Classification Models for Predicting Non-Alcoholic Fatty Liver Disease

Rishi Rokariya
MTech ICT ML
202411059

Aniruddha Shinde
MTech ICT ML
202411065

Adarsh Gupta
MTech ICT WCSP
202411083

***Abstract*—This project focuses on using machine learning classification models to predict the presence or absence of NAFLD based on physical parameters such as age, gender, BMI, waist circumference, and others. We employed three classifiers—Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF)—to analyse the dataset. After performing extensive exploratory data analysis (EDA) and pre-processing, the models were trained and evaluated. Random Forest outperformed SVM and Naive Bayes, achieving the highest accuracy. This project aims to demonstrate the feasibility of using machine learning to predict NAFLD and offers an affordable and scalable solution for healthcare in India. The goal is to develop a predictive model that can aid healthcare professionals in early detection, thereby facilitating timely interventions to mitigate the risks associated with advanced liver diseases.**

I. Introduction

Non-Alcoholic Fatty Liver Disease (NAFLD) is a condition where fat builds up in the liver without excessive alcohol consumption. It is a significant health issue affecting a large proportion of the population worldwide, particularly in India. NAFLD is a growing health concern, with studies showing that nearly 40% of Indians are affected by this condition. However, NAFLD often remains undiagnosed in its early stages due to the absence of noticeable symptoms, leading to delayed intervention and the potential progression to severe liver diseases, including fibrosis. Early detection of NAFLD is crucial for effective management and prevention of complications. Given the lack of awareness and the expensive nature of medical tests, such as liver biopsy, which costs approximately Rs.4500 in India, an affordable alternative is crucial. Machine learning offers a promising solution by leveraging widely available physical parameters (e.g., BMI, age, gender, waist circumference) to predict the presence of NAFLD. The goal is to build a model that can efficiently classify individuals into two categories: fibrosis present and no fibrosis, based on their physical attributes.

II. Dataset Information

The dataset used in this project can be accessed in raw format "raw dataset", and additional information is available in the Kaggle dataset "Kaggle dataset". Size of dataset of 605 (rows), 9 features and 1 output (10 columns). The 9 features are as follows:

- **Age**: Age of the patient (in years).
- **Gender**: Coded as 0 for Female and 1 for Male.
- **Height**: Height of the patient (in cm).
- **Weight**: Weight of the patient (in kg).
- **Body Mass Index (BMI)**: Calculated as weight (kg) divided by the square of height (m).
- **Waist Circumference**: Measured in cm.
- **Hip Circumference**: Measured in cm.
- **Diabetes**: Binary variable, 0 for No and 1 for Yes.
- **Smoking Status**: Coded as 0 for No Smoking and 1 for Smoking.
- **Fibrosis Status**: Binary variable, 0 indicates No Fibrosis, while 1 indicates Fibrosis (Fibrosis 1 and above).

Output classes are 0 (absent) and 1 (present).
Output class ratio is 196:409.

III. Packages required

Below mentioned are the packages used for the Project:

TABLE I
List of Python packages used and their versions.

Package Name	Version Used
`numpy`	1.26.0
`pandas`	2.1.3
`matplotlib`	3.8.0
`scipy`	1.11.3
`scikit-learn`	1.3.2
`imbalanced-learn`	0.12.4

IV. Exploratary Data Analysis

Exploratory Data Analysis (EDA) is a critical first step in the data analysis process, as it helps to uncover patterns, relationships, and anomalies within the dataset. By using various statistical and graphical techniques, such as histograms, box plots, and correlation matrices, EDA allows data scientists to gain a deeper understanding of the data's structure, distribution, and potential issues. It helps in identifying missing values, outliers, and trends that may not be immediately apparent. Moreover, EDA provides insights into the variables' relationships, which can inform the selection of appropriate models for further analysis. Ultimately, performing

EDA ensures that data is clean, well-understood, and ready for more sophisticated techniques like predictive modeling, leading to better and more reliable outcomes.

