Supervised learning techniques for classification of liver disease patients

1.Introduction

1.1 Classification of liver disease

The incidence of liver disease is increasing throughout the world. In addition to the effects of alcohol consumption, harmful gases and drugs on the liver, the obesity and diabetes epidemics will likely invoke a liver disease epidemic as a result of their contribution to non-alcoholic fatty liver disease (NAFLD), which is increasingly recognized as the most common chronic liver disease in the western world (Kemmer, 2013).

Liver disease covers a diverse group of pathologies, that notoriously present late, due to the vague symptom profile in early disease and the ability of the liver to continue to function normally even when partially damaged (Kun-Hong, 2008).

An early diagnosis of liver problems will increase patient's survival rate and reduce burden on healthcare services. Analysis of liver function blood tests are used to aid liver disease diagnosis. However, the tests can be difficult to interpret, as they can be affected by other conditions and generally don't correlate well to disease severity. Therefore, automated tools to aid classification of liver disease from blood test results could be useful as a screening tool (Ramana, 2011).

1.2 The dataset

The Indian Liver Patient Dataset (ILPD) is an open source dataset available on the UCI machine learning repository and has been used to develop classification algorithms for liver disease. The data was collected from the north east of Andhra Pradesh, India. It is composed of 583 instances and 11 attributes, one of which is a selector class used to divide the instances into binary sets - liver disease (416 patients) or no liver disease (142 patients) – as labelled by experts. The remaining 10 attributes are listed and described in table 1 along with the distribution of the feature, as visualised by histograms and boxplots in section 1.4 of the attached notebook.

Feature	Distribution	Description	Correlation with Class
Age	Normal	Average age: 45. Age range: 4-90.	0.14
Gender	Categorical	75% male and 25% female patients.	0.082
Total Bilirubin	Right Skewed	A product of red blood cell breakdown, processed by the liver, increases with liver damage.	0.3
Direct Bilirubin	Right Skewed	Processed form of bilirubin only, increases with liver damage.	0.29
AlkPhos	Right Skewed	Alkaline Phosphatase: An enzyme found in bone and liver. Increases with liver damage and bone disease.	0.27
Sgpt/ALT	Right Skewed	Alanine aminotransferase: Parenchymal liver enzyme, increases with liver damage.	0.29
Sgot/AST	Right Skewed	Aspartate aminotransferase: Parenchymal liver enzyme, increases with liver damage.	0.31
Total_Protein	Normal	Measure of proteins produced by liver and elsewhere, can increase or decrease in liver disease.	-0.032
Albumin	Normal	A protein produced by the liver, usually decreases late in liver disease.	-0.17
A/G ratio	Normal	Albumin/Globulin – ratio of albumin to globulin proteins produced in liver, can decrease or increase with liver disease.	-0.19
Class Label	Categorical	1 = Liver Disease, 2 = No Liver Disease	n/a

