Predicting Cirrhosis of the Liver using Machine Learning Techniques

Daniel G. Barstow
Department of Mathematics,
The University of Illinois at Urbana-Champaign
Dr. Bruce Reznick

April 26, 2022

Abstract

The liver is an essential organ of the human body. It filters out toxins, aids in maintenance of the metabolism, and helps break down food in the digestive tract by producing bile. Cirrhosis is a late-stage scarring of the liver caused by conditions like hepatitis, chronic alcoholism, and obesity. The goal of this project was to successfully classify and determine the most important factors contributing to cirrhosis using machine learning techniques. The data was acquired from Kaggle's online machine learning library, containing 18 clinical features (sample size of $n = 418$). Descriptive statistics and visualizations were used to determine existing relationships in the data and select features for the models. Using Python and its associated libraries, five different machine learning models were trained ($n_{\text{train}}= 334$) and tested ($n_{\text{test}}= 84$) on the dataset. The models were subject to a 10-fold cross validation procedure to reduce overfitting, tune hyperparameters, and increase generalization potential. After the initial predictions were obtained, the training set was altered using SMOTE (Synthetic Minority Oversampling TEchnique) and new prediction accuracies were obtained.

The best performing model was a Naive Bayes Classifier with an accuracy of 84.52% on the test set. The remaining models, Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors achieved testing accuracies of 80.95%, 82.14%, 77.38%, and 80.95%, respectively. The attributes in the data were then analyzed for their contribution toward the Logistic Regression and Naive Bayes models using SHAP (SHapley Additive exPlanation) values. 9 features were determined to contribute to the Logistic Regression model predictions, and 11 to the Naive Bayes' predictions.

Background

Between 1974 and 1984, a trial was conducted by the Mayo Clinic to determine if the drug D-penicillamine was effective in treating liver disease. Ultimately, the drug was found to only be effective in chronic liver patients who were not in need of a liver transplant. 312 patients participated in the randomized trial of the drug and 112 only consented to have their measurements recorded on an ongoing basis, of which 6 eventually opted out (thus are not present in the dataset). In total there are 418 samples of patient data from blood tests and consultations with hepatologists.

- `ID` : The case number.
- `N_Days` : The number of days between registration and the earlier of death, liver transplantation, or study analysis time in July, 1986.
- `Status` : Whether the patient was still alive at the time of the statistical analysis in 1986.
- `Drug` : Whether the patient was given the drug D-penicillamine or a placebo in the trial.
- `Age` : The patient's age in days.
- `Sex` : Whether the patient is male or female.
- `Ascites` : Presence of ascites which is a build up of fluid in the abdomen.
- `Hepatomegaly` : Presence of an enlarged liver.
- `Spiders` : Presence of dilated blood vessels which have a spider-like appearance.
- `Edema` : Presence of tissue swelling (typically in the feet or ankles).
- `Bilirubin` : Amount of Bilirubin present in blood measured in mg/dl (breaks down heme from old red blood cells).
- `Cholesterol` : Amount of cholesterol present in blood measured in mg/dl (provides structure to cell membranes).
- `Albumin` : Amount of albumin present in blood measured in mg/dl (keeps fluid from leaking into bloodstream).
- `Copper` : Amount of copper present in urine measured in μg/day (regulates nerve cells and immune system).
- `Alk_Phos` : Amount of Alkaline phosphatase present in blood measured in IU/L (an enzyme which can leak into the bloodstream and cause issues).
- `SGOT` : Amount of SGOT (also called AST) present in blood measured in IU/L (another enzyme which can leak into the bloodstream and cause damage).
- `Triglycerides` : Amount of triglycerides in blood measured in mg/dl (a type of fat which if too high can become an issue).
- `Platelets` : Number of platelets per-cubic-millimeter of blood / 1000 (helps blood to clot).
- `Prothrombin` : Time in seconds required for blood clots to begin forming.
- `Stage` : Stage of liver disease (stage 4 is Cirrhosis).

Figure 1: A description of the twenty clinical features measured in the original study.

A brief description of the clinical features measured in the study are shown in fig. 1. Five years after the conclusion of the study in 1989, Dickson, Fleming, et al. created the Mayo-Model, which was used to predict the survival rate among patients with advanced liver disease [3]. This paper was important because it was the first time a model had been created to predict the survival of liver disease without requiring a biopsy, only a blood test. Later that year, Markus, Dickson, et al. used the surviving population from the original study to conduct another, assessing the efficacy of liver transplantation in patients with advanced cirrhosis [4]. They had to be clever about this though because they could not ethically implement a control group for a transplant. What they did instead was use their previous model to simulate control group data. Liver transplantation was eventually determined to be effective in this case and researchers started looking elsewhere for data as it became more readily available in the following decades.

Two datasets that have risen to popularity in this area are the Indian Liver Patient (ILP) dataset and the Hepatitis C Virus (HCV) dataset which are available on the University of California Irvine's machine learning repository. The former was studied by Joloudari et al. in [1], using a Particle-Swarm Support Vector Machine (PSO-SVM), achieving a remarkable 95.17% prediction rate. It was also studied by Abdar et al. in [2], using a Multilayer Perceptron and Boosted Decision Tree, achieving an accuracy of 87.91%. The latter was studied by Hoffman et al. in [5], using a Decision Tree model, obtaining an accuracy of 83.6%. It was also studied by Safardi et al. in [6], using many techniques, but the highest reported model and accuracy were a Random Forest model with an astonishing prediction rate of 99.8%.

One suggestion from other researchers was modifying the training set using a technique called Synthetic Minority Oversampling Technique (SMOTE) to address imbalances in the target prediction class. Another suggestion was using a Support Vector Machine (SVM) with hyperparameter tuning to increase prediction accuracy. A third suggestion was to impute missing data with a measure of central tendency to minimize the effect of the missing data. And a final suggestion

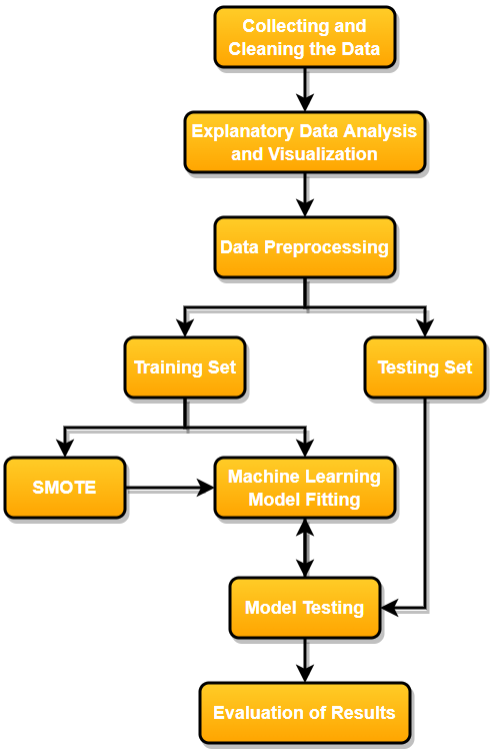

Figure 2: The steps followed in this project.

was to analyze feature importance in terms of contribution toward the model predictions using SHapley Additive exPlanation (SHAP) values. All of these suggestions were taken into account and a treatment of these concepts will be given later on in the methodology and results sections. A flowchart outlining the steps of this project is shown in fig. 2.

Methodology

Data Collection and Cleaning

The data was collected from [7] and contained 418 patient samples with 20 features for each patient. The 'ID', 'N_days, 'Status', and 'Drug' variables were dropped from the dataset because they either were determined have no predictive value, or would introduce data leakage into the models. The units of the 'Age' variable were also changed from days to years. One issue with the dataset was that it was missing some values, as shown in fig. 3.

Since the size of the dataset was relatively small, it was determined that simply leaving out the samples with missing data would impact the predictions too much. So, the missing data in the categorical variables such as 'Ascites', 'Hepatomegaly', and 'Spiders' was imputed with the mode of those features. And the missing data in the numerical variables such as 'Cholesterol', 'Copper', and 'Alk_Phos' was imputed with the median of those features. This was done to preserve the original measures of central tendency in the existing data without needing to discard entire samples. Further, the 'Edema' variable originally had three classifications: 'N' (no Edema), 'S' (Edema present without diuretics, or resolved by diuretics), and 'Y' (Edema presents despite treatment

```
In [10]: data.isna().sum()

Out[10]: Age              0
         Sex              0
         Ascites        106
         Hepatomegaly   106
         Spiders        106
         Edema            0
         Bilirubin        0
         Cholesterol    134
         Albumin          0
         Copper         108
         Alk_Phos       106
         SGOT           106
         Trygliceride   136
         Platelets       11
         Prothrombin      2
         Stage            6
         dtype: int64
```

Figure 3: Determining the number of missing features from the dataset.

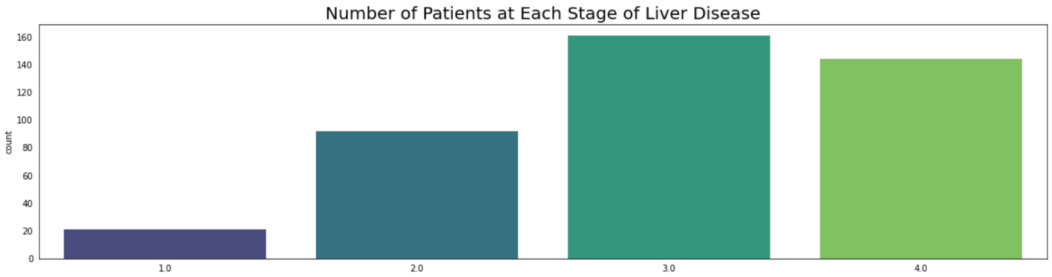

Figure 4: Original breakdown of 'Stage' variable.

with diuretics). In this case, 'S' was reclassified as 'Y', due to 'Edema' still being present, and also to make the binary classification slightly easier.

