Liver Disease Detection using Blood Attributes from Clinical Data

Akshit Chhabra
Brown University
Link to repository: https://github.com/akshitchhabra22/DATA1030Project

Introduction

Hepatitis C virus (HCV) affects more than 2.4 million people in the United States alone[1]. The current method of testing for whether a person has HCV is to use an anti-HCV antibody assay followed by confirmation using a more specific serologic test like recombinant immunoblot assay[2]. Both these tests must be prescribed by a physician but that is likely to occur when the patient presents with relevant symptoms. HCV remains asymptomatic until very long[2], and therefore, undiagnosed until very long. The medical community would greatly benefit from an easy preliminary method to see if a patient requires more specific HCV testing. The motivation for this project stems from the fact that blood tests are part of routine check-ups and if we could use attributes from those tests to predict if a patient was at risk of HCV, then that would not only automate and streamline the liver disease detection process, but also lead to better treatment outcomes for patients.

In this project, we use a clinical dataset of 615 patients obtained from the UCI Machine Learning repository[3]. The dataset contains basic patient information such as their sex and age, and then multiple blood attribute values: Albumin (ALB), Alkaline Phosphatase (ALP), Bilirubin (BIL), Choline Esterase (CHE), Gamma Glutamyl-Transferase (GGT), Aspartate Amino-Transferase (AST), Alanine Amino-Transferase (ALT), Creatinine (CREA), Protein (PROT), and Cholesterol (CHOL). Our aim was to solve a classification problem and the target variable "Category" initially had 5 classes: 0=Blood Donor, 0s=suspect Blood Donor, 1=Hepatitis, 2=Fibrosis, and 3=Cirrhosis signifying healthy patients and different stages of liver disease. Since the dataset was not large enough for a multi-classification problem (with some classes only having as little as 7 values), we combined four classes into one class signifying liver disease. The final two classes for the binary classification problem were: 0=Blood Donor, 1=Non-Blood Donor.

While there has been previous work conducted with the same dataset[4], they used various forms of imputation for missing values and achieved high F1 scores of 0.98. We believe that imputation should not be used to handle missing values in a critical healthcare dataset as even a slightly skewed result can prove detrimental. Therefore, in this project we worked our models around missing values using techniques like looping over various unique feature patterns and using packages like XGBoost. Thus, our F1 scores may not be comparable but might prove to be clinically more relevant.

Exploratory Data Analysis

We conducted extensive exploratory data analysis (EDA) to understand our dataset. As a first step, we used value_counts() to explore our categorical target variable. We visualized it using a horizontal bar plot (Fig. 1a). The data was slightly imbalanced with 533 patients being in class 0 and 82 patients being in class 1. The patient ID column was dropped since it was not relevant to the machine learning pipeline. We then used boxplots, violin plots, and category-specific histograms to visualize the numerical features (age and all blood attributes) against the target variable. We used stacked bar plots to visualize the categorical feature (sex). AST came out looking as an important feature, with the category-specific histogram showing clearly higher values of AST for patients with liver disease (Fig. 1b). Interestingly, sex and age did not seem to play a very important role (Figs. 1c and 1d).

