Predictive Modeling in Chronic Kidney Disease

Class of Bioinformatics

Group: GiRRRls

Celia Rivera, Dakota Coloroso, Inês Salema, Nicole Erwin

Universitat de Barcelona and Universitat Politècnica de Catalunya

December 18, 2024

Abstract

Liver disease ranks among the top 10 causes of mortality worldwide, with major contributors including viral hepatitis, alcohol consumption, non-alcoholic fatty liver disease (NAFLD), and drug-induced liver injury. The asymptomatic nature of early-stage liver disease often leads to delayed diagnoses, despite timely diagnostics being critical for saving lives. Regions such as Africa bear a particularly high burden of liver disease, compounded by limited access to advanced diagnostic tools such as ultrasounds, CT scans, and liver biopsies [1].

Prediction models offer a promising solution by identifying at-risk individuals early, requiring only basic patient data and computational tools. This work presents a two-part approach: the development of methods for a predictive model and the design of a user-friendly front end for an adaptable model. The predictive model leverages machine learning algorithms, which demonstrated strong performance in this study. In addition, logistic regression is explored as a customizable option, enabling healthcare professionals to tailor predictions based on available patient information.

By integrating these methods, this project aims to provide accessible, efficient, and scalable solutions for liver disease prediction, particularly in resource-constrained settings.

Also, this project presents a shiny app for another project, regarding Diabetes.

Contents

1 Methods to build prediction model

1.1 Data Collection

The dataset was obtained from Kaggle: Chronic Kidney Disease Dataset [2].

This dataset was chosen because it is of high quality, with a usability score of 10, 1,500 records, and 11 variables, which include both environmental and genetic factors. It is purportedly preprocessed and ready for data science applications, and it is available under the CC BY 4.0 license. Additionally, the ”Diagnosis” class, which is the ultimate goal of our work (to build a model to predict this class), is balanced, a crucial factor that is difficult to find and very important for model performance. After conducting the literature review, we concluded that the dataset contains many valuable features worth analyzing.

There are no missing values in the dataset, and no outliers were identified. This was confirmed using Tukey’s Interquartile Range (IQR) method. Additionally, we checked for data type consistency by reviewing the data types of all variables to ensure they made sense.

All variables had zero missing values, and no outliers were detected, which is a positive sign. Furthermore, there were no duplicate records, and no highly correlated or irrelevant features were found.

1.2 Exploratory Data Analysis

Performing Exploratory Data Analysis (EDA) is essential before building a predictive model, as it helps identify potential data issues that could affect model performance or lead to data leakage. For example, if the dataset is imbalanced with respect to certain variables, it is important to determine whether any model failures arise due to data issues, which can be addressed and mitigated during the preprocessing phase.

As part of the EDA, we generated count plots for five categorical variables: gender, smoking, genetic risk, diabetes, hypertension, and diagnosis (figure 1). We observed notable data imbalances, particularly with the smoking, diabetes, and hypertension variables. These imbalances should be taken into consideration throughout the analysis and modeling process, as they may influence the model’s ability to generalize effectively.

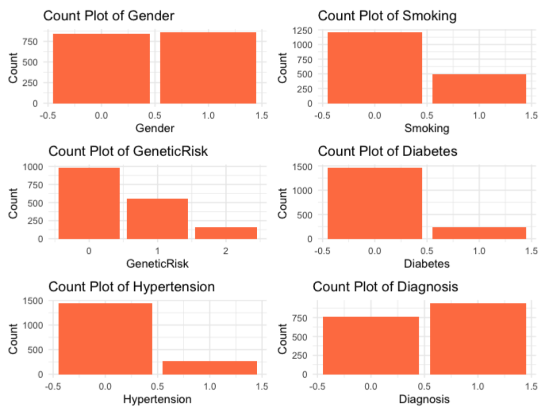


Figure 1: Count plot for the five categorical variables

Histograms were generated to examine the distribution of continuous variables, including age, BMI, alcohol consumption, physical activity, and liver function test (figure 2). No significant imbalances were observed in these variables, which is advantageous for the subsequent model training, as it indicates that the variables are well-distributed and do not require additional transformations or adjustments.

