Module Code: CS3AM
Assignment Title Report: Liver Disease Dataset Analysis
Student Number: 30008472
Student Name: Reagan Dias
Convenor Name: Muhammad Shahzad
Date (When work completed): 8/12/2024
Actual hrs spent on the assignment: 30

Abstract:

In this report two machine learning models are evaluated, XGBoost and a Feedforward Neural Network (FNN) for predicting liver disease based on several different features. Both models were trained on pre-processed data, with Synthetic Minority Oversampling Technique (SMOTE) applied to training data to address target imbalance. XGBoost achieved outstanding performance with an accuracy of 99.78% on the test dataset following cross validation and hyperparameter tuning. The deep learning model similarly showed strong performance on the test dataset with an accuracy of 99.73% and generalized extremely well.

Background & Problem Statement

Liver disease contributes to approximately two million deaths annually, accounting for 4% of global mortality, and this number continues to rise (Harshpreet Kaur, 2021). This alarming statistic underscores the urgent need for improved methods of early detection and diagnosis (Shahid Mohammad Ganie, Dutta and Zhao, 2024). Traditional diagnostic techniques, such as imaging and liver biopsies, while effective, are often time-consuming, invasive, and carry risks, such as internal bleeding or infection (Cleveland Clinic, 2021). These limitations highlight the necessity of innovative, non-invasive solutions. Machine Learning (ML) offers a promising alternative for early diagnosis by identifying patterns and trends in biomedical data that may not be immediately evident to clinicians (Srilatha Tokala et al., 2023). Leveraging its capabilities, this report aims to classify whether a patient has liver disease using biomarkers from a Kaggle dataset containing over 32,000 records. The objective is to evaluate and compare the performance of two classification models, XGBoost and a Feedforward Neural Network (FNN), to determine the most effective approach for predicting liver disease.

Exploratory Data Analysis (EDA):

Dataset Overview

The 'Liver Patient Dataset' on Kaggle has 32,801 records along with 11 features. These features show several clinical and general factors of patients such as Age and Albumin. More specifically, the dataset also contains biomarkers which are indicative of liver health. This dataset contains many numeric features which are all the flaot64 datatype and one categorical feature, 'Gender', which is an object datatype. Summary of features:

Feature	Description
Age	Age of a patient
Gender	Gender of a patient ('Male' / 'Female') – Needs to be converted to numeric
Total Bil	Total bilirubin level in the blood (mg/dL)
Alk Phos	Alkaline Phosphate levels (U/L)
SGPT	Serum Glutamate Pyruvate Transaminase level (U/L)
SGOT	Serum Glutamate Oxaloacetate Transaminase level (U/L)
Total Proteins	Total protein levels in blood (g/dL)
Albumin	Albumin level in the blood (g/dL)
AG Ratio	Albumin to Globulin level
Result	Liver disease patient (2: Liver disease, 1: No Liver disease) – Target feature

Table 1: Feature Summary

```
Msinng Value %'s:

Age              0.006097
Gender           2.749916
Total_Bil        1.975550
Direct_Bil       1.710314
Alk_Phos         2.426755
SGPT             1.640194
SGOT             1.408494
Total_Protein    1.411542
Albumin          1.506052
AG_Ratio         1.752995
Result           6.432731
```

