Machine Learning in Practice: Indian Liver Patient Records

Andrew Infantino

November 17, 2020

Abstract

Chronic liver disease has risen to become the tenth leading cause of death worldwide. This paper examines the medical records of 416 liver patients and 167 non-liver patients from India to examine the relationship between liver disease diagnosis and several demographic and biochemical variables. The goal was to develop efficient algorithms for predicting diagnosis such as to reduce the burden on medical personnel in making diagnoses. Exploratory use of unsupervised machine learning found no predictively meaningful clusters. Seventeen supervised algorithms — based mainly on logistic regression, linear discriminant analysis, quadratic discriminant analysis, local regression (LOESS), k-Nearest Neighbors, and random forest — were trained to predict liver disease diagnosis and evaluated. Most of them were directly trained via two forms of cross validation, bootstrap and K-fold. Three consisted of ensembles of the others. One supervised algorithm was based on k-means-identified clusters. The two forms of cross validation yielded mixed results in affecting predictive performance. Quadratic discriminant analysis, logistic regression, and ensemble algorithms delivered the most promising results while differing in how well they confirmed and disconfirmed diagnoses. Future research should test the approaches of this report on larger and more diverse data while striving to develop consistent approaches that proficiently detect both positive and negative liver disease cases.

Introduction

Chronic liver diseases caused an estimated 1.32 million deaths across the world in 2017, comprising appromixately 2.4% of global deaths that year. These statistics mark a significant increase from 899,000 total deaths and 1.9% of global deaths in 1990. [1] As such, liver disease outranked road injuries, tuberculosis, and HIV/AIDS as the 10th leading worldwide cause of death in 2017. [2][3] The estimated global death toll of chronic liver disease was attributed to five causes: hepatitis B, hepatitis C, alcohol-related liver disease, and non-alcoholic steatohepatitis (NASH). Chronic liver disease itself refers to the long term (over at least six months) progression of liver disease toward the destruction of liver tissue, [4][5] such as via cirrhosis (severe scarring and replacement of functional tissue) and hepatocellular carcinoma (cancer). [6][7]

The primary causes and burdens of chronic liver disease vary by geographic location and demographic group. Hepatitis B and C are the most prominent worldwide causes, especially in low-income countries, but the Global Burden of Disease (GBD) research program estimated in 2017 that they will decline and NASH will overtake them in prevalence. [1] The age-standardized mortality rate due to chronic liver disease is highest in sub-Saharan Africa and lowest in high-income nations. It remained stable or decreased in most world regions but increased in Eastern Europe and Central Asia; increasing prevalence of alcohol-related liver disease is the primary culprit of the latter trend. [1]

In the United States, chronic alcohol-related liver disease — once considered "an old man's disease" — became more prevalent and deadly from 1999 to 2016, especially among women and people aged 25-34. [8] During that time period, alcohol-related mortality increased most significantly among Native, White, and Hispanic Americans. [30] Alcohol-related deaths have notably increased along with other "deaths of despair" — such as suicide and drug overdoses — among working class and middle-aged White Americans. [9]

The GBD recommends cost-effective interventions to prevent chronic liver disease, treat viral hepatitis, and achieve early diagnosis for chronic liver disease due to alcohol and NASH. [1] Mellinger et al. (2018) have also stressed the "urgent need to more effectively detect and prevent [alcoholic liver disease]." [10] The University of California, Irvine Machine Learning Repository compiled a database of 416 liver patient records and 167 non-liver patient records from Northeastern Andhra Pradesh, India. The dataset, available on Kaggle and GitHub, exists for algorithm developers to predict liver disease more efficiently and thereby reduce burdens on doctors. [11][12]

The data consist of eleven variables for all 583 patients — age, gender, diagnosis (liver disease or no disease), and eight chemical compounds present in the human body. The chemicals — all detectable via blood tests — include total bilirubin, direct bilirubin, alkaline phosphatase, alamine aminotransferase, aspartate aminotransferase, total proteins, albumin, and albumin:globulin ratio. [11]

Alkaline phosphatase (ALP), aspartate aminotransferase (AST or SGOT), and alanine aminotransferase (ALT or SGPT) are enzymes the liver and some other organs produce; liver damage elevates their concentration in the bloodstream. [13][14][15] Bilirubin is a pigment produced in the destruction of red blood cells; the liver absorbs it such that most is then excreted via feces as bile. It can be measured in the bloodstream as indirect (pre-liver process), direct (post-liver process), and total bilirubin. Unusually high levels of bilirubin indicate either an unusually high rate of red blood cell decomposition, a flaw in the excretory process between liver and stool, or a failure of the liver to properly break down waste and absorb bilirubin from the blood. [16][17] Albumin and globulin are proteins the liver produces and secretes into the bloodstream; they are collectively referred to as total protein. Low levels of total protein and low ratios of albumin to globulin can indicate liver disease. [18][19]

The purpose of this project was to examine the relationship between liver disease diagnosis and the other ten variable via machine learning and develop algorithms that would predict diagnosis on the basis of those variables as accurately as possible. Machine learning is the study and training of computer algorithms that automatically recognize patterns in data, improving themselves through experience. [20][21] An algorithm is a well-ordered sequence of unambiguous, computable, and finite operations that can solve a specifically defined problem. [22][23]

Two general forms of machine learning include supervised and unsupervised learning. The former approach involves training algorithms to discover unknown clusters of data. The latter approach relies on partitioning datasets into training and testing sets — algorithms are trained to infer statistical relationships between independent variables and outcomes in the former set; they are tested to accurately predict unprocessed outcomes from independent variables in the latter set. Subsequently, in practice, the supervised algorithms are trained on full datasets to predict outcomes in newly acquired data. [24][25]

All analyses are computed in in R 3.6.2 for this project. Algorithms that perform hierarchical clustering, heatmap visualization, and principal component analysis are forms of unsupervised machine learning that were used in the exploratory analysis of this report. [25][26] Supervised machine learning algorithms — including logistic regression, k-nearest neighbors (kNN), quadratic discriminant analysis (QDA), linear discriminant analysis (LDA), locally estimated scatterplot smoothing (LOESS), and random forests — were used to ultimately predict patient diagnosis. These supervised algorithms all used two forms of cross-validation, the repeated partitioning of training sets into sub-training and validation sets in order to achieve more accurate results. [27]

k-Means clustering, an unsupervised algorithm [25], was used to infer two clusters of data — corresponding to diagnosis — from the training set; a custom supervised algorithm was then employed to predict the clusters that would fit each testing set patient. Finally, all predictive algorithms were compiled into three different ensembles to predict the diagnosis of testing set patients. The algorithms were judged on the basis of their total accuracy, sensitivity (accuracy in predicting positive liver disease diagnoses), and specificity (accuracy in predicting negative liver disease diagnoses).

Methodology and Analysis

Data Examination and Cleaning

The Indian Liver Patient Records dataset is available on Kaggle and GitHub as a ZIP file. [11][12] After downloading it and extracting its contents, one is free to directly examine and analyze the data. Initial observations of the dataset `il_data` reveal 583 observations for 11 variables. Each observation corresponds to a patient. The variable names follow: Age, Gender, Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase, Alamine_Aminotransferase, Aspartate_Aminotransferase, Total_Protiens, Albumin, Albumin_and_Globulin_Ratio, Dataset.

`Dataset` refers to the patients' diagnoses. A value of 1 indicates liver disease whereas a value of 2 indicates no disease. As such, the data consist of 416 liver disease patients (liver patients) and 167 non-liver disease (healthy) patients. Before continuing the analysis, one must search the dataset for missing values — early attempts at machine learning without doing so have resulted in algorithmic failure. Indeed, 4 missing values were identified in the making of this report. They all were determined to belong to the Albumin_and_Globulin_Ratio variable for 4 different patients.

After those patients were removed, the dataset consisted of 414 liver patients and 165 healthy patients. For visual purposes, a clone `v_data` of the dataset was created in which `Dataset` was converted to the class "character" under the new variable `Diagnosis`:

```
v_data <- il_data %>%
  mutate(Diagnosis = ifelse(Dataset == 1, "Liver Disease", "No Disease"))
```

All `il_data` variables except for gender are of the class "integer;" gender is of the class "character." Gender was converted to a factor variable such that it could be analyzed statistically alongside all other variables:

```
il_data <- il_data %>%
  mutate(Gender = as.integer(as.factor(Gender)))
```

71.5 % and 28.5% of the remaining patients respectively had liver disease and no disease. The patients are 24.18% female and 75.82% male. Male patients are somewhat more likely than female patients are to be diagnosed with liver disease, composing a somewhat larger majority among liver patients than among healthy patients.