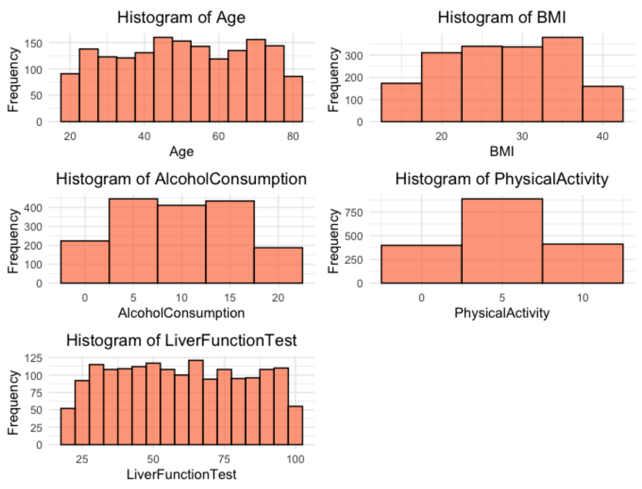


Figure 2: Histograms for continuous variables

From a statistical perspective, understanding the correlations between variables is crucial. Highly correlated variables may not contribute additional information to the model and could introduce bias by giving undue weight to certain predictors. Therefore, constructing a correlation matrix is a valuable step to identify potential issues related to multicollinearity, helping to ensure that the model remains robust and interpretable (figure 3).

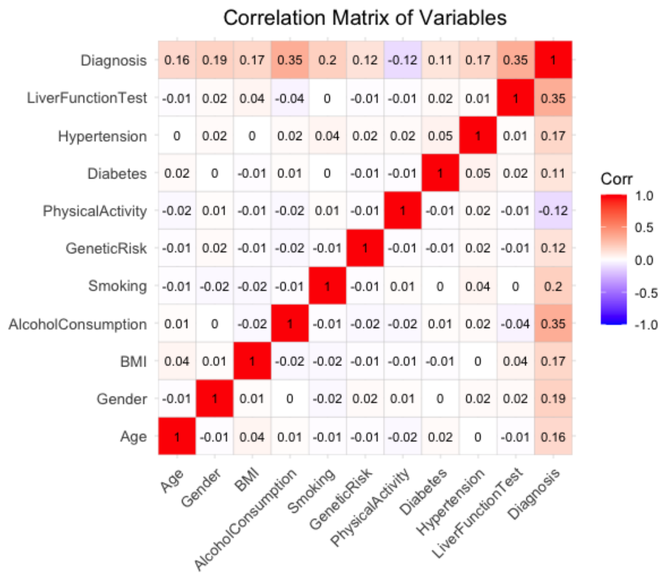


Figure 3: Correlation Matrix between variables

The correlation matrix indicates that the variables are not strongly correlated with each other, which is a positive outcome for model development. This suggests that each variable will provide distinct information to the model, minimizing redundancy. Additionally, the correlation matrix offers insights into how the variables relate to the diagnosis class. For instance, as anticipated from the literature review, alcohol consumption appears to play a significant role in predicting the outcome.

By generating individual plots for each predictor against the target class (”diagnosis”), we gain further understanding of how each variable might influence the predictive model. From these plots (figure 4), it is evident that some variables are intuitively correlated with the diagnosis class. These include liver function test, alcohol consumption, genetic risk (scoring 2), smoking, and gender. It is important to emphasize that at this stage, we are not making predictions; we are simply observing how these variables are correlated with the target class. This is not a causal analysis, which can only be conducted by manipulating independent variables and observing their effects on the dependent variable (in this case, the diagnosis class).

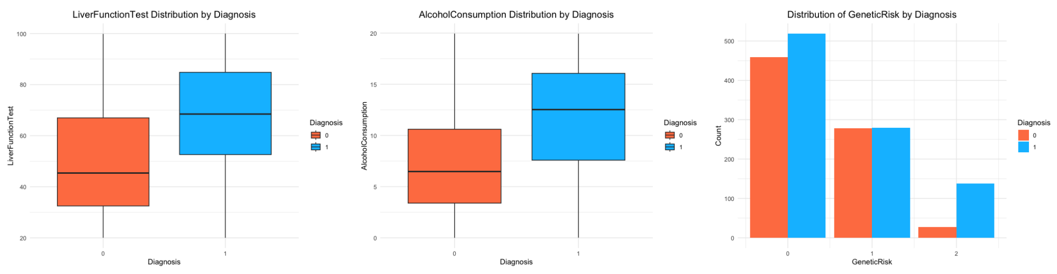


Figure 4: (a) Liver x Dx, (b) Alcohol x Dx, (c) Genetic Risk x Dx

An interesting analysis involves examining the interaction between alcohol consumption, liver function test, and genetic risk (figure 5). A clear distinction between the two classes of Diagnosis emerges when we explore these interactions. The class with no liver disease contains only one patient with semi-high alcohol consumption, a high liver function test, and a genetic risk score of 2. In contrast, this pattern is not observed in the liver disease class, where the distribution of these variables differs significantly.