TABLE II
DESCRIPTION OF THE DATASET VARIABLES.

#	Column	Non-Null Count	Dtype
0	Age	605 non-null	int64
1	Height (cm)	605 non-null	int64
2	Weight (kg)	605 non-null	int64
3	Body Mass Index	605 non-null	float64
4	Waist Circumference (cm)	576 non-null	float64
5	Hip Circumference (cm)	598 non-null	float64
6	Diabetes (No=0, Yes=1)	605 non-null	int64
7	Fibrosis status (No=0, Yes=1)	605 non-null	int64
8	Gender (Female=0, Male=1)	605 non-null	int32
9	Smoking Status (Not Smoking=0, Smoking=1)	605 non-null	int32

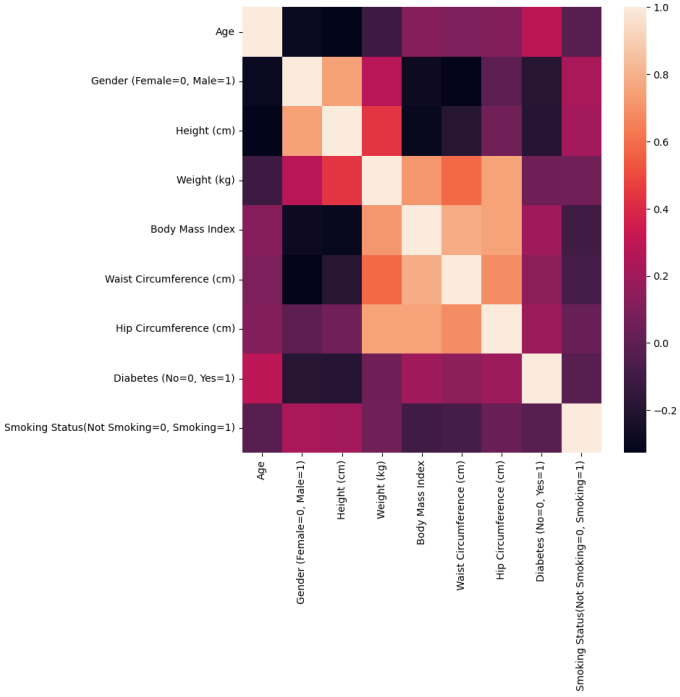

Fig. 1. Feature Correlation Heatmap

A. Missing Values Detection

We can see we have some missing values in ”Waist Circumference” and ”Hip Circumference”

B. Missing value treatment

Since Missing Value are very less we can impute the missing Values with mean data for that particular column

C. Outlier Detection

The method used here for outlier detection is based on the Interquartile Range (IQR). The IQR is a measure of statistical dispersion, calculated as the difference between the 75th percentile (Q3) and the 25th percentile (Q1) of the data.

TABLE III
MISSING VALUES IN EACH COLUMN

Column	Missing Values
Age	0
Height (cm)	0
Weight (kg)	0
Body Mass Index	0
Waist Circumference (cm)	29
Hip Circumference (cm)	7
Diabetes (No=0, Yes=1)	0
Fibrosis Status (No=0, Yes=1)	0
Gender (Female=0, Male=1)	0
Smoking Status (Not Smoking=0, Smoking=1)	0

```
# Fill missing values with the mean
data = data.fillna(data.mean())  # Replace missing values with the mean

# Verify that there are no missing values left
print("\nMissing values after treatment:")
data.isnull().sum()
```

Fig. 2. Missing Value Treatment Code

For each numerical column, this approach identifies potential outliers as values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5* IQR. This method is robust against non-normal distributions and effectively highlights extreme deviations in the data. The total number of outliers is computed for each column, and a summary of outliers is returned for further analysis.

