Data Science Machine Learning 'Choose Your Own' Project - Indian Liver Patient Records

Overview

Being an important organ of the human body, diseases of the liver can be life-threatening. It is therefore highly important to diagnose and subsequently treat them as soon as possible. In clinical practice, liver function tests are used to this end. In liver function tests, a patient's blood is analyzed with regard to different biomarkers, e.g. the activity of enzymes which are specific to liver functions. Those biomarkers can be used as a proxy measure of the patient's liver's health.

The present data set contains the results of liver function tests of Indian patients and is available from the Machine Learning Repository of the University of California, Irvine (Dua and Graff, 2019). The data is classified into liver-patient ('affected' in the remainder of the report) and non-liver-patient ('unaffected' in the remainder of the report).

The aim of this project is to use the available biomarker data to predict a person's affection status based on those same biomarkers. Firstly, standard methods of data exploration will be applied to become familiar with the data and visualize the data and the marker distribution. Subsequently, different machine learning algorithms will be trained on a part of the data and the best model will be used on the validation data set (split from the original data set in the first step of the analysis).

Data exploration (exploratory data analysis, EDA)

The data set

The data set is contained in a comma-separated file (.csv), which is downloaded from the UCI repository. Since this file does not contain a header, the relevant column names are be taken from the description website (https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)) and added to the resulting data frame. Here are the first three entries (note that the last column has been recoded as mentioned above):

Age	Gender	Total_bilirubin	Direct_bilirubin	Alkaline_phosphatase	Alanine_aminotransferase	Aspartate_aminotransferase	Total_protein	Albumin	Albumin_globulin_ratio	Affection_status
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	Affected
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Affected
62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Affected

In total, there are 583 observations of 11 variables in the data set. To make sure that the algorithms created later on will be valid, an important step is to see whether any data is missing. Using R's `is.na` function, it becomes apparent that information on the albumin-globulin-ratio is missing for four patients. These will be moved to a new data frame and analyzed separately in the end:

Age	Gender	Total_bilirubin	Direct_bilirubin	Alkaline_phosphatase	Alanine_aminotransferase	Aspartate_aminotransferase	Total_protein	Albumin	Albumin_globulin_ratio	Affection_status
45	Female	0.9	0.3	189	23	33	6.6	3.9	NA	Affected
51	Male	0.8	0.2	230	24	46	6.5	3.1	NA	Affected
35	Female	0.6	0.2	180	12	15	5.2	2.7	NA	Unaffected
27	Male	1.3	0.6	106	25	54	8.5	4.8	NA	Unaffected

The new total number of observations in the data set is 579. The following table shows the distribution of the affection status in the data:

.	Freq
Affected	0.7150259
Unaffected	0.2849741

There is a slight imbalance in the data, with more affected than unaffected patient's available. However, this is still moderate and should not have consequences on the model generation.

To assess the accuracy of the different models, the data set is split into a `model` data set and a `validation` data set in a 9:1 ratio. This ensures that sufficient data is available for the training on the `model` data, while at the same time preventing overtraining. All further data visualization will be performed on the `model` data. The distribution of the affection status is similar between the two generated data sets:

`Model data`:

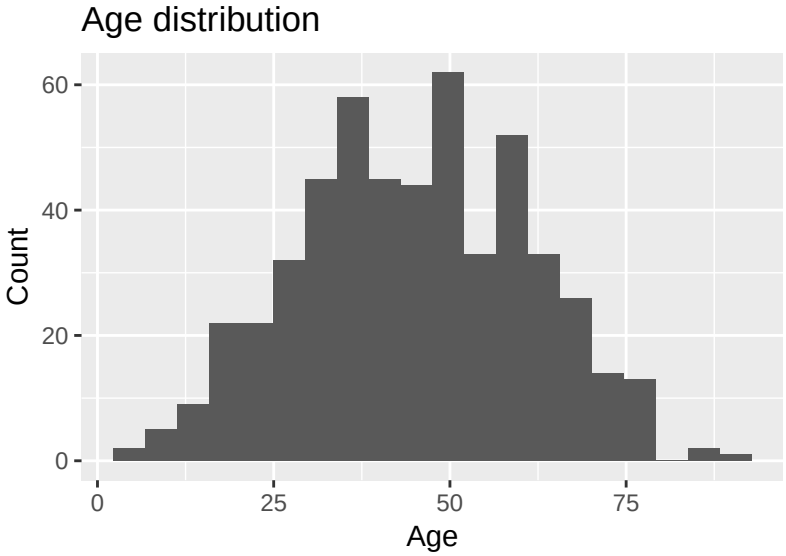
.	Freq
Affected	0.7153846
Unaffected	0.2846154