This suggests that there may be a distinctive pattern in the non-liver disease patients that is markedly different from those diagnosed with liver disease. The interaction of these variables provides valuable insight into how the combination of alcohol consumption, liver function, and genetic risk influences the likelihood of liver disease, highlighting potential areas for further investigation in predictive modeling.

Performing Principal Component Analysis (PCA) on the dataset can provide valuable insights. PCA transforms the data into a set of uncorrelated variables called Principal Components (PCs). The first principal component (PC1) captures the direction of maximum variance in the data, meaning it explains the majority of the variance. The second principal component (PC2) captures the direction of maximum variance orthogonal to PC1, and so on.

We do not need to use all principal components; typically, the first few components capture a large portion of the total variance. For instance, using just the first few PCs, we can often explain 80% or even 90% of the variance in the dataset.

This analysis differs from previous analyses as it does not involve the Diagnosis class. Instead, PCA helps identify patterns in the data, highlights redundant features, and reveals potential clusters between different classes.

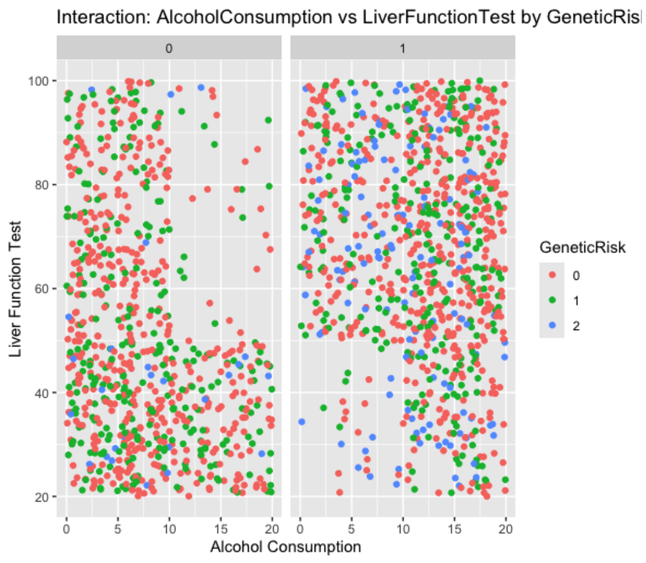


Figure 5: Interaction between Alcohol Consumption, FunctionLiverTest and GeneticRisk

In our initial analysis, we assessed how many principal components were needed to explain 80% of the variance. According to the scree plot (figure 6), the first four principal components account for 80% of the total variance in the data.

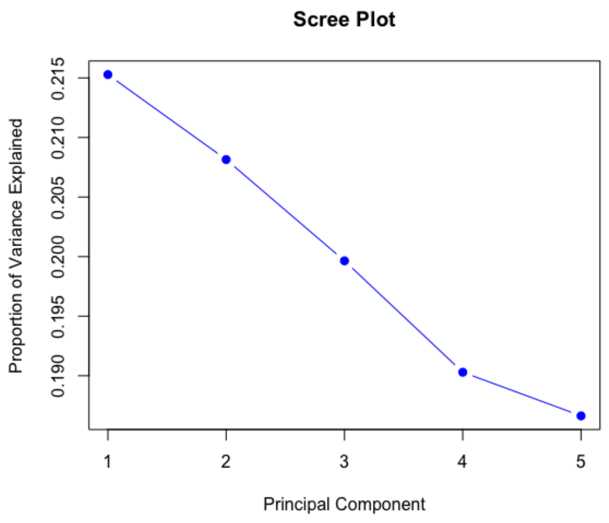
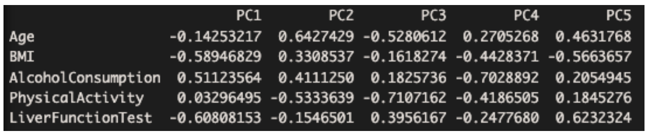


Figure 6: Variance of each PCA

It is also important to understand which variables contribute the most to each Principal Component (PC). The results are shown in the figure 7. The contribution of each variable to a PC is interpreted based on the absolute value of the loadings — the larger the absolute value, the greater the contribution of that variable to the component.

For instance, PC1 can be viewed as a new variable formed by a linear combination of the original variables, capturing the largest portion of the data's variability. In this case, PC1 is primarily driven by LiverFunctionTest, BMI, and AlcoholConsumption, as indicated by the high absolute values of their loadings.