Figure 1: Percentage of Patients by Diagnosis

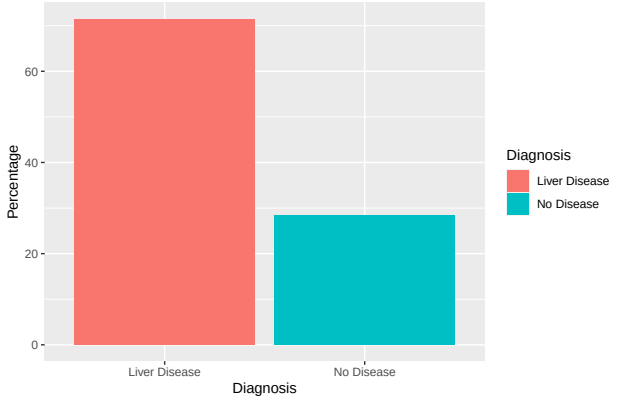


Figure 2: Percentage of Patients by Gender

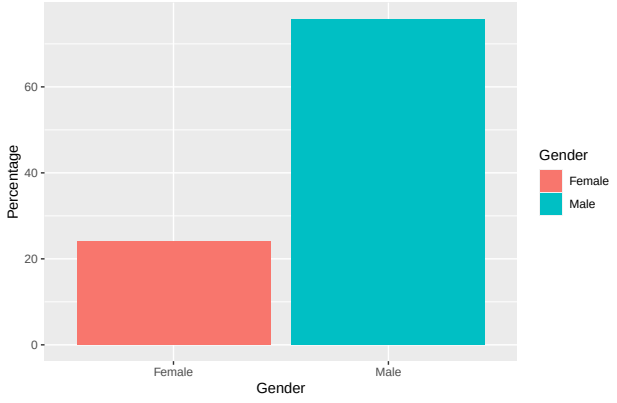


Figure 3: Liver Disease Rate by Gender

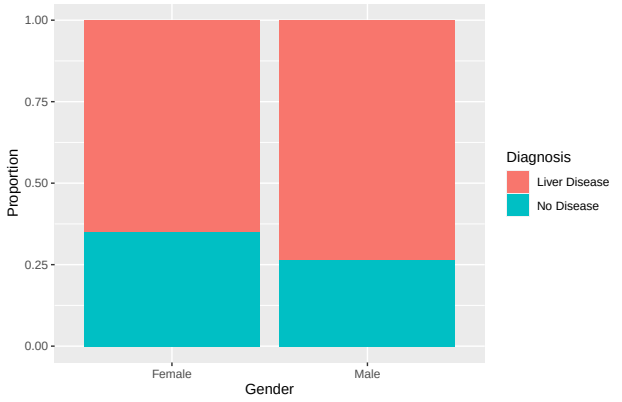


Figure 4: Gender Demographics of Diagnostic Groups

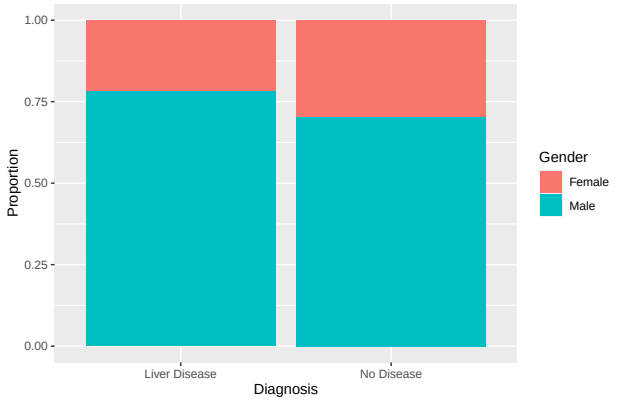


Table 1 contains the average, median, mode, minimum, maximum, and standard deviations of patient ages in each diagnostic category. Note, patients aged older than 90 were recorded as being aged 90. There appears to be somewhat more age variability in healthy patients than in liver patients.

Table 1: Summary Statistics of Patient Ages in Years

Diagnosis	Mean	Median	Mode	Min.	Max.	S.D.
All Patients	44.78	45	60	4	90	16.22
Liver Patients	46.14	46	60	7	90	15.69
Healthy Patients	41.36	41	38	4	85	17.06

Liver patients generally tend to be somewhat older than healthy patients, but this difference is negligible among female patients.

Figure 5: Median Patient Age by Diagnosis

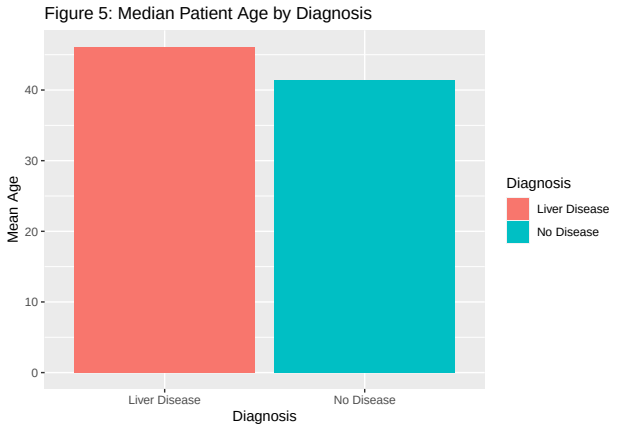
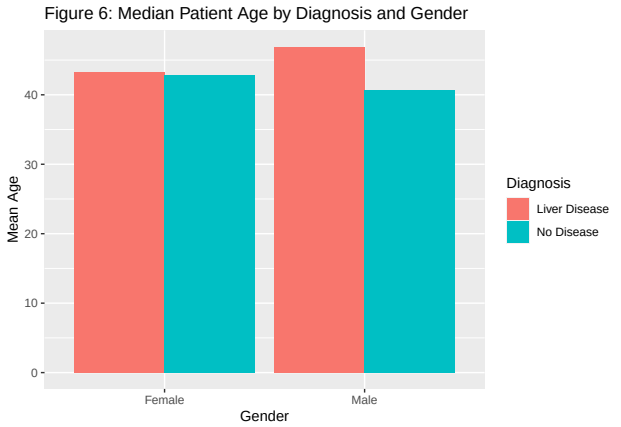


Figure 6: Median Patient Age by Diagnosis and Gender



Figures 7 and 8 plot the patient ages (in total and differentiated by diagnosis, respectively) in density distributions. Due to graphic smoothing, the modal spikes at age 60 among all patients and liver patients are not visible in either figure. The modal spike at age 38 among healthy patients is visible in Figure 8. Density plots with smoothing were used because they conveyed the most visually discernible age distributions. Early attempts to use histograms instead of density plots resulted in distorted figures.

Figure 7: Patient Age Distribution

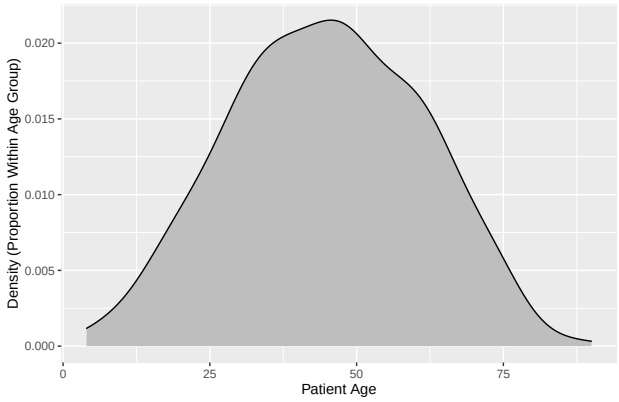
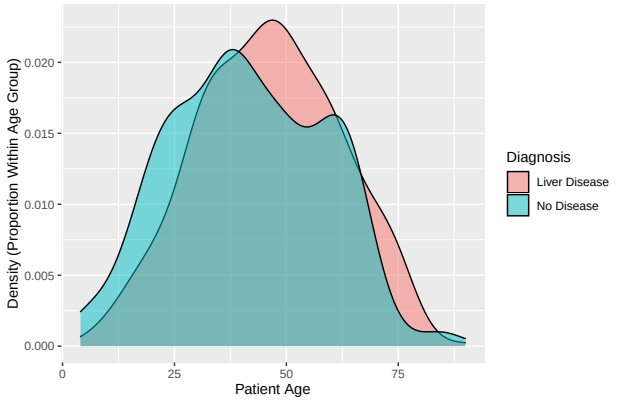


Figure 8: Patient Age Distribution by Diagnosis



Matrix Creation and Standardization

Table 2 demonstrates that, while the `il_data` variables correlate with diagnostic outcome (`Dataset`) on comparable orders of magnitude, the variables themselves yield values that differ more significantly from each other in scale. As such, the predictors (variables excluding diagnostic outcome) are not readily comparable to one another. Mathematical attempts to compare them may process low-scale variables, such as age and gender, as insignificant in comparison to high-scale variables such as alkaline phosphotase and alamine aminotransferase, generating misleading inferences of relationships between the variables. Figure 11, a heatmap of the unstandardized predictor variables, presented in the section below, demonstrates such an outcome.

Table 2: Variable Correlation w/ Dataset and Summary Statistics.

	Correlation	Min.	Max.	Mean	Median	S.D.
Age	-0.1331636	4.0	90.0	44.7823834	45.00	16.2217857
Gender	-0.0813494	1.0	2.0	1.7582038	2.00	0.4285417
Total_Bilirubin	-0.2202179	0.4	75.0	3.3153713	1.00	6.2277164
Direct_Bilirubin	-0.2462729	0.1	19.7	1.4941278	0.30	2.8164986
Alkaline_Phosphotase	-0.1833633	63.0	2110.0	291.3661485	208.00	243.5618633
Alamine_Aminotransferase	-0.1631169	10.0	2000.0	81.1260794	35.00	183.1828449
Aspartate_Aminotransferase	-0.1518345	10.0	4929.0	110.4145078	42.00	289.8500344
Total_Protiens	0.0336138	2.7	9.6	6.4816926	6.60	1.0846411
Albumin	0.1597696	0.9	5.5	3.1385147	3.10	0.7944347
Albumin_and_Globulin_Ratio	0.1631314	0.3	2.8	0.9470639	0.93	0.3195921
Dataset	1.0000000	1.0	2.0	1.2849741	1.00	0.4517924

Standardization is a form of pre-processing that converts a value x to the number z of standard deviations σ by which it stands above or below the average μ of a measured variable.[28][29] A variable would yield negative standardized scores for below-average values and positive standardized scores for above-average values. The formula for doing so follows: [29]

$$z = \frac{x - \mu}{\sigma}$$

Standardizing the `il_data` variables allows one to compare them on a more consistent scale and derive relationships between them more accurately. In order to do so and compute other linear algebra operations in R, one must convert the dataset to a matrix `x`. This is done via the following command:

```
x <- as.matrix(il_data[,1:10])
```

The matrix `x` is then be converted to a matrix `X` in which every variable is standardized:

```
X <- sweep(x, 2, colMeans(x), FUN = "-")
X <- sweep(X, 2, colSds(x), FUN = "/")
```

Note, the matrices consist only of the predictor variables, not the outcome variable. The average value and standard deviation for every standardized variable is equal to approximately 0 and 1 respectively. Table 3 demonstrates that standardization does not alter the correlations between standardized predictors and outcomes:

Table 3: Correlations Between Standardized Predictors and Diagnosis.