Table 1: Feature Set Description & Distribution

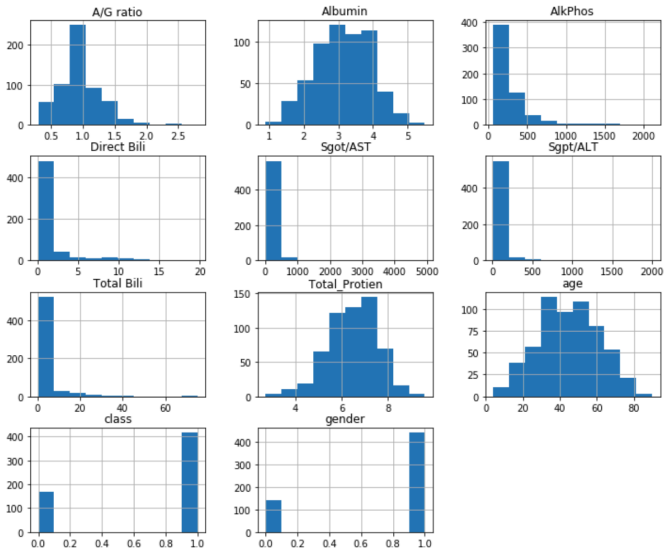


Figure 1a: Feature Space Histograms

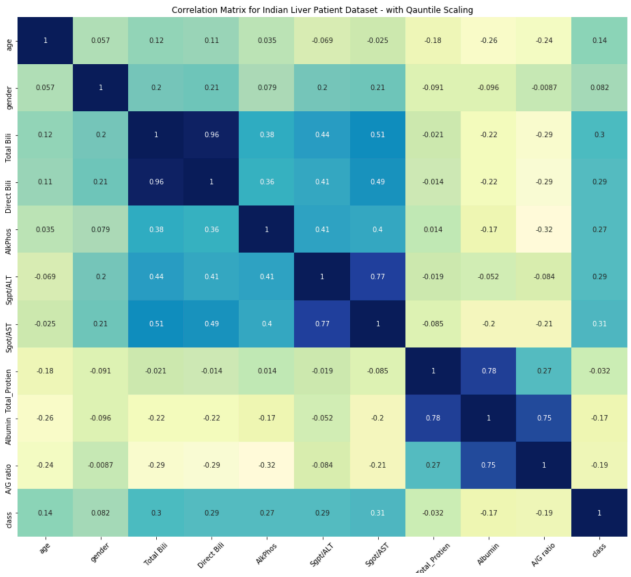


Figure 1b: Feature Space Correlation Matrix

1.3 The effect of right skewed distributions

Total Bili, Direct Bili, Sgot/AST, Agpt/ALT and AlkPhos (which all rise as a result of liver damage) have a **right skewed distribution** with a long tail at higher values. This is a known distribution of these products (Johnston, 1999), as can be seen in the figure 2. These liver products can increase to very large values in certain stages of liver disease - e.g. in acute hepatic injury, values can be > 10 times the upper reference range (Giannini, 2005). Therefore, the outlier values for these features are most likely to represent real values, rather than input or measurement error and were all included in the models. The effect of this distribution, however, is that the large outlier values dominate the feature space and make it hard to visualise (see figure 3a). Quantile scaling is a form of scaling that spreads out the most frequent values and reduces the impact of the outliers (scikit-learn.org, 2017), better revealing the feature space in 3D (see figure 3b).

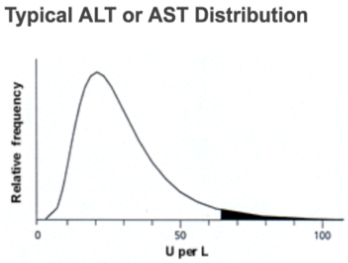


Figure 2: ALT/AST Distribution (Johnston, 1999)

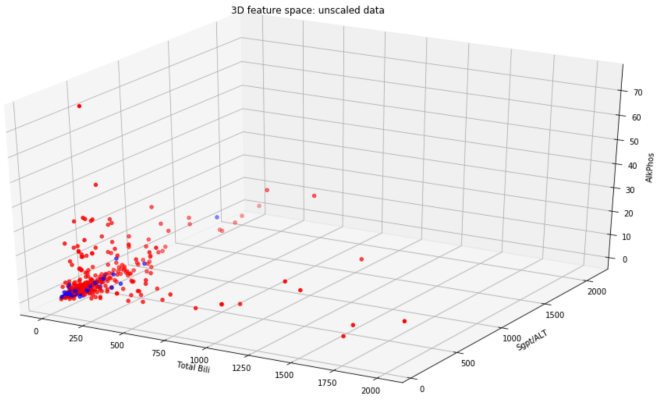


Figure 3a: 3D unscaled feature space scatter plot

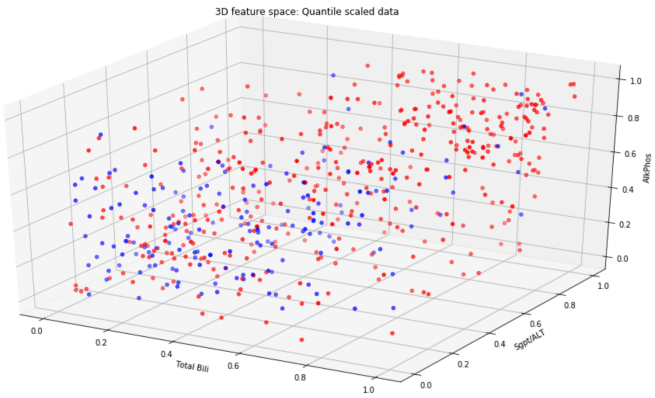


Figure 3b: 3D quantile scaled feature space scatter plot

2. Method

2.1 Data Cleansing

Non-liver patient class value was changed from 2 to 0, to resemble common classification values. Gender was converted to numerical categorical values: female=0, male=1. **Imputation** with mean values was used on the 4 instances with missing values in the albumin/globulin ratio column. Both **min-max scaling** and **quantile scaling** were applied to the data and used to optimise pipelines. To avoid overfitting of the algorithms and reduce bias, the dataset was **split 70%/30%** into training/testing samples. **Random over sampling** of the non-liver group was used to balance the dataset - as machine learning models often misclassify data when faced with imbalanced datasets due to minority classes being treated as noise (Analytics Vidhya, 2017). Under sampling was not used as the dataset is already small.