TABLE IV
OUTLIER DETECTION SUMMARY

Column	Outliers Count
Age	0
Gender (Female=0, Male=1)	0
Height (cm)	0
Weight (kg)	15
Body Mass Index	16
Waist Circumference (cm)	19
Hip Circumference (cm)	11
Diabetes (No=0, Yes=1)	0
Smoking Status (Not Smoking=0, Smoking=1)	0

D. Outlier Treatment

According to 3 sigma rule, any data point that lies more than 3 standard deviations away from the mean is considered an outlier. This is based on the assumption that the data follows a normal (Gaussian) distribution, where 99.7% of the data points should lie within three standard deviations from the mean.

As seen from the distribution of waist circumference data points which had most amount of outliers, The nature of distribution has not changes much so the treatment of Outliers did not change the distribution of data since the number of outiers were small.

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Function to cap outliers at 3-sigma
def cap_outliers_3_sigma(data):
    capped_df = data.copy()
    for column in data.select_dtypes(include=[np.number]):  # Apply only to numeric columns
        mean = data[column].mean()
        std = data[column].std()
        lower_limit = mean - 3 * std
        upper_limit = mean + 3 * std
        capped_df[column] = np.clip(data[column], lower_limit, upper_limit)
    return capped_df

# Cap outliers in the data
data_2 = cap_outliers_3_sigma(data)
```

Fig. 3. Outlier Treatment Code

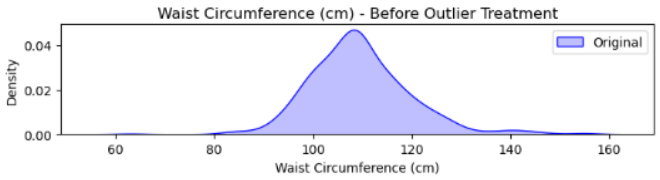

Fig. 4. Waist Circumference Distribution Before Treatment

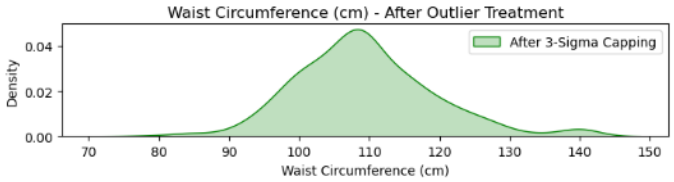

Fig. 5. Waist Circumference Distribution After Treatment

E. Standardization

Standardization of data is the process of rescaling features to have a mean of zero and a standard deviation of one. This is typically done by subtracting the mean of each feature and dividing by its standard deviation.