`Validation data`:

.	Freq
Affected	0.7118644
Unaffected	0.2881356

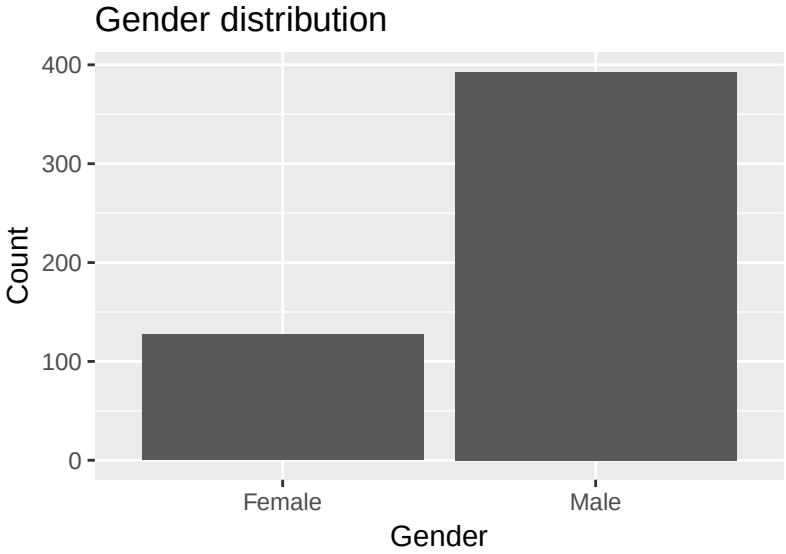
Data visualization

The first variable is the age of the patients:



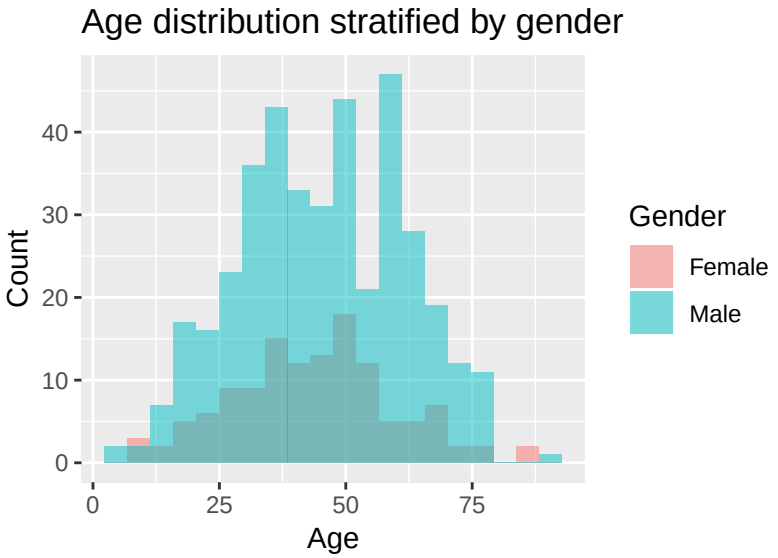
As can be seen, the age looks somewhat normally distributed. The youngest patient is 4 years old. The age of the oldest patient is given as 90 years. However, the data repository states that the age of every patient above the age of 90 was set to 90. The median age of the patients is 45.