	Correlation
Age	-0.1331636
Gender	-0.0813494
Total_Bilirubin	-0.2202179
Direct_Bilirubin	-0.2462729
Alkaline_Phosphotase	-0.1833633
Alamine_Aminotransferase	-0.1631169
Aspartate_Aminotransferase	-0.1518345
Total_Protiens	0.0336138
Albumin	0.1597696
Albumin_and_Globulin_Ratio	0.1631314

One can compute a matrix of Euclidian distances between all 579 remaining patients. All diagonal values of the 579x579 matrix are equal to 0. The average distance between healthy patients is 3.1 while the average distance between liver patients is 3.77. Transposing the matrix before calculating the distances creates a 10x10 matrix, which will serve in the visualization of heatmaps below.

Unsupervised Machine Learning

Three forms of unsupervised machine learning — hierarchical clustering, heatmap visualization, and principal component analysis — were used to continue the exploratory analysis. Note, these algorithms were not used to predict patient diagnoses.

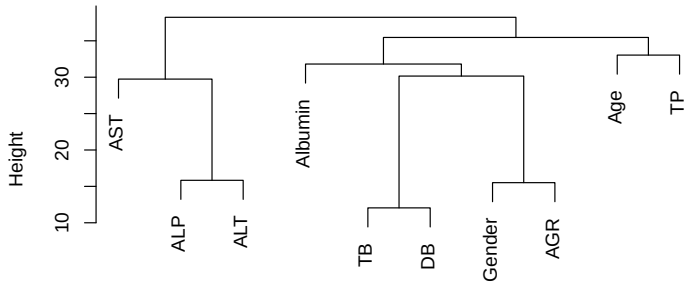
Hierarchical Clustering

Using a matrix of distances, hierarchical clustering defines each observation as a distinct group and clusters it with its closest neighbor into a higher-order group. The process is repeated until a single highest-order group contains all observations. A dendrogram, as seen in Figure 9, visually represents this process. The height at which observations are split into two different groups — from top to bottom — represents the distance between them. [25] Figure 9 is a dendrogram of clusters derived from distances between predictors.

To conserve visual space, "Total_Proteins" was abbreviated as TP, "Total_Bilirubin" as TB, "Direct_Bilirubin" as DB, "Alkaline_Phosphotase" as ALP, "Aspartate_Aminotransferase" as AST, "Alanine_Aminotransferase" as ALT, and "Albumin_and_Globulin_ratio" as AGR. The most closely related predictors appear to be total and direct bilirubin. Gender appears to be relatively close to albumin:globulin ratio while alkaline phosphotase is apparently similarly close to alanine aminotransferase;

the two pairs are not closely related to one another. Alaline aminotransferase is relatively distant from aspartate aminotransferase; albumin is similarly distant from albumin:globulin ratio; total protein is even more distant from albumin and AGR.

Figure 9: Patient Feature Dendrogram



Heatmaps

A heatmap is a grid-like image that uses color to represent correlations between two sets of hierarchical clusters. The columns and rows of a heatmap are ordered with respect to the dendrograms of the clustering algorithm used to derive it. [25] As seen in Figures 10-11, the hiearchical clustering algorithm employed above was used to derive heatmaps of correlations between hierarchically clustered predictors. Red colors represent positive correlations, blue colors represent negative correlations, and white represents zero or negligible correlations. Dark colors represent strong correlations; light colors represent weak ones.

Figure 11 represents the results of heatmap visualization performed on unstandardized matrices; most of the correlations are weak or negligible, even along the diagonal line, which should consist of perfect correlations. The two figures dmonstrate that standardization affects the order in which the predictors are clustered. Figure 10 — computed on standardized matrices — yields more appropriate values for diagonal correlations. Interestingly, most correlations outside of the diagonal in Figure 10 are negative; the rest are weak or negligible.

Figure 10: 10x10 Standardized Heatmap

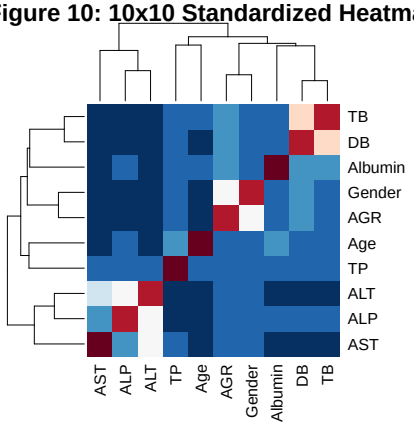
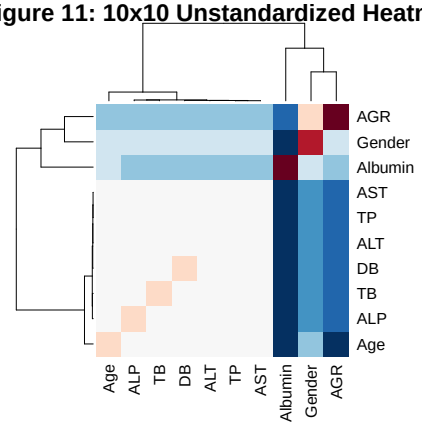


Figure 11: 10x10 Unstandardized Heatmap



Principal Component Analysis

Principal component analysis (PCA) is the process of applying orthogonal transformations to a matrix such as to summarize its variance into a series of principal components. The first principal component should contain the largest amount of variance in the matrix; each following principal component should contain successively less variance until accumulatively accounting for all of it. PCA reduces the matrix's dimensionality while preserving as much of its data as possible, providing for more efficient analysis. [26] R provides an algorithm for performing PCA via a simple command:

```
pca <- prcomp(X)
```

Corresponding to ten predictors, ten principal components are derived for the matrix. They are summarized as follows:

Table 4: Principal Component Analysis Summary (continued below)

	PC1	PC2	PC3	PC4	PC5
Standard deviation	1.666	1.424	1.17	1.03	0.9584
Proportion of Variance	0.2775	0.2027	0.1369	0.1061	0.09186
Cumulative Proportion	0.2775	0.4802	0.6172	0.7232	0.8151

	PC6	PC7	PC8	PC9	PC10
Standard deviation	0.8963	0.813	0.4511	0.3545	0.2352
Proportion of Variance	0.08034	0.0661	0.02034	0.01257	0.00553
Cumulative Proportion	0.8954	0.9616	0.9819	0.9945	1

As shown in Figure 12, liver patients tend to yield higher values in the second principal component and lower values in the first principal component than healthy patients do. However, as Table 4 indicates, those two principal components account for less than half of the variance. Such a proportion, while significant, is not sufficient to represent the full matrix. 96.16% of the variance is contained in the first seven principal components; such a proportion is more appropriate. Plotting the first seven principal components in Figure 13 demonstrates more nuance. It appears that differences between healthy and liver patients are most pronounced in terms of extreme values. They generally appear to be small in terms of average and even interquartile range values.

Figure 12: Principal Components 1, 2 and Diagnosis

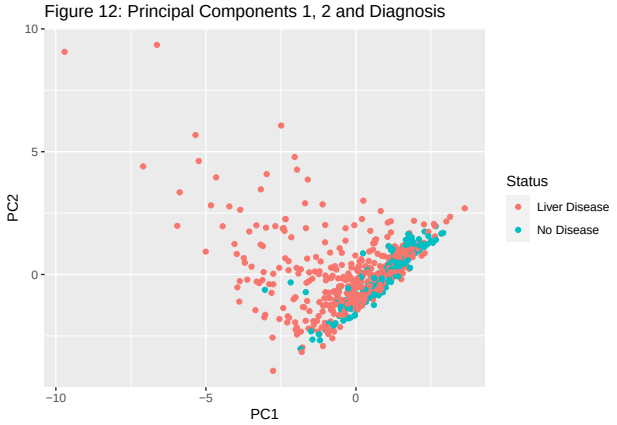
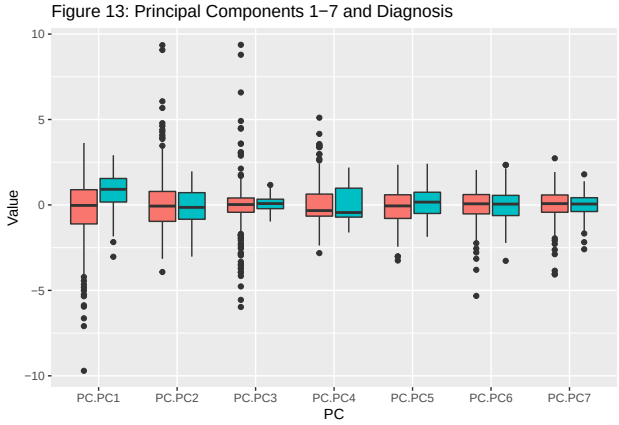


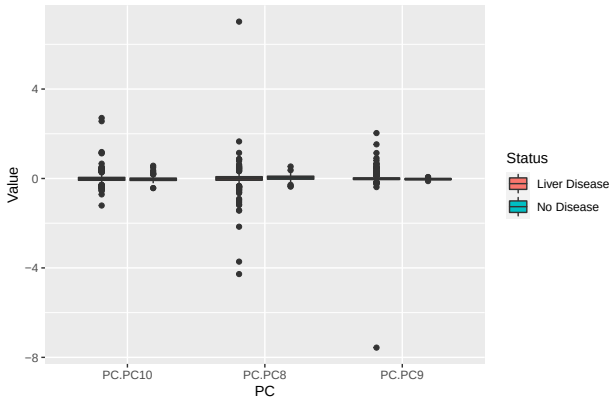
Figure 13: Principal Components 1−7 and Diagnosis



The last three principal components contain so little variance that plotting them presents little discernible

or useful information:

Figure 14: Principal Components 8–10 and Diagnosis



Set Splitting for Supervised Machine Learning

The lack of strong positive correlations between hierarchically clustered predictors in Figure 10 suggests that the method would not directly predict liver disease diagnosis reliably. Likewise, the significant overlap between liver and healthy patients across principal components in Figure 12 suggests that PCA would not directly predict liver disease diagnosis reliably either. Indeed, the relationship between the dataset's chemicals and liver disease is complicated; the chemicals may additionally vary for reasons unrelated to liver disease. For example, alkaline phosphatase are higher in children and teenagers than in adults due to continuing bone development. Abnormal levels of the protein may also indicate bone disease rather than liver disease; they may also indicate no health problems. [15]

Supervised machine learning is the next logical candidate for attempting to predict liver disease diagnosis. To use it, one must partition the dataset into training and testing sets. Google and Amazon recommend allocating 70-80% of a full dataset to a training set and the rest to a test set in machine learning applications. [31][32] Training sets are generally larger than testing sets because machine learning models are always trained on the former. The larger the training set, the more accurately it represents the original dataset and the more accurately the model learns, as per the law of large numbers. [33] The testing set, however, must also be large enough to yield statistically meaningful results and sufficiently represent the original dataset. That is, it should not have different structural characteristics from the training set. [31]

Indeed, varying the training:testing set proportion ratio varies the extend to which the testing set over-represents patients of one diagnostic category and, conversely, under-represents patients of the other. The code below calculates the difference between the average proportion diagnosed with liver disease for the training and testing sets. As the results are expected to yield random variance, each calculation is reiterated 1000 times and averaged; the `set.seed()` command is used to obtain consistent results.