Similarly, for PC2, the variables with the highest loadings are age and physical activity, suggesting that these variables explain the second most significant direction of variance in the data.



```
                           PC1        PC2        PC3        PC4        PC5
Age                -0.14253217  0.6427429 -0.5280612  0.2705268  0.4631768
BMI                -0.58946829  0.3308537 -0.1618274 -0.4428371 -0.5663657
AlcoholConsumption  0.51123564  0.4111250  0.1825736 -0.7028892  0.2054945
PhysicalActivity    0.03296495 -0.5333639 -0.7107162 -0.4186505  0.1845276
LiverFunctionTest  -0.60808153 -0.1546501  0.3956167 -0.2477680  0.6232324
```

Figure 7: Loading of each variable

After completing this EDA, we are well-positioned to proceed with developing a predictive model for liver disease based on this dataset.

1.3 Statistical Modeling

As seen in the Exploratory Data Analysis (EDA), the dataset includes categorical variables that assume binary values (example, 0 for male and 1 for female). In R, it is essential to convert these variables into factors, which are specifically designed to handle categorical data. Without this conversion, a binary variable like Gender could be misinterpreted as numerical, leading the model to infer a direct relationship between the higher value (1) and the outcome. Using factors ensures that the model treats these variables appropriately as categorical, and the output is displayed in terms of factor levels, making interpretation more intuitive. Therefore, converting categorical variables into factors is an important starting point for this analysis.

Dividing the data into training and test sets is crucial, even before selecting or building a model. This separation allows us to fairly evaluate model performance using metrics like accuracy, precision, and others. Testing on unseen data ensures a more realistic assessment of the model's ability to generalize. If we evaluate the model on the same data it was trained on, there is a risk of overfitting, where the model memorizes patterns specific to the training set, resulting in overly optimistic performance metrics. While this practice is more commonly associated with machine learning algorithms, it is equally important for statistical models like logistic regression to validate their performance.

The dependent variable in this case is the class variable, which is binary, divided into two classes: 0 and 1, indicating the absence or presence of liver disease, respectively. Since the variable is dichotomous, there are no outliers or high correlations between predictors, making logistic regression (LR) an appropriate choice for analysis. Logistic regression doesn't predict a continuous y value but instead estimates the probability of each possible outcome (liver disease or no liver disease).

In logistic regression, we calculate the odds ratio, which compares the probability of having the disease to the probability of not having it. The coefficients of logistic regression represent the log odds, indicating how a unit increase in a predictor affects the likelihood of the outcome. By exponentiating these coefficients, we obtain the odds ratios, which can then be interpreted alongside their confidence intervals to evaluate their significance. Logistic regression assumes independence of observations, which holds true in our case, allowing us to proceed safely.

We cannot determine the best model upfront; it is identified by building and evaluating multiple models. The p-value of predictors is a common indicator, but models with similar p-values may perform differently when evaluated on the test dataset. To choose the best model, we rely on comprehensive evaluation metrics, including cross-validation, to ensure robust performance.

To build and refine models, we will use the stepwise selection method. This technique systematically evaluates a variety of models to identify the most predictive one. Starting with either no predictors (forward selection) or all predictors (backward elimination), stepwise selection iteratively adds or removes variables based on a criterion such as AIC (Akaike Information Criterion). This approach ensures that we explore a wide range of models, ultimately selecting the one that performs best based on rigorous evaluation.

1.4 Stepwise Method

Our dataset consists of 10 variables in addition to the class variable. To build a logistic regression model from this dataset, we face numerous combinations of variables that could potentially yield a good model. To efficiently identify an optimal model without testing all possible combinations, we employed the stepwise method using a backward approach. This method begins with a model that includes all variables and systematically removes one variable at a time based on a selection criterion. Essentially, the algorithm removes the variable that contributes the least to the model's performance (as determined by the criterion) and repeats this process iteratively.

After performing this process, we evaluated the final model, which achieved an accuracy of approximately 83%.

Interestingly, the stepwise regression concluded with the model that includes all variables. This result suggests that each variable contributes positively to the model's performance, indicating that all predictors add unique and useful information for predicting the class "Diagnosis."