In the original dataset, the 'Stage' variable was broken up into Stage 1, Stage 2, Stage 3, and Stage 4, as shown in fig. 4. Since the goal of this project was to use binary classification to correctly identify Cirrhosis ('Stage' = 4), the first 3 stages were reclassified as 'Stage' = 0, and stage 4 was reclassified as 'Stage' = 1, as shown in fig. 5. After the reclassification, it was noted that the target prediction class was imbalanced toward 'Stage' = 0. This caused the models later on to have a better prediction rate for 'Stage' = 0 because of the relative availability of the data. This issue was addressed after the first model fitting procedure with SMOTE, although it still hindered prediction accuracy overall.

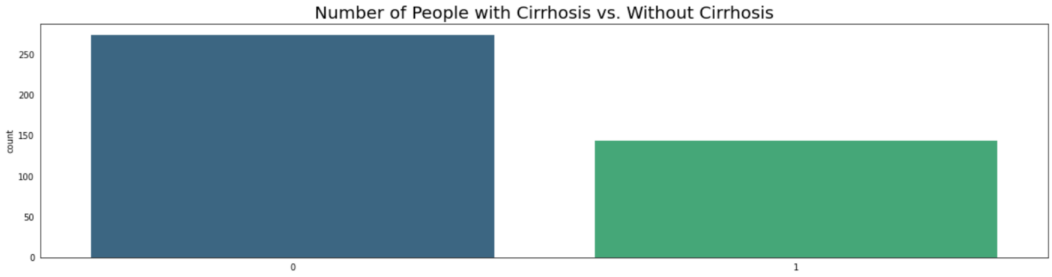

Figure 5: Reclassification of 'Stage' variable.

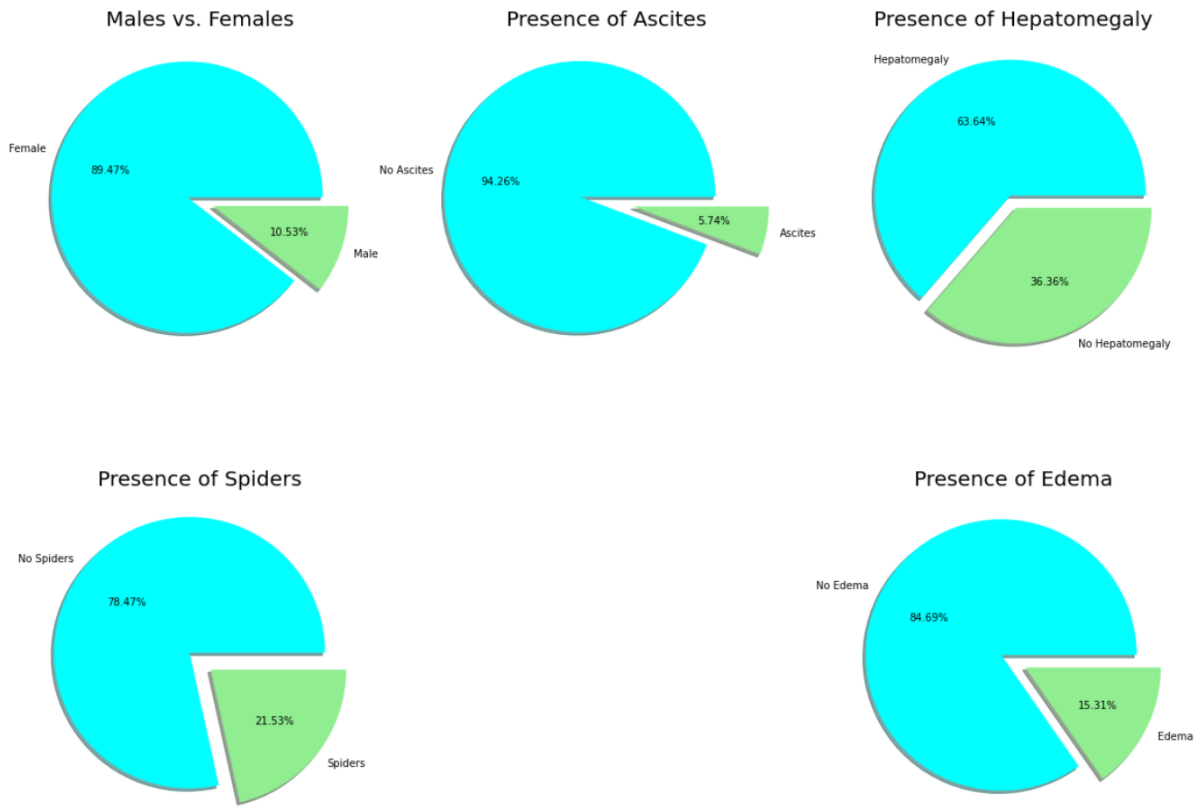

Figure 6: Presence of binary features in the dataset.

Explanatory Data Analysis and Visualization

A breakdown of the 5 binary features is shown in fig. 6. Recall that 106 values were imputed for 'Ascites', 'Hepatomegaly', and 'Spiders'. A few things can be noted from the graph in fig. 6. There were many more females than males in the dataset. The presence of 'Spiders', 'Edema', and 'Ascites' were quite low. And two thirds of the patients exhibited 'Hepatomegaly'.

The distributions of 10 numerical attributes were plotted and are shown in fig. 7. Recall that 134 values were imputed for 'Cholesterol', 108 values for 'Copper', 106 values for 'Alk_Phos' and 'SGOT', 136 values for 'Triglycerides', 11 values for 'Platelets', and 2 values for 'Prothrombin'. In general, normal distributions are observed, with some large peaks near the mean, indicative of the imputation process, although no large skewing is apparent. Patient age was also normally distributed with a mean of about 50 years.

The features were then analyzed against the target predictor, 'Stage', to see if there were any relationships between them. Bar graphs of the 5 binary features are shown in fig. 8. From these graphs, it was noted that presence of 'Ascites' is slightly more common in patients with cirrhosis. Presence of 'Spiders' and 'Edema' are slightly more common in patients with cirrhosis. And non-presence of 'Hepatomegaly' is slightly more common in patients with cirrhosis.

Distribution plots of the numerical features vs. 'Stage' are shown in fig. 9. A number of insights can be discerned from these plots. Lower values of 'Cholesterol', 'Bilirubin', 'Triglyc-

erides‘, ‘Copper‘, ‘Platelets‘, ‘Albumin‘ and ‘SGOT‘ are more common in patients with cirrhosis. And higher values for ‘Age‘ and ‘Prothrombin‘ are more common in patients with cirrhosis.

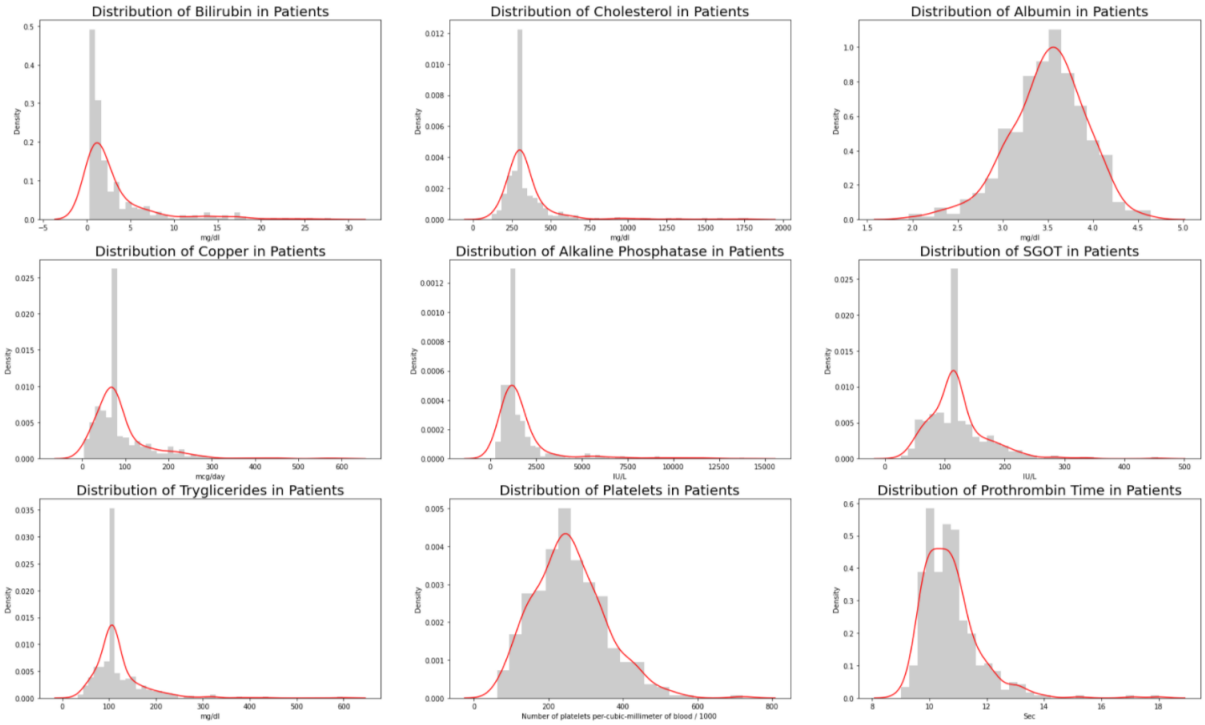

Figure 7: Distributions of the numerical features in the dataset.