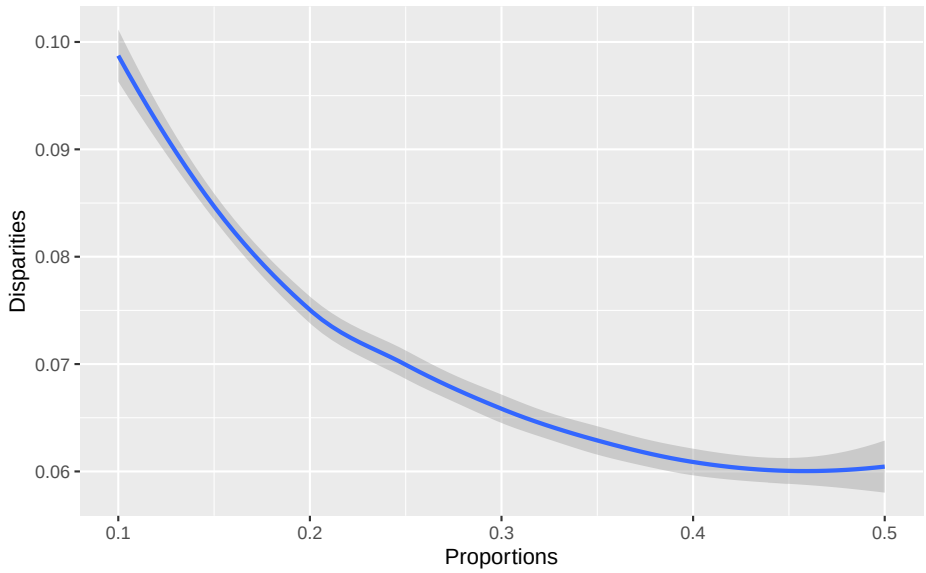
41 proportions allocated to the test set — ranging from 0.10 to 0.50 — were used, yielding a total of 41,000 calculations. Due to the time required to compute 41,000 calculations, the number of reiterations was not raised above 1000. Different proportions variously under-represented or over-represented liver patients in the testing set relative to the training set; as such, the absolute values of the differences were calculated to yield a consistent relationship. The final values were doubled to account for the converse representation of healthy patients.

```
props <- seq(0.1, 0.5, 0.01)
set.seed(1, sample.kind = "Rounding")
disp2 <- sapply(props, function(p){
  disp <- replicate(1000, {
    ind <- createDataPartition(il_data$Dataset,
```

```
                                times = 1, p = p,
                                list = FALSE)
    trainY <- as.factor(il_data$Dataset[-ind])
    testY <- as.factor(il_data$Dataset[ind])
    abs(mean(trainY == 1) - mean(testY == 1))
  })
  return(mean(2*disp))
})
```

The results of the calculations above are visually represented in Figure 15. Large test set proportions yield lower disparities in representation between training and testing sets, but diminishing returns are observable as one approaches a 50:50 ratio. Conversely, the disparities approach 0.10 as the testing set proportions approach 0.10. The recommended test set proportions of 0.2 and 0.3 yield respective disparities of 0.075 and 0.065.

Figure 15: Diagnostic Disparity Between Training and Test Sets, by Test Set Proportion



Ultimately, the decision was to use a training:testing ratio of 80:20, as per Google's recommendation. [31] Its disparity is higher than that of a 70:30 ratio but by a relatively small difference. The advantage of a larger training set was believed to be more compelling than the disadvantage of a smaller testing set, but such can be debated. The training and test sets were coded below. `trainX` and `testX` contain the matrix variables split with respect to the partition; `trainY` and `testY` contain the diagnoses corresponding to the respective patients in the previous datasets.

```
set.seed(1, sample.kind = "Rounding")
ind <- createDataPartition(il_data$Dataset, times = 1, p = 0.2, list = FALSE)
trainX <- X[-ind,]
testX <- X[ind,]
trainY <- as.factor(il_data$Dataset[-ind])
testY <- as.factor(il_data$Dataset[ind])
```

Maintaining a representative test set is important because the final accuracy results may be erroneous for models that are extremely imbalanced in sensitivity (accuracy in predicting positive cases) and specificity

(accuracy in predicting negative cases). For example, by virtue of 71.5 % of the patients having liver disease, an algorithm that simply predicts every case to be positive would yield an accuracy rating of 71.5 % for the entire dataset. However, one can mitigate this potential effect by calculating the sensitivity and specificity for all models. To do so, the indices of liver and healthy patients in the test set are catalogued via the following command:

```
ldi <- which(as.numeric(testY) == 1)
ndi <- which(as.numeric(testY) == 2)
```

Cross Validation

A common goal of machine learning is to find an algorithm that produces predicted outcomes $\hat{Y}$ for real outcomes Y that minimize mean squared error (MSE) for N outcomes i: [27]

$$MSE = E\{\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2\}$$

$E\{\}$ denotes the expectation value of the true MSE while $\hat{MSE}$ denotes the observed or apparent mean squared error: [27]

$$\hat{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

Due to the randomness of data examined in machine learning exercises, apparent error varies randomly from true error. Some algorithms may yield different apparent errors due only to chance. The data examined in this exercise is a small sample of a large population; patterns in the observed sample may not resemble patterns in the population. Training an algorithm on the same dataset used to calculate the apparent error may constitute overtraining, often yielding underestimates of the true error. [27]

Cross validation is an approach that mitigates those risks by randomly and repeatedly partitioning training sets into sub-training and validation sets. The true error is reformulated as an average of many random errors obtained by applying the algorithm to B random samples b of the original dataset: [27]

$$MSE = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i^b - Y_i^b)^2$$

Some algorithms feature independently variable tuning parameters λ, thus one may repeat the entire process of cross validation for different values of λ. As such, MSE varies as a function of λ: [27]

$$MSE(\lambda) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i^b(\lambda) - Y_i^b)^2$$

The value of λ that yields the lowest MSE, or, conversely, the highest percent accuracy, should be chosen as the final tuning parameter for the algorithm. In theory, B can be infinite. In practice, cross validation uses as many random samples or partitions as possible. During the process of cross validation, algorithms are trained on data in sub-training sets to predict outcomes in validation sets. Repeating the process allows one to obtain accuracy estimates from within the training data without overtraining and thereby refine the algorithm before training it to predict outcomes in the testing set. [27]

One method, known as K-fold cross validation, entails partitioning the training set into K equally sized, non-overlapping subsets; each subset is sequentially employed as a validation set while the others are collectively

used to train the algorithm in K trials. Each subset consists of $M = N/K$ observations, hence the apparent error for each subset and tuning parameter in K-fold cross validation follows: [27]

$$\hat{MSE}_b(\lambda) = \frac{1}{M}\sum_{i=1}^{M}(\hat{Y}_i^b(\lambda) - Y_i^b)^2$$

Another method, known as bootstrap cross validation, randomly partitions the training data and resamples it (into any selected ratio) during the process of repitition; that is, the different validation sets for each trial may overlap. [27] The R function `caret::train()` is capable of performing cross-validation through a single command. By default, it bootstraps 25 validation sets that each contain 25% of the original data. Via the function's argument `method` and the function `caret::trainControl()`, it may perform K-fold cross validation as well. [34]

During the HarvardX course PH125.8x, students were instructed to use it in training several algorithms — logistic regression, LDA, QDA, LOESS, kNN, and random forest — via bootstrapping to predict breast cancer diagnoses. This exercise will experiment with the same methods by employing K-fold alongside bootstrap cross validation. K will be set to 10 in every application, as per the default `caret` arguments. Due to the time required to compute certain algorithms, larger values of K will not be used. Preliminary testing has found the computation time to be equal between default bootstrap and K-fold algorithms.

The formula for MSE is known as a loss function. [24] Equivalently, one can assess algorithms by computing their percent accuracy manually or via the R function `caret::confusionMatrix()`, as shall be done in this report. [34] Percent accuracy ratings were used for their presentability and accessibility to readers not familiar with the subject studied.

k-Means Clustering

After one defines a number k of clusters and respective centers, the k-means algorithm iteratively assigns each observation in a dataset to a cluster with whose center is closest to it. Subsequently, each cluster is redefined with respect to a centroid or average between its center and newly collected observations. The process is repeated until the centers converge. [25] Two clusters can be defined from the Indian Liver Patient Records database — liver patients and healthy patients. The following algorithm was derived in PH125.8x to predict which clusters — derived from training set observations via the R function `stats::kmeans()` — would fit testing set observations:

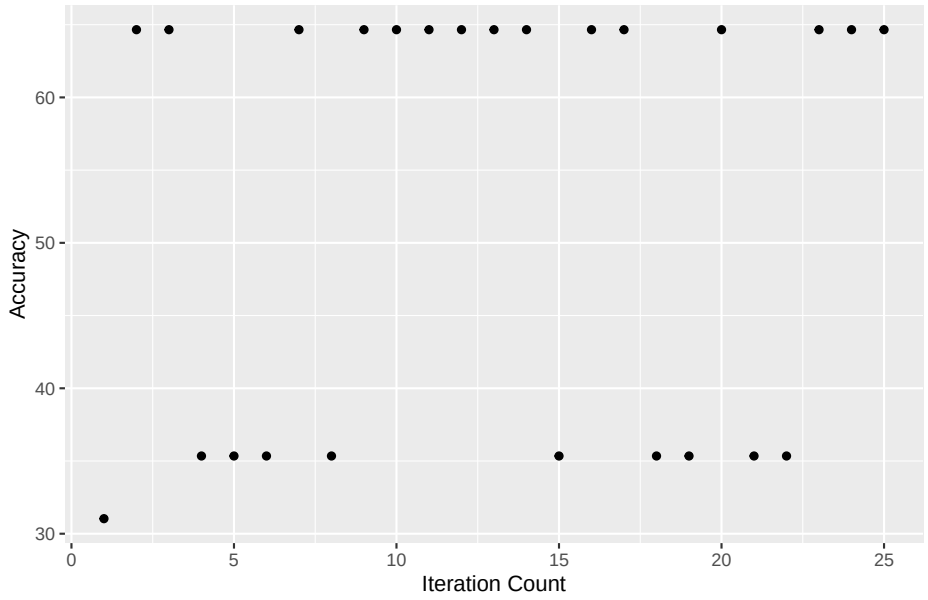
```
predict_kmeans <- function(x, k){
  c <- k$centers
  d <- sapply(1:nrow(x), function(i){
    apply(c, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(d))
}
```

As such, the distinction between supervised and unsupervised machine learning is blurred. Patients would be assigned to clusters whose centers are least distant from their conditions. Due to the custom programming of the method, `caret::train()` does not support it in a pre-programmed capacity. One can randomly partition the training set into a 75:25 subtraining:validation ratio and predict patient diagnoses in the validation set with 31% accuracy. `stats::kmeans()` selects a random value to be the first cluster center. However, by altering the function's `nstart` argument, one can repeat the process of k-means clustering with each repitition using a different starting point. The following code determines the percent accuracy for different numbers, ranging from 1 to 25, of repititions and respective starting points:

```
set.seed(1, sample.kind = "Rounding")
Accuracies <- sapply(1:25, function(i){
  k <- kmeans(trainX[-sub_ind,], centers = 2, nstart = i)
  p <- predict_kmeans(trainX[sub_ind,], k)
  100*mean(p == as.numeric(trainY[sub_ind]))
})
```

The results of the code above are plotted in Figure 16. It appears that different `nstart` values yield randomly fluctuating binary accuracies. The maximum value of `nstart`, 25, yields the high value of 64.7% accuracy and is chosen to be consistent with the standard cross-validation methods associated with `caret::train()`.

Figure 16: k−Means Accuracy by Number of Random Starting Points



The k-means algorithm is trained on the entire, non-partitioned training set to predict outcomes in the testing set via the following code:

```
set.seed(1, sample.kind = "Rounding")
k <- kmeans(trainX, centers = 2, nstart = 25)
p_km <- predict_kmeans(testX, k)
```

The methods used to train the algorithm bear some resemblence to bootstrap cross validation in principle but not in entirety; the training set was not repeatedly partitioned. Accordingly, the training process may not yield as accurate a model as proper cross-validation would. The accuracy, sensitivity, and specificity of the final k-means algorithm are coded below:

```
acc_km <- mean(p_km == testY)
sen_km <- mean(p_km[ldi] == 1)
spe_km <- mean(p_km[ndi] == 2)
```

Logistic Regression

Logistic regression is a type of generalized linear model that predicts a binary outcome. As a probabilistic curve, it estimates the conditional probablity that an outcome Y would yield a specific value given certain

input values x_i for the predictor variables X_i. As the dataset consists of one outcome and eleven predictor variables, the conditional probability $p(x_i)$ that a patient has liver disease is expressed as such: [35]

$$p(x_i) = Pr(Y = 1|X_1 = x_1, X_2 = x_2, ..., X_i = x_i, ..., X_{11} = x_{11})$$

Due to the presence of more than one predictor, the conditional probability is converted to a logarithmic odds g via logistic transformation: [35]

$$g(p) = log\left(\frac{p}{1-p}\right)$$

The logistic transformation allows one to model the conditional probability as a linear regression: [35]

$$g\{p(x_i)\} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i + ... + \beta_{11} x_{11}$$

`caret::train()` calculates the regression coefficients β_i to yield the best logistic fit after one sets its argument `method = "glm"`:

```
set.seed(1, sample.kind = "Rounding")
b_fit_glm <- train(trainX, trainY, method = "glm")
```

By default, the code above trains the logistic algorithm via bootstrapping, yielding 68.8% accuracy in the process. The final parameters are then used to predict the outcomes in the testing set:

```
b_p_glm <- predict(b_fit_glm, testX)
b_acc_glm <- confusionMatrix(b_p_glm, testY)$overall["Accuracy"]
b_sen_glm <- mean(b_p_glm[ldi] == 1)
b_spe_glm <- mean(b_p_glm[ndi] == 2)
```

Below, the algorithm is re-coded to train via K-fold cross validation with a K of 10:

```
set.seed(1, sample.kind = "Rounding")
k_fit_glm <- train(trainX, trainY, method = "glm",
                   trControl = trainControl(method = "cv"))
```

An accuracy of 70.6% was observed in the process. The final K-fold parameters for the logistic regression algorithm were reapplied to predict the testing set outcomes:

```
k_p_glm <- predict(k_fit_glm, testX)
k_acc_glm <- confusionMatrix(k_p_glm, testY)$overall["Accuracy"]
k_sen_glm <- mean(k_p_glm[ldi] == 1)
k_spe_glm <- mean(k_p_glm[ndi] == 2)
```

Quadratic Discriminant Analysis

Bayes theorem dictates that the conditional probability of yielding an output A given an input B follows: [36]

$$Pr(A|B) = \frac{Pr(B|A)Pr(A)}{Pr(B)}$$

$Pr(A)$ denotes the absolute probability of A being true; $Pr(B)$ denotes the absolute probability of B being true; $Pr(A|B)$ denotes the conditional probability the input equals B when the output equals A. Let $-A$

denote the negative of the output A and $-B$ denote the negative of the input B. The theorem can be alternatively expressed as: [36]

$$Pr(A|B) = \frac{Pr(B|A)Pr(A)}{Pr(B|A)Pr(A) + Pr(B| - A)Pr(-A)}$$

Naive Bayes is a statistical approach that lets one approximate the probabilities $Pr(B|A)$ and $Pr(B| - A)$ respectively as independent distribution functions $f_{B|A}$ and $f_{B|-A}$. However, the distributions are often unknown. Quadratic discriminant analysis (QDA) is an application of Naive Bayes that assumes the distributions to be multivariate normal, that is, normally distributed in multiple dimensions. Such allows one to approximate the distributions by estimating their means and standard deviations, which pragmatically serve as algorithmic parameters. In the context of this report, the distributions can be termed $f_{X_1,\ldots,X_{11}|Y=1}$ and $f_{X_1,\ldots,X_{11}|Y=2}$. [37]

The QDA algorithms are trained via the following code:

```
set.seed(1, sample.kind = "Rounding")
b_fit_qda <- train(trainX, trainY, method = "qda")
set.seed(1, sample.kind = "Rounding")
k_fit_qda <- train(trainX, trainY, method = "qda",
                   trControl = trainControl(method = "cv"))
```

Their accuracy ratings measure 57.1% in the process of bootstrapping and 55.1% in the process of K-fold cross validation. Subsequently, the final models were used to predict testing set outcomes from all training set outcomes. Their ultimate performance will be evaluated in the *Results* section; the same shall apply to the test set predictive accuracy of all other algorithms in this reporrt.

Linear Discriminant Analysis

Linear discriminant analysis simplifies the QDA approach by assuming the correlation structure to be the same for all outputs, reducing the number of parameters one needs to estimate. [37]

```
set.seed(1, sample.kind = "Rounding")
b_fit_lda <- train(trainX, trainY, method = "lda")
set.seed(1, sample.kind = "Rounding")
k_fit_lda <- train(trainX, trainY, method = "lda",
                   trControl = trainControl(method = "cv"))
```

One trains the LDA algorithms via the command `caret::train(method = "lda")`. The respective accuracies determined via bootstrap and K-fold cross validation are 68.6% and 69.8%.

Locally Estimated Scatterplot Smoothing

Locally Estimated Scatterplot Smoothing (LOESS), or local regression, is a model that approximates an apparently randomly fluctuating curve as a piecewise smooth function composed of linear regressions, that is, a series of straight lines pointed in different directions. The function is modelled over a series of "windows" or "bins" that contain equal numbers of observations. A minimum independent value x_0 sets the range for each window such that $|x_i - x_0|$ is less than or equal to a certain maximum value for an independent variable x_i. For each such window, the expectation value of the outcome Y_i conditioned on the value of each predictor X_i is modelled: [38]

$$E[Y_i|X_i = x_i] = \beta_0 + \beta_1(x_i - x_0)$$

Local regression defines a new window and fits a line within it for every point x_0 on the domain, achieving the characteristic smoothness of the model. The fitted outcome at each x_0 is denoted as $\hat{f}(x_0)$. In fitting each local regression, LOESS aims to minimize a weighted version of the sum of least squares: [38]

$$\sum_{i=1}^{N} w_0(x_i)[Y_i - \{\beta_0 + \beta_1(x_i - x_0)\}]^2$$

The process of defining weights involves the Turkey tri-weight function: [38]

$$W(u) = \begin{cases} (1 - |u|^3)^3 & \text{if } |u| \leq 1 \\ 0 & \text{if } |u| > 1 \end{cases}$$

Denoting the window size as $2h$, the weights are defined as such: [38]

$$w_0(x_i) = W\left(\frac{x_i - x_0}{h}\right)$$

The window size is adjustable via the argument `span`. `degree` sets the order of the model equation; the default 1 sets it to a monomial equation while any higher value sets it to a polynomial. Attempts to use higher values for `degree` resulted in system failure, indicating that polynomial functions may not fit the data. In training LOESS to predict patient diagnosses via bootstrapping, `span` was independently varied from 0.45 to 0.55 while `degree` was kept at a constant value of 1:

```
s <- seq(0.45, 0.55, 0.01)
set.seed(1, sample.kind = "Rounding")
b_fit_loess <- train(trainX, trainY, method = "gamLoess",
                     tuneGrid = data.frame(span = s, degree = 1))
```

A `span` value of 0.52 yielded the highest accuracy 68.1% during the training process. Figure 16 presents the full relationship between bootstrap LOESS fit accuracy and span. Training LOESS via K-fold cross validation yielded the highest accuracy at a `span` of 0.45, 68.9%, regardless of whether the `span` values varied from 0.45 to 0.55 or from 0.35 to 0.45. Figure 17 represents the relationship between `span` values in the latter range and K-fold LOESS fit accuracies. Originally, `span` sequences ranged from 0.45 to 0.65 and from 0.25 to 0.45, yielding the same results observable in this report; the sequences were shortened to conserve computation time.