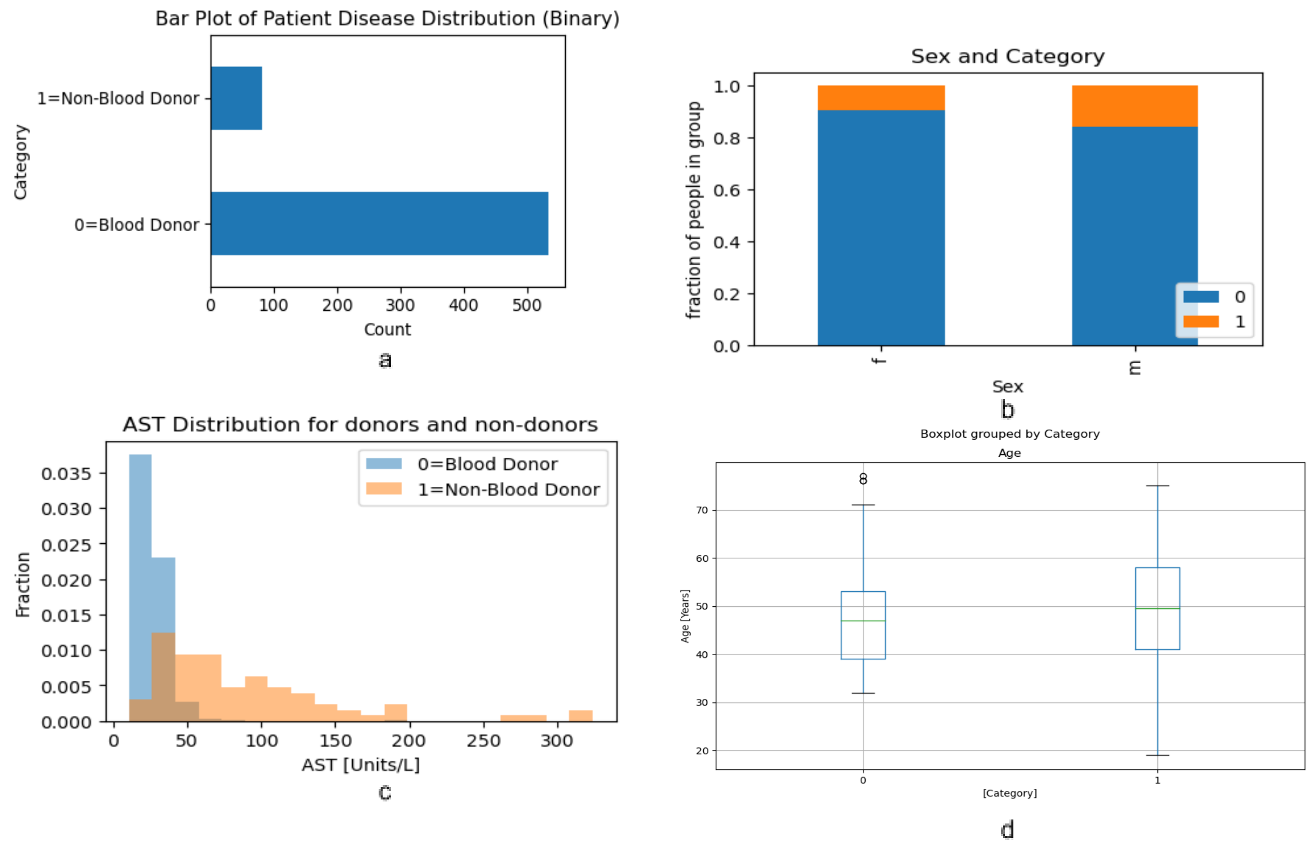


Figure 1: Interesting EDA findings. (a) The target variable is visualized, (b) category-specific histogram of AST attribute, (c) stacked bar plot of sex and the target variable, and (d) boxplot of age and target variable.

We also plotted a Pearson correlation matrix and none of the features had a very high correlation (Fig. 2). These were just a few of the interesting findings from our EDA. The rest can be found in the repository.

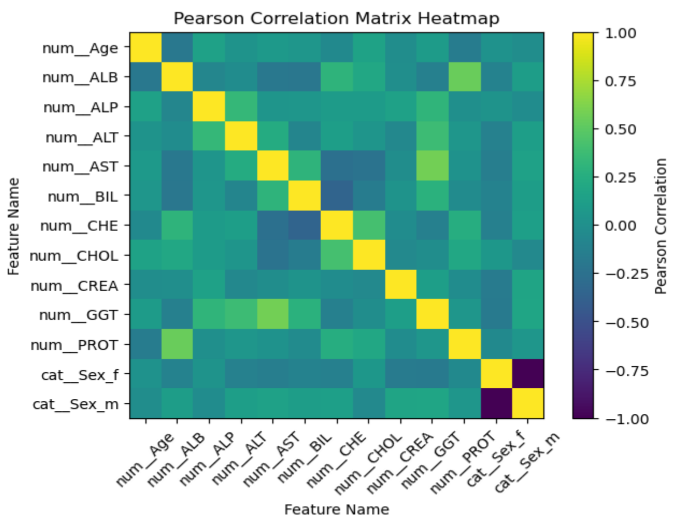


Figure 2: Pearson Correlation Matrix of features.