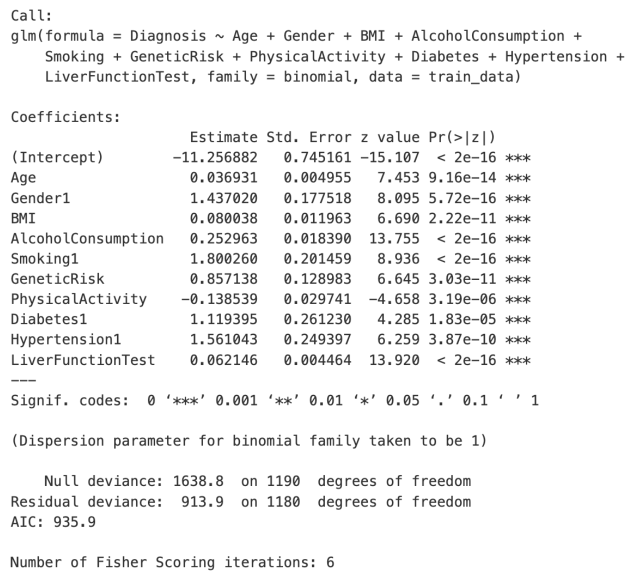
Initially, this outcome raised concerns about potential issues in the model-building process, such as overfitting. However, considering the low collinearity between predictors (as shown earlier in the EDA in figure 3) and the relatively large size of the dataset, overfitting is unlikely.

Indeed, the variables exhibit very little correlation with one another, except for their relationship with the class "Diagnosis" (figure 3). This supports the possibility that each variable contributes unique and valuable information for predicting the "Diagnosis" class.

Furthermore, an examination of the summary of our final model reveals that all p-values are very small, further indicating that every variable meaningfully contributes to predicting the class (figure 8).

Besides that, when comparing with the literature, this makes sense: liver disease is a very complex disease which has a relationship with all 10 presented variables:

- Age: The older someone is, the more likely they are to have liver disease, according to our model. Additionally, purely analyzing the relationship between age and the class diagnostic, we can clearly see that for the class "1" individuals have higher ages (Figure 9). When comparing to the literature, this is also supported [3]. Alcohol consumption among the elderly has increased. Alcohol metabolism changes with age, and the elderly are more sensitive to the toxic effects; this increased consumption is, therefore, of great clinical relevance. Furthermore, older individuals are generally more susceptible to diseases, which makes it expected that liver function declines with age.



```
Call:
glm(formula = Diagnosis ~ Age + Gender + BMI + AlcoholConsumption +
    Smoking + GeneticRisk + PhysicalActivity + Diabetes + Hypertension +
    LiverFunctionTest, family = binomial, data = train_data)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -11.256882   0.745161 -15.107  < 2e-16 ***
Age                  0.036931   0.004955   7.453 9.16e-14 ***
Gender1              1.437020   0.177518   8.095 5.72e-16 ***
BMI                  0.080038   0.011963   6.690 2.22e-11 ***
AlcoholConsumption   0.252963   0.018390  13.755  < 2e-16 ***
Smoking1             1.800260   0.201459   8.936  < 2e-16 ***
GeneticRisk          0.857138   0.128983   6.645 3.03e-11 ***
PhysicalActivity    -0.138539   0.029741  -4.658 3.19e-06 ***
Diabetes1            1.119395   0.261230   4.285 1.83e-05 ***
Hypertension1        1.561043   0.249397   6.259 3.87e-10 ***
LiverFunctionTest    0.062146   0.004464  13.920  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1638.8  on 1190  degrees of freedom
Residual deviance:  913.9  on 1180  degrees of freedom
AIC: 935.9

Number of Fisher Scoring iterations: 6
```

Figure 8: Summary final model of the stepwise method with all variables

- Gender: In our model, individuals are more likely to have liver disease if they are female. In the dataset, we can clearly observe the relationship between these two variables (Figure 9). This is also supported by the literature [4] with explanations such as differences in sex hormones and the more acute effects of alcohol in women, despite generally consuming less alcohol.
- BMI: Higher BMI is associated with an increased risk of liver disease, according to our prediction model. Additionally, we can observe the relationship between BMI and liver disease (Figure 9). This relationship is well-documented in the literature [5].
- Alcohol consumption: This is one of the most evident factors when discussing liver disease. It was already identified as one of the key variables more strongly related to the class Diagnostic, as seen previously in Figures 3 and 4. This is widely recognized in the literature and makes sense as a variable included in the model [6].
- Genetic Risk: There are numerous genetic factors associated with liver health [7]. While this dataset does not explicitly specify which genetic risks are considered, we observe a significant correlation, especially in individuals with a genetic risk factor of 2 (Figure 4).
- Physical activity: Decreased physical activity is linked to liver disease and is supported by literature [8].
- Diabetes: Diabetes is well-established as a contributing factor to liver disease. This relationship stems from various factors such as increased inflammation, high blood sugar levels leading to liver dysfunction, and compromised immune systems increasing susceptibility to liver infections like hepatitis B and C [9].
- Hypertension: There is substantial evidence linking hypertension to liver disease [10].
- Liver Function Test: This variable directly impacts the outcome of predictions, as it indicates liver health.