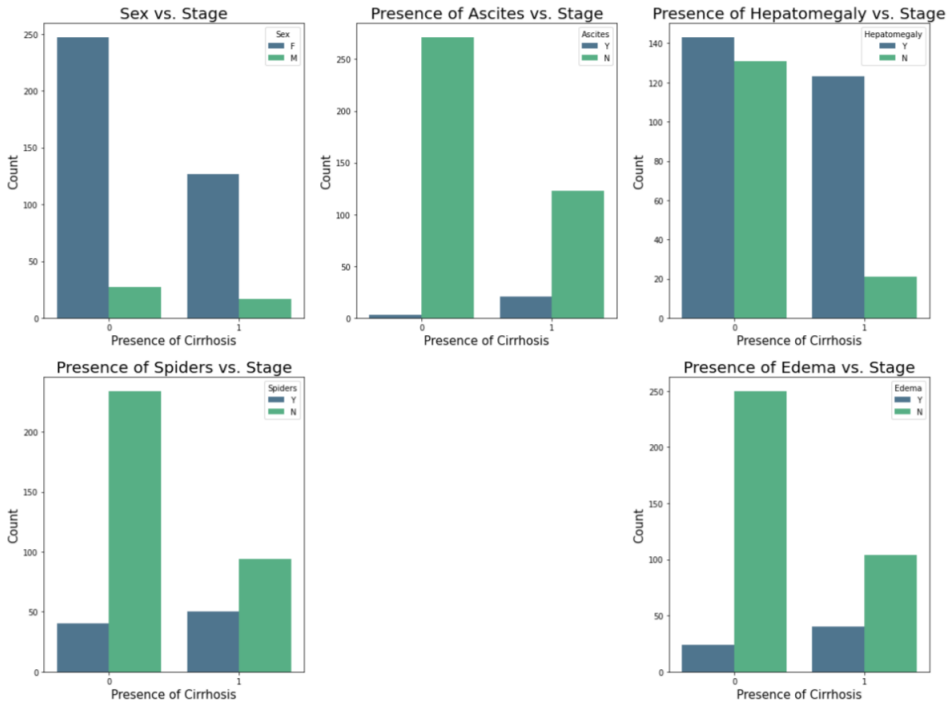

Figure 8: Bar plots of the binary features vs. the target ‘Stage‘.

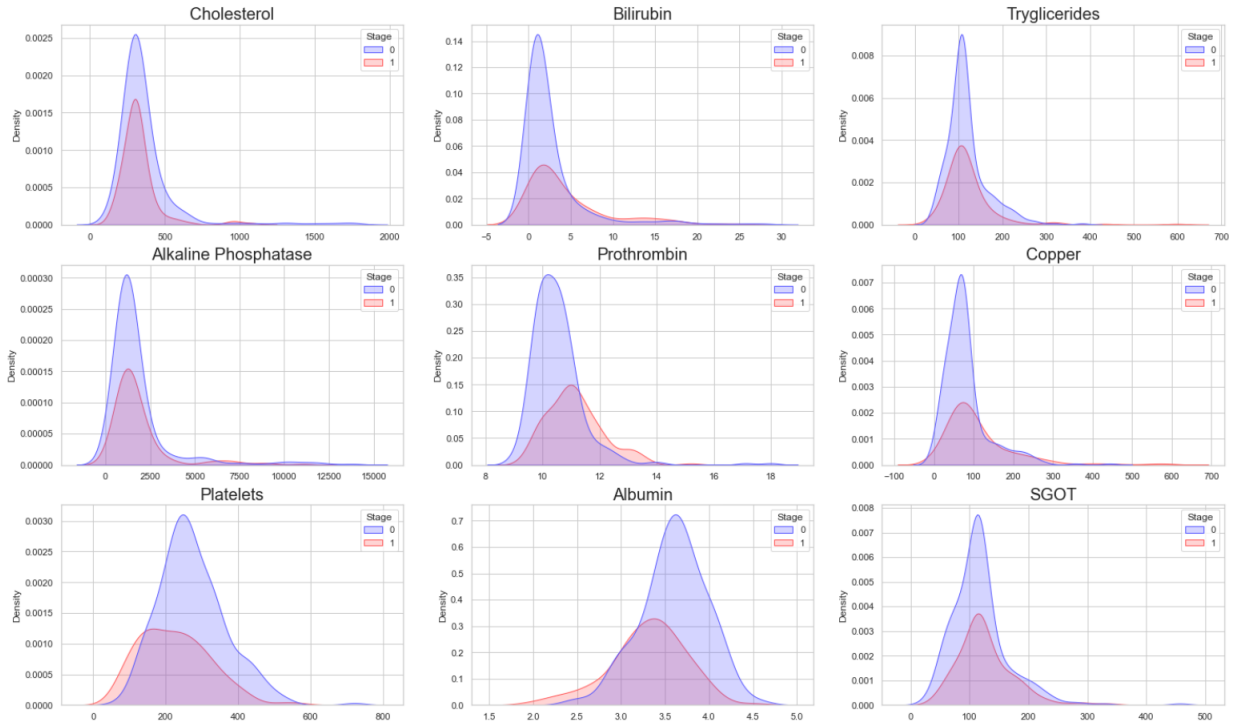

Figure 9: Distributions of the numerical features vs. the target 'Stage'.

Data Preprocessing and Feature Selection

At this point, the categorical features in the dataset were transformed from 'N' and 'Y' to 0 and 1, respectively so they would work with each machine learning model. Then the correlation matrix was plotted to determine the correlation between features in the dataset, as shown in fig. 10. An arbitrary threshold of 0.08 was chosen to select features for the machine learning models. So, values having a correlation with 'Stage' above 0.08 were used in the training set to fit the machine learning models. The 12 features selected along with their correlation with 'Stage' are shown in fig. 11. The data was then standardized to have a mean = 0 and variance = 1. This was done to prevent features with wider ranges from distorting the predictions and is common practice. Finally, the data was randomly split into a training ($n_{\text{train}} = 334$) and a testing ($n_{\text{test}} = 84$) set.

SMOTE was a technique first described by Chawla et al in [9], and was an attempt to address the issue of class imbalance in datasets, such as the one in fig. 5. The basic idea is to create synthetic data of the target class by slightly changing the values of the features, according to the 5 (arbitrary) nearest neighbors in the feature space. Chawla writes, "Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general" [9]. In this project, two versions of the training set were created to compare the results of the models trained on the data with SMOTE, and without SMOTE.

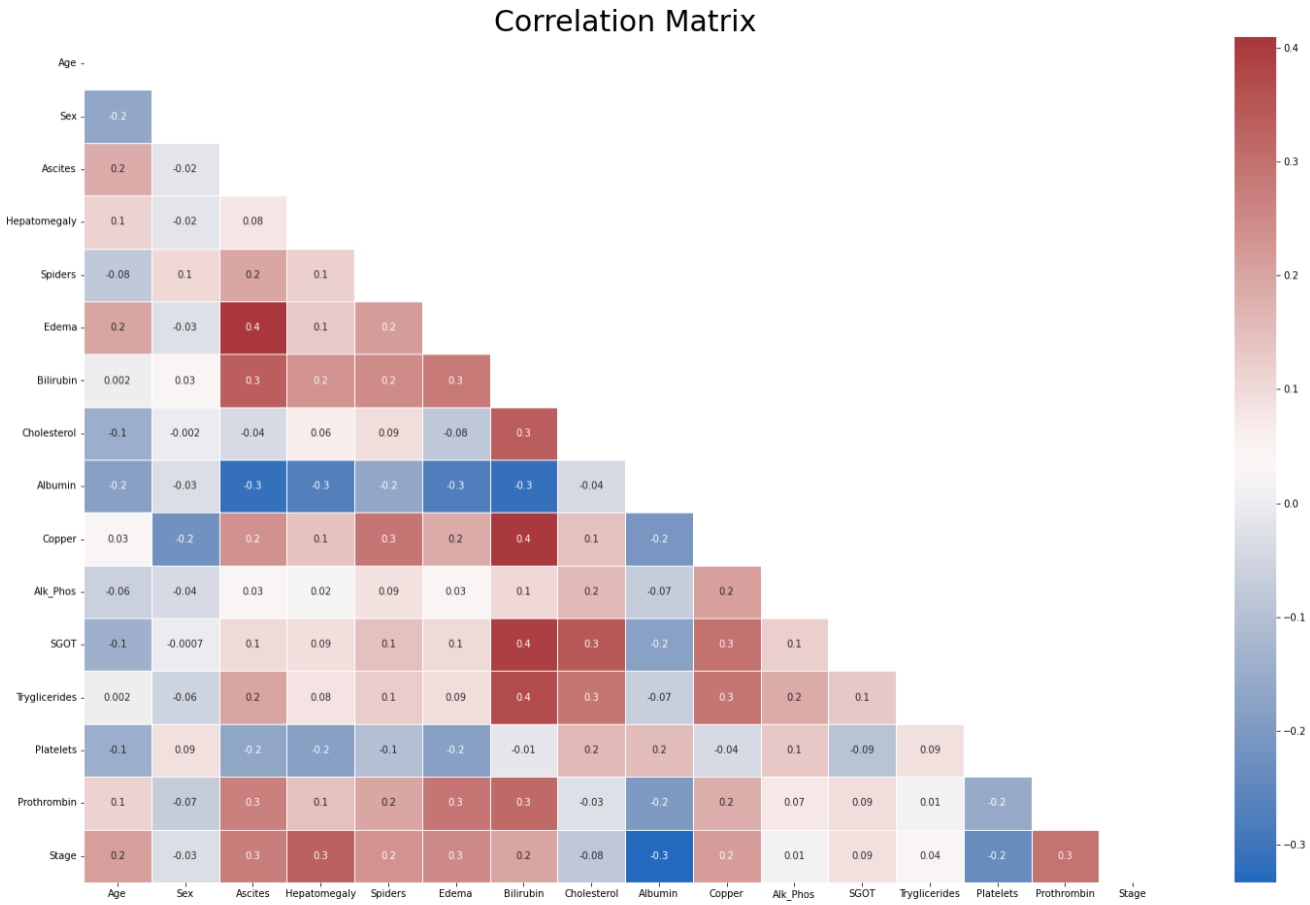

Figure 10: Correlation matrix between all features in the dataset.