Methods

The data points were independent and identically distributed - each row corresponded to data from a single patient - and therefore, we could use a simple train_test_split to create standard 60:20:20 train:validation:test splits. For preprocessing, the categorical feature sex was one hot encoded and the rest of the numerical features were standard scaled. This increased one feature in the feature matrix. There were missing values in the data, specifically in the ALP, ALT, and CHOL features, amounting to a total of 4%. To account for these, we had to use special techniques. We trained an XGBoost model, which has the ability to work around these missing values. We also created functions to loop over various unique patterns of missing data, create smaller “reduced-features (RF)” datasets without the missing values, and trained models, finally using the aggregate test score over these patterns as the final one. Using this technique, we trained an XGBoost model, a logistic regression model, a random forest model, a support vector classifier (SVC), and a k-nearest neighbors (KNN) classifier. For our evaluation metric, we chose the f_beta score over accuracy to place more emphasis on precision and recall, i.e., the false positives and false negatives, since they are more important for clinical deployment. To decide the beta value and choose whether to put more weight on precision or recall would require consultation with a domain expert and for this project, we went with beta = 1, consistent with previous work in the field[4]. We tuned various hyperparameters for each model to account for over and underfitting. These hyperparameters, along with the best ones, and the corresponding validation scores are summarized in Table 1. To account for uncertainties due to splitting and due to non-deterministic machine learning (ML) methods, we looped over 5 different random states and collected the mean and standard deviation of the F1 score.

Model Name	Hyperparameters to tune	Best Parameters	Validation Score
XGBoost	"reg_alpha": [0e0, 1e-2, 1e-1, 1e0, 1e1, 1e2], "reg_lambda": [0e0, 1e-2, 1e-1, 1e0, 1e1, 1e2], "max_depth": [1,3,10,30,100],	{'colsample_bytree': 0.9, 'learning_rate': 0.03, 'max_depth': 100, 'missing': nan, 'n_estimators': 10000, 'reg_alpha': 1.0, 'reg_lambda': 100.0, 'seed': 0, 'subsample': 0.66}	0.90 ± 0.01
RF XGBoost	"reg_alpha": [0e0, 1e-2, 1e-1, 1e0, 1e1, 1e2], "reg_lambda": [0e0, 1e-2, 1e-1, 1e0, 1e1, 1e2], "max_depth": [1,3,10,30,100]	{'colsample_bytree': 0.9, 'learning_rate': 0.03, 'max_depth': 100, 'missing': nan, 'n_estimators': 10000, 'reg_alpha': 1.0, 'reg_lambda': 100.0, 'seed': 0, 'subsample': 0.66}	0.90 ± 0.04
RF Logistic Regression	penalty': ['l1', 'l2', 'elasticnet'], 'C': [0.001, 0.01, 0.1, 1, 10], 'l1_ratio': [0.1, 0.5, 0.9]	{'penalty': 'elasticnet', 'l1_ratio': 0.1, 'C': 0.001}	0.75 ± 0.10
RF Random Forest	n_estimators': [3, 5, 7, 10, 50], 'max_depth': [1, 3, 5, 10, 20]	{'n_estimators': 5, 'max_depth': 1}	0.85 ± 0.06
RF SVC	C': [0.1, 1, 10], 'kernel': ['linear', 'rbf', 'poly'], 'gamma': np.logspace(-1,2,5)	{'kernel': 'linear', 'gamma': 0.1, 'C': 0.1}	0.82 ± 0.09
RF KNN	n_neighbors': [1,2,3,5,10,30,100], 'weights': ['uniform', 'distance']	{'weights': 'uniform', 'n_neighbors': 1}	0.61 ± 0.11

Table 1: Summary of hyperparameter tuning for all models.