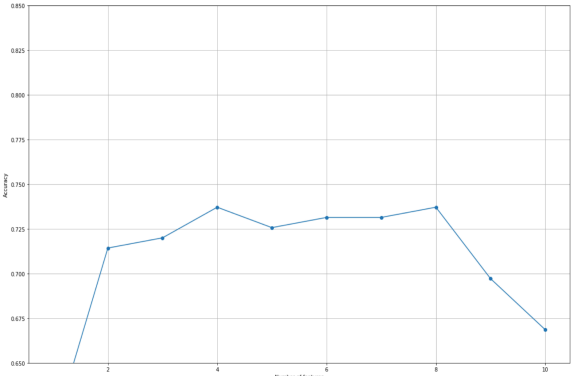


Figure 4a: Accuracy with increasing features

2.2 Feature Selection

A combination of 'filter' (algorithm independent) and 'wrapper' (sequential backward selection – SBS) techniques were used to select a subset of features to reduce redundancy/noise and improve model accuracy. Combining the results of SBS and Weka attribute ranking carried out on the ILPD by Ramana et al. (Ramana, 2011) four features were chosen for exclusion from the dataset (gender, direct bili, total_protein, A/G ratio). This is discussed in more detail in section 3 of the notebook.

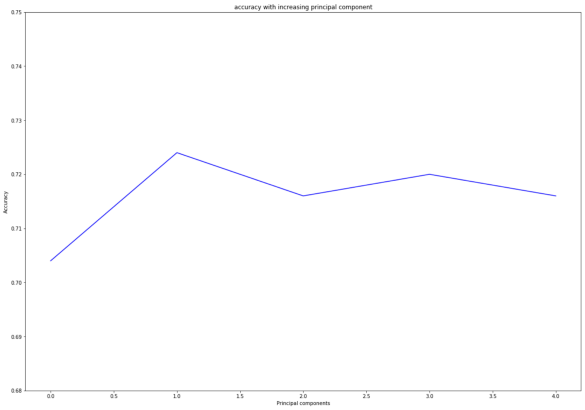


Figure 4b: Accuracy with increasing components

2.3 Feature Extraction; Dimensionality Reduction

A basic un-tuned naïve-bayes algorithm was used to evaluate accuracy with increasing principal components (see figure 4b). Reducing to 2 principal components appears to provide best accuracy – most likely by reducing noise in the model.

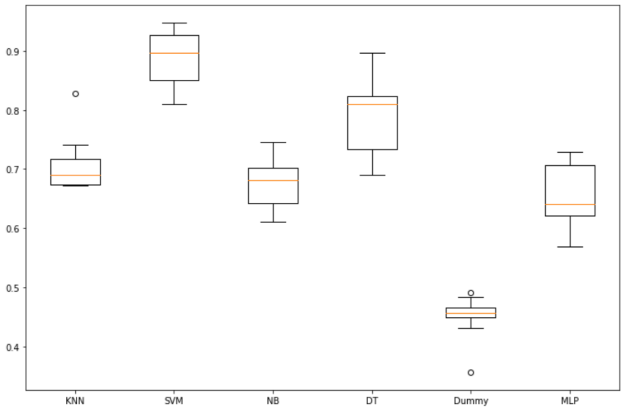


Figure 4c: Comparing accuracy of un-tuned classifiers

2.4 Algorithm Selection

An array of un-tuned pipelines was used to compare the accuracy of different classification algorithms and provide as a basis for selecting two to focus on (see figure 4c). Support vector machine and decision tree performed well and were thought to fit the needs of the dataset as will now be discussed.

2.5 Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm that discriminates between two classes by generating a separating hyperplane. A hyperplane is an object that has one less dimension than the

dataset. In non-linearly separable datasets, hyperplanes are determined by applying non-linear kernel functions.