Albumin	0.333060
Hepatomegaly	0.328234
Prothrombin	0.293754
Ascites	0.275527
Edema	0.250983
Platelets	0.236666
Spiders	0.232651
Copper	0.217266
Age	0.210092
Bilirubin	0.198645
SGOT	0.090118
Cholesterol	0.083042

Figure 11: The 12 features selected for the machine learning models and correlation with the target 'Stage'.

Machine Learning Models for Binary Classification

5 machine learning models were first trained on the data without SMOTE and tested, then trained on the data with SMOTE and again tested. The models used for this binary classification task were: a Logistic Regression, Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbors (KNN), and Naive Bayes. The Python library scikit-learn was used to create, train, and test the models. The mathematical treatment of these models is adapted from [8], and will follow the conventions and style therein.

Logistic Regression

Logistic Regression is a modification of Linear Regression, which is a technique commonly used in statistics to find a line of best-fit for continuous data to make predictions. Given n independent, identically distributed samples with k features, $\{(x_i, y_i) : x_i \in \mathbb{R}^k, y_i \in \mathbb{R}, i \leq n\}$, the output of the Linear Regression model $\hat{y}$ is a weighted sum of the inputs, $\hat{y_i} = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}$ for $i \leq n$, $j \leq k$. The goal is to find coefficients β, which minimize the squared error between the true values y_i and the predicted values $\hat{y_i}$. Equation (1) represents the optimization problem to solve in Linear Regression.

$$\min_{\beta} \ \text{error}(\beta) = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \ \ \text{where} \ \ \hat{y_i} = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \ \forall i \leq n, j \leq k. \tag{1}$$

While there exists a closed form solution for this problem, it isn't always the best solution. For datasets with high correlation between features, this model performs poorly unless it is penalized in some way. Equation (2) is an alteration of the Linear Regression problem with an added penalty term to the error function, $\lambda \sum_{j=1}^{k} \beta_j^2$. This term artificially increases the error in the predictions by a factor of λ to penalize values of β which are significantly larger than others, leading to more accurate predictions. Notice that in the case where $\lambda = 0$, (2) simplifies to (1).

$$\min_{\beta} \ \text{error}(\beta) = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 + \lambda \sum_{j=1}^{k} \beta_j^2, \ \text{s.t.} \ \lambda \geq 0 \ \ \text{where} \ \ \hat{y_i} = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \ \forall i \leq n, j \leq k. \tag{2}$$

While Linear Regression works great for finding best-fit lines and predicting continuous variables, it can't be used for classification problems. The logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is useful because it maps values between 0 and 1, so it can be used to model the probability that an input belongs to a certain class. If an arbitrary cutoff value is chosen (typically 0.5), then the output of the Logistic Regression can be compared against the cutoff, and can be used for classification.

Given some patient data x_i, we want to model the conditional probability that the patient belongs to the class 'Stage' = 1. Suppose this probability is given by $P(Y = 1|X = x_i) = p(x_i; \beta)$ for $i \leq n$. Then let the logit transform of $p(x; \beta)$ be given by $\ln(\frac{p(x)}{1-p(x)}) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}$ for $i \leq n, j \leq k$. Solving for p, we arrive at the Logistic Regression model, $p(x) = \frac{1}{1+e^{-(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}}$ for $i \leq n, j \leq k$.

When trained on the data, $p(x)$ is compared against the threshold, and an output of 0 or 1 is given. The problem now becomes estimating the parameters β of the model which give the maximum likelihood of observing the existing data. In a sequence of Bernoulli trials with probability of success p, the maximum likelihood of observing the existing data is given by (3). In practice, the negative log-likelihood (sometimes called 'log-loss') is used, which is given by (4). Note that for Logistic Regression, $y_i \in \{0, 1\}$.

$$\max_{\beta} \ L(\beta) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} \tag{3}$$

$$\min_{\beta} \ -\ln(L) = -\sum_{i=1}^{n} [y_i \ln(p(x_i)) + (1 - y_i) \ln(1 - p(x_i))]. \tag{4}$$

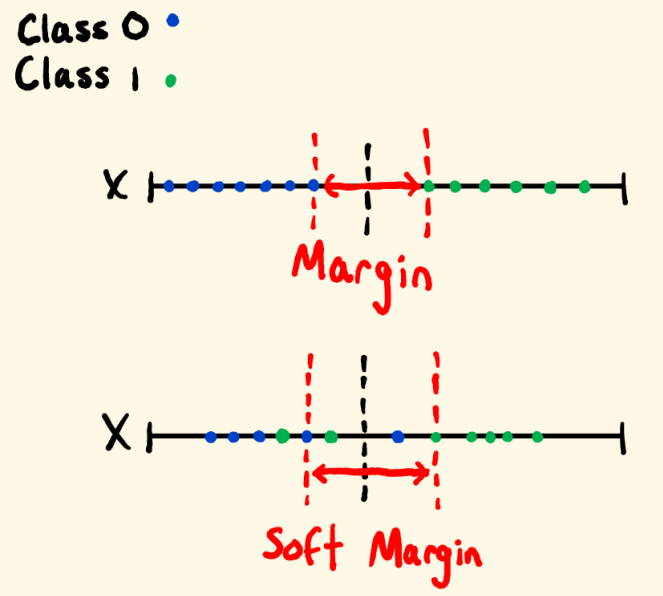

Figure 12: A 1-dimensional Support Vector Classifier with a hard margin and a soft margin, which allows for some misclassification.

Prediction accuracy can be increased with the addition of a penalty term. The full optimization problem for Logistic Regression is given by (5). Since this is a transcendental function, the solution is found by numerically using solvers integrated into the scikit-learn Python library. A value for λ is found via cross-validation on the training set.

For $i \leq n, j \leq k$ and $p(x_i) = \frac{1}{1+e^{-(\beta_0+\sum_{j=1}^{k}\beta_j x_{ij})}}$,

$$\min_{\beta} \; -\ln(L) = -\sum_{i=1}^{n}[y_i \ln(p(x_i)) + (1-y_i)\ln(1-p(x_i))] + \lambda \sum_{j=1}^{k} \beta_j^2 \text{ s.t. } \lambda \geq 0. \tag{5}$$

Support Vector Machine

Logistic Regression assumes that log-likelihood is a linear function of the input variables, and attempts to model the probability that a sample $x_i \in \mathbb{R}^k$ belongs to a certain class, however this assumption is not always true and can lead to poor classification performance. Support Vector Machines are an extension of Support Vector Classifiers (SVC), which plot every sample in a k-dimensional vector space, looking for a hyperplane which explicitly separates the samples by target class in $(k-1)$ dimensions, maximizing the distance between the decision boundary and the closest points of each class. These distances are called *margins* and the closest samples are called *support vectors*. This margin can be modified (or "softened") to allow for a certain number of misclassifications to maintain overall prediction accuracy. An example of a SVC is shown in 1 dimension in fig. 12.

The problems with SVC are that there will always be many hyperplanes which satisfy these constraints, and that the samples in k dimensions are not guaranteed to be linearly separable. In the event that a boundary cannot be found because of too much overlap, a trick can be employed which maps the samples to a higher dimensional space, increasing the computational complexity of

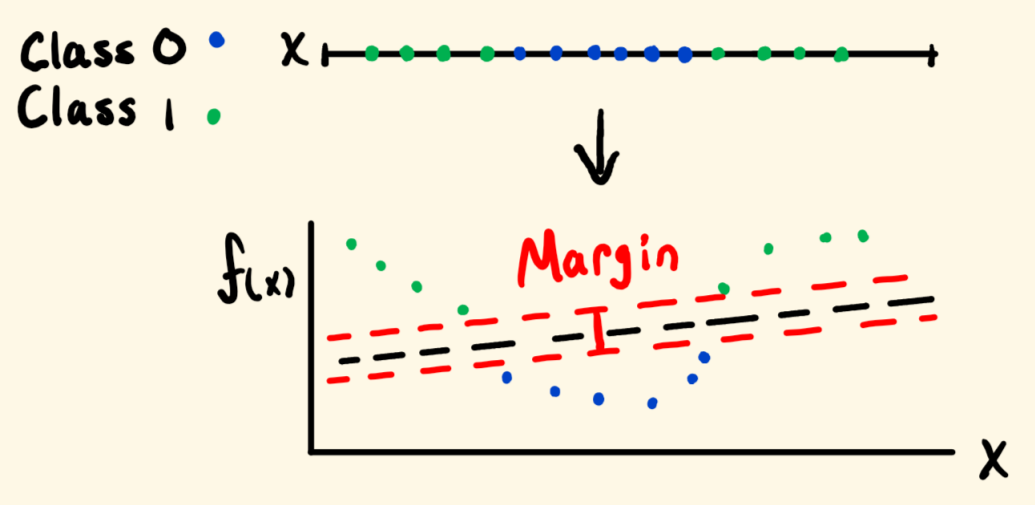

Figure 13: A case where SVC is useless in 1 dimension, so the data is mapped to 2 dimensions where a decision boundary can be found. Note that $f(x)$ need not be linear.

the problem, but making finding the decision boundary easier. This trick is called finding a *kernel function* and a visualization is shown in fig. 13. This is the key idea behind SVM.