The second variable is the gender:

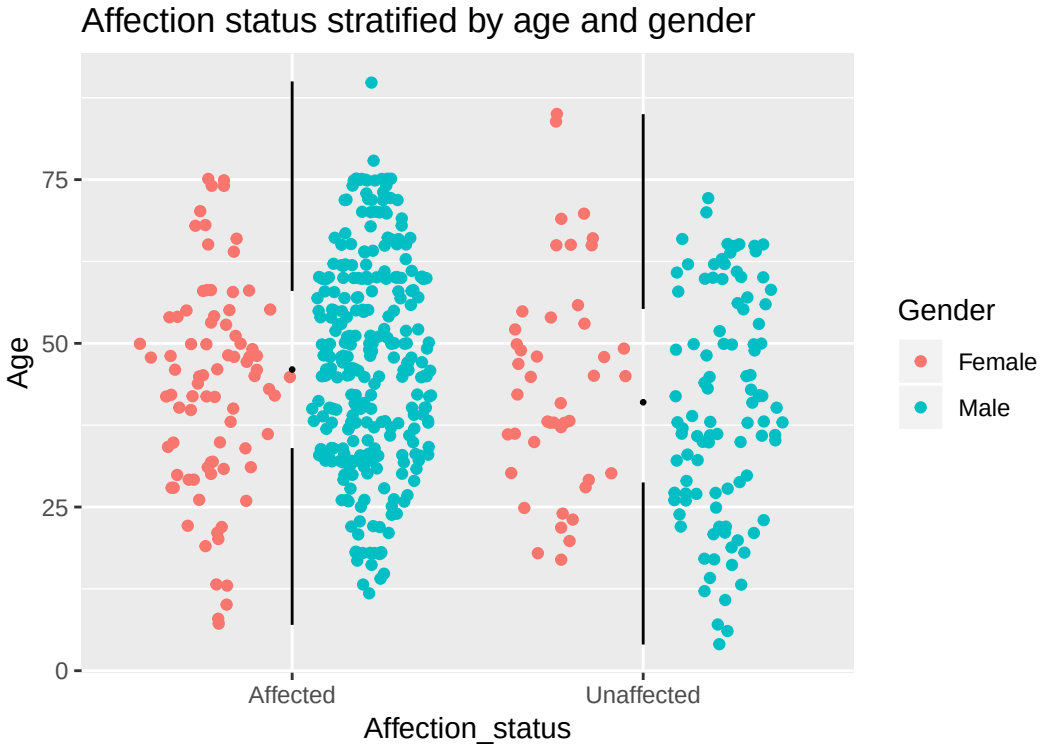


There are approximately three times as many men as women in the data. As it has been shown that there are gender differences in liver disease prevalence (Buzzetti et al., 2017), this might introduce a bias into the data.

Plotting both of the first two variables at once shows that the age distribution between the genders mimic each other fairly well:



As the distribution of the affection status has been demonstrated in the tables above, we can instead complement this by the following visualization, which incorporates all the previous dimensions of information into one plot. In this plot, the points show the age and affection status stratified by gender (color), while the respective Tufte boxplots (minimalistic versions of standard boxplots; black) show this distribution irrespective of gender, i.e. age versus affection status:



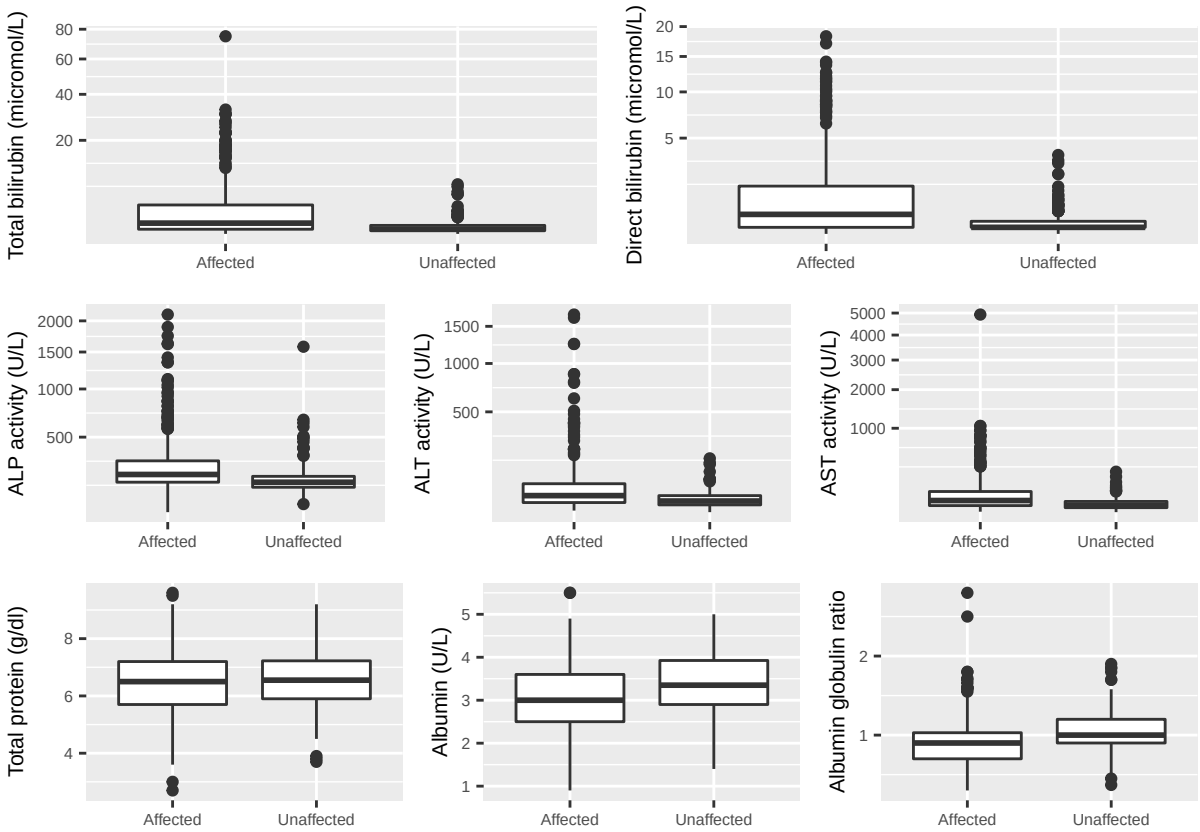
This plot shows that neither age nor gender are perfectly matched between the two groups of affection status. This might be due to an imbalance in the data but might also reflect real differences in disease etiology and prevalence.

Next up is the biomarker data. Present in the data set are:

- Total bilirubin (degradation product of hemoglobin)
- Direct bilirubin (sub-portion of the above)

- Alkaline phosphatase activity (ALP; enzyme)
- Alanine aminotransferase activity (ALT; enzyme)
- Aspartate aminotransferase activity (AST; enzyme)
- Total protein (protein content in the blood)
- Albumin (a protein family)
- Albumin-globulin-ratio (ratio between albumin and another protein family, globulin)

The following patchwork plot visualizes all the above mentioned data at once for an easier overview. Note that both bilirubin plot y-axes and all three enzymes plot y-axes have been `sqrt` transformed.



As can be seen from this plot combination, there are strong differences between affected and unaffected persons for the two bilirubin biomarkers and the three enzyme activity biomarkers. However, this difference is far more pronounced for the outliers in each category than for the median values. For total protein, albumin content and albumin-globulin-ratio, the two groups are very similar. This strongly indicates that bilirubin and the enzyme activities will be most useful for the model generation to distinguish between affected and unaffected persons.

Modelling and model assessment

To be able to generate different models and test them, while at the same time disregarding the `validation` data set, the `model` data set will again be split in a 9:1 ratio to obtain a `training` data set and a `test` data set.

As can be seen, the affection status distribution is still similar between the `training` data set and the `test` data set:

Training data set affection status distribution:

.	Freq
Affected	0.7152034
Unaffected	0.2847966

Test data set affection status distribution:

.	Freq
Affected	0.7169811
Unaffected	0.2830189

To create a baseline algorithm, we can use a model which (like tossing a coin) has a 50 % chance for predicting either 'affected' or 'unaffected':

```
guess = function(data){
  n = nrow(data)
  prediction = sample(c("Affected", "Unaffected"), n, replace=TRUE)
  prediction = factor(prediction, levels=c("Affected", "Unaffected"))
  return(prediction)
}
```

To keep track of all model's performances, a separate data frame containing the obtained accuracies on the `test` data set is created:

Model	Accuracy
Guessing	0.4528302

A second option for a baseline algorithm is one which always predicts 'affected'. This will naturally have better performance then the guessing algorithm, due to the slightly skewed data:

```
affected = function(data){
  n = nrow(data)
  prediction = rep("Affected", n)
  prediction = factor(prediction, levels=c("Affected", "Unaffected"))
  return(prediction)
}
```

Model	Accuracy
Guessing	0.4528302
All affected	0.7169811

As expected, this algorithm is indeed superior to plain guessing.