Figure 16: Loess Fit Accuracy by Span (Bootstrap)

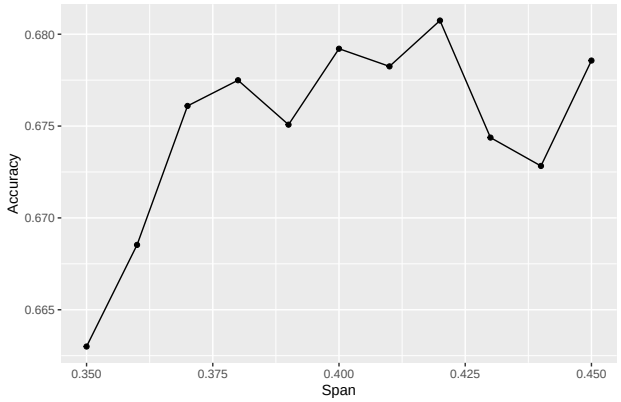
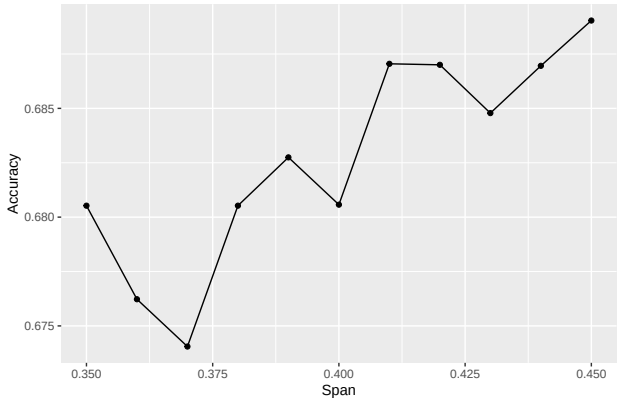


Figure 17: Loess Fit Accuracy by Span (K−fold)



k-Nearest Neighbors

To estimate the conditional probability $p(x_i)$ that the outcome fits a certain value given several predictors x_i, the k-Nearest Neighbors (kNN) algorithm defines the distance between all observations with respect to their features. For every point $(x_1, ..., x_i, ..., x_{11})$ with an unknown outcome $p(x_i)$, kNN searches for a number k of the points, or *neighbors*, nearest to it and averages their respective outcomes; the result is the predicted outcome $\hat{p}(x_i)$. In principle, it functions similarly to the bin smoothing method of LOESS. Each set of neighbors is referred to as a neighborhood. [27]

If k is equal to 1, each neighborhood would consist only of one point, that which is studied. Perfect accuracy would be apparent in the training set because each point is used to predict itself. Such a situation is called overfitting and needs to be avoided in machine learning. Similarly small values of k also lead to overfitting. Conversely, overly large values of k lead to over-smoothing. Over-smoothed algorithms approximate linear regressions. Linear regressions are inappropriate when the relationships between predictors and outcomes are not linear. [27]

Accuracy is an important metric for evaluating different values of k, but, as demonstrated in Figures 18-19, not the sole evaluating metric one should use. `caret::train()` computes Cohen's kappa statistic for each model it derives. Kappa is a measure of the extent to which two different methods of categorical assessment assign the same outcome to a given observation. A kappa value of 1 indicates perfect agreement whereas one of 0 indicates unpredictable agreement, that is, agreement equivalent to chance. [39][40] In a sense, it is the converse of a p-value, which yields the positive probability that a statistical outcome is a product of chance. Figures 18-19 plot the accuracy ratings and kappa statistics for each value of k used in bootstrap and K-fold cross validation. Both methods are trained via the code below:

```
k <- seq(1, 100, 1)
set.seed(1, sample.kind = "Rounding")
b_fit_knn <- train(trainX, trainY, method = "knn", tuneGrid = data.frame(k = k))

set.seed(1, sample.kind = "Rounding")
k_fit_knn <- train(trainX, trainY, method = "knn", tuneGrid = data.frame(k = k),
                   trControl = trainControl(method = "cv"))
```

As with all other algorithms trained via `caret::train()`, the K-fold version is trained by adjusting the `trControl` argument. It yielded a peak accuracy of 0.7% and a kappa of 0.12 for a k value of 34. Both forms of cross-validation yield diminishing returns with respect to accuracy as k approaches 100. More importantly, as k approaches 100, kappa approaches 0. The bootstrap training yields its highest accuracy of 69.7% for a k of 87; however, it corresponds to a kappa statistic of 0.02. To demonstrate the effects of over-training, one bootstrap algorithm is trained on a k of 87:

```
b_p_knn_87 <- predict(b_fit_knn, testX)
```

Figure 18: kNN Fit Results by k (Bootstrap)

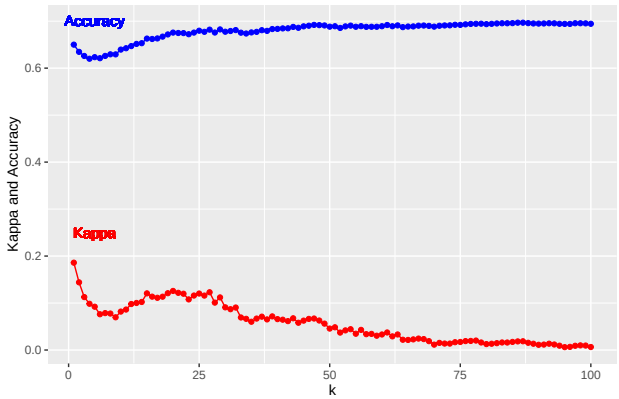
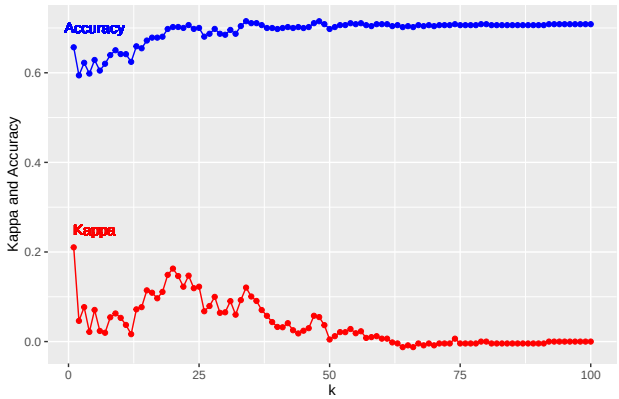


Figure 19: kNN Fit Results by k (K−fold)



Another bootstrap algorithm is trained, for predictive purposes, on a k of 27. The value yields a similarly high accuracy of 68.2% and a kappa statistic of 0.12:

```
set.seed(1, sample.kind = "Rounding")
b_fit_knn <- train(trainX, trainY, method = "knn", tuneGrid = data.frame(k = 27))
b_p_knn <- predict(b_fit_knn, testX)
b_acc_knn <- mean(b_p_knn == testY)
b_sen_knn <- mean(b_p_knn[ldi] == 1)
b_spe_knn <- mean(b_p_knn[ndi] == 2)
```

Random Forest

A decision tree, graphically represented as a flow chart, is an algorithm that partitions the predictors of a dataset and evaluates them sequentially in predicting an outcome for an observation. The tree begins with a single predictor variable and determines whether an observation passes a certain threshold value. Afterward, the tree either predicts an outcome for the observation or determines whether it passes a threshold for another predictor. The following predictor may vary with respect to whether the observation passed the preceding predictor's threshold. The process repeats until an outcome is predicted for the observation. Each discrete outcome — following one or a series of predictor evaluations — is displayed at an endpoint, or node, of a decision tree diagram. [37]

Random forest algorithms combine the predictions of multiple decision trees that differ from each other randomly. When based on regression trees, the predicted outcome is given as the average of each tree's prediction for a given observation. As the data in this project consist of binary categorical outcomes, the random forests used are based on classification trees; the prediction of each algorithm is the most common prediction of all trees for a given observation. To ensure randomness, each tree is trained on a bootstrap aggregation (bagging) of the training data. [37] As caret::train() performs cross validation in addition to bagging, it yields relatively long processing time in computing random forest algorithms. As such, the tuning parameter `mtry` is varied from only 1 to 5 in increments of 1:

```
m <- seq(1, 5, 1)
set.seed(1, sample.kind = "Rounding")
b_fit_rf <- train(trainX, trainY, method = "rf",
                  tuneGrid = data.frame(mtry = m),
                  importance = TRUE)
set.seed(1, sample.kind = "Rounding")
k_fit_rf <- train(trainX, trainY, method = "rf",
                    tuneGrid = data.frame(mtry = m),
                    trControl = trainControl(method = "cv"),
                    importance = TRUE)
```

Figures 20-21 visualize the relationship between `mtry` and algorithm accuracy. The random forest algorithm yielded peak accuracies at an `mtry` of 1 for both bootstrap and K-fold cross validation at 70.8% and 73% respectively. Early operations varied `mtry` from 1 to 17 and yielded the same results; the sequences were shortened to conserve computational time.

