**VIETNAM NATIONAL UNIVERSITY – HCM**

**INTERNATIONAL UNIVERSITY**

A blue and white logo

Description automatically generated**DEPARTMENT OF COMPUTER SICENCE & ENGINEER**

SEMESTER 2 (2023-2024)

**IT159IU**

**Artificial Intelligence Project**

**Topic name:**

*May 2024*

**Group 7:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Full name - Student name** | **Student ID** | **Contribution (%)** |
| 1 | PHAM VU TUYET ANH | ITDSIU21073 |  |
| 2 | NGUYEN NHAT KHIEM | ITDSIU21091 |  |
| 3 | LIEU PHONG SON | ITITIU18200 |  |

**Table of Contents**

[I. INTRODUCTION 3](#_Toc166961965)

[1 Import needed libraries & data 3](#_Toc166961966)

[2 Description 3](#_Toc166961967)

[3 Visualization 3](#_Toc166961968)

[II. Preprocessing (TUYET ANH) 4](#_Toc166961969)

[1 Processing missing value 4](#_Toc166961970)

[**2** Encoding 4](#_Toc166961971)

[**3** Normalization 4](#_Toc166961972)

[III. APPLYING MODEL (KHIEM): 5](#_Toc166961973)

[**1. Aim** 5](#_Toc166961974)

[**2. Methodology** 5](#_Toc166961975)

[IV. METRICS (KHIEM) 7](#_Toc166961976)

[1 Compare 7](#_Toc166961977)

[2 Recommendation 7](#_Toc166961978)

[V. Exited model (SON) 9](#_Toc166961979)

# INTRODUCTION (TUYET ANH)

## Import needed libraries & dataset.

It is necessary to import the needed libraries for

* Manipulation data:

import numpy as np

import pandas as pd

* Visualization:

import matplotlib.pyplot as plt

import seaborn as sns

* Applying models:

from sklearn.model\_selection import KFold, GridSearchCV,  StratifiedKFold, RepeatedStratifiedKFold, cross\_val\_score, train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.neural\_network import MLPClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

* Evaluating performance

from sklearn.metrics import accuracy\_score,f1\_score, recall\_score, classification\_report

from sklearn.metrics import roc\_auc\_score, confusion\_matrix

import time

import warnings

warnings.filterwarnings('ignore')

pd.options.display.float\_format = '{:.2f}'.format

The dataset can be imported to analyze by the command code with name “df” and path\_line(“heart.csv”):

df = pd.read\_csv("heart.csv")

## Description

Using “head” to see top five content of dataset:

A screenshot of a computer

Description automatically generateddf.head()

Dataset is divided into input data (used to predict and independent) and target output (will be predicted and dependent on input data).

* To predict the patient whether heart disease or normal which based on some information below:
* Age: age of patient [year]
* Sex: sex of the patient [M: Male, F: Female]
* Chest Pain Type: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
* RestingBP: resting blood pressure [mm Hg]
* Cholesterol: serum cholesterol [mm/dl]
* Fasting BS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
* Resting ECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
* Max HR: maximum heart rate achieved [Numeric value between 60 and 202]
* Exercise Angina: exercise-induced angina [Y: Yes, N: No]
* Oldpeak: oldpeak = ST [Numeric value measured in depression]
* ST\_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
* Target output: Heart disease [ 1: heart disease, 0: normal]
* In conclusion, it seems to be a task as a classification problem in supervised learning based on two main criteria. Firstly, the target output is categorical - 2 classes (heart disease / normal). Then, accessing to a labeled dataset for training.
* When the task is satisfied, supervised learning algorithms are recommended as well as Logistic Regression, Random Forests, SVM, KNN, Naive Bayes, Neural Networks.
* The shape of dataset:

print("The shape of dataset is: ",df.shape)

* Dataset records 12 information of 918 patient.
* Using “df.info()” to quickly get a summary of dataset

A screen shot of a computer

Description automatically generated

* This summary is useful for showing that there are no missing values in any columns; therefore, it is not necessary to preprocess for missing values. Additionally, there are three types of data: float, int, and categorical. This helps determine that the dataset needs to be scaled and encoded in the preprocessing step.
* It is easily to get a comprehensive statistical summary of dataset by using describe method.
* Numerical columns:

A screenshot of a graph

Description automatically generated

* The ranges of the numerical variables differ significantly, such as between FastingBS and Cholesterol. It seems to all numerical input variables need to be scaled which help to reduce biased results and improve the performance of machine learning models.
* A screenshot of a computer

  Description automatically generatedObject columns:
* Since these are not ordinal variables and do not have a ranking, one-hot encoding is a better choice than label encoding.

## Visualization

* First, it is necessary to visualizing the distribution of target output through pie chart by using command line:

# Overall about output of database

partition = *list*(df['HeartDisease'].value\_counts())

# returns a list (easier to work) containing frequency table of target output.

plt.pie( # Creating the pie chart

        partition, # Adding the value of pie chart

*labels* = ['Nomarl','Heart Disease'], # Adding the label

*autopct*='%1.1f%%', # Specifies the format for displaying the percentage values on the pie chart.

*startangle* = 90, #  It starts at 90 degrees (which is at the top).

*explode* = (0.1,0), # This parameter determines the fraction of the radius with which to offset each slice. Here, the first slice ('Normal') is exploded (offset) by 0.1 times the radius.