A final, relatively simple, algorithm, which is not based on machine learning itself is looking at the expected values for the present biomarkers and deciding on 'affected' status, once a certain number of biomarkers are outside of their given range. To do this, the following values could be used (Gowda et al., 2009):

Parameter	Unit	Reference_lower	Reference_upper
Total bilirubin	µmol/L	2	21
Direct bilirubin	µmol/L	0	8
Alkaline phosphatase	U/L	41	133
Alanine aminotransferase	U/L	7	56
Aspartate aminotransferase	U/L	0	35

The following algorithm can then be used to predict the affection status based on the number of biomarkers exceeding the threshold:

```
clinical = function(data=test_data_x, ref=ref_values, threshold=1) {
  data %>% mutate(ToBi = ifelse(Total_bilirubin < ref_values[1,3] |
                                  Total_bilirubin > ref_values[1,4], 1, 0)) %>%
    mutate(DiBi = ifelse(Direct_bilirubin < ref_values[2,3] |
                           Direct_bilirubin > ref_values[2,4], 1, 0)) %>%
```

```
    mutate(AlPh = ifelse(Alkaline_phosphatase < ref_values[3,3] |
                         Alkaline_phosphatase > ref_values[3,4], 1, 0)) %>%
    mutate(AlAm = ifelse(Alanine_aminotransferase < ref_values[4,3] |
                         Alanine_aminotransferase > ref_values[4,4], 1, 0)) %>%
    mutate(AsAm = ifelse(Aspartate_aminotransferase < ref_values[5,3] |
                         Aspartate_aminotransferase > ref_values[5,4], 1, 0)) %>%
    mutate(All = ToBi + DiBi + AlPh + AlAm + AsAm) %>%
    mutate(prediction = ifelse(All >= threshold, "Affected", "Unaffected")) %>%
    pull(prediction)
}
```

Setting this threshold to any value from one to five extends the accuracy table by five more values:

Model	Accuracy
Guessing	0.4528302
All affected	0.7169811
Clinical parameters (1)	0.7169811
Clinical parameters (2)	0.6603774
Clinical parameters (3)	0.6226415
Clinical parameters (4)	0.4716981
Clinical parameters (5)	0.3018868

Here, even the best-performing version of the algorithm (which predicts 'affected' once one of the used biomarkers exceeds its respective threshold) performs equal to predicting 'affected' for everyone (i.e. accuracy = proportion of affected patients).

To remedy this, 'true' machine learning algorithms will be used, which can determine more intricate relationships between the individual biomarkers (as well as age and gender). The `train` function of the `caret` package will be used for the generation of each model. Where applicable 50-fold cross-validation (9:1 ratio) is used.

First, five different machine learning models are trained. A k-nearest neighbor model (kNN); a random forest model (rf); a generalized linear model (glm); a support vector machine model (SVM); and a regularized discriminant analysis model (RDA):

```
control = trainControl(method="cv", number=50, p=0.9)
set.seed(3)
fit_knn = train(train_data_x, train_data_y, method="knn",
                tuneGrid=data.frame(k=seq(11, 31, 2)),
                trControl=control)
set.seed(5)
fit_rf = train(train_data_x, train_data_y, method="rf",
               tuneGrid=data.frame(mtry=seq(1, 10, 1)),
               trControl=control)
set.seed(8)
fit_glm = train(train_data_x, train_data_y, method="glm")
set.seed(13)
fit_svm = train(train_data_x, train_data_y, method="svmRadialSigma",
                tuneGrid=data.frame(expand.grid(C=c(2**seq(-5, 1, 2)),
                                                sigma=c(2**seq(-5, 1, 2)))),
                trControl=control)
set.seed(21)
fit_rda = train(train_data_x, train_data_y, method="rda")
```

For the k-nearest neighbors model, `k`-values between 11 and 31 were tested. A standard approach is to choose `k` as `sqrt(n)`, which in this case would be 21.6101828. Hence, sufficient values around this were chosen. For

binary classification approaches, `k` should preferably be an uneven number. For the random forest model, `mtry` is tested for every possible value (based on the number of relevant variables). In a support vector machine, `C` is the penalty factor and `sigma` is the distance between the used training spheres. Both are usually rather small values.

