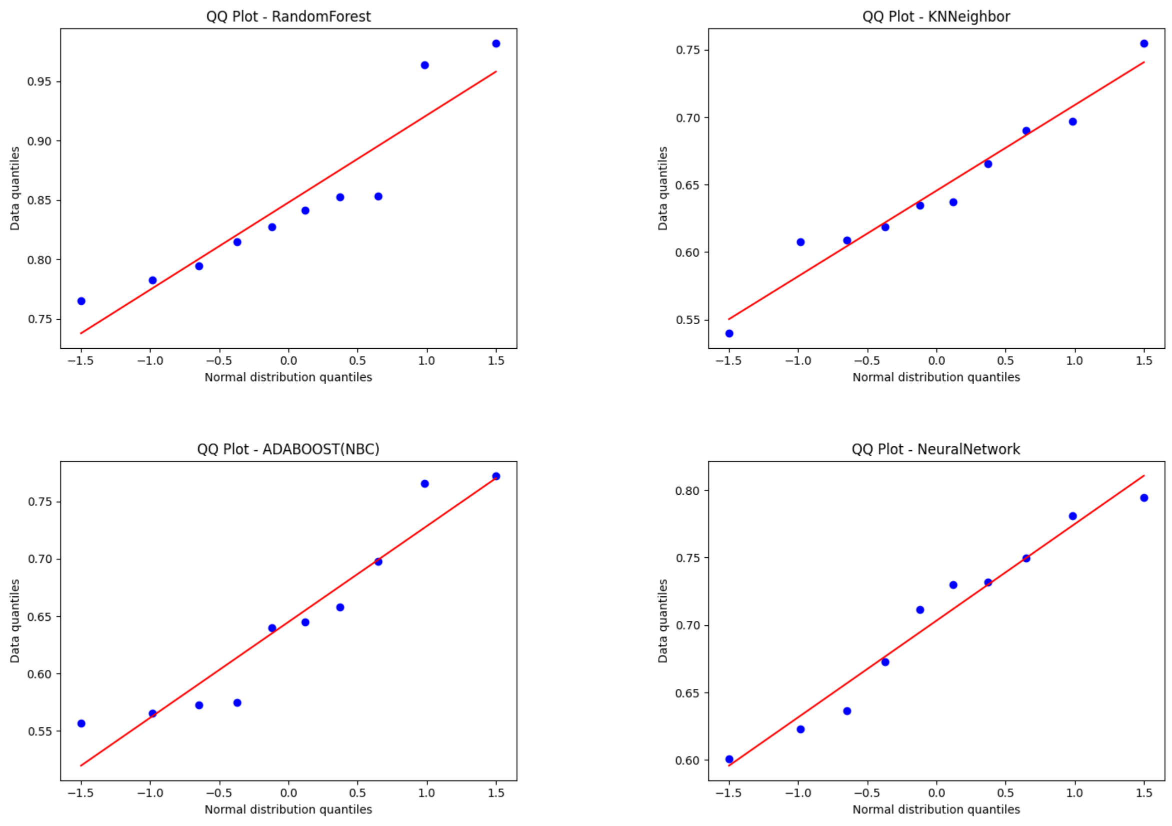


Figure 4: Q-Q plots for the different models.

After seeing this, the equal variances hypothesis can be proved by the **Levene's test**:

