Machine Learning Programming Assignment

Title: Heart Disease Prediction using Machine Learning
Name: Herath H.M.R.K
Student ID: IT20665548
Date: 18 April 2025

1. Introduction

The World Health Organization estimates that cardiovascular illnesses cause 17.9 million deaths annually, making them the world's top cause of mortality. These conditions are frequently brought on by a confluence of risk factors, including smoking, heavy alcohol use, poor diet, and inactivity. It is essential to diagnose cardiac disease as soon as possible in order to save lives and lower medical expenses.

Machine learning (ML) has become a potent instrument in the healthcare sector in recent years, allowing for the creation of prediction models that help physicians diagnose and treat illnesses more successfully. Large volumes of medical data can be analyzed by ML algorithms, which can also find patterns that humans might not see right away.

Using a clinical dataset that is openly accessible, we use machine learning in this assignment to forecast the existence of heart disease. Building accurate predictive models and comparing their performance are our goals utilizing two supervised learning algorithms: Random Forest and Logistic Regression. Finding the best algorithm for identifying patients at risk of heart disease based on a variety of clinical markers is the aim.

2. Dataset Description

This project uses the "Heart Failure Prediction Dataset" from Kaggle. It includes 12 characteristics, including both numerical and categorical variables, and 918 patient data entries. The objective is to use these characteristics to predict a patient's likelihood of developing heart disease.

Source: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction
Target Column: HeartDisease (0 = No, 1 = Yes)

Context of the Dataset : This dataset includes important medical metrics that are frequently gathered during cardiac examinations and diagnostic procedures. It contains physiological data (blood pressure, cholesterol, resting blood pressure), exercise-related metrics, demographic characteristics (age, sex), and kinds of chest discomfort. Cardiologists usually utilize these factors to assess the risk of heart disease.

Size and Type:

- **Number of records (rows):** 918
- **Number of features (columns):** 12 (11 independent features + 1 target)
- **Data types:** Mixed (categorical and numerical)

Features:

- Age (numeric)
- Sex (categorical: M/F)
- ChestPainType (categorical: typical angina, atypical angina, non-anginal pain, asymptomatic)
- RestingBP (Resting blood pressure - numeric)
- Cholesterol (numeric)
- FastingBS (fasting blood sugar > 120 mg/dl - binary)
- RestingECG (categorical: normal, ST-T wave abnormality, left ventricular hypertrophy)
- MaxHR (maximum heart rate achieved - numeric)
- ExerciseAngina (binary: Yes/No)
- Oldpeak (numeric - ST depression induced by exercise)
- ST_Slope (categorical: upsloping, flat, downsloping)

Target:

- HeartDisease: 0 (No), 1 (Yes)

This dataset is perfect for implementing and assessing machine learning models in the healthcare industry since it is well-suited for binary classification tasks and offers a real-world clinical context.

Sample data:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	HeartDisease
0	40	M	ATA	140	289	0	0
1	49	F	NAP	160	180	0	1

3. Data Preprocessing

The dataset underwent a number of pretreatment procedures to make sure it was appropriate for machine learning methods. These actions are essential for enhancing model functionality and guaranteeing compatibility with scikit-learn methods.

Handling Categorical Variables: The dataset included a number of categorical characteristics, such as "Sex," "ChestPainType," "RestingECG," "ExerciseAngina," and "ST_Slope." Since ML models couldn't use them directly, we used pd.get_dummies() to implement one-hot encoding. By employing drop_first=True to prevent multicollinearity, this converted categorical data into a numerical format and produced distinct binary columns for each category.

- **Checking for Missing Values:** We used df.isnull().sum() to confirm that there were no missing or null values in the dataset. This prevented models from experiencing mistakes as a result of insufficient data during training.
- **Feature Scaling:** We utilized StandardScaler to standardize all numerical variables because features such as "RestingBP," "Cholesterol," and "MaxHR" were on various scales. By taking this step, the learning algorithm was kept from being dominated by features with wider ranges.
- **Correlation Analysis:** The correlation between numerical features was visualized by creating a heatmap with sns.heatmap(). This made it easier to see any redundant or highly correlated features, however in this instance, every feature was kept for model training.
- **Train/Test Split:** Train_test_split() was used to split the data into training and testing sets in an 80-20 ratio following preprocessing. As a result, the model might be trained on a subset of the data and tested on an additional, hidden subset to determine its generalizability. The scikit-learn function train_test_split() was used to divide the data into 80% training and 20% testing.

4. Algorithms Used

The Random Forest Classifier and Logistic Regression were the two supervised learning algorithms chosen for comparison in this study. Both have distinct advantages in terms of robustness, accuracy, and interpretability and are frequently employed in classification issues.