Figure 1: Percentage of missing values

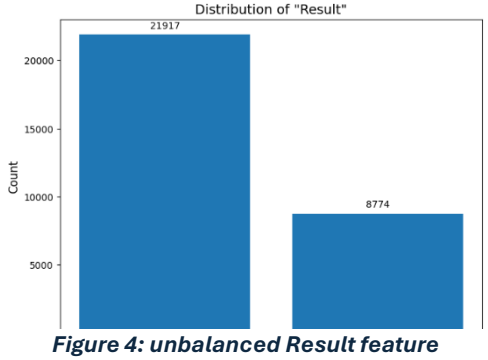

Figure 4: unbalanced Result feature

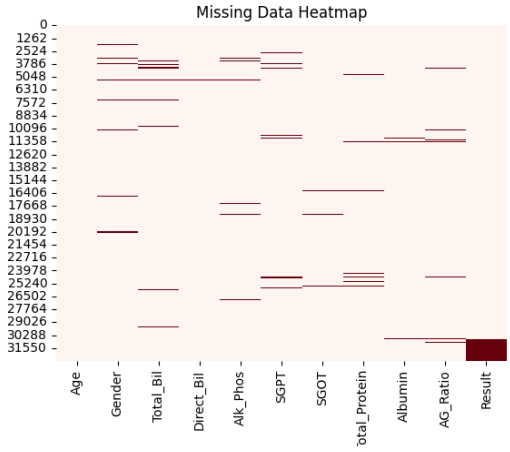

Figure 2: Heatmap of missing values

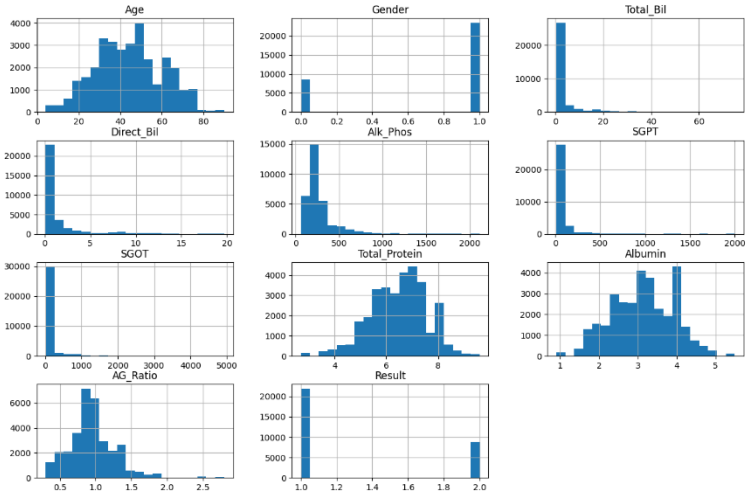

Figure 3: distribution of numeric features

Missing Values - Figure 1 illustrates that nearly all features have a low percentage of missing values, apart from 'Result', which has the most at 6.43%. Figure 2 reveals that most features appear to have missing values at random. However, 'Result' has a cluster of missing values towards the end of the dataset, suggesting that the missing values for 'Result' aren't random but systematic. This might be due to errors during data collection or potential bias during sampling, which would influence model training. Imputation for most features seems plausible with the low percentages and dropping the missing result values would be the most appropriate solution as this is the target feature and imputation of this feature would introduce noise.

Feature Distribution - Figure 3 shows that the dataset contains a mix of both normally distributed features such as 'Age' and skewed features like 'SGOT' and 'SGPT'. The features skewed to the right are biomarker features like 'Direct Bil'. These features reflect that most patients have normal biomarker levels, while a smaller subset of patients with extreme cases of liver disease show elevated levels. Applying transformations, such as logarithmic scaling, will be applied to skewed features to reduce their range and better handle outlier detection. For categorical features, 'Gender' is imbalanced with 73.4% being males and only 26.6% being females. This imbalance is unlikely to significantly affect predictions due to its low correlation with the target as seen in Figure 5. Nevertheless, it's still important to acknowledge and monitor this imbalance during modelling.

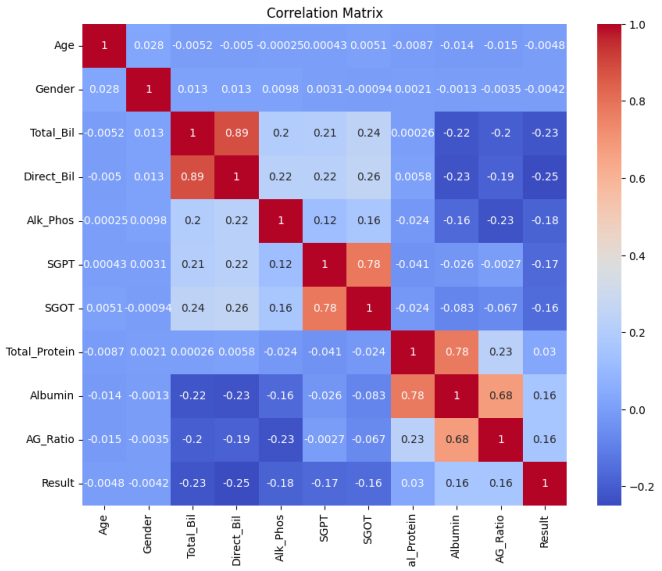

Figure 5: Correlation Matrix for all feature

Correlation Analysis - The biomarkers had the strongest correlation which shows their importance for predicting liver disease, and this aligns with domain knowledge as SGOT and SGPT are elevated together during liver dysfunction (Cohen and Kaplan, 1979). Highly correlated features will be removed during feature selection to avoid multicollinearity.