These models add upon the results data frame:

Model	Accuracy
Guessing	0.4528302
All affected	0.7169811
Clinical parameters (1)	0.7169811
Clinical parameters (2)	0.6603774
Clinical parameters (3)	0.6226415
Clinical parameters (4)	0.4716981
Clinical parameters (5)	0.3018868
k-nearest neighbors	0.6792453
Random forest	0.7547170
Generalized linear model	0.7735849
Support vector machine	0.7169811
Regularized discriminant analysis	0.7169811

While some of those models perform better than the approaches based on clinical parameters/threshold or simply assigning 'affected' to everyone, kNN performs worse and SVM and RDA show the exact same accuracy. This implies that those two models also predict 'affected' for every patient.

To further optimize the algorithms, dimension reduction via principal component analysis might help:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Standard deviation	302.67	232.43	104.83	15.93	6.36	1.36	1.24	0.43	0.41	0.12
Proportion of Variance	0.58	0.34	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cumulative Proportion	0.58	0.93	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

This shows that the first two principal components already account for more than 90 % of the variance in the data. Adding the third principal component increases the explained variance to more than 99 %.

Therefore, only three dimensions will be used for the models trained on dimension-reduced data; the remaining parameters are kept unchanged with regard to the previous model training:

```
set.seed(34)
fit_knn_dim = train(x_train_dim, y_train_dim, method="knn",
                    tuneGrid=data.frame(k=seq(11, 31, 2)),
                    trControl=control)
set.seed(55)
fit_rf_dim = train(x_train_dim, y_train_dim, method="rf",
                   tuneGrid=data.frame(mtry=seq(1, 3, 1)),
                   trControl=control)
set.seed(89)
fit_glm_dim = train(x_train_dim, y_train_dim, method="glm")
set.seed(144)
fit_svm_dim = train(x_train_dim, y_train_dim, method="svmRadialSigma",
                    tuneGrid=data.frame(expand.grid(C=c(2**seq(-5, 1, 2)),
                                                    sigma=c(2**seq(-5, 1, 2)))),
                    trControl=control)
set.seed(233)
fit_rda_dim = train(x_train_dim, y_train_dim, method="rda")
```

This adds five more data points to the results data frame:

Model	Accuracy
Guessing	0.4528302
All affected	0.7169811
Clinical parameters (1)	0.7169811
Clinical parameters (2)	0.6603774
Clinical parameters (3)	0.6226415
Clinical parameters (4)	0.4716981
Clinical parameters (5)	0.3018868
k-nearest neighbors	0.6792453
Random forest	0.7547170
Generalized linear model	0.7735849
Support vector machine	0.7169811
Regularized discriminant analysis	0.7169811
k-nearest neighbors (dim. red.)	0.6981132
Random forest (dim. red.)	0.6792453
Generalized linear model (dim. red.)	0.7169811
Support vector machine (dim. red.)	0.7169811
Regularized discriminant analysis (dim. red.)	0.7169811

Apparently, most of the five algorithms perform worse than or equal to the original model. Only the kNN model shows a slight increase in accuracy.

Since the performances so far vary widely, an ensemble model might leverage the strengths of each of the trained models. The following ensemble model is based on the models trained with the original data (before dimension reduction). The `threshold` parameter determines how many 'unaffected' predictions by the individual models are necessary for the ensemble model itself to predict 'unaffected':

```
ensemble = function(data=test_data_x, threshold=1){
  output = data.frame(knn = rep(NA, nrow(data)))
  output$knn = predict(fit_knn, data)
  output$rf = predict(fit_rf, data)
  output$glm = predict(fit_glm, data)
  output$svm = predict(fit_svm, data)
  output$rda = predict(fit_rda, data)
  output$count = rowCounts(as.matrix(output), value="Unaffected")
  output = output %>% mutate(prediction = ifelse(count>=threshold,
                                                 "Unaffected", "Affected"))
  output$prediction = factor(output$prediction, levels=c("Affected", "Unaffected"))
  return(output)
}
```