Logistic Regression

A statistical model for binary classification problems is called logistic regression. It uses a logistic function to predict the likelihood that a given input falls into a specific category. When there is a roughly linear relationship between the input features and the target variable, this technique works best.

Interpretability: Each feature's impact on the prediction can be understood by interpreting the coefficients that logistic regression delivers.

- **Efficiency:** When the data is linearly separable, it works well and is computationally efficient.
- **Output:** generates probabilities that can be thresholded to provide predictions that are binary.

For challenges involving binary classification, logistic regression provides a solid baseline model that is simple to use and comprehend.

Random Forest Classifier

Random Forest is an ensemble learning technique that generates the class mode for classification tasks by training several decision trees. In order to decrease variance and improve accuracy, it is predicated on the idea of bagging, or bootstrap aggregating.

- **Robustness:** averaging makes it less prone to overfit than individual decision trees.
- **Feature Importance:** aids in feature selection by offering insights regarding feature relevance.
- **Performance:** produces greater accuracy in general, particularly when working with big datasets that contain non-linear connections.

Random Forest is a powerful model for predicting heart disease in situations where feature connections may not be strictly linear because of its exceptional ability to handle high-dimensional data and capture intricate patterns.

These two algorithms offer a useful comparison between a more sophisticated, potent model (Random Forest) and a straightforward, interpretable model (Logistic Regression).

enhances accuracy and decreases overfitting by averaging the output of several trees.

5. Results

Accuracy, precision, recall, and F1-score were among the important classification measures used to assess the models' performance. A thorough understanding of each model's capacity to accurately categorize cases of heart disease is offered by these measurements.

Logistic Regression Performance:

- **Accuracy:** 86% — percentage of overall accurate forecasts.
- **Precision:** 85% — proportion of expected positives that turn out to be positive.
- **Recall:** 87% — proportion of true positives that are accurately detected.
- **F1-Score:** 86% — precision and recall harmonic mean.

All measures were balanced, and logistic regression did rather well. Its ease of use makes it appropriate for situations where interpretability is crucial as well as for rapid baseline testing.

Random Forest Performance:

- **Accuracy:** 90%
- **Precision:** 91%
- **Recall:** 89%
- **F1-Score:** 90%

The Random Forest model outperformed the Logistic Regression approach. It was better able to generalize to the test set because of the ensemble technique, which enabled it to capture more intricate correlations in the data.

Confusion Matrix Analysis: There were less false positives and false negatives with Random Forest than with Logistic Regression, according to the confusion matrix. This adds to its overall robustness and greater F1-score.

Visualizations: The comparison of the two models was visualized using heatmaps and bar charts. These visual aids helped pinpoint the advantages and disadvantages of each strategy while also validating the numerical results.

The more successful Random Forest model for this dataset was confirmed by the generation and analysis of confusion matrices and classification reports for both models.

6. Comparison and Discussion

Accuracy, precision, recall, and F1-score were the main performance indicators that showed the Random Forest classifier outperforming Logistic Regression. Because Random Forest is an ensemble model and can handle non-linear relationships, this finding implies that it is superior at capturing complicated patterns in the dataset.

A linear relationship between the input features and the result is assumed by logistic regression, notwithstanding its simplicity and ease of interpretation. On the other hand, Random Forest is typically less susceptible to data noise and uses the combined strength of several decision trees to produce reliable forecasts.

Logistic regression may still be the better option from a practical standpoint in healthcare settings where model transparency is essential. It helps physicians explain choices to patients and other medical professionals by enabling them to comprehend how specific factors affect the forecast. In contrast, Random Forest offers a better option for cases where interpretability is less important than predictive accuracy.

Furthermore, the Random Forest model's feature importance analysis can assist in determining which factors—such as age, cholesterol, and kind of chest pain—have the greatest influence on heart disease. This knowledge may prove useful for developing more effective diagnostic instruments or for more medical research.

All things considered, each model has advantages. Random Forest performs exceptionally well and can model intricate feature interactions, but Logistic Regression is useful for its ease of use and interpretability. The particular requirements of the application setting ultimately determine which model is best, whether the emphasis is on interpretability, accuracy, or a combination of the two. in all important parameters, particularly accuracy and F1-score. Because it can model intricate non-linear interactions between variables, Random Forest offered superior prediction power even though Logistic Regression is simpler to understand.

7. Limitations and Future Work

Notwithstanding the encouraging outcomes of this experiment, a number of limitations should be taken into account when analyzing the results.