Target Feature Imbalance - Figure 4 highlights the severe imbalance of the 'Result' feature with ~71% of records having no liver disease while ~29% have liver disease. This emphasizes the importance of oversampling techniques, such as (Synthetic Minority Oversampling Technique) SMOTE, to balance out the classes and avoid bias towards the majority class. Oversampling is a better option in this scenario as under sampling would discard a large portion of the dataset to match the minority class leading to a smaller training dataset, which in turn would lead to the model not being able to generalize well. SMOTE retains the

majority records while generating synthetic samples for the minority class leading to a larger and more balanced dataset. This will lead to the model being able to learn both classes better.

Outlier Detection - Figure 6 visually identifies outliers using boxplots. Its notable that a lot of the biomarkers, such as 'SGPT', 'SGOT' and 'Alk Phos' from first glace contain a large portion of outliers. However, these outliers are biologically meaningful, showcasing extreme cases of liver disease, therefore they should be retained. Removing these records could risk losing valuable information about extreme cases, which a robust ML model should account for.

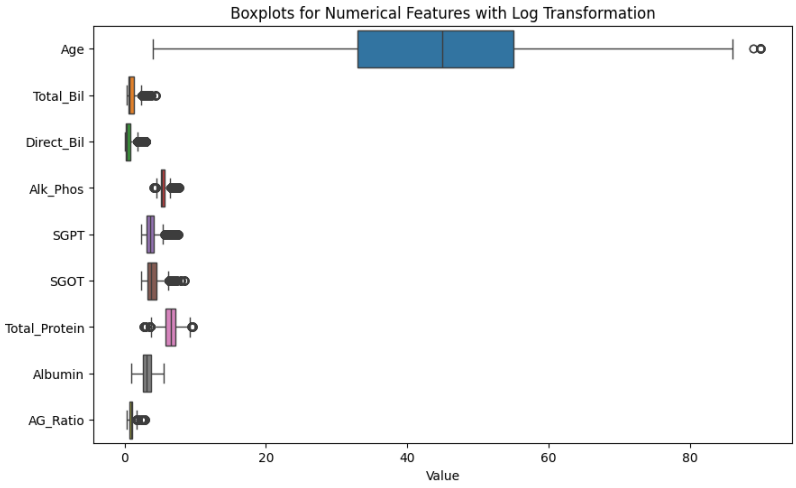

Figure 6: Boxplot showing outliers for numeric features

Duplicates - Figure 7 summarizes the count of duplicate rows vs non duplicate rows. There are ~11,000 duplicate rows, which is a substantial volume of the original dataset. While these duplicate rows could represent patients with similar or repeated entries. Removing them was decided as to avoid overfitting and to ensure that the model learns generalized patterns instead.

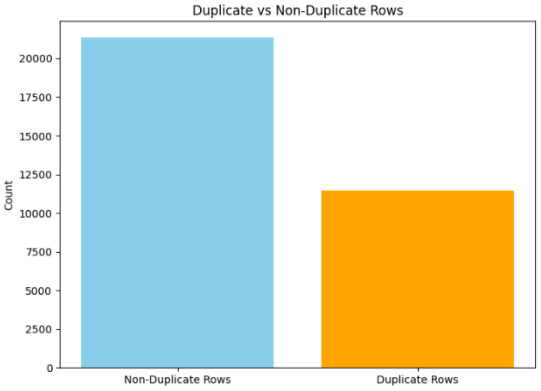

Figure 7: None duplicate rows Vs duplicate rows

Data Preprocessing and Feature Selection:

Duplicates - Removing the duplicates was the option that was chosen in the end and after removing the duplicates, the dataset was reduced to ~21,000 unique records. While this may be quite a chunk of the original data it is important to prioritize the model's generalization instead of overfitting and just memorizing patterns within the training data.

Missing Values - I researched into multiple different imputation methods such as using K Nearest Neighbours (KNN), linear regression and simple imputation like mean, median and mode. KNN imputation works by using the K nearest neighbours based on feature similarity. Since the dataset contains multiple highly correlated features, this may cause KNN to overly on these features, potentially introducing noise. Linear regression on the other hand, uses feature relationships to predict missing values. However, the issue with linear regression to impute missing values for this dataset is that is assumes a linear relationship between features, which may not hold true for some nonlinear features, such as biomarkers. Moreover, linear regression is more prone to overfitting if there are limited missing values for a feature. I decided to use mean, median and mode as they preserve the integrity of the dataset and avoids introducing additional complexity.