Given n samples $\{(x_i, y_i) : x_i \in \mathbb{R}^k, y_i \in \{-1, 1\}, i \leq n\}$, define a hyperplane given by $\{x : f(x) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} = 0\}$ for a unit vector $\beta : ||\beta|| = 1$. Then define a rule for classification $G(x) = \text{sign}(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})$. It can be shown that f gives the signed distance from x_i to the hyperplane and G gives the classification of x_i. The goal is to find an f such that the hyperplane separates the samples while maximizing the margin M. The optimization problem for SVC with non-overlapping classes is given by (6). If there are overlapping classes, then slack variables λ can be introduced to the constraints, allowing for a softer margin with threshold C, as shown in (7). Note that $M = \frac{1}{||\beta||}$, in this case.

$$\max_{\beta} \; M(\beta) \;\; \text{s.t.} \;\; y_i(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}) \geq M, \;\; \text{for } i \leq n, \;\; ||\beta|| = 1. \tag{6}$$

$$\min \;\; ||\beta|| \;\; \text{s.t.} \;\; \begin{cases} y_i(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}) \geq M(1 - \lambda_i) \;\; \forall i \leq n, \\ \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i \leq C. \end{cases} \tag{7}$$

This is a quadratic convex optimization problem, so techniques like Lagrange multipliers can be used to transform it into a dual nonlinear program given by (8).

$$\begin{aligned} \max_{\beta} \; L_D &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'} \\ \text{subject to} \; \sum_{i=1}^{n} \alpha_i y_i x_i &= \beta, \\ \sum_{i=1}^{n} \alpha_i y_i &= 0, \\ C - \mu_i &= \alpha_i, \; \forall i \leq n, \\ 0 \leq \alpha_i &\leq C. \end{aligned} \tag{8}$$

SVM goes one step further and maps each x_i to a higher dimensional space by a function h so that the solution f is in terms of inner products of h, $f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle h(x), h(x_i) \rangle$, $0 < \alpha_i < C$. The nice thing about this is that we need not know what h is, only what the kernel function is $K(x, x') = \langle h(x), h(x') \rangle$. In practice, K is required to be a symmetric, positive (semi-) definite function, and is usually chosen to be either a d^{th}-degree polynomial, $K(x, x') = (1 + \langle x, x' \rangle)^d$, or a radial basis, $K(x, x') = e^{-\gamma ||x - x'||^2}$. With a linear kernel, smaller values of the parameter C lead to a smoother decision boundary in the high-dimensional feature space. With a radial basis kernel, γ determines the curvature of the Gaussian decision boundary. These parameters are found via cross-validation the training set.

Decision Tree

Decision Trees can be computationally simpler than SVM (in the greedy case) and have the advantage of being more interpretable. Decision Trees consist of *nodes* representing the features of the input data and are recursively split according to a predetermined rule and threshold. Each node contains a proportion of the samples which follow that decision path in the tree. The algorithm greedily looks for splits which minimize the *impurity*, or inhomogeneity in the final classifications. The number of layers in the tree is called the *depth* and the terminal nodes are called *leaves*, which give the output for the classifications. The depth and the number of leaves and splits are found via cross-validation on the training set. A Decision Tree model used in this project is shown in fig. 14.

Given N samples with k features $\{(x_i, y_i) : x_i \in \mathbb{R}^k, y_i \in \{0, 1\}, i \leq N\}$, and a partition of the feature space $X = \{x_i \in \mathbb{R}^k, i \leq n\}$ into M regions $R_1, R_2, ..., R_M$. In a given node m representing region R_m with N_m samples, let the proportion of samples belonging to class $c \in \{0, 1\}$ be given by $\hat{p}_{mc} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = c)$, where I is an indicator function. Classify the samples in leaf t by the majority class given by $\hat{c}(t) = \text{argmax}_c \, \hat{p}_{tc}$. Finally, let the impurity of node m be given by the Gini index $Q_m = \sum_{c=0}^{1} \hat{p}_{mc}(1 - \hat{p}_{mc})$.

The goal is to delete (or *prune*) leaves from the initial tree T_0 while greedily minimizing the impurity Q_m of the classifications. For a *subtree* $T \subset T_0$ with $|T|$ leaves with cost of pruning $C_\alpha(T)$, the optimization problem to solve is given by (9). The parameter α determines the trade-

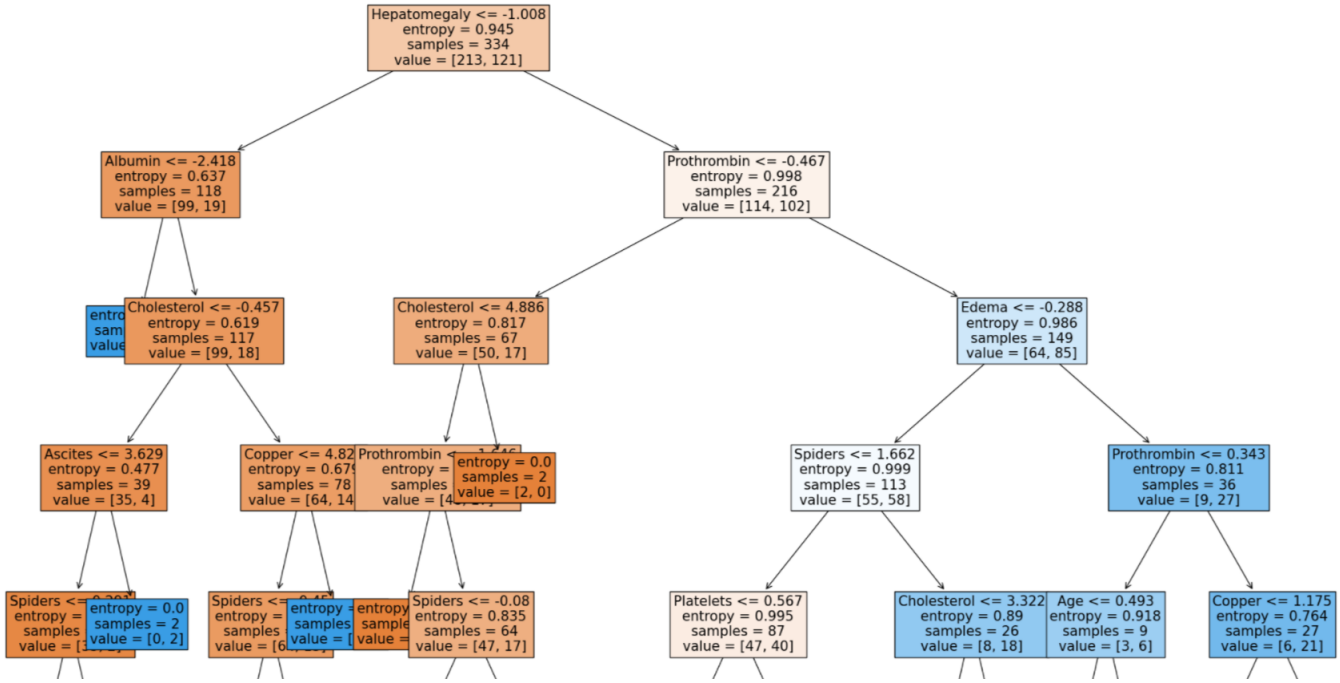

Figure 14: The first Decision Tree model used in this project. The samples are first split by 'Hepatomegaly', then by 'Albumin' and 'Prothrombin', and so on. The 'value' variable in the tree counts the number samples in each class.

off between tree size and prediction accuracy, and is also found via cross-validation.

$$\min_T \; C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m + \alpha|T| \;\; \text{s.t. } \alpha \geq 0. \tag{9}$$

K-Nearest-Neighbors

The KNN algorithm is rather simple. Given n samples $\{(x_i, y_i) : x_i \in \mathbb{R}^m, y_i \in \{0, 1\}, i \leq n\}$, define the *Minkowski distance of order p* between points in $\mathbb{R}^m$ by $D_p(x_i, x_{i0}) = (\sum_{i=1}^m |x_i - x_{i0}|^p)^{\frac{1}{p}}$. Then any new sample $x_0 \in \mathbb{R}^m$ is classified as the most common among its k-nearest-neighbors, according to the Minkowski distance of order p. In other words, if $K = \{(x_i, y_i) : i \leq k \leq n\}$ is the set of samples closest to x_0 for some order p, then $y_0 = \text{argmax}_{y_i} K$. The parameters k and p are found via cross-validation and p is typically either the *Manhattan distance* ($p = 1$), or *Euclidean distance* ($p = 2$). A simple example is shown in fig. 15.