```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
# Standardization
scaler = StandardScaler()
data_2_standardized = scaler.fit_transform(data_2[input_col]) # Pass input_col directly
data_2_standardized = pd.DataFrame(data_2_standardized, columns=input_col) # Pass input_col directly
df = data_2_standardized
data_2.head(5)
```

Fig. 6. Standardization Code

V. Model

A. Navie Bayes Classifier

The Naive Bayes classifier is a probabilistic classification algorithm based on Bayes' Theorem. It applies the principle of conditional probability with a strong assumption that the feature used to predict the class are independent, which is why it is termed as "naive". It calculates the probability of a class given the input features and selects the class with the highest probability.

Bayes' Theorem is expressed as:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Where:

- $P(C|X)$ is the posterior probability of class C given feature vector X.
- $P(X|C)$ is the likelihood, i.e., the probability of observing features X given class C.
- $P(C)$ is the prior probability of class C.
- $P(X)$ is the evidence, i.e., the total probability of observing features X.

We used the Gaussian Naive Bayes (GaussianNB) from scikit-learn for this project. Gaussian Naive Bayes is one of the versions of naïve bayes classifier which is used when the features are continuous and follows a Gaussian (normal) distribution. The likelihood is computed using the probability density function of the normal distribution:

The likelihood for a Gaussian distribution is given by:

$$P(X_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(X_i - \mu_C)^2}{2\sigma_C^2}\right)$$

Where:

- X_i is the i-th feature of the data.
- μ_C is the mean of the feature X_i for class C.
- σ_C is the standard deviation of the feature X_i for class C.

B. Support Vector Machine

A Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks, but it is most commonly used for classification. The goal of an SVM is to find a decision boundary (or hyperplane) that best separates data points of different classes while maximizing the margin between them. In an n-dimensional space, a hyperplane is a flat affine subspace of one dimension less than the space itself, that acts as a boundary that separates different classes. For example, in a 2D space, the hyperplane is a line, and in 3D space, it is a plane.

Mathematically, a hyperplane can be expressed as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Where:

- w is the weight vector (normal to the hyperplane),
- x is the feature vector of a data point, and
- b is the bias term.

The margin is the distance between the hyperplane and the closest data point from either class. The goal of SVM is to maximize this margin to ensure that the classifier is as confident as possible in making predictions. The larger the margin, the better the model generalizes to unseen data, as it is less likely to overfit.

The support vectors are the data points that are closest to the decision boundary (hyperplane). These points are crucial in determining the optimal hyperplane because they directly affect the position and orientation of the hyperplane. In fact, only the support vectors are needed to define the decision boundary. Data can be either linearly separable or non-linearly separable.

SVMs can handle non-linearly separable data by using a technique called the kernel trick. Instead of finding a linear decision boundary in the original input space, SVM using kernel trick, maps the data into a higher-dimensional feature space where a linear hyperplane can separate the data.

There are many kernels out of which RBF (Radial Basis Function) kernel and Linear kernel are most popular. To decide which one to use, we need to know whether the data is linearly separable or not. For this we used the Principal Component Analysis (PCA) which is a dimensionality reduction technique that transforms data into a set of orthogonal axes (principal components) that capture the maximum variance. In this project we captured 95% of variance and got the following:

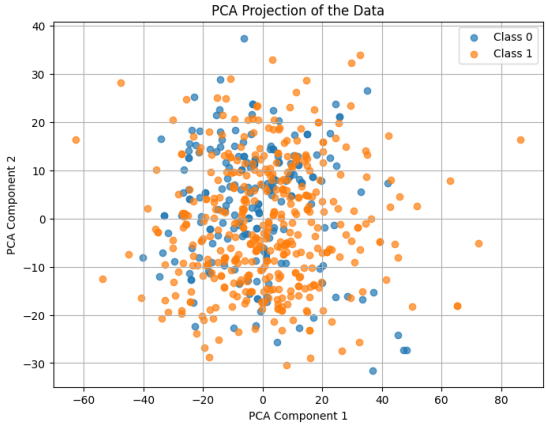

Fig. 7. Scatter plot of PCA-projected data with two components

As seen the data is not linearly separable. So we used RBF kernel which maps the data into a higher-dimensional space, where it is easier to find a linear separation between the classes. The RBF kernel is based on the Gaussian function, which is mathematically expressed as:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$$

Where:

- $K(x, x')$ is the kernel function that computes the similarity between two data points x and x'.
- $\|x - x'\|^2$ is the squared Euclidean distance between the two data points.
- σ is a parameter that controls the width of the Gaussian distribution.
- Sometimes, the parameter $\gamma = \frac{1}{2\sigma^2}$ is used, and it is often referred to as the gamma parameter in the context of SVM (Support Vector Machines).

While training the model we faced an issue called imbalanced class ratio, in which the SVM gives only one class (which is higher in ratio) for any given input which in this case is the 1 or 'yes'. To solve this issue, we used SMOTE (Synthetic Minority Over-sampling Technique) analysis which is a popular technique used to address class imbalance in machine learning datasets. It works by generating synthetic examples for the minority class, which helps the model learn more about the minority class and improve its generalization.

For further model enhancement and improved accuracy, we did hyperparameter tuning. The two main hyperparameters that typically need tuning in SVM are:
1. C (Regularization parameter) that Controls the trade-off between maximizing the margin and minimizing classification errors. We used 4 values for C i.e. 0.1, 1, 10 and 100.
2. Gamma (kernel coefficient) that defines the influence of each training point on the decision boundary. It controls the shape of the decision boundary by determining how far the influence of a single training example reaches. We used 4 values of Gamma i.e. 'auto', 'scale', 0.1 and 1.

To evaluate the above hyperparameters we did Grid Search Cross-Validation (GridSearchCV) which is a technique used to tune hyperparameters of a machine learning model by exhaustively searching through a specified hyperparameter grid. It evaluates all possible combinations of hyperparameters using cross-validation to find the best model configuration based on performance. GridSearchCV helps to avoid overfitting by selecting the model with the best generalization ability. The result is the hyperparameter set that yields the highest cross-validation score.
Here, we performed 5 folds of CV, i.e. 4x4x5 a total of 80 fits. It gave C=10 and gamma=0.1 as best hyperparameters.

C. Random Forest

A Random Forest is an ensemble learning method, which means it combines multiple models to improve performance and accuracy. Specifically, it is made up of many decision trees and uses them to make predictions. The idea behind Random Forest is to create a "forest" of decision trees, each trained on a different subset of the data. The final prediction is made by aggregating the predictions of all the individual trees. This ensemble approach generally leads to a more robust model compared to a single decision tree, reducing the overfitting problem. It can be computationally expensive and harder to interpret, especially when dealing with large datasets.
We defined n=100 i.e. the number of trees in the forest.

VI. Results

To evaluate the performance of the machine learning models, we use the following:
1. Confusion Matrix:
Confusion matrix is a table used to evaluate the performance of a classification model, showing the actual vs. predicted classifications. It breaks down the prediction results into four categories:
• True Positive (TP): The number of instances that were correctly predicted as positive.