The advantage of using SVM is that it is 'model free' (Yu, 2010)– in contrast to logistic regression which relies on a pre-determined model. This makes SVMs powerful for classification with small sample sizes (such as the ILPD) and large number of variables.

2.6 Decision Trees (DT)

A decision tree is a predictive modelling algorithm that maps a set of instances to a target function that will accurately classify the instances, by searching a set of function hypotheses. For classification trees, target variables have discrete values. In decision trees, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. The algorithm uses a top down approach, looking at predictors at each node to find a cost value that best splits the data into two groups. The cost value either tries to minimize the impurity of nodes (entropy) or maximize the information gained (gini index).

Support Vector Machine (SVM)	**Classification Decision Tree (DT)**
1. Identify classification category. 2. Determine closest points from one group of data points to the another**: 'support vectors'.** 3. Look for separating boundary (hyperplane) that maximizes distance between hyperplane and support vectors**: 'maximal separating hyperplane'.** 4. If data not linearly separable - apply non-linear kernel functions to transform data – **linear, polynomial, radial basis function or sigmoid** 5. Allow for some data points to be on wrong side of the hyperplane by specifying a **parameter 'C'** (Higher 'C' = harder margin, sensitive to noise). 6. Specify **parameter 'gamma'** that defines how far the influence of a single training example reaches (low values ='far', high values ='close'). 7. Classify new data points by determining which side of the hyperplane they fall into.	1. Choose criteria to split each node of decision tree: **entropy** (minimise impurity), **gini** (maximise information gain). 2. Specify splitting and stopping criteria – e.g. **maximum depth** of the tree, **minimum samples** for node split. 3. Create a root node by searching for decision attribute (A) among all input variables that produces best split for criteria. Assign A to root node. 4. For each value of A at root node – create descendent node. 5. Search and assign again at each descendent node for best decision attribute from remaining variables as in step 2. 6. Repeat splitting/searching/assigning as in steps 2-4 until instances are perfectly classified or stopping criteria met.

Table 2: SVM and Decision Tree algorithms in pseudocode.

2.7 Hyper-parameter Tuning

As discussed in the pseudocode table, the algorithms can be tuned with by altering parameter values. The optimal parameters for SVM and DT were determined by using sci-kit learn grid search. This was only done on training data so as not to introduce bias to the model by exposure to the test data. A comparison of the hyper-parameter results will be discussed in results.

2.8 Cross Validation, Supervised Learning and Model Evaluation

To further reduce potential for bias and overfitting, 10-fold cross validation was incorporated into the training algorithms (10 folds of the data were used to train on each fold and test on a different held out subset, then test accuracy on the whole training set).

The final SVM and DT models, with optimised hyper-parameters were fitted to the clean, feature selected, dimension-reduced data in a two-phase process: the training phase (with cross-validation) builds the classifier on the training set and the second phase classifies on the unseen test data. Several performance estimates were used to evaluate the models: area under the curve (AUC), accuracy, sensitivity, specificity, positive predictive value (PPV) and negative predictive value (NPV).

2.9 An Ensemble Approach to Decision Trees: Gradient Tree Boosting

An automated machine learning tool called TPOT was applied to the ILPD. It explores thousands of machine learning pipelines using genetic programming. The outcome showed a gradient boosting classifier was optimal.

Gradient tree boosting (GTB) is a generalisation of boosting ensemble learning. GTB achieves improved performance by combining decision trees and gradually minimising the loss function at each sequential combination using gradient descent method (an algorithm for finding the minimum of a function). GTB handles heterogeneous data well and is robust to outliers, which explain its strong predictive power for the ILPD. However, due to the sequential nature of boosting, it is not suitable to scaling/parallelisation.

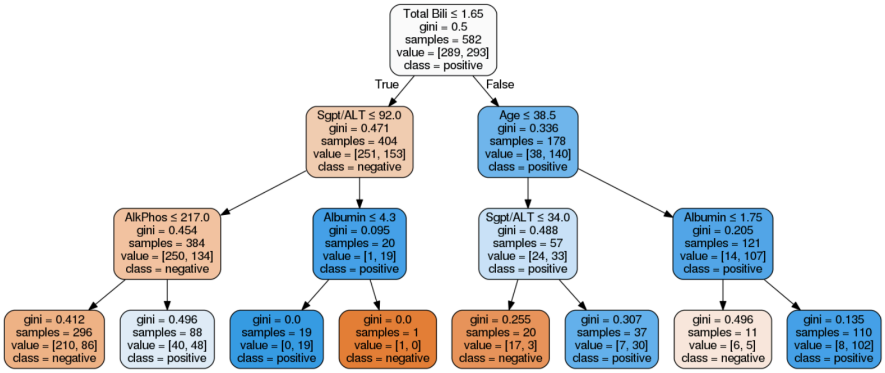


Figure 5: Decision Tree with max depth = 4 to show splitting using gini index. (Full decision tree in notebook)