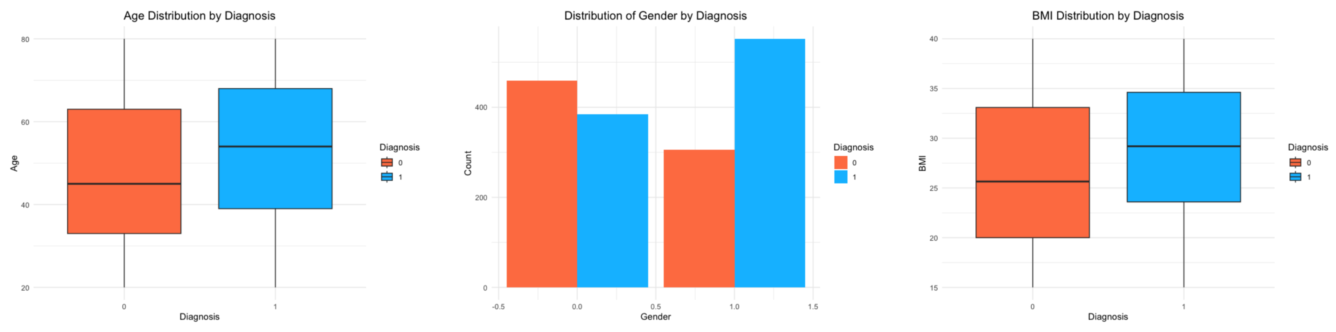


Figure 9: (a) Relationship between Age and the Class Diagnostic.
(b) Relationship between Gender and the Class Diagnostic.
(c) Relationship between BMI and the Class Diagnostic.

1.5 Costumizable Logistic Regression

Given that the variable LiverFunctionTest is almost entirely reliant on genetic and environmental factors, if we consider this project in a practical context, where a doctor seeks to predict a patient's health outcomes using limited information, it would be unreasonable to assume the availability of the LiverFunctionTest. This variable is essentially the outcome we aim to predict.

To address this, we developed a script allowing users to select the variables they wish to include in a logistic regression model. This model provides metrics for accuracy and enables predictions based on new input data. This approach is highly beneficial in real-world scenarios, as doctors often do not have access to all patient data when making clinical decisions.

For example, by excluding the LiverFunctionTest and using other available variables (which are commonly accessible in medical contexts), the model achieved an accuracy of 77%. This still provides a valuable level of confidence and assists doctors in making informed predictions based on the available data.

1.6 ML models

Since implementing machine learning (ML) models in R is straightforward, we performed a surface-level analysis of how these models might perform with this dataset in the given setting. The goal of this analysis is twofold: first, if we consider this work as a tool to assist doctors in making predictions about whether a patient has a disease or not, we want it to be customizable, as discussed in the previous section. This would allow a doctor to make predictions even when all data points (as in the dataset) are not available. Second, we aim to identify the best-performing model to ensure the predictions are as accurate as possible. This way, the doctor can decide how to utilize the assistance from the algorithm.

For this purpose, we evaluated the following ML models: KNN, Naive Bayes, Random Forest, SVM, and XGBoost. Below, we explain how each model works, their strengths, and the reasons behind their varying performance in this context:

- **KNN:** K-Nearest Neighbors is a simple yet effective ML model that classifies a data point based on the majority class of its K nearest neighbors in the feature space, often using metrics like the Euclidean distance. Although it achieved decent results (77% accuracy), it is the least accurate among the models tested. This could be due to the preprocessing techniques applied to the dataset, such as normalization, which are unknown to us. Additionally, KNN may suffer from the curse of dimensionality, which can degrade its performance in higher-dimensional spaces.

- **Naive Bayes:** This model estimates the probability for each class and predicts the one with the highest posterior probability. It assumes that features are independent given the class. Naive Bayes performed slightly better than KNN (79% accuracy), likely due to its ability to capture probabilistic relationships. However, the independence assumption may limit its accuracy for datasets where features are correlated.

- **Random Forest:** Random Forest constructs an ensemble of decision trees and aggregates their outputs (majority vote for classification). It is robust, versatile, and resistant to overfitting. It achieved an accuracy of 91.5%, making it the second-best performing ML model. This reflects its strength in capturing non-linear relationships and handling complex data patterns effectively.

- **SVM:** Support Vector Machine identifies the optimal hyperplane that maximally separates classes. SVM performed very well (87% accuracy), demonstrating its ability to model complex relationships. However, it was slightly less accurate than Random Forest and XGBoost, possibly due to sensitivity to parameter tuning and overlapping data distributions. SVM is particularly effective in high-dimensional spaces and cases where the number of features exceeds the number of samples, which is not the case with this dataset.