• False Positive (FP): The number of instances that were incorrectly predicted as positive (i.e., predicted positive but actually negative).
• True Negative (TN): The number of instances that were correctly predicted as negative.
• False Negative (FN): The number of instances that were incorrectly predicted as negative (i.e., predicted negative but actually positive).

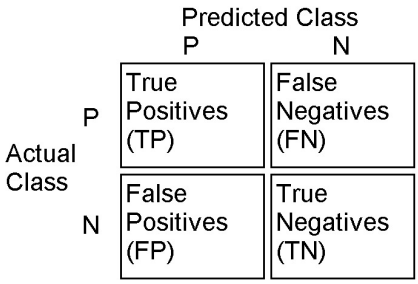

Fig. 8. Confusion Matrix

This matrix helps us calculate different performance metrics like accuracy, precision, recall, and F1-score.

2. Accuracy, which is the proportion of correct predictions (both positives and negatives) among the total predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3. Precision which is also called positive predictive value measures how many of the instances predicted as positive are actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

4. Recall which is also called sensitivity or true positive rate measures how many of the actual positive instances the model correctly predicted as positive.

$$\text{Recall} = \frac{TP}{TP + FN}$$

5. The F1-score which is the harmonic mean of precision and recall, providing a balance between them. It's especially useful when you need a single metric to evaluate a model's performance and when the class distribution is imbalanced.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A. Navie Bayes Classifier

Accuracy:	0.6923
Precision:	0.7742
Recall:	0.7742
F1 Score:	0.7742

B. Support Vector machine

Accuracy:	0.7363
Precision:	0.7436
Recall:	0.9355
F1 Score:	0.8286

C. Random Forest

Accuracy:	0.7418
Precision:	0.7619
Recall:	0.9032
F1 Score:	0.8266

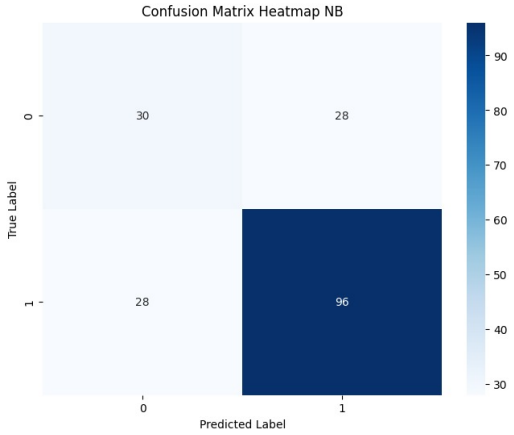

Fig. 9. Navie Bayes Classifier Heatmap

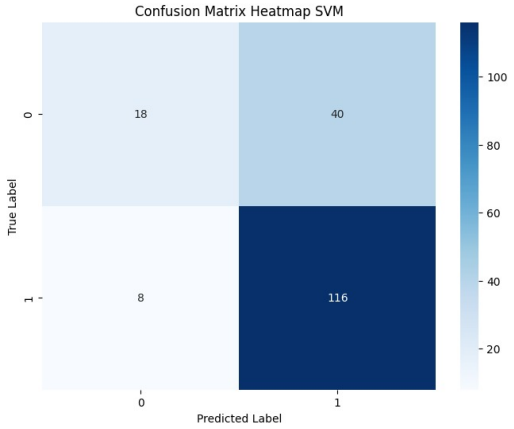

Fig. 10. Support Vector Machine Heatmap

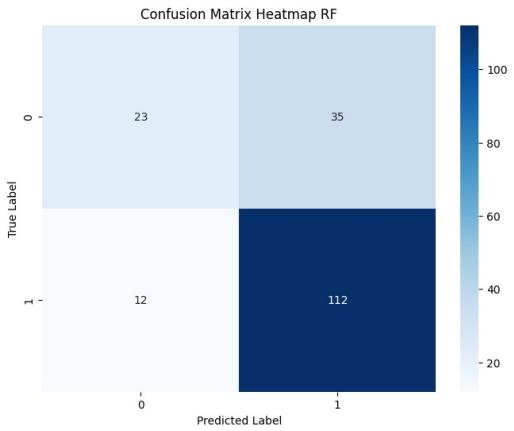

Fig. 11. Random Forest Heatmap

As it can be clearly seen the accuracy of Random forest (74.18%) is higher than the accuracy of SVM (73.63%) and Naïve Bayes (69.23%), Random forest outperforms SVM and Naïve Bayes Classifier.

CONCLUSION

In this project, we successfully implemented machine learning classifiers to predict the presence or absence of NAFLD based on physical parameters like age, gender, BMI, waist circumference, and others. After performing detailed EDA, we trained and evaluated three machine learning classifiers and the Random Forest model delivered the best performance, achieving the highest accuracy of 74.18%, along with a balanced precision, recall, and F1-score. The results indicate that machine learning can be a reliable and efficient tool for predicting NAFLD, offering a cost-effective and scalable alternative to traditional lab tests, especially in resource-limited settings like India. The use of physical parameters that can be easily measured, such as BMI and waist circumference, makes this approach highly accessible. The highlight of this project is that individuals can measure the physical parameters which does not require any medical intervention and a suitable model will give the predictive outcomes. Future work could be focused on developing real-time prediction systems to facilitate early diagnosis and intervention, further contributing to the management of NAFLD and potentially reducing the healthcare burden associated with liver diseases.

REFERENCES

[1] 40% in India suffer from non-alcoholic fatty liver: Doctors. https://economictimes.indiatimes.com/industry/healthcare/biotech/healthcare/40-in-india-suffer-from-non-alcoholic-fatty-liver-doctors/articleshow/92109732.cms?from=mdr.

[2] A Machine Learning Based Framework to Identify and Classify Non-alcoholic Fatty Liver Disease in a Large Scale Population Weidong Ji, 1 ,Mingyue Xue, 2 , Yushan Zhang, 3 Hua Yao, 4 and Yushan Wang 4 Published online 2022 Apr 4. doi: 10.3389/fpubh.2022.846118

[3] Fatty Liver Disease dataset. https://www.kaggle.com/datasets/tourdeglobe/fatty-liver-disease