Figure 6: Calculating average error using K-fold cross validation

3. Results

3.1 Optimal Hyper-Parameters

Grid search was performed on pipelines for both SVM and DT, yielding the optimal hyper-parameters as seen in table 3. The grid search was performed to optimise both accuracy and area under the curve (AUC) – metrics that will be discussed later.

For SVM, the optimal C-parameter was always small (1.0 or 10.0), creating a softer margin that is less sensitive to noise. The optimal kernel was 'rbf' which stands for radial basis function, a non-linear transformation function, indicating the non-linear nature of the dataset.

For DT, the optimal criterion was gini, meaning the nodes will be split to minimise the gini impurity index, and the tree should be grown to a max-depth of 9 with 4 samples needed at each node for a split.

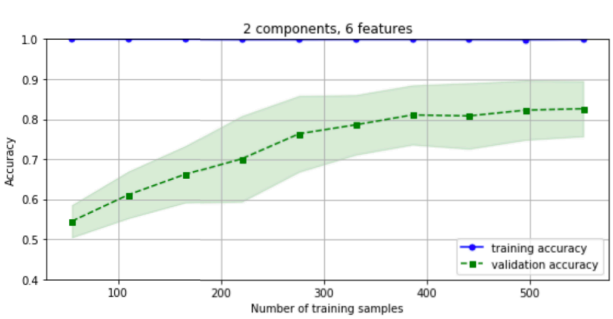
Model	Optimal Hyper-Parameters	Scoring Criteria
SVM, 10-fold CV, 2 principal components, standard scaler	C: 1.0, gamma: 1000.0, kernel: 'rbf'	Accuracy
SVM, 10-fold CV, 2 principal components, standard scaler	C: 10.0, gamma: 1000.0, kernel: 'rbf'	AUC
DT, 10-fold CV, 2 principal components, standard scaler	criterion: 'gini', max_depth: 9, min_samples_split: 4	Accuracy
DT, 10-fold CV, 2 principal components, standard scaler	criterion: 'gini', max_depth: 9, min_samples_split: 4	AUC

Table 3: Optimal Hyper-parameters for SVM and DT classifiers

3.2 Learning Curves; Bias and Variance

Learning curves are graphs that plot model performance on training and testing data against varying numbers of training instances. It allows training and testing performance to be viewed separately, to estimate how well models generalise to new data and allow diagnosis of bias and variance problems. High bias is when training/testing errors are high and converge, resulting in poor generalisation. High variance is when there is a large gap between the errors, which could indicate there is not enough data or the model is too complex with too many features (Ng, 2018).

As can be seen in figure 7a, for SVM: with 2 principal components the training line (blue) is at its maximum regardless of training samples showing severe overfitting. Testing score (green) increases over time and there is a large gap between the scores indicating high variance. Reducing the complexity of the model (to 1 principal component) can be seen to reduce the variance with a trade-off that bias is increased (accuracy reduces). Variance can also be seen to reduce for DT learning curve when going from 10 principal components + 10 features to 2 components + 6 features. Collection of more data will likely improve the variance of the models.



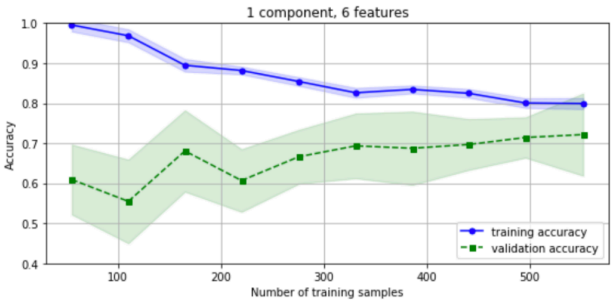
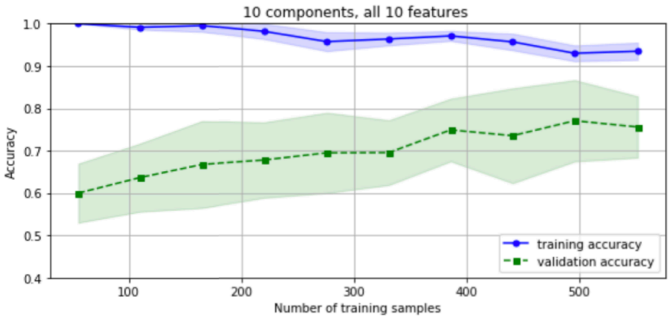