*shadow*= True, # Adding shadow effect

       )

plt.title('Heart Disease %'); # set title

A blue and orange pie chart

Description automatically generated

* Observing the total dataset, we note that 44.7% have heart disease, which is a substantial proportion. With many individuals affected by heart disease, there is a necessity to utilize machine learning to predict whether someone is at risk of developing the condition to intervene in a timely manner.
* Impact of Categorical Features on the output
* Impact of Numerical Features on the output

# **Preprocessing (TUYET ANH)**

## Processing missing value

A screenshot of a computer

Description automatically generated

* The check for missing values with the command above reveals that there are no missing values in any column of the data, so the step can be skipped.

## Encoding

Since dealing with nominal variables, it is logical to use one-hot-encoding by command conde:

A screenshot of a computer

Description automatically generated

This function from the pandas library is used to convert categorical variables into dummy/indicator variables. “drop\_first=True” means that it drops the first level of each categorical variable to avoid multicollinearity. For example, if there are two categories in a variable, only one dummy variable is created to represent them.

## Normalization

# APPLYING MODEL (KHIEM):

## **1. Aim**

The objective of this section is to train models for classifying heart failure patients into two categories: non-heart failure (0) and positive heart failure (1). Various machine learning models are employed to achieve this classification and conducted a thorough evaluation to determine the best model based on accuracy and other performance metrics.

## **2. Methodology**

* 1. Grid Search for Model Selection: A grid search is performed to identify the most suitable machine learning models based on their accuracy scores on the training set.

We utilized GridSearchCV to perform an exhaustive search over specified parameter values for different machine learning algorithms along with kfold train-test split. The following models and their respective parameter grids were considered: KNeighborsClassifier, GaussianNB, SVC, MLPClassifier, LogisticRegression, RandomForestClassifier.

After performing GridSearch using sklearn, the output yeilds:

* KNeighborsClassifier: 87.02%, {'leaf\_size': 1, 'metric': 'euclidean', 'n\_neighbors': 5, 'weights': 'uniform'}
* GaussianNB: 86.16%, {'var\_smoothing': 0.12328467394420659}
* SVC: 87.71%, {'C': 1, 'gamma': 0.1, 'kernel': 'poly'}
* MLPClassifier: 87.02%, {'activation': 'tanh', 'alpha': 0.05, 'early\_stopping': False, 'hidden\_layer\_sizes': (50, 100, 50), 'learning\_rate': 'constant', 'solver': 'sgd'}
* LogisticRegression: 85.64%, {'C': 5, 'penalty': 'l2', 'solver': 'lbfgs'}
* RandomForestClassifier: 87.54%, {'criterion': 'gini', 'max\_depth': 5, 'min\_samples\_leaf': 2, 'min\_samples\_split': 5, 'n\_estimators': 50}
  1. Model Implementation and Evaluation: Selected models are initialized, trained them on the dataset.

Each model was initialized with the optimal parameters determined from the grid search. The training process involved fitting the models on the training dataset, ensuring that they learned the underlying patterns and relationships between the features and the target variable (heart failure classification). The models were then prepared for subsequent evaluation to assess their performance on unseen test data.

The implementation process for each model is outlined below:

* KNeighborsClassifier: Initialized with the best parameters including leaf\_size, metric, n\_neighbors, and weights.
* GaussianNB: Configured with the optimal var\_smoothing parameter.
* SVC: Set up with the best values for C, gamma, and kernel.
* MLPClassifier: Defined with the optimal activation, alpha, early\_stopping, hidden\_layer\_sizes, learning\_rate, and solver.
* LogisticRegression: Implemented with the best C, penalty, and solver.
* RandomForestClassifier: Initialized with the best criterion, max\_depth, min\_samples\_leaf, min\_samples\_split, and n\_estimators.

# METRICS (KHIEM)

## Compare

A screen shot of a computer

Description automatically generated

Overall, all models performed well, each demonstrating strengths in various evaluation metrics. The RandomForestClassifier achieved the highest performance across several key metrics, with an accuracy of 91%, a recall of 94%, an F1 score of 92%, and a ROC AUC score of 90%. These metrics indicate that the RandomForestClassifier not only correctly identified the majority of positive cases but also maintained a high balance between precision and recall. Additionally, it showed robust generalization capability as evidenced by its cross-validation score of 92%.

The GaussianNB model also demonstrated strong performance, particularly in terms of training efficiency. It achieved an accuracy of 89%, a recall of 91%, an F1 score of 91%, and a ROC AUC score of 88%. Despite these metrics being slightly lower than those of the RandomForestClassifier, GaussianNB had the shortest training time at just 0.09 seconds. This makes it an attractive option for scenarios where computational resources or time are limited.

The remaining models, including KNeighborsClassifier, SVC, MLPClassifier, and LogisticRegression, also performed well, with accuracies ranging from 86% to 88% and consistent cross-validation scores around 92%. Each model has its unique advantages, such as the simplicity and interpretability of LogisticRegression or the flexibility of the MLPClassifier with its ability to capture complex patterns through neural networks.

## Recommendation

Based on the evaluation metrics and training time, the RandomForestClassifier is recommended for heart failure classification. It achieved the highest accuracy, recall, F1 score, and ROC AUC score. Although it takes longer to train than some other models, its performance makes it suitable for this classification task.

For scenarios where training time is critical, GaussianNB can be considered as it offers a good balance of performance and efficiency, with the second-highest accuracy and the shortest training time.

# Exited model (SON)