- **XGBoost:** XGBoost builds an ensemble of decision trees using gradient boosting, sequentially reducing errors by correcting the residuals of the previous trees. It is robust to overfitting due to the incorporation of regularization. XGBoost is particularly suited for tabular datasets, as is the case here. It achieved the highest accuracy (93%), owing to its ability to fine-tune decision boundaries and capture intricate patterns in the data.

To conclude, simpler models, such as KNN and Naive Bayes, provided decent results but were outperformed by more sophisticated techniques like Random Forest and XGBoost, which excelled in leveraging the dataset's complexity to deliver higher accuracy.

2 API and Front-End Instructions

In the second part of the project, we developed a Shiny app for another group's project on diabetes and integrated it into a Docker container for seamless project collaboration. Below are detailed instructions to replicate our setup.

2.1 Shiny App Development

We cloned the project's GitHub repository and added a Shiny app file. After adjusting file paths in the API as per the README instructions, we verified the API was operational. The Shiny app was then configured to work with the API, creating a user-friendly interface for diabetes-related predictions.

After ensuring the API was running, we:

1. Opened a new RStudio session.

2. Executed the Shiny app code, ensuring the API port matched the port in the Shiny app code.

The Shiny app functioned seamlessly, enabling smooth interaction with the API. To integrate the API into the Shiny app, we used the following code snippet:

```r
api_base_url <- Sys.getenv("API_BASE_URL", "http://api:8001")
```

This approach ensures flexibility by allowing the API base URL to be overridden through an environment variable. The fallback value `http://api:8001` is used when the environment variable is not set, which ensures compatibility across different deployment environments. For instance, during local development, a custom API endpoint can be specified without modifying the app code.

Important Notes regarding Docker :

1. In a Docker environment, the working directory must align precisely with the container's file system structure because the paths inside the container are isolated from the host system. This means you must configure the setwd command to point to the correct directory within the container, such as /usr/src/app/random-E11, as mapped during container setup. Conversely, for local development, the working directory should match the project structure on the host machine, such as C:/Users/dakot/Documents/Biomedical Informatics/random-E11. Proper alignment ensures seamless operation across environments and prevents errors like missing files or incorrect paths. This alignment must be made in both the API and Shiny app.

2. The Shiny app file must be named `app.R` or placed in a subdirectory to work with the Docker base image for Shiny. This is because the `rocker/shiny` image is configured to automatically look for `app.R` in specific locations, such as the root directory or within a subdirectory. This convention simplifies deployment by standardizing file locations.

2.2 Docker Overview

We set up two Docker containers to host the Shiny app and API, connecting them via a custom Docker network. This configuration ensured seamless communication between the containers.

2.3 Shiny Dockerfile

The Dockerfile for the Shiny container is shown below:

```

# Use the R base image with Shiny support
FROM rocker/shiny:latest

# Install R packages required by the Shiny app
RUN install2.r --error \
    shiny \
    httr \
    jsonlite \
    ggplot2 \
    png \
    shinythemes

# Install system dependencies for handling PNGs
RUN apt-get update && apt-get install -y \
    libpng-dev \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Set the environment variable for API base URL
ENV API_BASE_URL=http://2api_container:8001

# Copy your Shiny app code to the container
COPY app.R /srv/shiny-server/app.R

# Set permissions for the Shiny app directory
RUN chown -R shiny:shiny /srv/shiny-server/app.R

# Expose the default Shiny port
EXPOSE 3838

# Run the Shiny server
CMD ["/usr/bin/shiny-server"]
```

2.4 API Dockerfile

The Dockerfile for the API container is:

```
FROM rocker/r-ver:4.4.2

# Set the working directory inside the container
WORKDIR /usr/src/app/2random-E11

# Copy the entire project into the container
COPY . /usr/src/app/2random-E11

# Print the directory structure for debugging purposes
RUN ls -R /usr/src/app/2random-E11

# Install necessary R packages
RUN R -e "install.packages(c('ggplot2', 'dplyr', 'shiny', 'plumber',
   'caret', 'pROC', 'PRROC'))"

# Expose the API port
EXPOSE 8001

# Run the R script to start the API
CMD ["Rscript", "04_RunAPI.R"]
```

2.5 Building Docker Images

To build the Docker images:

```
docker build -t t2dapi .
docker build -t t2dshiny -f Dockerfileshiny .
```

Ensure the Dockerfiles are in the same folder as the application files. Verify the images using `docker images`.

