CSE449: High Performance Computing

Leveraging High-Performance Computing for Heart Disease Prediction

Submitted by:

Shouvik Banerjee Argha (20301118)
Arian Wazed (20301039)
Group: 19-B

Table of Contents

1.Introduction:

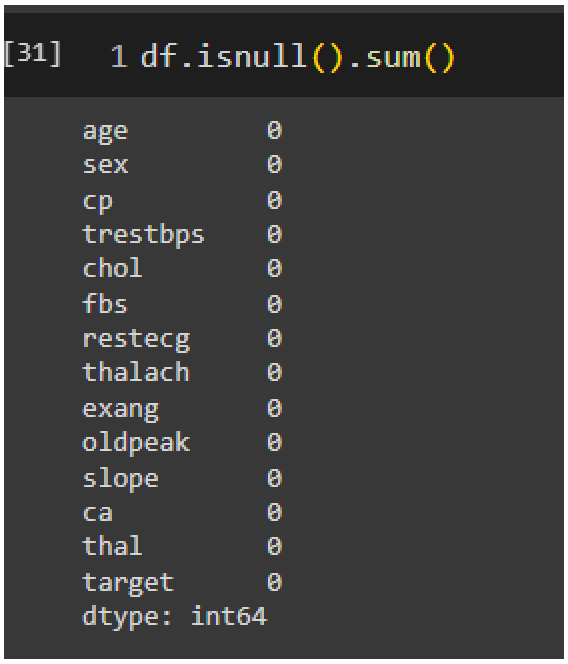
Heart disease poses significant health challenges, necessitating accurate prediction for timely intervention. Leveraging High-Performance Computing (HPC) in tandem with Machine Learning (ML) algorithms offers a promising avenue for enhancing predictive accuracy. This study conducts a comparative analysis of K-Nearest Neighbors (KNN), Naive Bayes, Logistic Regression, and Decision Trees, using biological parameters like cholesterol levels, blood pressure, sex, and age for heart disease prediction. By evaluating the accuracy of these algorithms, we aim to identify the most effective approach, contributing to advancements in proactive healthcare interventions and precision medicine.

2. Dataset Description:

The dataset comprises 303 rows and 14 columns, where each row represents a data point and each column corresponds to a feature or attribute. Among these features, there are 13 predictors and one target variable, making it a classification problem with a binary outcome. The features encompass both quantitative aspects like age, resting blood pressure, serum cholesterol, maximum heart rate achieved, and ST depression induced by exercise relative to rest, as well as categorical attributes such as sex, chest pain type, fasting blood sugar, resting electrocardiographic results, exercise-induced angina, slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, and thalassemia. Understanding correlations between these features is facilitated by a correlation heatmap matrix, where darker points signify stronger correlations and lighter points indicate weaker or no correlations. Additionally, the dataset exhibits class imbalance, meaning that certain classes within the target variable have significantly fewer samples compared to others. Addressing this imbalance involves imputing missing values with column means and employing algorithms tailored for imbalanced data to enhance model performance and ensure fair representation of all classes, measured through metrics like F1-score, precision, and recall.

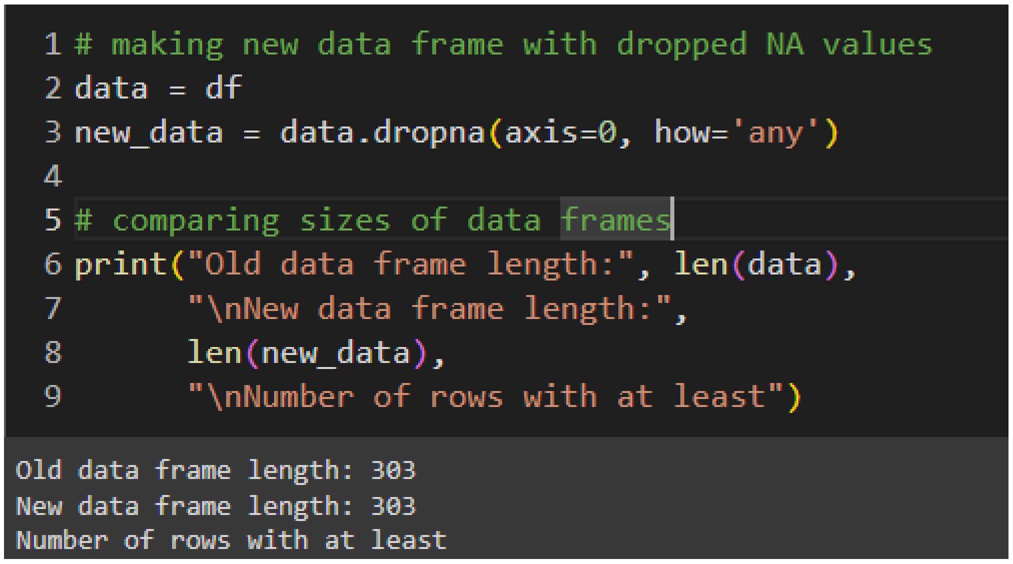
3.Dataset Preprocessing:

Faults: In the dataset there are some null values which we imputed later.
Solutions: Identified the null values and removed the row which contained Null values.



```
[31]  df.isnull().sum()

age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```



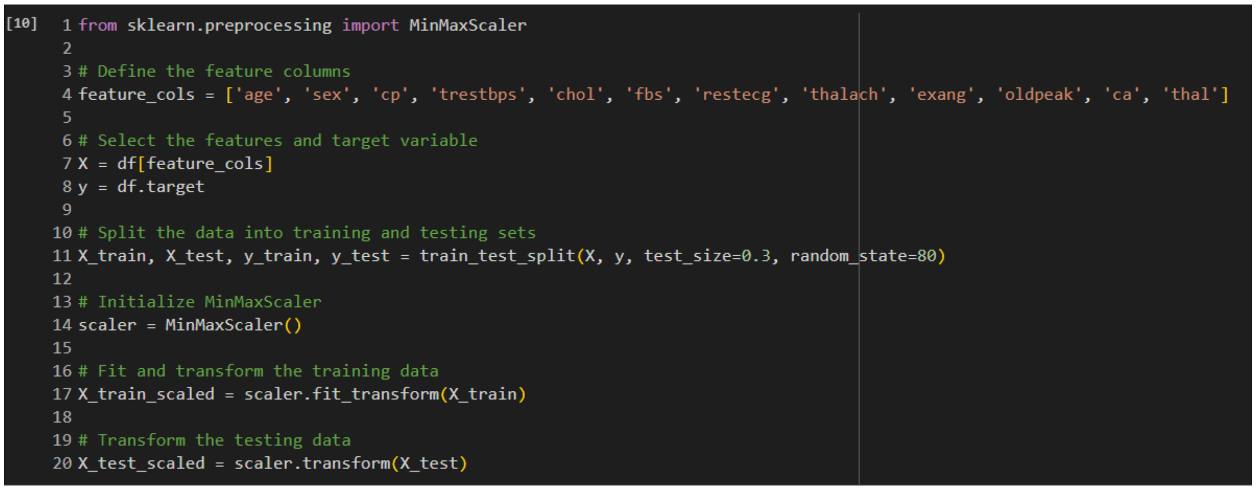
```
# making new data frame with dropped NA values
data = df
new_data = data.dropna(axis=0, how='any')

# comparing sizes of data frames
print("Old data frame length:", len(data),
      "\nNew data frame length:",
      len(new_data),
      "\nNumber of rows with at least")
```

```
Old data frame length: 303
New data frame length: 303
Number of rows with at least
```

4. Feature Scaling :

MinMaxScaler is useful when the data has a bounded range.Scaling these values using MinMaxScaler ensures that the values are within a fixed range and contributes equally to the analysis. In the dataset, some of the values were outliers that's why it's needed.


```
[10]
from sklearn.preprocessing import MinMaxScaler

# Define the feature columns
feature_cols = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'ca', 'thal']

# Select the features and target variable
X = df[feature_cols]
y = df.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=80)

# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the testing data
X_test_scaled = scaler.transform(X_test)
```

5. Dataset Splitting:

The dataset which has been used, the test size is 0.3 So, the training set ratio is 70%, testing set ratio 30% and random_state is 80.

6.Conclusion:

In conclusion, this study undertook a comprehensive approach to predict heart disease using high-performance methods. Through meticulous preprocessing, feature scaling, and dataset splitting, we have laid a robust foundation for accurate predictions. Leveraging a well-curated dataset with detailed descriptions, we aimed to develop a predictive model that can assist in early detection and prevention of heart diseases. Moving forward, the application of advanced machine learning algorithms on this prepared dataset holds promise for enhancing diagnostic accuracy and improving patient outcomes. This research underscores the significance of utilizing sophisticated techniques in healthcare analytics to address critical challenges such as cardiovascular diseases.