Naive Bayes

Bayes' Theorem states that given some data $x_i, i \leq n$, the conditional probability that y belongs to a certain class is $P(y|x_1, x_2, ..., x_n) = \frac{P(y)P(x_1,x_2,...,x_n)}{P(x_1,x_2,...,x_n)}$. The "naive" assumption of Naive Bayes is that each feature is independent, so $P(y|x_1, x_2, ..., x_n) = \frac{P(y)\prod_{i=1}^n P(x_i|y)}{P(x_1,x_2,...,x_n)}$. This assumption does not always hold, nevertheless Naive Bayes tends to outperform other models as the feature space becomes very large. The complexity only scales linearly in the number of features, so it is also computationally inexpensive. Since $P(x_1, x_2, ..., x_n)$ is constant (i.e. the training data doesn't change), the goal is to maximize the likelihood of simultaneously observing the data and y belonging to a

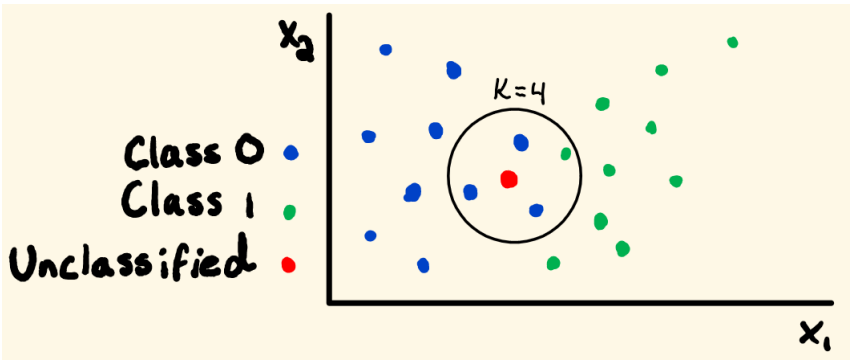

Figure 15: A KNN classifier where $k = 4$. The new point is classified as '0'.

particular class. The optimization problem to for Naive Bayes is given by (10).

$$\text{argmax}_y \ \hat{y} = P(y) \prod_{i=1}^{n} P(x_i|y) \quad \text{for } i \leq n. \tag{10}$$

If we also assume that this conditional probability follows a normal distribution, then the problem is given by (11). $P(y)$ is estimated via cross-validation on the training set and the solution is again found using the scikit-learn library.

$$\text{argmax}_y \ \hat{y} = P(y) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{x_i^2}{2}} \quad \text{for } i \leq n. \tag{11}$$

Cross Validation

Cross validation is a procedure where training data is partitioned into subsets and multiple models are trained on a subset then compared to find the best one, shown in fig. 16. This process can also be used to tune *hyperparameters* and give good heuristics to the models and make better predictions. One common tactic is to define a grid with an arbitrary step value and iterate through each point along the grid, choosing hyperparameters which yield the best predictions over all subsets of the training data, as shown in fig. 17.

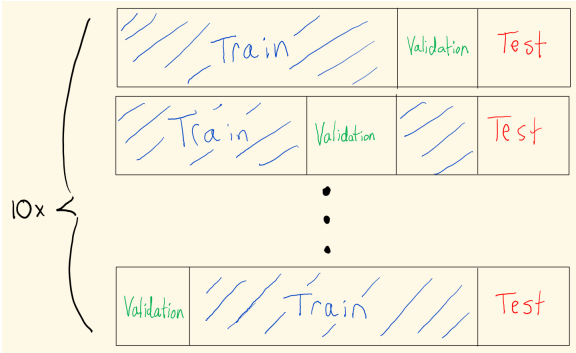

Figure 16: A 10x cross-validation over training data. Note that the testing data is only used at the end.

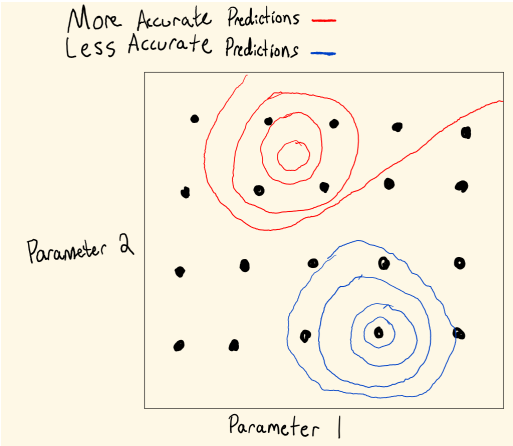

Figure 17: An exhaustive *grid search* for hyperparameter tuning.

Model	Test Accuracy (Before SMOTE)	Test Accuracy (After SMOTE)
Logistic Regression	80.95%	78.57%
Support Vector Machine	82.14%	65.48%
Decision Tree	77.38%	63.10%
K-Nearest Neighbors	80.95%	66.66%
Naive Bayes	84.52%	80.95%

Figure 18: Testing accuracy of each model before and after applying SMOTE. Note the general decrease in accuracy.

Results

After both rounds of fitting the training data with and without SMOTE, the results shown in fig. 18 were obtained. Across each model, we see a decrease in prediction accuracy after applying SMOTE. These accuracies don't tell us the full story, though. To get a better understanding of the idiosyncrasies of each model, we look at their *confusion matrices*. The confusion matrix of a binary classifier shows the number of *true positives, true negatives, false positives, and false negatives* made on the test set. This allows us to visualize the number of $Type - I$ and $Type - II$ errors. The confusion matrices of each model before and after applying SMOTE are shown in figures 19-23.

SHAP values are a concept from game theory discussed in [10]. To determine which features contribute most to the model predictions, SHAP values were calculated for the features in the Logistic Regression and Naive Bayes models before and after the application of SMOTE. The plots obtained are shown in figures 24-28. Note that brighter colors indicate more feature importance for a particular sample.

Discussion

There were many potential sources of error in this project. We imputed missing data for roughly a third of the dataset. We saw that SMOTE decreased the overall accuracy of the models, so it is possible that the SVM or KNN models overfit synthetic data. The fact that the dataset was

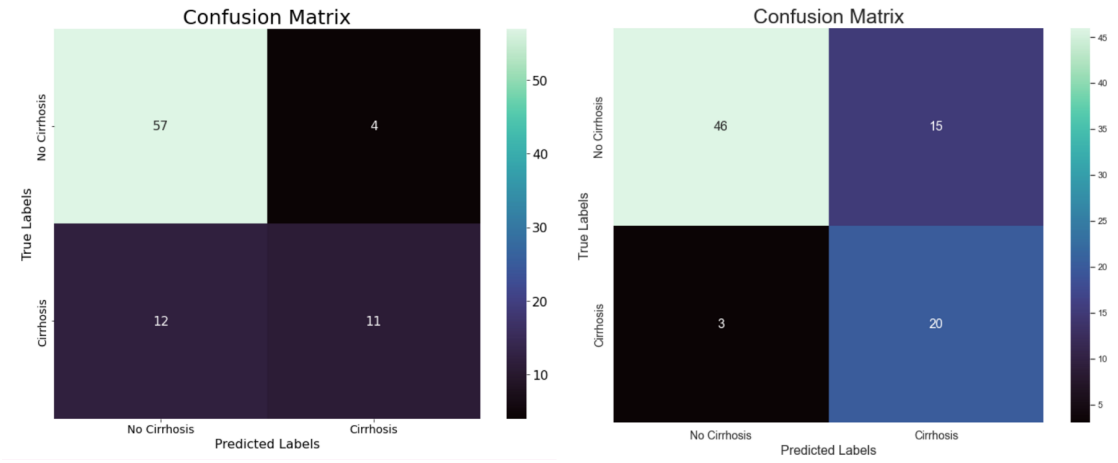

Figure 19: Confusion matrices for Logistic Regression before and after SMOTE. We see an increase in the true positive rate and a decrease the false negative rate, at the cost of higher false positive and lower true negative rate.

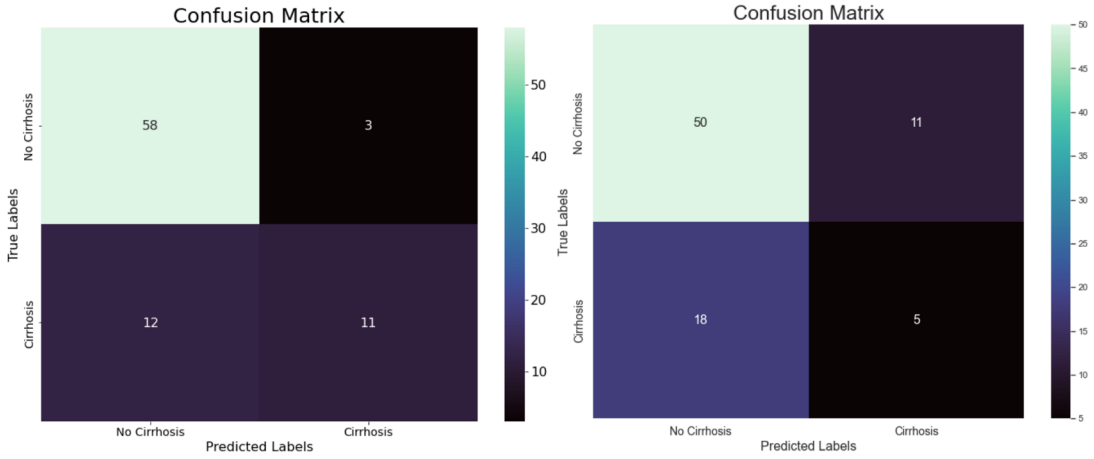

Figure 20: Confusion matrices for SVM before and after SMOTE. We see a decrease in performance by all metrics.