A similar model can be constructed for the models trained on the data after dimension reduction (based on the same underlying principle as described above):

```
ensemble_dim = function(data=x_test_dim, threshold=1){
  output = data.frame(knn = rep(NA, nrow(data)))
  output$knn = predict(fit_knn_dim, data)
  output$rf = predict(fit_rf_dim, data)
  output$glm = predict(fit_glm_dim, data)
  output$svm = predict(fit_svm_dim, data)
  output$rda = predict(fit_rda_dim, data)
  output$count = rowCounts(as.matrix(output), value="Unaffected")
  output = output %>% mutate(prediction = ifelse(count>=threshold,
                                                 "Unaffected", "Affected"))
  output$prediction = factor(output$prediction, levels=c("Affected", "Unaffected"))
  return(output)
```

```
}
```

Using those two ensemble models (with varying thresholds from one to five) further extends the results data frame to its final dimensions:

Model	Accuracy
Guessing	0.4528302
All affected	0.7169811
Clinical parameters (1)	0.7169811
Clinical parameters (2)	0.6603774
Clinical parameters (3)	0.6226415
Clinical parameters (4)	0.4716981
Clinical parameters (5)	0.3018868
k-nearest neighbors	0.6792453
Random forest	0.7547170
Generalized linear model	0.7735849
Support vector machine	0.7169811
Regularized discriminant analysis	0.7169811
k-nearest neighbors (dim. red.)	0.6981132
Random forest (dim. red.)	0.6792453
Generalized linear model (dim. red.)	0.7169811
Support vector machine (dim. red.)	0.7169811
Regularized discriminant analysis (dim. red.)	0.7169811
Ensemble (1)	0.7547170
Ensemble (2)	0.7358491
Ensemble (3)	0.7169811
Ensemble (4)	0.7169811
Ensemble (5)	0.7169811
Ensemble (dim. red.) (1)	0.6603774
Ensemble (dim. red.) (2)	0.7169811
Ensemble (dim. red.) (3)	0.7169811
Ensemble (dim. red.) (4)	0.7169811
Ensemble (dim. red.) (5)	0.7169811

These results show that the best performing model is still the GLM without dimension reduction, followed by kNN before dimension reduction and the ensemble before dimension reduction (using `threshold = 1`). The GLM also shows high values for further performance parameters:

	Value
Sensitivity	0.9736842
Specificity	0.2666667
Pos Pred Value	0.7708333
Neg Pred Value	0.8000000
Precision	0.7708333
Recall	0.9736842
F1	0.8604651
Prevalence	0.7169811
Detection Rate	0.6981132
Detection Prevalence	0.9056604
Balanced Accuracy	0.6201754

GLM is therefore chosen as the final model and trained on the `model` data set:

```
set.seed(377)
fit_glm_final = train(model_data_x, model_data_y, method="glm")
```

This final model could now be tested on the `validation` data. However, as a final check, the performance on the data set created from the observations with missing values of the albumin-globulin-ratio is tested. To prevent errors due to the `NA`s for this variable, the mean albumin-globulin-ratio of the `model` data will be used as a proxy value (0.9477115).

Here, GLM achieves an accuracy of 0.75, i.e. three out of four are predicted correctly.

Now for the performance of the GLM on the validation data:

The accuracy of this model is 0.7966102. Further performance parameters can be seen in the following table:

	Value
Sensitivity	0.9761905
Specificity	0.3529412
Pos Pred Value	0.7884615
Neg Pred Value	0.8571429
Precision	0.7884615
Recall	0.9761905
F1	0.8723404
Prevalence	0.7118644
Detection Rate	0.6949153
Detection Prevalence	0.8813559
Balanced Accuracy	0.6645658

While the accuracy of the model is close to 0.8, the very high sensitivity close to 1 in combination with the still low specificity indicates that predicting 'affected' still occurs very frequently, regardless of the actual outcome.

Final results

Training the GLM on the `training` data and testing it on the `test` data led to an accuracy of about 0.77, thereby outperforming all other models and algorithms. The sensitivity was close to 1, with specificity being 0.27. Training this model on the whole `model` data (i.e. the combined `training` data and `test` data), made predictions on the `missing` data and validation on the `validation` data possible.

For the `missing` data, the prediction accuracy was 0.75. However, since there were only a total of four observations in this data set, the meaningfulness of this is rather limited.

More importantly, the accuracy on the `validation` data showed a slight increase. Simultaneously, the specificity increased from 0.27 to below 0.35.