Feature	Age	Gender	Total Bil	Direct Bil	Alk Phos	SGPT	SGOT	Total Protein	Albumin	AG Ratio	Result
Method	Mean	Mode	Median	Median	Median	Median	Median	Mean	Mean	Median	Dropped

Table 2: Imputation Method

Mean – uses central tendency for normal distribution because it captures the centre of the data accurately.

Median – for skewed features or features that have heavy outliers. Median is not affected by extreme values making it idea for imputation for features that are skewed such as SGOT.

Mode – for categorical features such as Gender this is the best imputation method due to it selecting the most frequent value.

Class Imbalance 'Result' - Figure 8 shows the before and after of applying SMOTE. The count of records ended up being 21,917.

Feature Encoding: - 'Gender' had to be converted into a numeric format to be used for training. Label Encoding was used for this ('Male': 1, 'Female': 0).

Correlated Features - As discussed in the EDA, highly correlated features such as 'Direct_Bil', 'Albumin' and 'SGPT' were dropped to avoid multicollinearity. Figure 9 is a correlation matrix after dropping the features named, as can been seen in the figure, no highly correlated features are present after dropping those features.

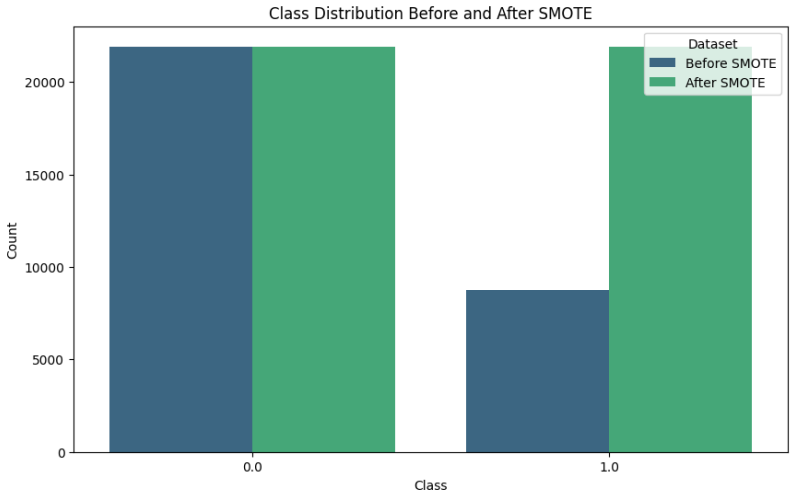

Figure 8: Result before and after SMOTE

Splitting dataset – Using stratified sampling, the clean dataset was split into train (80%) and test (20%). Stratified sampling was used as it preserve the class distributions. Similar to the real world, you'd expect to have a high number of people without liver disease compared to liver disease, which is what the original dataset shows as previous seen from Figure 4. Preserving this was crucial choice as it mimics real world scenarios.

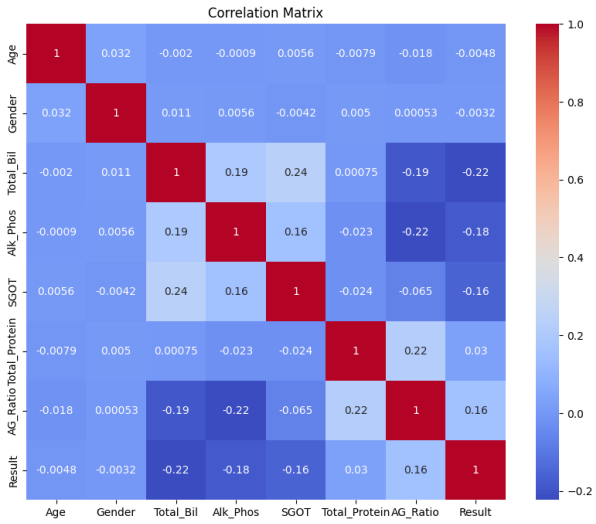

Figure 9: Correlation Matrix after dropping features

Modelling

XGBoost:

To solve the binary classification problem of identifying if a patient has liver disease or not, XGBoost was selected as the first model. XGBoost was specifically chosen because of a couple of factors, such as the nature of the dataset being structured and the data containing features such as biomarkers, which are more susceptible to outliers and skewness, which XGBoost can effectively handle (Chen and Guestrin, 2016). One final key reason as to why XGBoost was chosen was because of the feature importance metric. This metric will show insights as to which features are indicative of liver disease, allows deeper understanding and analysis of the dataset. XGBoost works by utilises gradient boosting for improved performance on structured data, handling outlier and missing data effectively (Shwartz-Ziv and Armon, 2021).