Figure 20: Random Forest Accuracy by mtry (Bootstrap)

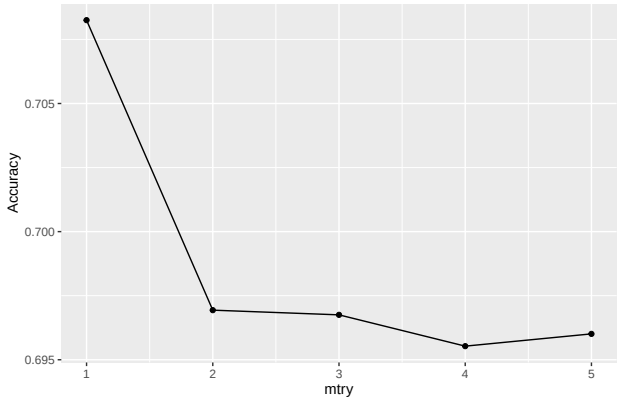
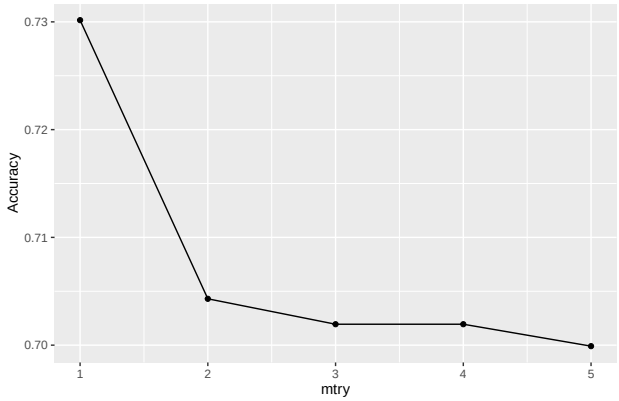


Figure 21: Random Forest Accuracy by mtry (K−fold)



Random forests yield more stable results than decision trees do, but their processes are more difficult to interpret. One cannot visualize them as flow charts. [37] To mitigate the disadvantage of random forests, `caret::train()` allows one to evaluate the importance of each variable via the `importance` argument, as seen in the code above. A rating of 100 indicates the highest importance while one of 0 indicates the lowest importance. Tables 5 and 6 represent the predictor rankings for random forests trained via bootstrap and K-fold cross validation:

Table 6: Bootstrap Random Forest Variable Importance

	Importance
Direct_Bilirubin	100.0000000
Alkaline_Phosphotase	87.1277562
Alamine_Aminotransferase	84.6746026
Aspartate_Aminotransferase	83.0581476
Total_Bilirubin	82.3211683
Age	48.2620235
Albumin_and_Globulin_Ratio	13.9361151
Gender	7.6565122
Total_Protiens	0.4659184
Albumin	0.0000000

Table 7: K-fold Random Forest Variable Importance

	Importance
Aspartate_Aminotransferase	100.000000
Direct_Bilirubin	97.592271
Alamine_Aminotransferase	95.553286
Total_Bilirubin	93.796862
Alkaline_Phosphotase	75.237770
Age	42.714696
Gender	15.510670
Albumin	6.234122
Albumin_and_Globulin_Ratio	4.532212
Total_Protiens	0.000000

Ensemble

An ensemble combines the results of multiple algorithms and can lead to increased accuracy. [41] Three ensembles were programmed for this project — one to combine the results of all bootstrap-derived algorithms, another to combine those of all K-fold-derived algorithms, and a third to combine those of the bootstrap-derived, K-fold-derived, and k-means algorithms. Based off an ensemble designed for PH125.8x, they predict a given patient to have liver disease if at least 50% of its algorithms predict so; otherwise, they predict a patient to have no disease. All three ensembles are coded below:

```
b_ensemble <- cbind(b_p_glm, b_p_qda, b_p_lda, b_p_loess, b_p_knn, b_p_rf)
b_vote <- sapply(1:nrow(testX), function(i){
  ifelse(mean(b_ensemble[i,] == 1) >= 0.5, 1, 2)
})

k_ensemble <- cbind(k_p_glm, k_p_qda, k_p_lda, k_p_loess, k_p_knn, k_p_rf)
k_vote <- sapply(1:nrow(testX), function(i){
  ifelse(mean(k_ensemble[i,] == 1) >= 0.5, 1, 2)
})

c_ensemble <- cbind(k_p_glm, k_p_qda, k_p_lda, k_p_loess, k_p_knn, k_p_rf,
                    b_p_glm, b_p_qda, b_p_lda, b_p_loess, b_p_knn, b_p_rf, p_km)
c_vote <- sapply(1:nrow(testX), function(i){
  ifelse(mean(c_ensemble[i,] == 1) >= 0.5, 1, 2)
})
```

Results

Table 8: Model Results

Method	Accuracy	Sensitivity	Specificity	Balance
k-Means	55.17	50.00	70.00	60.00
Logistic Regression (Bootstrap)	73.28	84.88	40.00	62.44
LDA (Bootstrap)	70.69	89.53	16.67	53.10
QDA (Bootstrap)	60.34	47.67	96.67	72.17
LOESS (Bootstrap)	75.00	81.40	56.67	69.03
kNN (Bootstrap)	72.41	91.86	16.67	54.26
Random Forest (Bootstrap)	71.55	90.70	16.67	53.68
Logistic Regression (K-fold)	73.28	84.88	40.00	62.44
LDA (K-fold)	70.69	89.53	16.67	53.10
QDA (K-fold)	60.34	47.67	96.67	72.17
LOESS (K-fold)	73.28	79.07	56.67	67.87
kNN (K-fold)	70.69	93.02	6.67	49.84
Random Forest (K-fold)	70.69	86.05	26.67	56.36
Ensemble (Bootstrap)	75.00	91.86	26.67	59.26
Ensemble (K-fold)	73.28	89.53	26.67	58.10
Ensemble (Complete)	72.41	86.05	33.33	59.69
kNN (k = 87)	74.14	100.00	0.00	50.00

After the final parameters were trained, all algorithms were used to predict diagnoses in the test data from all those in the training data. The table above summarizes their results in four variables — accuracy, sensitivity, specificity, and balance. Balance is introduced as an average between sensitivity and specificity

to compensate for the fact that most of the patients have liver disease.

The 87-Nearest Neighbors model, (over-)trained via bootstrap, is included at the bottom as an example of what not to use. While yielding a high accuracy of 74.14%, it yielded a sensitivity of 100% and specificity of 0%, resulting in a balance of 50%. One may as well simply predict liver disease for every case, which, in practice, is no better than flipping a coin! As the purpose of this project is to predict diagnoses more accurately and efficiently, such that the burden on doctors is reduced, such a method is inappropriate. Prescriptions made on the false assumption of liver disease may also have adverse effects on patient health. The K-fold kNN algorithm, trained to a k of 34, yielded results that were comparably polarized, less accurate, and slightly less balanced. Over-training may have occurred.

In some algorithms — logistic regression, LDA, and QDA — training via K-fold or bootstrap cross validation made no difference. The K-fold version of LOESS simply yielded inferior sensitivity to that of its bootstrap counterpart. K-fold cross validation generated somewhat less sensitivity but notably more specificity than bootstrapping did in the random forest algorithm, yielding lower accuracy but higher balance. Both versions of LDA and random forest, along with the bootstrap version of kNN, only marginally exceeded 50% in terms of balance. The bootstrap ensemble yielded the highest absolute accuracy and higher balance than the K-fold ensemble did. However, the complete enseble yielded slightly higher balance and sensitivity than the bootstrap ensemble did. The K-fold kNN algorith may have driven the K-fold and complete ensembles' disadvantages.

Quadratic discriminant analysis yielded the highest balance of 72.17% for a modest accuracy of 60.34%. It yielded the highest specificity of 96.67% and the lowest sensitivity of 47.67%. In contrast, the ensembles yielded 72.41-75.00% accuracy and 58.10-59.69% balance, rivalling the random forest and kNN algorithms with sensitivities of 86.05-91.86% and outperforming them with specificities of 26.67-33.33%. Ensembles are the most balanced algorithms for identifying liver disease while quadratic discriminant analysis is best at ruling it out. Medical professionals may differentially value either strategy depending on their situations, hence the most appropriate choice may vary.

LOESS and k-means were the only algorithms to yield sensitivity and specificity measures simultaneously in excess of 50%. k-Means outperformed LOESS in specificity while bootstrap LOESS markedly outperformed k-means in sensitivity, accuracy, and balance.

Conclusion

The purpose of this project was to examine the relationships between demographic and biochemical variables and liver disease among 583 patients from Northeastern Andhra Pradesh, India via machine learning and develop algorithms to predict diagnosis. Four patients were excluded due to missing data. Most of the patients were male and exhibited positive diagnoses. Male patients appeared to have somewhat higher rates of positive diagnosis. Liver patients were somewhat older than healthy patients were, but this difference appeared to be marginal among women. The apparent gender differences in diagnostic results may exist due to sampling error and imbalance. They may also represent sex differences in liver function or gender differences in social circumstances that facilitate liver disease. Unsupervised machine learning algorithms — hierarchical clustering, heatmaps, and principal component analysis — did not appear to generate predictively useful clusters. Supervised machine learning was the primary focus.

Six supervised algorithms were trained — logistic regression, linear discriminant analysis, quadratic discriminant analysis, local regression (LOESS), k-Nearest Neighbors, and random forest — via bootstrap and K-fold cross validation to predict liver disease diagnoses. Three ensembles were programmed to compile their results. One supervised algorithm based on k-means identified clusters was included independently and as part of one of the ensembles. An over-trained version of bootstrap kNN was also included for demonstrative purposes. In total, seventeen algorithms were evaluated. The two forms of cross validation yielded no predictive differences in logistic regression, linear discriminant analysis, or quadratic discriminant analysis. K-fold cross validation yielded inferior results for LOESS and extremely imbalanced results in kNN, but the latter case may have arisen from overtraining. It yielded marginally less sensitivity and notably greater

specificity in the random forest algorithms, but they both yielded marginally greater balance than a coin flip would have. Both kNN algorithms yielded kappa values below 0.2 for every value of k except, via K-fold cross validation, 1. These results suggest that algorithm may not be reliable.

All tested algorithms demonstrated polarized performance in terms of sensitivity and specificity. Only k-means and LOESS simultaneously exceeded 50% in both metrics; the former was significantly more specific than sensitive while the opposite was true of the latter. Bootstrap LOESS generally performed petter. Quadratic discriminant analysis and the ensembles respectively performed best in terms of balanced accuracy and absolute accuracy, but they were significantly more polarized in terms of sensitivity and specificity. Each algorithm may be preferable in different circumstances depending on whether medical professionals are more interested in confirming liver disease or ruling it out in their patients.