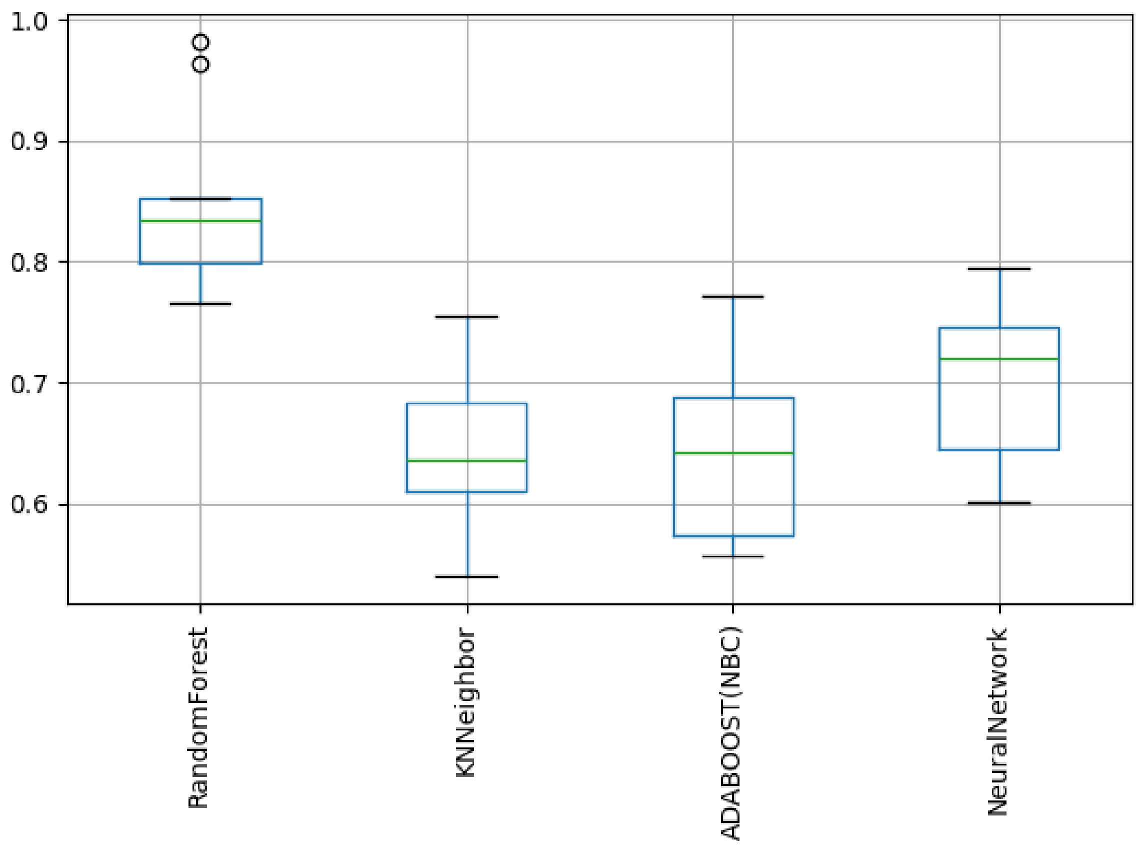
```
Levene's statistic: 0.3345409486210277
p value: 0.8004176795500553
There is no evidence to reject the null hypothesis of equal variances.
```

6 Final Results

The results of the pipelines showed that overall the RandomForest approach **performed better** (all the results in the appendix).

The following are the best parameters for both the pipelines:

```
{
    'undersampler__k_neighbors': 9,
    'preprocessing__num__imputer__n_neighbors': 8,
    'oversampler__k_neighbors': 5,
    'oversampler__borderline': False,
    'model__estimator__max_depth': 5,
    'model__estimator__criterion': 'gini',
    'feature_selector__k': 12
}
```



7 Appendix

```
Kruskal-Wallis statistic: 22.780975609756126
p value: 4.485831149755783e-05
There is a statistically relevant difference between at least two groups.
Comparing RandomForest and KNNeighbor
Wilcoxon statistic: 0.0
p value: 0.001953125
There is a statistically relevant difference between RandomForest and KNNeighbor
----
Comparing RandomForest and ADABOOST(NBC)
Wilcoxon statistic: 0.0
p value: 0.001953125
There is a statistically relevant difference between RandomForest and ADABOOST(NBC)
----
Comparing RandomForest and NeuralNetwork
Wilcoxon statistic: 0.0
p value: 0.001953125
There is a statistically relevant difference between RandomForest and NeuralNetwork
----
Comparing KNNeighbor and ADABOOST(NBC)
Wilcoxon statistic: 24.0
p value: 0.76953125
There is no statistically relevant difference between KNNeighbor and ADABOOST(NBC)
----
Comparing KNNeighbor and NeuralNetwork
Wilcoxon statistic: 7.0
p value: 0.037109375
There is a statistically relevant difference between KNNeighbor and NeuralNetwork
----
Comparing ADABOOST(NBC) and NeuralNetwork
Wilcoxon statistic: 12.0
p value: 0.130859375
There is no statistically relevant difference between ADABOOST(NBC) and NeuralNetwork
----
```

8 References

1. Medical News Today, "Cirrhosis of the liver: Life expectancy," https://www.medicalnewstoday.com/articles/cirrhosis-of-the-liver-life-expectancy
2. Cirrhosis Prediction Dataset https://www.kaggle.com/datasets/fedesoriano/cirrhosis-prediction-dataset
3. SMOTE-ENC algorithm https://arxiv.org/abs/2103.07612
4. Pipeline Code https://github.com/cletopellegrino/Mixed-Data-Pipeline