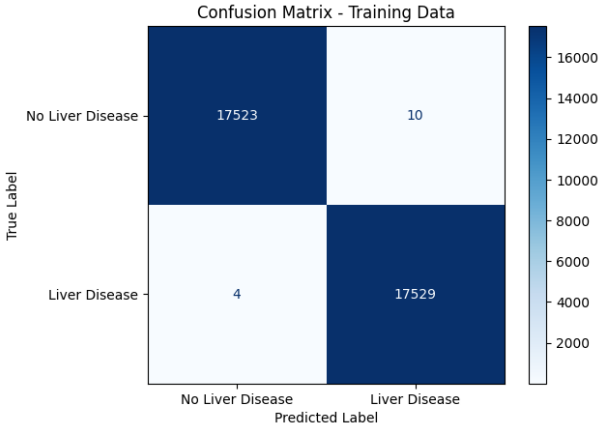

Figure 10: XGBoost Train (SMOTE)

Training and Evaluation:

Throughout the rest of the report a combination of setting a seed and random state was used to get finer control of consistent result. Seeding was used to have more refined control over the random number generators of different modules such as TensorFlow and NumPy, while random state was used to ensure reproducibility for isolated machine learning functions. An arbitrary number of 42 was chosen for both the seed and random state. The XGBoost model was first trained on the SMOTE balanced training dataset to address the issue of class imbalance in the target feature, this ensured that the minority class was well represented during the training phase. The model was evaluated using cross validated, hyperparameter tuning and finally on the test dataset. The initial model on the training dataset used these parameters: learning rate = 0.3, max depth = 6, number

of estimators = 100 and minimum child weight = 1. With these parameters the model received an accuracy of 99.96% with recall, F1 score and precision all being 1.00. Figure 10 showcases that the model had 4 false positives and 10 false negatives. While the initial model did achieve great results, these results were based on one train test split and doesn't demonstrate how the model would perform on unseen data. To address this concern, a more robust evaluation technique is required. Cross validation with 10 folds was applied to ensure generalizability and remove any concerns of overfitting. A pipeline incorporating SMOTE and XGBoost was used to balance each training fold dynamically. This meant that the class balance for each training subset was maintained without contaminating the validation data. The results were [99.59%, 99.75%, 99.91%, 99.75%, 99.71%, 99.63%. 99.67%, 99.83%, 99.83% and 99.83%]. The mean cross validation accuracy was 99.75% showcasing consistent performance across all folds with slight deviation, demonstrating the model's ability to generalize. Although the accuracy was high, an issue that needs to be addressed is the number of false positives and negatives. On the training dataset there were 4 false positives and 10 false negatives (Figure 10). While these number may seem comparatively low; in the medical domain this is quite significant. Patients shouldn't be told that they have liver disease when they don't or worse they don't have liver disease when they do. In order to decrease the number of false positives and negatives the parameters of the model need to be configured to get to most optimal results which have a low rate of false positives and negatives while maintaining generalization. Hyperparameter tuning addresses this issue as it will optimize the model's performance based on the best parameters. Evaluating different methods of hyperparameter tuning, GridSearch was decided upon due to its ability to exhaustively test all parameters with each other and finds the optimal combinations of these parameters without manually having to try them all, which would be very time consuming. These parameters that were evaluated were: Learning Rate = [0.2,0.3,0.4], Max Depth = [3,5,7], Max number of estimators = [100,200,300], scale Pos Weight = [1,2,3] and min child weight = [1,5,10]. The best parameters were Learning rate = 0.2, Max Depth = 7, Number of estimators = 300, Scale Pos weight = 2 and min child weight = 1. Running the model on these parameters the accuracy was 99.99% and the recall, precision and F1 scores were 1.00. Figure 11 shows the confusion matrix after hyperparameter tuning on the train dataset. The false positives dropped from 4 to 1 and the number of false negatives dropped from 10 to 1, which shows that hyperparameter tuning was very effective in decreasing the false positives and negatives. After getting the best hyperparameters and verifying that they improved the model's performance. The model was reinitialized, and the test dataset was run through the model. The accuracy on the test dataset was 99.73%, the F1 score, recall and precision were all 1.00 apart from the recall for the class with liver disease which had a recall of 0.99. Figure 12 shows the confusion matrix for the test dataset.