Limitations:

- There are just 918 records in the dataset, making it minimal. This restricts how broadly the models may be used to more diverse or sizable populations.
- Neither the Random Forest nor the Logistic Regression hyperparameters were adjusted. As a result, the models might not be performing at their peak efficiency.
- The dataset may not accurately reflect populations with diverse genetic, lifestyle, and environmental characteristics worldwide because it is based on a particular demographic.
- The model does not use domain-specific weighting and assumes that all features contribute equally, which may not be practical for medical diagnosis.
- External validation and real-world clinical testing were not used to confirm the model's applicability outside of the dataset.

  Future Work:

  - **Model Improvement:** Use methods such as Grid Search or Random Search to adjust hyperparameters in order to improve the accuracy and resilience of the model.
  - **Algorithm Expansion:** Try more sophisticated methods like deep learning models, gradient boosting (like XGBoost), and support vector machines (SVM).
  - **Cross-Validation:** To increase the model performance estimates' dependability, use k-fold cross-validation.
  - **Feature Engineering:** Examine how using derived or constructed features based on medical knowledge might improve the accuracy of predictions.
  - **Real-World Testing:** To make sure the model is reliable in a variety of medical contexts, validate it using various datasets or real-time clinical data.
  - **Integration into Applications:** Examine how to incorporate the model into mobile or web-based healthcare apps so that patients or physicians can use it in real-world situations.

The system may develop into a more reliable, accurate, and clinically beneficial instrument for the early diagnosis of heart disease if these drawbacks are fixed and the suggested future improvements are implemented.

Future Work:

- Use more algorithms, such SVM and XGBoost.
- Use cross-validation
- Perform hyperparameter tuning
- Work with a larger and more diverse dataset

8. References

- Kaggle Heart Failure Dataset: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction — gives the real-world dataset—which includes patient characteristics and heart disease labels—that was used in this experiment.

- Scikit-learn Documentation: https://scikit-learn.org — Official documentation for putting machine learning methods like Random Forest and Logistic Regression into practice.

- World Health Organization (WHO) Cardiovascular Disease Facts: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) — for background data on cardiovascular disease prevalence and effects worldwide.

- Python Pandas Documentation: https://pandas.pydata.org/docs/ — used to manipulate data and perform preparation operations such as managing missing values and encoding.

- Seaborn Documentation: https://seaborn.pydata.org/ — used to visualize data, such as bar charts and correlation heatmaps.

- Matplotlib Documentation: https://matplotlib.org/stable/contents.html — Used for generating plots to visualize model performance and insights.https://scikit-learn.org

9. Appendix - Source Code

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    classification_report
)
```

Load Dataset

```
df = pd.read_csv("heart.csv")
df.head()
df.info()
df.describe()
df.isnull().sum()
```

Show heatmap

```
sns.heatmap(df.select_dtypes(include='number').corr(), annot=True, cmap="coolwarm")
```

Data types

```
df.dtypes
```

Encode Categorical Variables

```
df_encoded = pd.get_dummies(df, drop_first=True)
```

Define Features and Target

```
X = df_encoded.drop("HeartDisease", axis=1)
y = df_encoded["HeartDisease"]
```

Feature Scaling

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Split Data into Train and Test

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

Train Logistic Regression

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
y_pred_lr = lr.predict(X_test)
```

Train Random Forest Classifier

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

Print Reports

```
print("Logistic Regression Report:")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))

print("Random Forest Report:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

Compare Metrics

```
metrics = {
    "Accuracy": [accuracy_score(y_test, y_pred_lr), accuracy_score(y_test, y_pred_rf)],
    "Precision": [precision_score(y_test, y_pred_lr), precision_score(y_test, y_pred_rf)],
    "Recall": [recall_score(y_test, y_pred_lr), recall_score(y_test, y_pred_rf)],
    "F1 Score": [f1_score(y_test, y_pred_lr), f1_score(y_test, y_pred_rf)]
}
```

Bar Chart for Comparison

```
labels = list(metrics.keys())
lr_scores = [score[0] for score in metrics.values()]
rf_scores = [score[1] for score in metrics.values()]
x = np.arange(len(labels))
width = 0.35

plt.figure(figsize=(10, 6))
plt.bar(x - width/2, lr_scores, width, label='Logistic Regression', color='skyblue')
plt.bar(x + width/2, rf_scores, width, label='Random Forest', color='salmon')
plt.ylabel('Score')
plt.title('Performance Comparison')
plt.xticks(x, labels)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Confusion Matrices

```
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap='Blues', ax=ax[0])
ax[0].set_title('Logistic Regression\nConfusion Matrix')
ax[0].set_xlabel('Predicted')
ax[0].set_ylabel('Actual')
```

```
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d', cmap='Greens', ax=ax[1])
ax[1].set_title('Random Forest\nConfusion Matrix')
ax[1].set_xlabel('Predicted')
ax[1].set_ylabel('Actual')

plt.tight_layout()
plt.show()
```