2.6 Running Containers

Start the containers with the following commands:

```
docker run -d -p 8001:8001 --name 2api_container t2dapi
docker run -d -p 3838:3838 --name 2shiny_container -e
    API_BASE_URL=http://2api_container:8001 t2dshiny
```

2.7 Connecting Containers with a Custom Network

To enable communication between containers:

1. Create a custom Docker network:

```
docker network create shiny-api-network
```

2. Attach both containers to the network:

```
docker network connect shiny-api-network 2api_container
docker network connect shiny-api-network 2shiny_container
```

2.8 Docker Compose

Use Docker Compose for simplified container management. The `docker-compose.yml` file:

```
services:
  api:
    container_name: 2api_container
    image: dakotacoloroso/t2dapi:latest
    ports:
      - "8001:8001" # Map API container's port 8001 to host's port 8001
    networks:
      - shiny-api-network # Ensure the API is on the correct network
    expose:
      - "8001" # Expose port internally for the Shiny container
    environment:
      - PLUMBER_PORT=8001 # Ensure API runs on port 8001

  shiny-app:
    container_name: 2shiny_container
    image: dakotacoloroso/t2dshiny:latest
    ports:
      - "3838:3838" # Map Shiny app's internal port to host's port 3838
    networks:
      - shiny-api-network # Ensure Shiny is on the same network as API
    depends_on:
      - api # Ensure API starts before Shiny app
    environment:
      - API_BASE_URL=http://2api_container:8001 # Use the correct
          internal hostname and port

networks:
  shiny-api-network:
    driver: bridge # Use the default bridge network driver
```

2.9 Building and Running with Docker Compose

To build and run the containers:

```
docker-compose build
docker-compose up -d
```

2.10 Accessing the Application

- **Shiny App**: Access at `http://localhost:3838`.
- **API**: Access at `http://localhost:8001`.

The Shiny app communicates with the API using the `API`$_{B}ASE_{U}RLenvironmentvariable.$

References

[1] H. Devarbhavi, S. K. Asrani, J. P. Arab, Y. A. Nartey, E. Pose, and P. S. Kamath, ``Global burden of liver disease: 2023 update,'' *Journal of hepatology*, vol. 79, no. 2, pp. 516--537, 2023.

[2] R. E. Kharoua, `` predict liver disease: 1700 records dataset,'' 2024. [Online]. Available: https://www.kaggle.com/dsv/8652019

[3] P. Meier and H. K. Seitz, ``Age, alcohol metabolism and liver disease,'' *Current Opinion in Clinical Nutrition & Metabolic Care*, vol. 11, no. 1, pp. 21--26, 2008.

[4] J. Guy and M. G. Peters, ``Liver disease in women: the influence of gender on epidemiology, natural history, and patient outcomes,'' *Gastroenterology & hepatology*, vol. 9, no. 10, p. 633, 2013.

[5] C. L. Hart, D. S. Morrison, G. D. Batty, R. J. Mitchell, and G. D. Smith, ``Effect of body mass index and alcohol consumption on liver disease: analysis of data from two prospective cohort studies,'' *Bmj*, vol. 340, 2010.

[6] O. Niemelä and P. Alatalo, ``Biomarkers of alcohol consumption and related liver disease,'' *Scandinavian journal of clinical and laboratory investigation*, vol. 70, no. 5, pp. 305--312, 2010.

[7] D. Ellinghaus, ``How genetic risk contributes to autoimmune liver disease,'' in *Seminars in immunopathology*, vol. 44, no. 4. Springer, 2022, pp. 397--410.

[8] S. Ryu, Y. Chang, H.-S. Jung, K. E. Yun, M.-J. Kwon, Y. Choi, C.-W. Kim, J. Cho, B.-S. Suh, Y. K. Cho *et al.*, ``Relationship of sitting time and physical activity with non-alcoholic fatty liver disease,'' *Journal of hepatology*, vol. 63, no. 5, pp. 1229--1237, 2015.

[9] N. A. Baig, S. K. Herrine, and R. Rubin, ``Liver disease and diabetes mellitus.'' *Clinics in laboratory medicine*, vol. 21, no. 1, pp. 193--207, 2001.

[10] D. Oikonomou, G. Georgiopoulos, V. Katsi, C. Kourek, C. Tsioufis, A. Alexopoulou, E. Koutli, and D. Tousoulis, ``Non-alcoholic fatty liver disease and hypertension: coprevalent or correlated?'' *European journal of gastroenterology & hepatology*, vol. 30, no. 9, pp. 979--985, 2018.