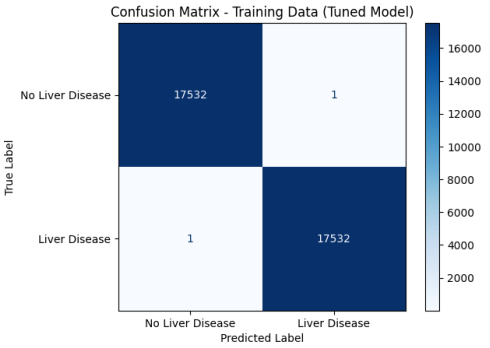

Figure 11: Confusion Matrix on Train with hyperparameter tuning

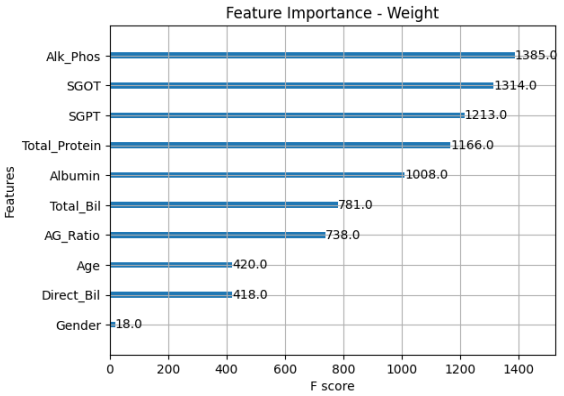

Figure 13: Feature Importance on test dataset

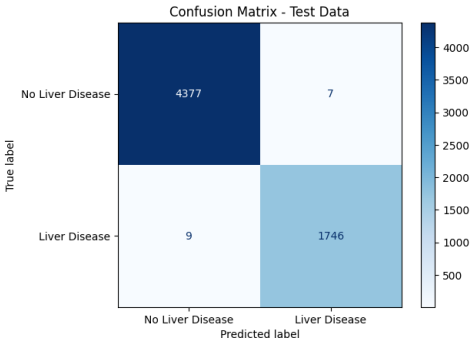

Figure 12: Confusion matrix on test dataset

Tabel 3: XGBoost Metrics

Metric	Train	Test
Accuracy	99.99%	99.73%
F1 Score	1.00	1.00
Recall	1.00	0.99
Precision	1.00	1.00

Results and Discussion:

In summary, XGBoost was successfully trained on unseen data and got a very high accuracy, recall, precision and F1 score. This was too be expected as it's known that XGBoost performs well on structured data as

discussed within this report. Using SMOTE was effective as can be seen from table 3, the test accuracy and nearly all other metrics were similar to the training model. The decrease in accuracy for test was to be expected, SMOTE was only applied to the training data and the test data was kept untouched. This was by design to preserve the distribution of classes and simulate the real world. The lower accuracy on test means that the model is good as generalizing on unseen data. Finally, figure 13 showcases that biomarker were crucial in the identification of liver disease and this is further support by medical knowledge.

Model 2: Deep Learning Model (Feedforward Neural Network):

Feedforward Neural Network (FNN) was used as the choice of deep learning model because of its capabilities to capture non-linear relationships between features (Goodfellow, Bengio and Courville, 2016). While it has been proven that other models outperform deep learning models on small tabular data, FNN can also perform very well if the data is pre-processed well and regularized (Robertmarnelius and Ikuo S.sogami, 2012).

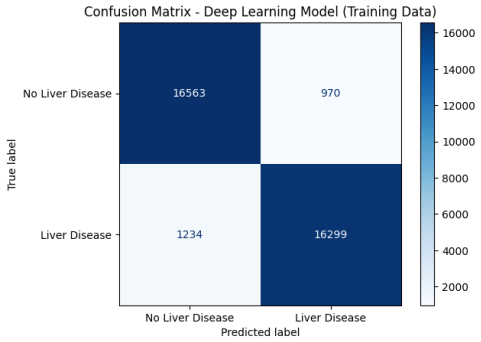

Figure 14: Confusion Matrix for FNN (SMOTE)

Training and Evaluation:

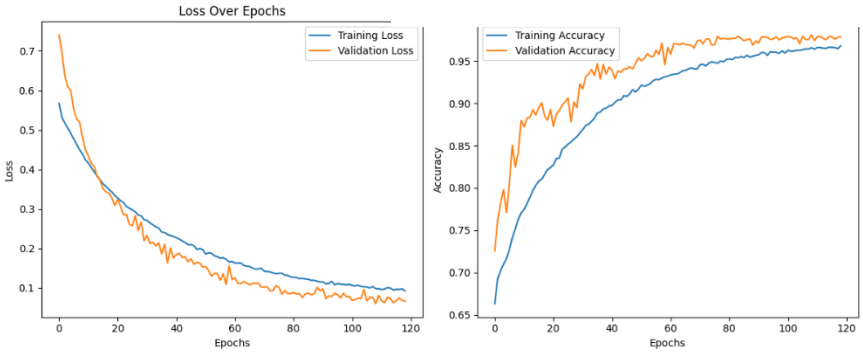

Figure 15: Loss and Accuracy over Epochs

Like XGBoost this model was also seeded and used the same random states to ensure consistency. The initial model architecture that the training data was going to be trained on consists of 3 layers (input, hidden and output) which are linked. The inputs layer is a dense layer containing 64 neurons, dropout rate of 0.2 and uses the RelU activation function. The second layer contains 32 neurons, dropout rate of 0.1 and RelU activation. Finally, the last layer contains 2 neurons and uses the sigmoid activation function. The first and second layers are used to introduce nonlinearity, allowing the model to learn patterns within the data. The third layer outputs probabilities of if the patient has liver disease or not. Before training started SMOTE was applied to the training dataset so the target imbalance issue could be addressed. StandardScaler was used to standardize the input data and stop feature scale from dominating during backpropagation. The model was initially trained using these parameters. Optimizer: Adam with a default learning rate of 0.0001, loss function: Binary cross entropy and early stopping was implemented to prevent overfitting by looking at the validation loss and stopping training if over 10 consecutive epochs it didn't improve. Figure 14 shows that the model learned quite well from the train dataset since the classes are balanced and the accuracy of the model was 93.71% after 81 epochs. The Recall, F1 score and Precision were roughly the same between the classes as can been seen in Table 4. Similar to XGBoost, the number of false positives and negatives needs to be addressed as they are quite high if this model were to diagnose patients. Hyperparameter tuning was conducted to find the best combination of parameter for the FNN. Table 5 shows the key highlights from experimenting with different parameters. The model showed Figure 15, which shows the loss and accuracy over the number of epochs. Due to early stopping the model stopped at 119 epochs. Some of the key discoveries from hyperparameter tuning: Smaller learning rate ensures fine-tuned convergence, a batch size of 64 is a intermediate spot which doesn't lead to noise, going from 2 to 3 layers improved accuracy potentially showing that the original architecture was underfitting.

Tabel 4: FNN Metrics

Metric	Train	Test
Accuracy	93.71%	98.73%
F1 Score	0.94	0.99
Recall	0.93	0.99
Precision	0.94	0.98

Epochs	Learning Rate	Batch Size	Drop Out Rate	Neurons/Layer	No. Hidden Layers	Accuracy	Comments
100	0.001	64	[0.2,0.1]	[128,64]	2	0.895	Baseline
100	0.001	64	[0.2,0.1]	[128,64]	2	0.912	Improved Generalisation
100	0.001	64	[0.2,0.1]	[256,128,64]	3	0.927	Optimal Neurons
100	0.005	64	[0.1,0.1]	[256,128,64]	3	0.950	Best Parameters

Table 5: Hyperperameter Tuning for FNN

Using the test dataset with the best parameters from hyperparameter tuning I got an accuracy of 99.73%. The precision, recall and F1 scores were all nearly 1.00, which shows the model has balanced performance across both classes. From figure 16, which is a correlation matrix for the test dataset, we can see that the model has a great ability to generalize.

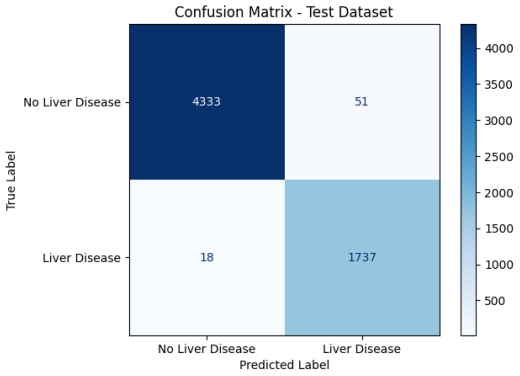

Figure 16: FNN on test dataset

Results and Discussion:

Table 4 shows the results from training the model from on train and test data. The model performed better on the test data than train due to more layers which introduced more dropout regularization. This combined with early stopping prevents the model from overfitting. Figure 16 shows the correlation matrix for the test data which was also expected since SMOTE wasn't applied to the test data to simulate the distribution of the real world and of the original dataset there would be more instances of one class compared to another.