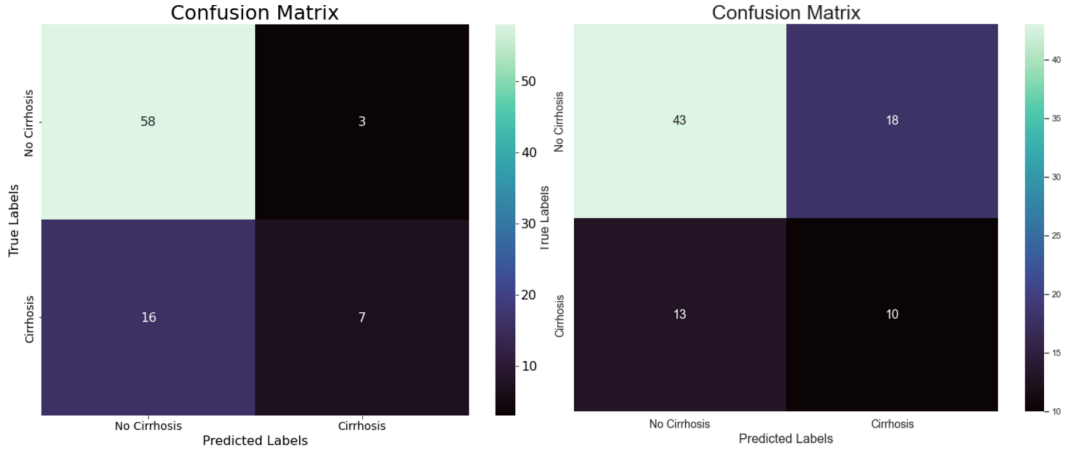

Figure 21: Confusion matrices for the Decision Tree before and after SMOTE. We see slight improvements in true negative and false negative rates, at a disproportionate cost of true positive and false positive rates.

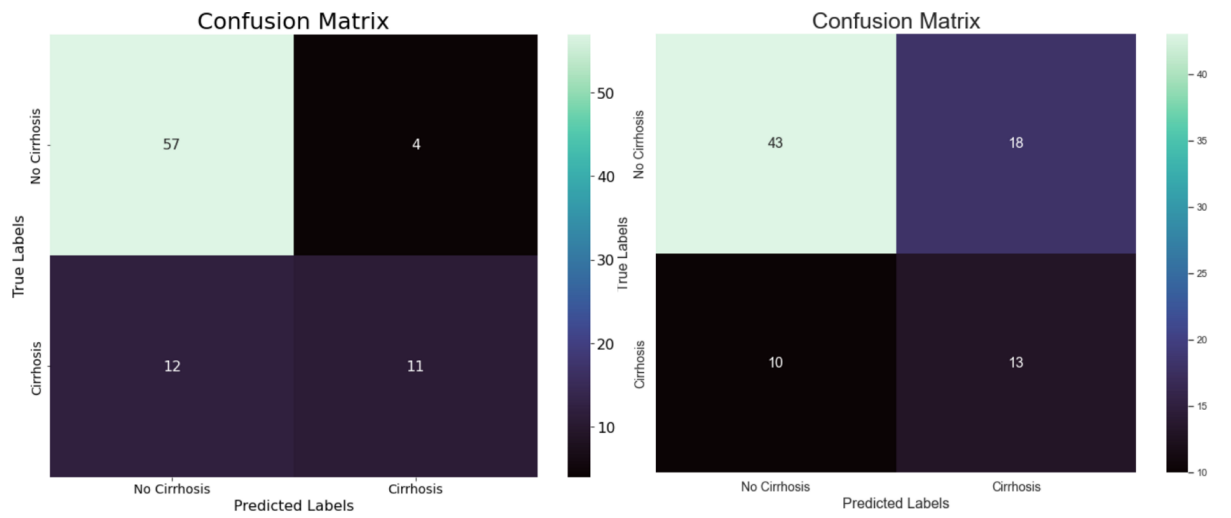

Figure 22: Confusion matrices for KNN before and after SMOTE. We see slight improvements with regard to true positive and false negative rates, at a disproportionate cost of a false positive and true negative rates.

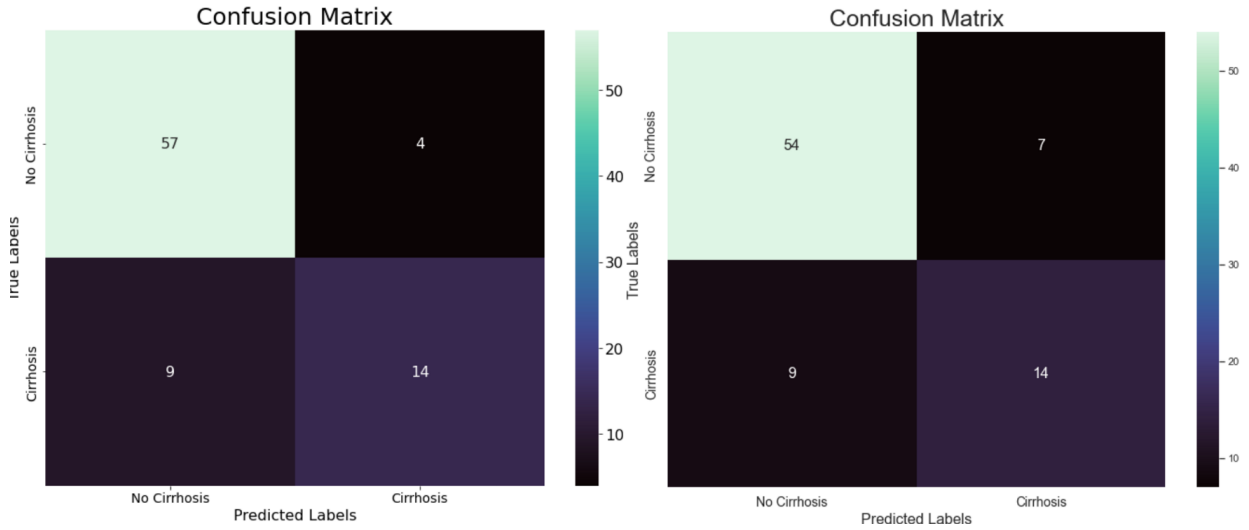

Figure 23: Confusion matrices for Naive Bayes before and after SMOTE. We only see a slight decrease in true negative rate, and increase in false positive rate. This model was the most robust toward SMOTE, maintaining most of its predictive capability.

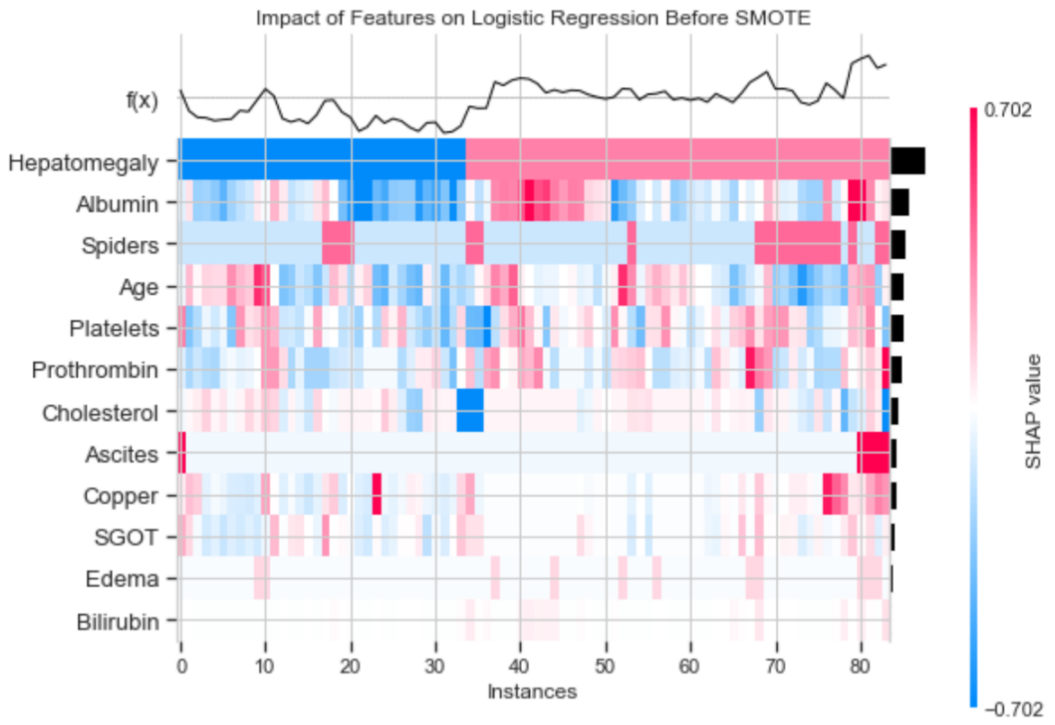

Figure 24: Impact of features on Logistic Regression model before SMOTE. Note the minimal contribution of 'SGOT', 'Edema', and 'Bilirubin', and the large magnitudes of the SHAP values on the right.

Figure 25: Impact of features on Logistic Regression model after SMOTE. 'Platelets', 'Prothrombin', 'Bilirubin', and 'Copper' increased in importance, while 'Age', 'Spiders', and 'Cholesterol' decreased in importance.