Results

The baseline F1 score was calculated by assigning the minority class for all predictions and came out to be 0.249±0.037. <u>All models performed multiple standard deviations above the baseline score.</u> The test scores are summarized in Table 2 and visualized in Fig. 3. The best performing model was the XGBoost model with a test score of 0.922±0.038. A normalized confusion matrix was plotted to inspect the model further (Fig. 4).

Model Name	Mean Test Score	Std Test Score
XGBoost	0.922	0.038
RF XGBoost	0.884	0.049
RF Logistic Regression	0.791	0.085
RF Random Forest	0.856	0.045
RF SVC	0.851	0.042
RF KNN	0.596	0.122
Baseline	0.249	0.037

Table 2: Summary of test scores for all models vs. baseline.

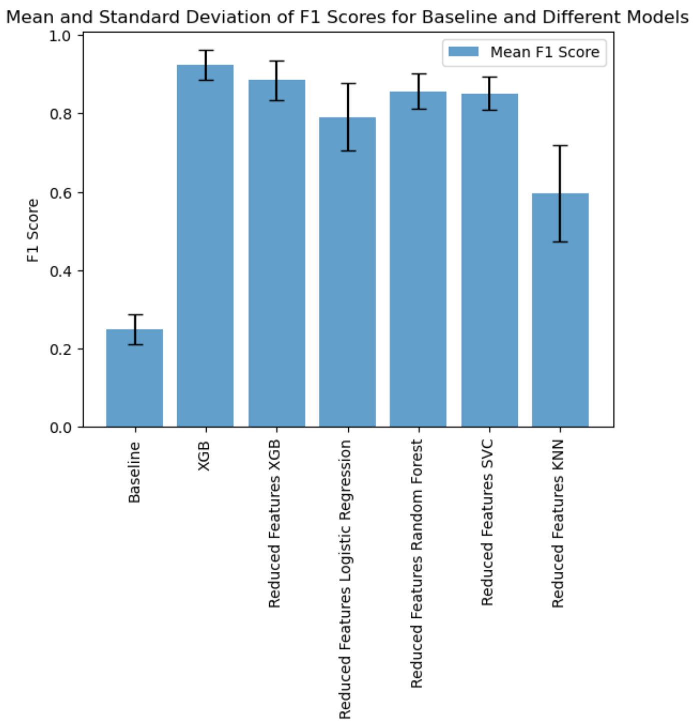


Figure 3: Summary of test scores for all models vs. baseline.