Comparison and Results:

Table 6: Comparing XGBoost Vs Feedforward neural network metrics

	XGBoost		Feedforward Learning Model	
	Liver Disease	No Liver Disease	Liver Disease	No Liver Disease
Recall	0.99	1.00	0.99	0.99
F1 Score	1.00	1.00	0.99	0.99
Precision	1.00	1.00	0.97	1.00
Accuracy	99.78%		98.73%	

Both models performed exceptionally well receiving very high accuracy, F1 scores, recall and precision. Table 6 shows the performance metrics for both XGBoost and the FNN on the test dataset. All metrics were identical for XGBoost were higher than FNN other than the recall for liver disease which was the same and the precision for no liver disease. This is further supported by looking at the confusion matrix on the test dataset for both models (Figure 12 and Figure 16). XGBoost performed slightly better only getting 9 false positives and 7 false negatives while the FNN got 18 false positives and 51 false negatives. The very slight increase in accuracy shows that XGBoost is better at generalizing compared to the FNN. XGBoost performing better than the FNN, all be it very slightly, was too be expected due to the structured nature of the dataset and unique strengths that differentiate XGBoost from other models. The liver dataset is tabular in nature, and there have been several studies, as previously mentioned in the report, that demonstrate that XGBoost excels with these typers of datasets. It is also important to note that given the simplistic nature of the dataset, as seen from the features importance graph (Figure 13), the FNN didn't learn any complex or nonlinear patterns within the dataset, which led to it performing quite well. The performance of the FNN was further helped by the small size of the dataset. After preprocessing, there was ~21,000 unique records which is relatively small for a deep learning model to reach its full potential.

Conclusion and Recommendations:

To conclude, two models, XGBoost and FNN, were successfully trained and tested on unseen data. Both models showed exceptional results with high accuracy, recall, precision and F1 scores. This success demonstrates the importance and significance that machine learning can have in a clinal setting, specifically identifying if a patient has liver disease or not based on a couple of clinical features. The report also highlights the importance of hyperparameter tuning for machine learning models. Particularly, there was an increase of ~5% with hyperparameter tuning for the FNN which bought the accuracy a lot close to the XGBoost model. While the results were exceptionally high, there is always room for improvement and further testing. Testing the model on unseen liver disease datasets would be the next recommendation as it would expose the model to broader set of patient data which would further test the model's generalizability. Potentially exploring cost sensitive techniques to punish false negatives could be another improvement given that this is to do with clinical data, being accurate and precise is crucial.

References:

Harshpreet Kaur, G.S. (2021). The Diagnosis of Chronic Liver Disease using Machine Learning Techniques. *INFORMATION TECHNOLOGY IN INDUSTRY*, 9(2), pp.554–564. doi:https://doi.org/10.17762/itii.v9i2.382.

Shahid Mohammad Ganie, Dutta, K. and Zhao, Z. (2024). Improved liver disease prediction from clinical data through an evaluation of ensemble learning approaches. *BMC medical informatics and decision making*, 24(1). doi:https://doi.org/10.1186/s12911-024-02550-y.

Cleveland Clinic (2021). *Liver Disease: Types*. [online] Cleveland Clinic. Available at: https://my.clevelandclinic.org/health/diseases/17179-liver-disease.

Srilatha Tokala, Koduru Hajarathaiah, Sai, Srinivasrao Botla and Murali Krishna Enduri (2023). Liver Disease Prediction and Classification using Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications*, [online] 14(2). doi:https://doi.org/10.14569/IJACSA.2023.0140299.

Cohen, J.A. and Kaplan, M.M. (1979). The SGOT/SGPT ratio?An indicator of alcoholic liver disease. *Digestive Diseases and Sciences*, 24(11), pp.835–838. doi:https://doi.org/10.1007/bf01324898.

Chen, T. and Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp.785–794. doi:https://doi.org/10.1145/2939672.2939785.

Shwartz-Ziv, R. and Armon, A. (2021). Tabular Data: Deep Learning is Not All You Need. *arXiv:2106.03253 [cs]*. [online] Available at: https://arxiv.org/abs/2106.03253.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. [online] Deeplearningbook.org. Available at: https://www.deeplearningbook.org/.

Robertmarnelius and Ikuo S.sogami (2012). GENERALIZED BRST QUANTIZATION AND MASSIVE VECTOR FIELDS. *International Journal of Modern Physics A*, [online] 13(18). doi:https://doi.org/10.1142/S0217751X98001530.

www.kaggle.com. (n.d.). *Liver Disease Patient Dataset 30K train data*. [online] Available at: https://www.kaggle.com/datasets/abhi8923shriv/liver-disease-patient-dataset.