Figure 7a: Learning curves for SVM (top left and above)



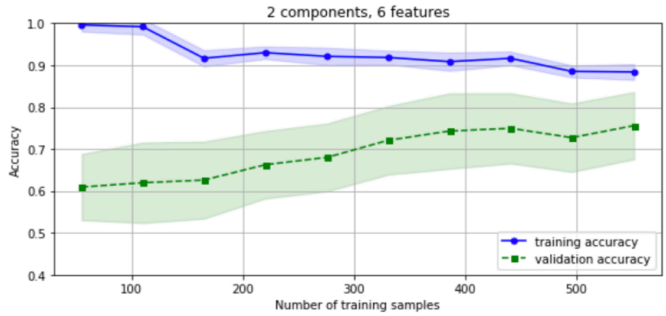
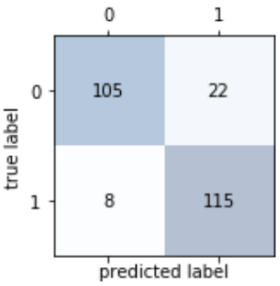


Figure 7b: Learning curves for DT (top right and above)

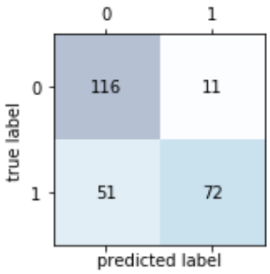
3.3 Model Performance

As a simple measure of performance, classification accuracy was used throughout to allow quick comparison between pipelines. This is a ratio of the number of correct predictions out of all predictions that were made. However, accuracy can be misleading, particularly on imbalanced data and it is important to look at a range of performance metrics.

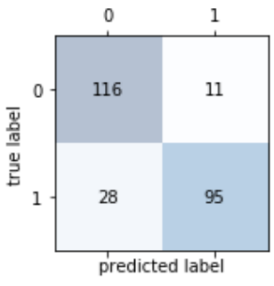
A confusion matrix provides a summary of the predictions made compared to expected actual values. A perfect set of predictions is shown as a diagonal line from the top left to the bottom right. Confusion matrices for final 3 classifiers are show in in figure 8.



Final SVC



Final DT



Final GTB

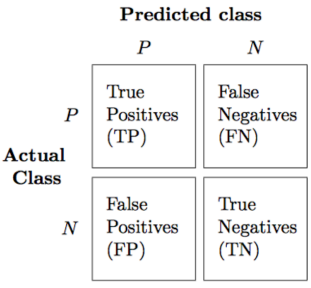


Figure 8: Confusion Matrices for final optimised classifiers.

SVM produces the 'best' confusion matrix with the most number of true-positive and true-negative results identified. DT has a high false-positive rate, and consequently a lower true-negative rate. These figures were used to calculate a range of performance metrics shown below.

Overall, SVM performs the best for all metrics except PPV and NPV, for which GTB performs slightly better. SVM has a high sensitivity (ability to correctly classify as ′disease′) and high specificity (ability to correctly classify as 'disease- free').

DT performs the worst for all metrics except NPV. It has a particularly low specificity – as expected from the high number of false positives.