The following table shows all obtained accuracies; note that those were obtained with different models, trained and tested on different fractions of the whole data set:

Model	Accuracy
Guessing	0.453
All affected	0.717
Clinical parameters (1)	0.717
Clinical parameters (2)	0.660
Clinical parameters (3)	0.623
Clinical parameters (4)	0.472
Clinical parameters (5)	0.302
k-nearest neighbors	0.679
Random forest	0.755
Generalized linear model	0.774
Support vector machine	0.717
Regularized discriminant analysis	0.717
k-nearest neighbors (dim. red.)	0.698
Random forest (dim. red.)	0.679
Generalized linear model (dim. red.)	0.717
Support vector machine (dim. red.)	0.717
Regularized discriminant analysis (dim. red.)	0.717
Ensemble (1)	0.755
Ensemble (2)	0.736
Ensemble (3)	0.717
Ensemble (4)	0.717
Ensemble (5)	0.717
Ensemble (dim. red.) (1)	0.660
Ensemble (dim. red.) (2)	0.717
Ensemble (dim. red.) (3)	0.717
Ensemble (dim. red.) (4)	0.717
Ensemble (dim. red.) (5)	0.717
GLM on missing data	0.750
GLM on validation data	0.797

Conclusion

The aim of this project was to predict the affection status of liver patients based on different biomarkers. The available data (besides age and gender) included bilirubin and protein amounts in the patients' blood as well as the activity of three liver-specific enzymes. Based on the data visualization, the most striking differences could be observed for those enzymes as well as the bilirubin levels.

Training a number of machine learning models and ensembles on the data led to the emergence of a generalized linear model (GLM) as the best-performing model in terms of accuracy. This model also showed a very high sensitivity, although the specificity was rather low. Re-training this model on a larger fraction of the data set led to an acceptable performance on a subset of incomplete data, which had been filled with the mean of the data set (accuracy of 0.75). Finally, using the `validation` data showed that the model performed with an accuracy of 0.797. The sensitivity remained very high (0.976), while the specificity was slightly higher than before (0.351).

It became obvious that all models (including the final GLM) perform far better predictin the 'affected' status than predicting the 'unaffected' status. A possible reason for this is the slight imbalance in the data set. There were approximately two times as many affected patients as there were unaffected patients. Secondly, the visualization and comparison of the biomarker data implies that not every patient who is affected by a liver disease shows highly abnormal values for the biomarkers contained in this data set. It has been shown previously that patients with (chronic) liver disease may present without clinical symptons and values of liver function tests well within the accepted range (Ahmed et al., 2018), which is in line with these findings. Thus,

affected patients which do not show significantly different levels of bilirubin, proteins or enzyme activity severely increase the difficulty of distinguishing between those. Furthermore, the affected patients with low levels of the biomarkers are likely to have lead the machine learning approaches to falsely flag unaffected patients as affected.

In the future, these shortcomings could be remedied by different approaches. First and foremost would be an increase in the number of observations. Since machine learning is a data-driven approach, a larger number of available data often results in better results (as long as over-training is prevented). Another option, based on the present data set, would be either under-sampling of the data with the 'affected' flag or over-sampling of the data with the 'unaffected' flag. This way, the models might be able to better distinguish between those two outcomes. A third approach, which has shown promise in many areas, would be the application of deep learning. All those approached, or a combination thereof, would likely increase the prediction performance and therefore resolve the limitations of this project.

References

Ahmed Z, Ahmed U, Walayat S, Ren J, Martin DK, Moole H, Koppe S, Yong S, Dhillon S. Liver function tests in identifying patients with liver disease. 2018. Clin Exp Gastroenterol. doi: 10.2147/CEG.S160537

Buzzetti E, Parikh PM, Gerussi A, Tsochatzis E. Gender differences in liver disease and the drug-dose gender gap. 2017. Pharmacol Res. doi: 10.1016/j.phrs.2017.03.014

Dua D, Graff C. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. 2019. University of California, School of Information and Computer Science. doi: NA

Gowda S, Desai PB, Hull VV, Math AAK, Vernekar SN, Kulkarni SS. A review on laboratory liver function tests. 2009. Pan Afr Med J. doi: 10.11604/pamj.25/11/2009.3.17.125

Hall P, Cash J. What is the Real Function of the Liver 'Function' Tests? 2012. Ulster Med J. doi: NA