Due to the dataset's inherent limitations, future research should not immediately discount the other algorithms. This paper did not examine whether the algorithms yielded differential results by gender or age group. Under-representation of healthy patients in the data set may have resulted in the mostly low specificity rates. Furthermore, the 579 patients examined are a small sample of Andhra Pradesh's population of more than 49 million. [42] Accordingly, the algorithms may not yield the same results among large, random samples of the general population or incoming patients. Less so should the results be generalized onto other populations due to differential causes — such as alcohol abuse and vinereal infection — of liver disease by culture and region. The data do not contain individual causes, lifestyle choices, other relevant health variables, or even specific types of liver diseases among the patients. Causes, moderators, and mediators of liver disease may vary sociologically and biologically by demographic group, hence the results may vary by population.

References

[1] Sepanlou, Sadaf G et al. "The global, regional, and national burden of cirrhosis by cause in 195 countries and territories, 1990–2017: a systematic analysis for the Global Burden of Disease Study 2017." The Lancet Gastroenterology & Hepatology, Volume 5, Issue 3, 245 - 266. Retrieved October 15, 2020, from https://www.thelancet.com/journals/langas/article/PIIS2468-1253(19)30349-8/fulltext

[2] Ritchie, Hannah. "Causes of Death" (2018). Our World in Data. Retrieved on October 15, 2020, from https://ourworldindata.org/causes-of-death

[3] "The top 10 causes of death" (2018). World Health Organization Newsroom. Retrieved October 15, 2020, from https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

[4] "Chronic Liver Disease" (2020). Stanford Health Care. Retrieved October 15, 2020, from https://stanfordhealthcare.org/medical-conditions/liver-kidneys-and-urinary-system/chronic-liver-disease.html

[5] Sharma A, Nagalli S. "Chronic Liver Disease." 2020 Jul 5. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2020 Jan–. PMID: 32119484. Retrieved October 15, 2020, from https://pubmed.ncbi.nlm.nih.gov/32119484/

[6] "Chronic Liver Disease/Cirrhosis" (2020). Johns Hopkins Medicine. Retrieved October 15, 2020, from https://www.hopkinsmedicine.org/health/conditions-and-diseases/chronic-liver-disease-cirrhosis

[7] "Hepatocellular carcinoma" (May 4, 2019). Mayo Clinic. Retrieved October 15, 2020, from https://www.mayoclinic.org/diseases-conditions/hepatocellular-carcinoma/cdc-20354552

[8] Wisely, Rene. "Doctors Are Seeing More Alcoholic Liver Disease in Young Adults" (January 22, 2019 7:00 AM). Michigan Heatlh. Retrieved October 15, 2020, from https://healthblog.uofmhealth.org/digestive-health/doctors-are-seeing-more-alcoholic-liver-disease-young-adults

[9] Case, Anne, and Angus Deaton. 2017. "Mortality and Morbidity in the 21st Century." Brookings Papers on Economic Activity, Spring 2017. Retrieved on October 15, 2020, from https://scholar.princeton.edu/accase/publications/mortality-and-morbidity-21st-century

[10] Mellinger, J.L., Shedden, K., Winder, G.S., Tapper, E., Adams, M., Fontana, R.J., Volk, M.L., Blow, F.C. and Lok, A.S. (2018), "The high burden of alcoholic cirrhosis in privately insured persons in the United States." Hepatology, 68: 872-882. doi:10.1002/hep.29887

[11] "Indian Liver Patient Records" (Septer 20, 2017). UCI Machine Learning, Kaggle. Retrieved on October 15, 2020, from https://www.kaggle.com/uciml/indian-liver-patient-records

[12] AnnoDomini2020. "IndianLiverProject" (November 16, 2020). GitHub repository. Retrieved on November 16, 2020, from https://github.com/AnnoDomini2020/IndianLiverProject/blob/main/archive.zip

[13] Felson, Sabrina. "What Is a Bilirubin Test?" (February 16, 2019). WebMD Medical Reference. Retrieved on October 19, 2020, from https://www.webmd.com/a-to-z-guides/bilirubin-test

[14] "Bilirubin test" (November 6, 2018). WebMD Medical Reference. Retrieved October 19, 2020, from https://www.mayoclinic.org/tests-procedures/bilirubin/about/pac-20393041

[15] Robinson, Jennifer. "What Is an Alkaline Phosphatase Test?" (May 15, 2019). WebMD Medical Reference. Retrieved on October 19, 2020, from https://www.webmd.com/digestive-disorders/alkaline_phosphatase_test#1

[16] Robinson, Jennifer. "What Is an Aspartate Aminotransferase (AST) Test?" (May 15, 2019). WebMD Medical Reference. Retrieved on October 19, 2020, from https://www.webmd.com/a-to-z-guides/aspartate_aminotransferse-test#1

[17] Robinson, Jennifer. "What Is an Alanine Aminotransferase (ALT) Test?" (May 14, 2019). WebMD Medical Reference. Retrieved on October 19, 2020, from https://www.webmd.com/digestive-disorders/alanine-aminotransferase-test#1

[18] Haldeman-Englert, C., Foley, M., and Turley, Raymond, Jr. "Total Protein and A/G Ratio" (2020). University of Rochester Medical Center. Retrieved on October 19, 2020, from https://www.urmc.rochester.edu/encyclopedia/content.aspx?contenttypeid=167&contentid=total_protein_ag_ratio

[19] DerSarkissian, Carol. "What Is a Total Serum Protein Test?" (May 10, 2019). WebMD Medical Reference. Retrieved on October 19, 2020, from https://www.webmd.com/a-to-z-guides/what-is-a-total-serum-protein-test#1

[20] Mitchell, Tom. "Machine Learning" (1997). Carnegie Mellon University. Retrieved on October 30, 2020, from http://www.cs.cmu.edu/~tom/mlbook.html

[21] "What Is Machine Learning?" (June 26, 2020). University of California, Berkely School of Information. Retrieved on October 30, 2020, from https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

[22] "Online CS Modules: The Definition Of An Algorithm." Virginia Tech Computer Science. Retrieved on October 30, 2020, from https://courses.cs.vt.edu/csonline/Algorithms/Lessons/DefinitionOfAlgorithm/index.html

[23] "Algorithm" (October 29, 2020). Wikipedia. Retrieved on October 30, 2020, from https://en.wikipedia.org/wiki/Algorithm

[24] Irizarry, RA (October 27, 2020). "Chapter 27 Introduction to machine learning." *Introduction to Data Science* (online). Retrieved on October 30, 2020, from https://rafalab.github.io/dsbook/introduction-to-machine-learning.html

[25] Irizarry, RA (October 27, 2020). "Chapter 34 Clustering." *Introduction to Data Science* (online). Retrieved on October 30, 2020, from https://rafalab.github.io/dsbook/clustering.html

[26] Irizarry, RA (October 27, 2020). "Chapter 33.5.4 Large datasets: Principal component analysis." *Introduction to Data Science* (online). Retrieved on October 30, 2020, from https://rafalab.github.io/dsbook/large-datasets.html

[27] Irizarry, RA (October 27, 2020). "Chapter 29 Cross validation." *Introduction to Data Science* (online). Retrieved on October 30, 2020, from https://rafalab.github.io/dsbook/cross-validation.html

[28] Irizarry, RA (October 27, 2020). "Chapter 32.1 Machine learning in practice: Preprocessing." *Introduction to Data Science* (online). Retrieved on November 5, 2020, from https://rafalab.github.io/dsbook/machine-learning-in-practice.html#preprocessing

[29] "Standard Score" (October 14, 2020). Wikipedia. Retrieved on November 5, 2020, from https://en.wikipedia.org/wiki/Standard_score

[30] Tapper, EB and Parikh, ND (June 11, 2018). "Mortality due to cirrhosis and liver cancer in the United States, 1999-2016: observational study." *BMJ* 2018; 362:k2817. doi:10.1136/bmj.k2817

[31] "Training and Test Sets: Splitting Data" (February 10, 2020). Google Developers Machine Learning Crash Course. Retrieved on November 10, 2020, from https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data

[32] "Splitting the Data into Training and Evaluation Data" (2020). Amazon Web Services Developer Guide. Retrieved on November 10, 2020, from https://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html

[33] Irizarry, RA (October 27, 2020). "Chapter 34.9 Random variables: Law of large numbers." *Introduction to Data Science* (online). Retrieved on November 10, 2020, from https://rafalab.github.io/dsbook/random-variables.html#law-of-large-numbers

[34] Irizarry, RA (October 27, 2020). "Chapter 30 The caret package." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/caret.html

[35] Irizarry, RA (October 27, 2020). "Chapter 31.3 Examples of algorithms: Logistic regression." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/examples-of-algorithms.html#logistic-regression

[36] Irizarry, RA (October 27, 2020). "Chapter 16 Statistical Models." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/models.html

[37] Irizarry, RA (October 27, 2020). "Chapter 31 Examples of algorithms." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/examples-of-algorithms.html

[38] Irizarry, RA (October 27, 2020). "Chapter 28 Smoothing." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/smoothing.html

[39] "18.7 - Cohen's Kappa Statistic for Measuring Agreement" (2018). Design and Analysis of Clinical Trials, STAT 509, Penn State Eberly College of Science (online). Retrieved on November 13, 2020, from https://online.stat.psu.edu/stat509/node/162/

[40] Viera AJ, Garrett JM. Understanding interobserver agreement: the kappa statistic. Fam Med. 2005 May;37(5):360-3. PMID: 15883903. Retrieved on Novemebr 13, 2020, from https://pubmed.ncbi.nlm.nih.gov/15883903/

[41] Irizarry, RA (October 27, 2020). "Chapter 32.5 Machine learning in practice: Ensembles." *Introduction to Data Science* (online). Retrieved on November 11, 2020, from https://rafalab.github.io/dsbook/machine-learning-in-practice.html#ensembles

[42] "Chapter 2 Demography" (May 2014). Government of Andhra Pradesh (Archive). Retrieved on November 17, 2020, from https://web.archive.org/web/20140714213923/http://www.ap.gov.in/AP%20State%20Statistical%20Abstract%20May%202014/2%20AP%20Demography.pdf