Predict Heart Failure using IBM AUTO AI Service

INTRODUCTION

Overview

Heart failure is a major cause of death among the people across the world. Eventhough there are several diagnosis methods for heart diseases, accurate prediction is lacking. This is because the reasons for heart failure vary from person to person. There are mainly two kinds of risk factors for heart diseases. Alterable risk factors, e.g., habit of smoking and time spend for excercise, and non-alterable risk factors, e.g., gender, age and family history. It is difficult to predict the chance for heart failure considering all these factors at the same time. Angiography and Electrocardiogram are the best available methods now. But these kinds of medical examinations as well as other tools are not ideal for the purpose of accurate prediction of heart failure. Moreover these methods are expensive, time consuming and demand techincal expertise. Therefore, for overcoming these difficulties, Machine learning techniques are very helpful.

In this work, an automatic predictor system for predicting heart failure is built using IBM auto AI service. It build a classification model automatically based on inputs and classifies the model using the random forest classifier with an accurate result. The model is deployed on a HTML page in which the users can enter the values of the predictors used and the results are visualized in the output.

Purpose

The purpose of this study is to build a machine learning model for predicting heart failure and build a web application where we can showcase the prediction of heart failure. Through this system, a user will be able to know the predicted result of Heart failure from the user given inputs.

Literature Survey

Existing problem

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Most heart diseases are highly preventable and simple lifestyle modifications(such as reducing tobacco use, eating healthily, obesity and exercising) coupled with early treatment greately improve their prognoses. Detecting the presence or absence of Heart Failure diseases earlier using associated input variables can reduce severe episodes.

Proposed solution

Machine learning (ML), due to its superiority in pattern detection and classification, proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry on heart disease. ThIS project aims to predict the HEART FAILURE of a person at the earliest using different variables by building a Machine Learning model,. The proposed solution uses machine learning and artificial intelligence based automatic prediction model using IBM auto AI service. It uses Random Forest Algorithm.

Theoretical Analysis

Block diagram

Flow diagram for Prediction of Heart Failure using Watson Studio, AutoAI and Flask App

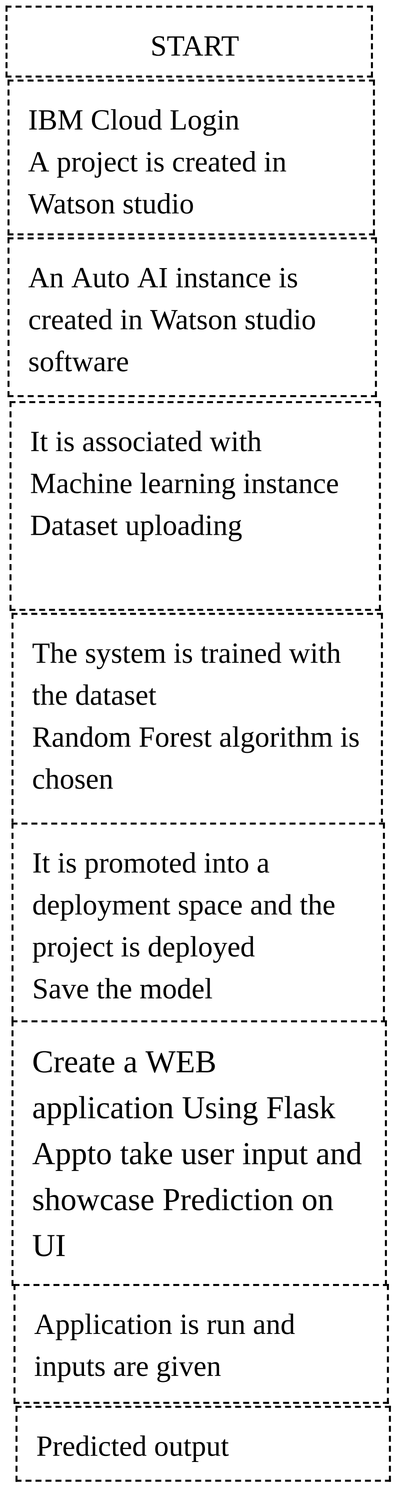
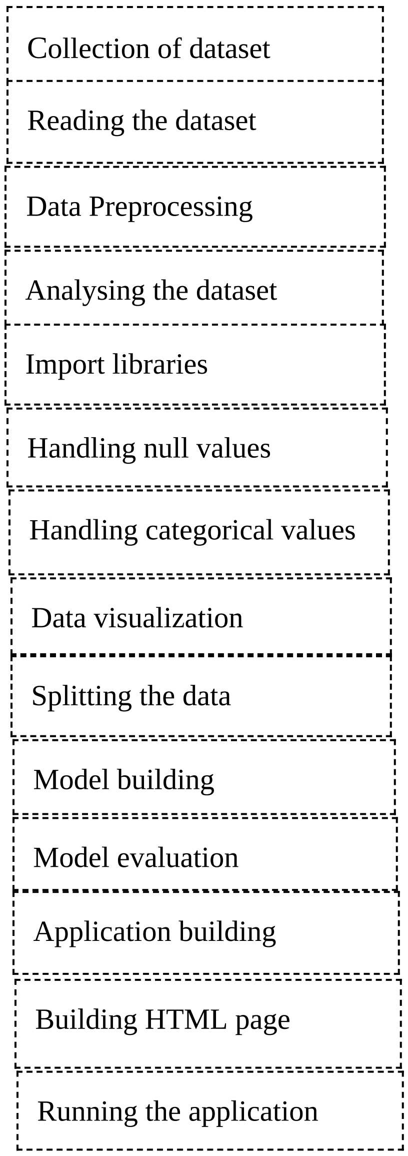