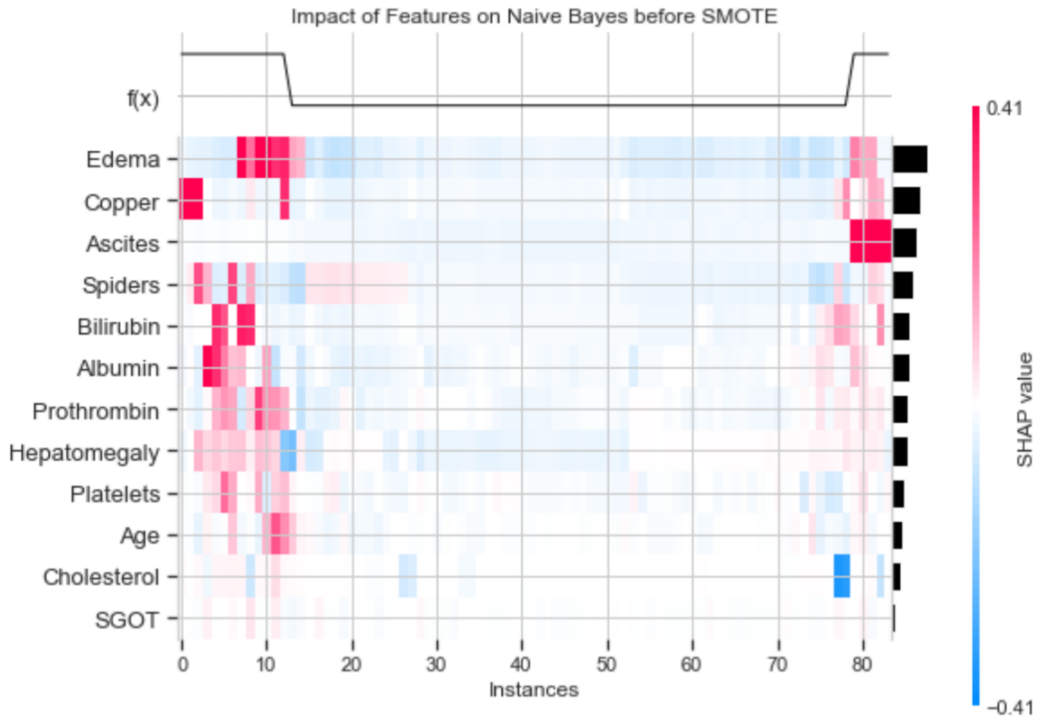

Figure 26: Impact of features on Naive Bayes model before SMOTE. Note that 'SGOT' barely contributes.

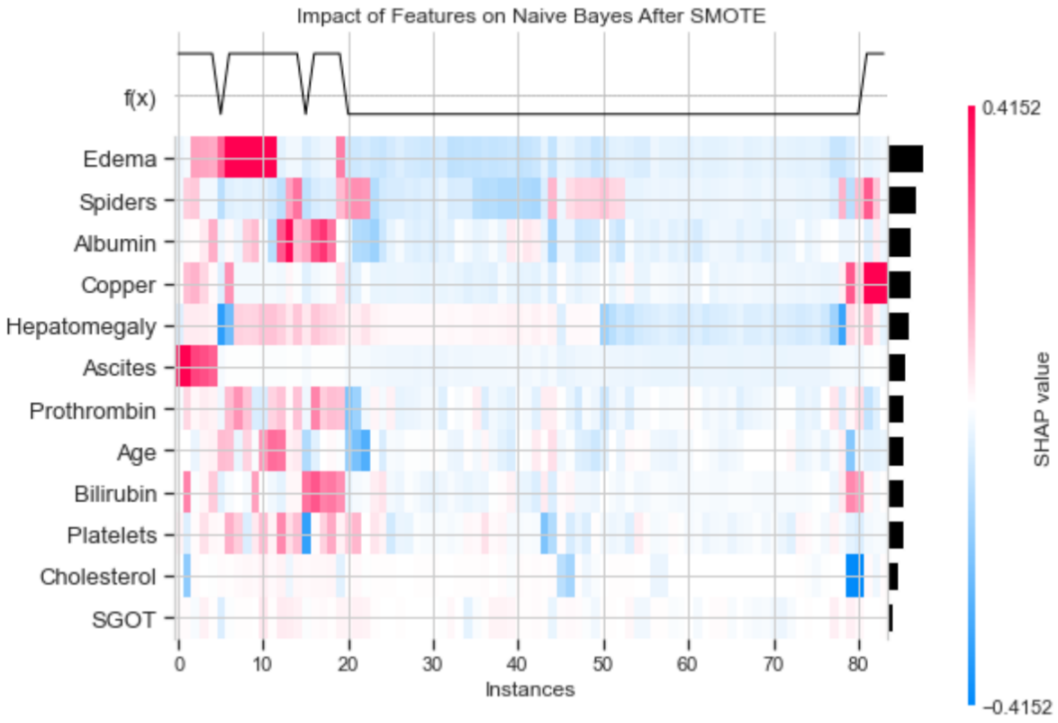

Figure 27: Impact of features on Naive Bayes model before SMOTE. 'Spiders', 'Albumin', 'Hepatomegaly', and 'Age' increased in importance, while 'Copper' and 'Ascites' decreased in importance.

imbalanced to begin with could have also affected the original choices of the models during cross-validation, leading to models with higher initial false negative rates. Since SVM is prone to the *curse of dimensionality*, it took a long time to train.

In a clinincal context, a sensible goal would be to minimize false positives. The SVM before SMOTE only had 3 false positives with a high overall accuracy on the original testing set, with some risk of a false negative. The Decision Tree before SMOTE also only had 3 false positives, but it had a higher false negative rate and lower true positive rate. SVM's are basically black boxes compared to Decision Trees, which are much more interpretable, so a hepatologist may opt for the Decision Tree due to being able to understand it better.

On the other hand, the best performing model overall was the Naive Bayes without any modification of the training set, with an accuracy of 84.52% on the test set. On the other other hand, the Logistic Regression (after modifying the training set with SMOTE) had the highest true positive rate, with fewer false negatives but more false positives. All these factors must be taken into account when using statistical tools to aid in the diagnosis of any disease. And of course, these techniques are no replacement for an actual doctor. The sample size of the dataset was only 418, so the results should be taken with a grain of salt. Nevertheless, 84.52% accuracy isn't too bad.

My motivation for this project was twofold. The first reason was because I wanted to have a better understanding of the tools I would use frequently as a Data Scientist, and to become more comfortable with programming these machine learning models. The second was because in December 2020, my dad passed away from a combination of cirrhosis and COVID-19, so this project felt personal, giving me more motivation to press forward in the face of challenges. If I had more time, I would have tried to use *principal component analysis* (PCA) or *linear discriminant analysis* (LDA) to reduce the dimensionality of the feature space so the SVM would take less time to train. I also would have looked into other classification metrics to further compare my models. Lastly, I would have tested my models on one of the other two datasets mentioned in the background section.

One thing I learned was that the flowchart in fig. 2 is somewhat farcical; it should be more of a cycle rather than a linear process, as there was a lot of going back and forth between training and testing phases (and sometimes all the way back to the cleaning step to account for some error further down the line). I also learned how not to become discouraged when my initial intuitive understanding doesn't quite match up perfectly with the mathematical underpinning (patience was important here). Lastly, I learned to manage my time better. Specifically, I got better at figuring out the point at which I'm getting diminishing marginal returns from the amount of effort I'm expending. I often found that going for a walk or talking with a friend for only 15 minutes would free up my mind again and allow me to think clearly, and be able to do more work than I would have been able to otherwise if I had just kept the grind going.

The Jupyter notebook with all the Python code and plots for this project can be found here.

References

[1] Javad Hassannataj Joloudari, Hamid Saadatfar, Abdollah Dehzangi, and Shahaboddin Shamshirband. Computer-aided decision-making for predicting liver disease using pso-based optimized svm with feature selection. *Informatics in medicine unlocked*, 17:100255, 2019.

[2] Moloud Abdar, Neil Yuwen Yen, and Jason Chi-Shun Hung. Improving the diagnosis of liver disease using multilayer perceptron neural network and boosted decision trees. *Journal of Medical and Biological Engineering*, 38(6):953–965, 2018.

[3] E Rolland Dickson, Patricia M Grambsch, Thomas R Fleming, Lloyd D Fisher, and Alice Langworthy. Prognosis in primary biliary cirrhosis: model for decision making. *Hepatology*, 10(1):1–7, 1989.

[4] Bernd H Markus, E Rolland Dickson, Patricia M Grambsch, Thomas R Fleming, Vincenzo Mazzaferro, Goran Bo G Klintmalm, Russell H Wiesner, David H Van Thiel, and Thomas E Starzl. Efficacy of liver transplantation in patients with primary biliary cirrhosis. *New England Journal of Medicine*, 320(26):1709–1713, 1989.

[5] Georg Hoffmann, Andreas Bietenbeck, Ralf Lichtinghagen, and Frank Klawonn. Using machine learning techniques to generate laboratory diagnostic pathways—a case study. *J Lab Precis Med*, 3:58, 2018.

[6] Reza Safdari, Amir Deghatipour, Marsa Gholamzadeh, and Keivan Maghooli. Applying data mining techniques to classify patients with suspected hepatitis c virus infection. *Intelligent Medicine*, 2022.

[7] fedesoriano. Cirrhosis prediction dataset. https://www.kaggle.com/datasets/fedesoriano/cirrhosis-prediction-dataset. Accessed March, 2022.

[8] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[10] Christoph Molnar. Shapley values. https://christophm.github.io/shapley. Accessed April, 2022.