Scoring Metric	**Calculation**
Accuracy	*(TP +TN) / (TP + FP + FN +TN)*
Specificity (true negative rate)	*TN / TN + FP*
Sensitivity / recall (true positive rate	*TP / (TP + FN)*
Positive Predictive Value (Precision)	*TP / (TP+FP)*
Negative Predictive Value (NPV)	*TN / TN + FN*
F1	*2* (precision *recall) / (precision + recall)*

Table 4: How to calculate different scoring metrics from confusion matrix results

Model	**Accuracy**	**Specificity**	**Sensitivity (recall)**	**PPV (precision)**	**NPV**	**F1**
SVC	0.88	0.93	0.88	0.89	0.83	0.88
DT	0.75	0.58	0.75	0.78	0.86	0.74
GTB	0.84	0.77	0.84	0.85	0.89	0.84

Table 5: Comparing different scores for final optimised classifiers

Model	Area Under Curve (AUC)			
Support Vector Machine	RBF	Linear	Sigmoid	Polynomial
	0.83	0.76	0.64	0.77
Decision Tree	Gini, depth=9	Entropy, depth=9	Gini, depth=6	Entropy, depth=6
	0.85	0.85	0.78	0.77
Gradient Tree Boosting	n_estimators=100, learning_rate=0.5, max_depth=6			
	0.95			

Table 6: Comparing AUC with different hyper-parameters

3.4 Receiving Operation Characteristic (ROC) and Area Under Curve (AUC)

AUC was calculated for each classifier and used to plot the ROC curve (Figure 9). AUC is used to compare the discriminatory powers of the models based on predicted outcome vs true outcome. A perfect classifier, dummy classifier and some un-tuned classifiers are also plotted alongside SVM, DT and GTB.

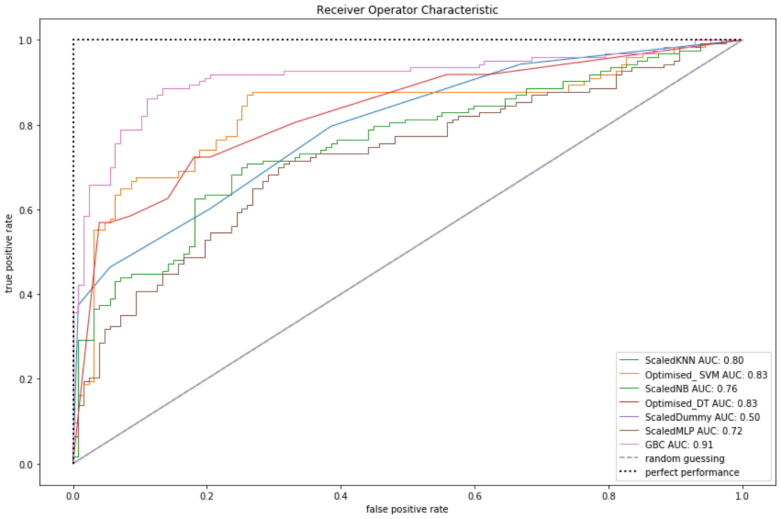


Figure 9: ROC Curve comparing classifier AUCs.

Table 6 compares AUC for SVM, DT and GTB with different hyper-parameters. All 3 classifiers have a high AUC, indicating good predictive power of the models. The best AUC for each model is seen with the optimised hyper-parameters discussed in section 3.1. GTB with n-estimators=100 produces the best AUC: 0.95.

It is noted that AUC for DT is high despite its poor specificity and lower accuracy score. Accuracy and other simple metrics are computed at single threshold values (default=0.5). While AUC is computed by averaging expected values across all threshold values (Stackexchange.com, 2018), which could explain this discrepancy. It also highlights the importance of using a range of model evaluation techniques.

4. Discussion and Conclusion

4.1 Comparison of results with other studies on ILPD

The study found GTB to be the best performing classifier when looking at AUC. SVM showed the highest specificity and sensitivity. While DT performed better than naïve algorithms, it's specificity was low. The algorithms generalised best with reduced model complexity and feature selection applied.

Ramana et al (Ramana, 2011) found SVM to have the best precision on the ILPD, in keeping with my results. In contrast, however, they found the accuracy improved with the addition of new attributes – but they did not comment on other evaluation metrics.

Gulia et al. (Gulia, 2014) found SVM algorithm to be the most accurate before applying feature selection (accuracy-71.3551%). But, Random Forest algorithm was optimal with the help of feature selection (accuracy-71.8696%). My accuracies for SVM are higher which could be explained by hyper-parameter tuning, which they do not discuss.

Pahariya et al. (Pahariya, 2014) also found Random Forest with over sampling 200% outperformed all the other techniques.

The success of random forests (an ensemble tree classifier) in 2 other papers, is in keeping with my finding that GTB performs the best on this dataset. Ensemble (boosting) tree techniques are widely found to be powerful off the shelf classifiers.