START
IBM Cloud Login
A project is created in Watson studio
An Auto AI instance is created in Watson studio software
It is associated with Machine learning instance
Dataset uploading
The system is trained with the dataset
Random Forest algorithm is chosen
It is promoted into a deployment space and the project is deployed
Save the model
Create a WEB application Using Flask Appto take user input and showcase Prediction on UI
Application is run and inputs are given
Predicted output

Fig: Building of ML model



Hardware / Software designing

Hardware used:

HP Notebook- 6GB RAM- 64bit- Windows10

Softwares used:

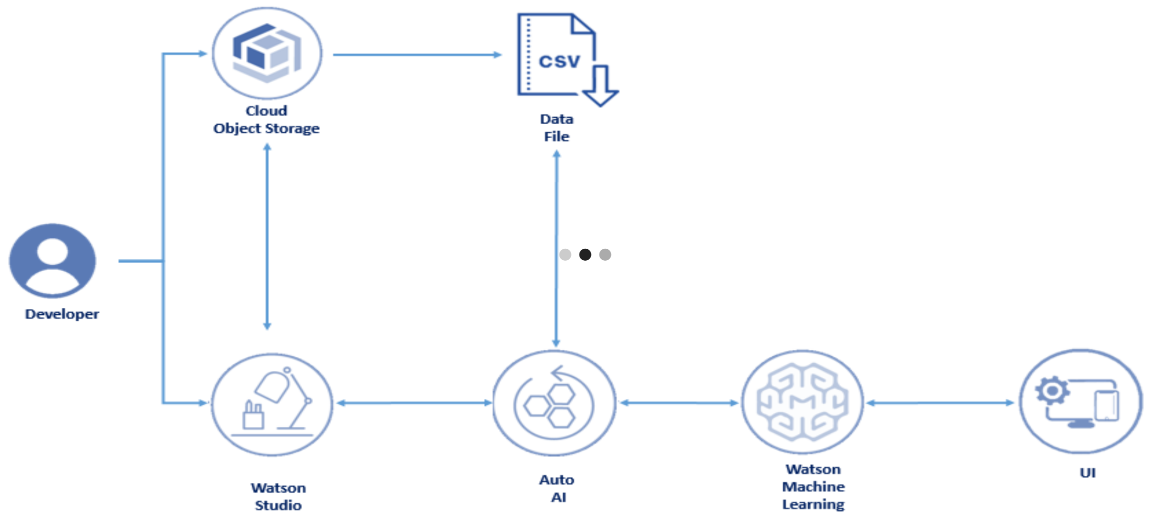
Python Notebook(anaconda 3), Anaconda Spyder, Microsoft excel

Experimental Investigations

CRISP-DM CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. This project follows the sequence of events followed in this methodology which are as follows:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modelling
5. Evaluation
6. Deployment

The figure below shows a block diagram of the events that were followed in the project.



Business Understanding/Proposed solution

Here the business problem is to predict the Heart Failure of a person by building a Machine Learning model, using Logistics Regression, Random Forest and Decision Tree algorithms. As per the results shown by the Model, the person can take actions.