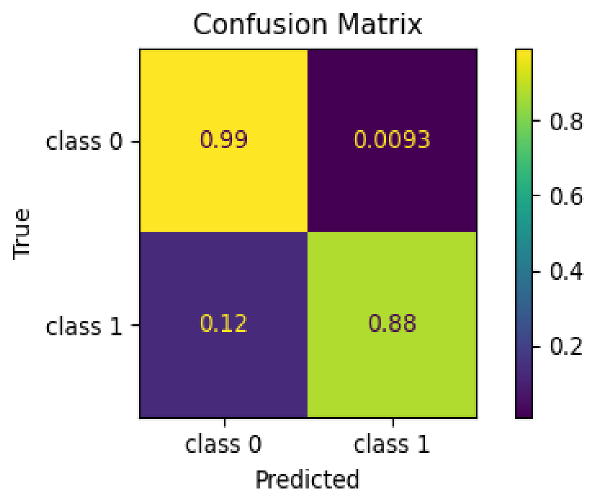
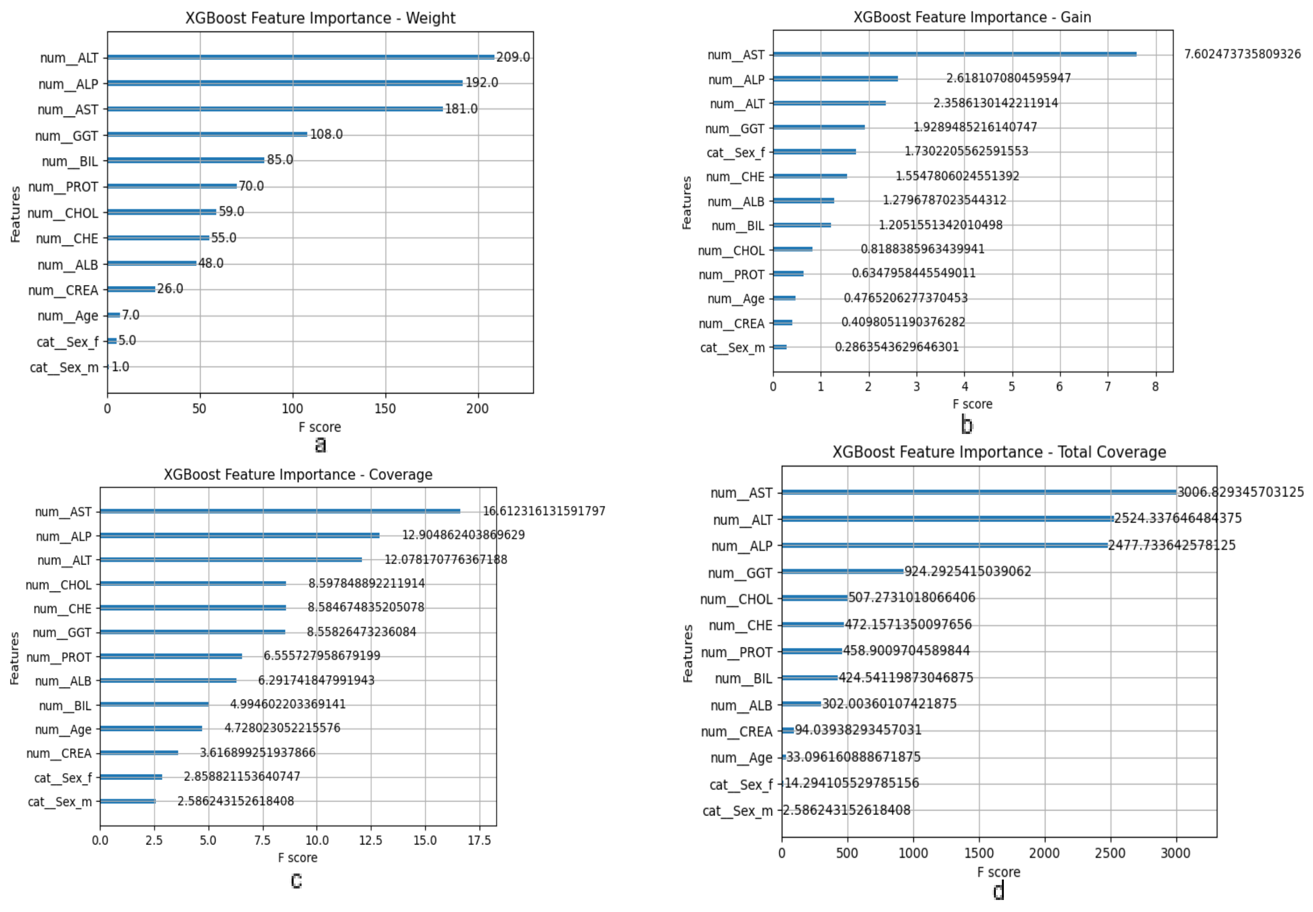


Figure 4: Normalized confusion matrix of predicted values from XGBoost model.

To increase interpretability of our best model, we also looked at global feature importances under different metrics of gain, cover, weight, total gain, and total cover (Fig. 5). We then calculated SHAP values and plotted global (Fig. 6) and local feature importances (Fig. 7) according to that.