4.2 Proposals for improving methodology

With more time, I would expand and improve this study with the following considerations:

• **Under Sampling:** Oversampling was used due to low sample size. However, under sampling of the liver disease group or a mixture of over and under sampling could be assessed.

• **Reduce variance:** By collecting more data or combining with other datasets.

• **Analyse varying cut-off values for evaluation metrics:** AUC gave a different picture to some of the other evaluation metrics – most likely due to being an average over different threshold values. Thresholds could be tuned to optimise other metrics such as sensitivity/specificity with the goal of meeting the requirements of a screening tool for liver disease.

4.3 Potential usage of ILPD classification algorithms in healthcare

The classifier could be used as a screening tool in primary care to refer for further testing e.g. an automated web-based risk calculator similar to the fib-4 index calculator for liver fibrosis currently used in UK to assess need for biopsy (Camdenccg.nhs.uk, 2018).

However, the ILPD is limited due to the lack of associated metadata about the patients. 'Liver disease' encompasses a wide range of conditions of differing aetiologies, requiring different screening methods and management. There is no information about the types of liver disease in the ILPD and the large age range indicates a mixed group. This produces a 'black box' situation often discussed in machine learning; our algorithm may be classifying well but we're unable to use the information to make informed decisions. Classification studies focused on particular categories of liver disease would be a more useful approach in future.

References

- Analytics Vidhya. (2017). Imbalanced Classification Problem. [online]. Available at: https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/ [Accessed 26 Mar. 18].
- Camdenccg.nhs.uk (2018), fib-4-calculator. [online]. Available at: https://gps.camdenccg.nhs.uk/fib-4-calculator [Accessed 26 Mar. 18].
- Giannini, E., Testa, R., Savarino, V. (2005). Liver enzyme alteration: a guide for clinicians. *Canadian Medical Association Journal (CMAJ)*, [online], Volume 172(3), pp. 367-379. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC545762/ [Accessed 26 Mar. 18]
- Github.com. (2018). *EpistasisLab/tpot.* [online]. Available at: https://github.com/EpistasisLab/tpot [Accessed 26 Mar. 18].
- Gulia, A., Vohra, R., Rani, P. (2014). Liver Patient Classification Using Intelligent Techniques. *(IJCSIT) International Journal of Computer Science and Information Technologies,* Volume 5(4), pp. 5110-5115.
- Johnston, D. (1999). Special Considerations in Interpreting Liver Function Tests. *American Family Physician,* [online], Volume 59(8), pp. 2223-2230. Available at: https://www.aafp.org/afp/1999/0415/p2223.html [Accessed 26 Mar. 18].
- Kemmer, N.[1,2], Neff, G.[1], Franco, E.[1], Osman-Mohammed, H.[1], Leone, J.[1], Parkinson, E.[1], Cece, E.[1]; Alsina, A**.** (2013). Nonalcoholic Fatty Liver Disease Epidemic and Its Implications for Liver Transplantation. *Transplantation,* Volume 96(10), pp. 860-862.
- Kun-Hong, L., De-Shuang, H. (2008). Cancer classification using Rotation forest. *Computers in biology and medicine*, Volume 38, pp. 601-610
- Ng, R., (2018). *Machine-learning-learning-curve.* [online]. Ritchieng.com. Available at: http://www.ritchieng.com/machinelearning-learning-curve/ [Accessed 26 Mar. 18].
- Pahareeya, J., Vohra, R., Makhijani, J., Patsariya, S. (2014). Liver Patient Classification using Intelligence Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering,* Volume 4(2).
- Ramana, B., Surendra Prasad Babu2, M., Venkateswarlu, N. (2011). A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis. *International Journal of Database Management Systems,* 3(2).
- Scikit-learn.org. (2017). Sklearn.preprocessing.QuantileTransformer. [online]. Available at: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html [Accessed 26 Mar. 18].
- Stackexchange.com (2014). *Why-is-auc-higher-for-a-classifier-that-is-less-accurate-than-for-one-that-is-more-accurate.* [online]. Available at: https://stats.stackexchange.com/questions/90659/why-is-auc-higher-for-a-classifier-that-is-less-accurate-than-for-one-that-is-mo [Accessed 26 Mar. 18].
- Yu, W., Liu, T., Valdez, R., Gwinn, M., Khoury, M. (2010). Application of support vector machine modelling for prediction of common disease: the case of diabetes and pre-diabetes. *BMC Medical Informatics and Decision Making*, Volume 10(16).