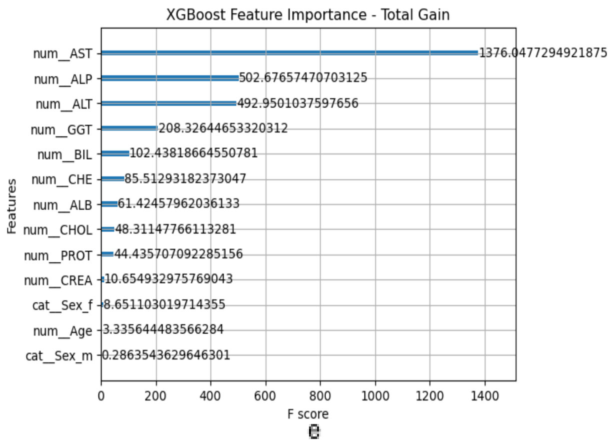


Figure 5: Global feature importances according to (a) Weight, (b) Gain, (c) Cover, (d) Total Cover, and (e) Total Gain.

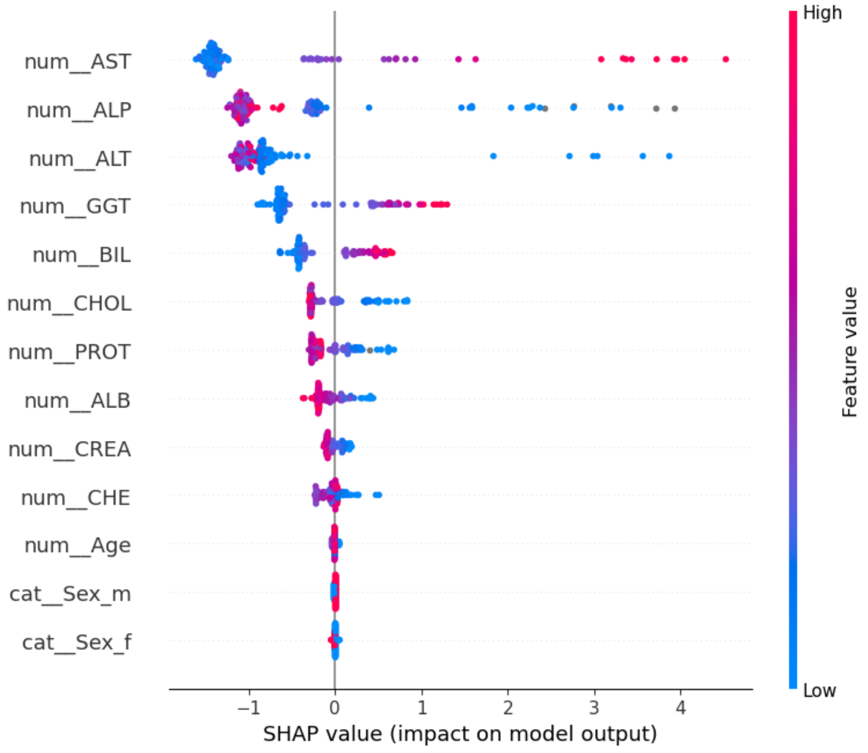


Figure 6: Global feature importances according to SHAP values.

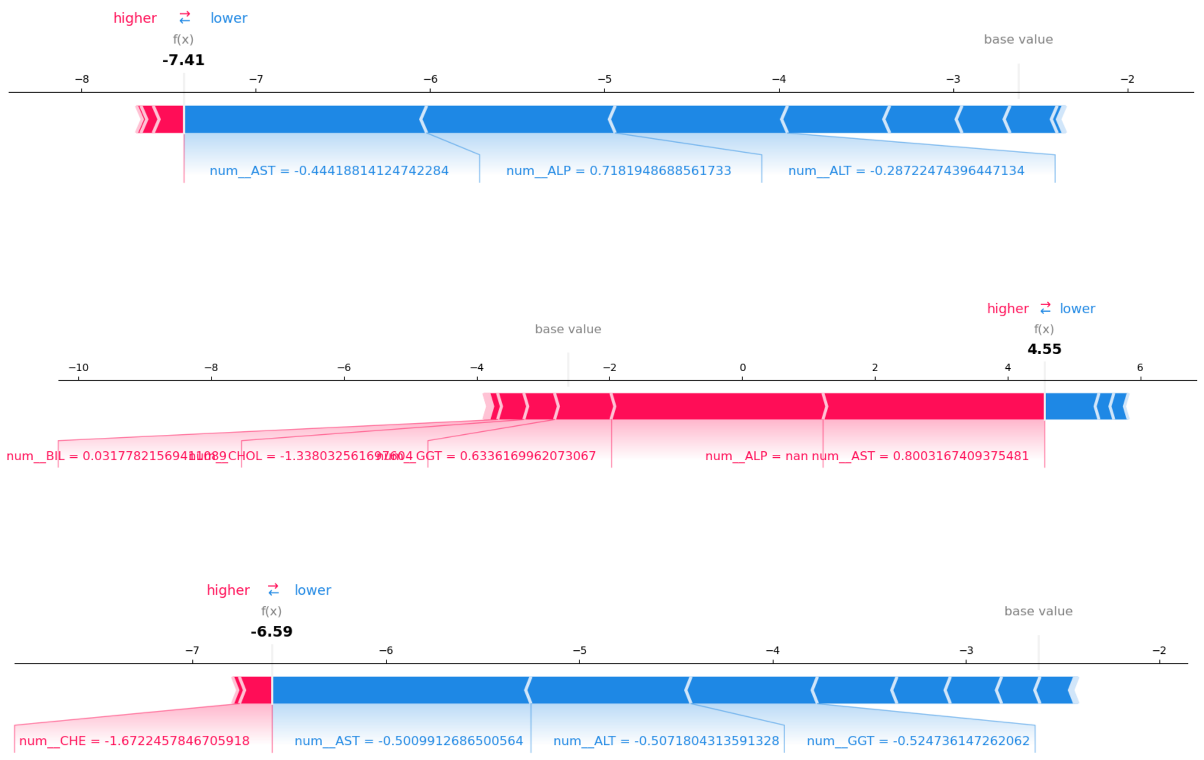


Figure 7: Local feature importances according to SHAP values for 3 data points.

Overall, our model gave us a lot of insight into the problem we were trying to solve and we achieved a much higher F1 score than the baseline. In the confusion matrix, the low false positive rate was promising but the high false negative rate is something which needs to be rectified in future iterations of the project before real world deployment. By plotting multiple feature importance graphs, we were able to derive some commonalities. Four features in particular: AST, ALT, ALP, and GGT stood out in all of these plots as the most important features. As predicted by our EDA, sex and age were not that important. These results make sense because these are liver enzymes and a badly functioning liver would not be able to make full use of these enzymes, causing them to leak into the bloodstream and show up higher on a blood test. Unsurprisingly, kidney enzymes such as creatinine and albumin are not that important for liver disease detection. The interpretability matches up with previous literature[4] and proves that we have a reasonable model.

Outlook

While the model does perform well, there is definitely scope for future improvements in terms of increasing predictive power and interpretability. Firstly, the dataset was very small. 615 patients are not enough and a major step forward could be to go to hospitals and collect more data to augment the dataset. In a similar vein, more enzymes (features) can be added. In terms of models to be used, we could try artificial neural networks and deep learning methods to see if they perform better. Even within XGBoost, we could use more advanced methods such as monotonic constraints and custom loss functions to fine-tune the model. Finally, consulting with a clinician would be a must before deployment.

References

[1] Fleurence, R. L., & Collins, F. S. (2023). A national hepatitis C elimination program in the United States: a historic opportunity. *Jama*, *329*(15), 1251-1252.
[2] Alter, M. J., Kuhnert, W. L., & Finelli, L. (2003). Guidelines for laboratory testing and result reporting of antibody to hepatitis C virus.
[3] Lichtinghagen,Ralf, Klawonn,Frank, and Hoffmann,Georg. (2020). HCV data. UCI Machine Learning Repository. https://doi.org/10.24432/C5D612.
[4] Mostafa, F. B., & Hasan, E. (2021). Machine Learning Approaches for Inferring Liver Diseases and Detecting Blood Donors from Medical Diagnosis. *medRxiv*, 2021-04.