Data understanding

The dataset used is a repository from IBM used for the purpose of heart failure prediction. It has 9 columns with 10801 rows with various features causing heart failure like smoking history, age, palpitation count, gender, walking habit etc.,

Independent variables

Avg_heartbeats_per_min, Palpitations_per_day, Cholestrol, BMI, Age, Sex, Family_history, Smoker_last_5years, Exercise_mins_per_week

Dependent variable

Heart_failure

Data Preparation

During this stage the following steps were taken to prepare the data for the building the model:

1. First checked if there where any missing values and then those where imputed
2. Renamed the columns and found correlation between the variables
3. Encoding the string categorical target variables using LabelEncoder
4. The data is then split into training and testing data
5. The variables are split into y train and y test dataset

Modelling

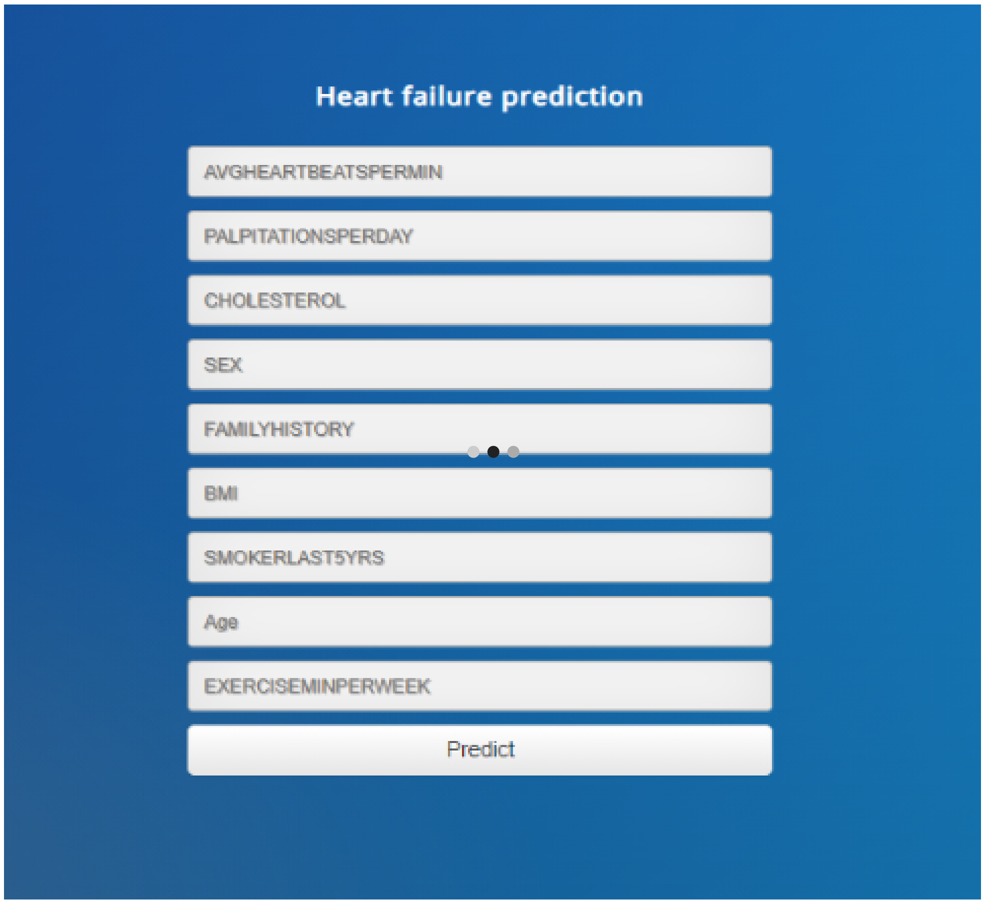
After the data preparation has been completed, the data has been tested on the two models mentioned earlier, Logistic Regression, Random forest and Decision Tree.

Evaluation

For each of the models created the classification report and the accuracy score is used to determine the best possible model that can be used for deployment.

Deployment

The model is deployed on a HTML page in which the users can enter the values of the predictors used and predict the chance of Heat failure of a person. To create the deployment, the UI and the predications made after the variables are entered in the UI are shown below.

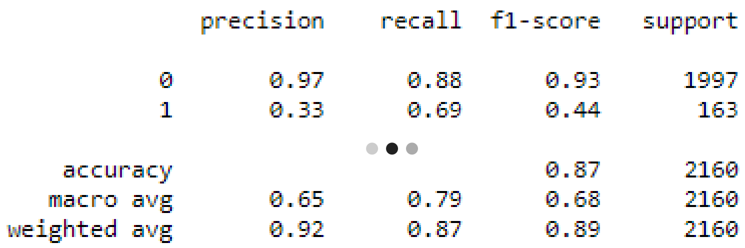


Model Interpretation

The tables below show the classification report and the confusion matrix of the test dataset. The values

that will be used to determine the fit of the model are precision, recall, f1 score, support and accuracy.
Precision is defined as the percentage of the total predicted results which are relevant/ have been correctly classified, i.e True Positive/ (True Positive + False Positive).
Recall is defined as the percentage of actual results that have been correctly classified by the model, i.e True Positive/ (True Positive + False Negative).
F-1 score is a harmonic mean of precision and recall which is used to identify the weighted mean of the classification model, with 1.0 being the best fit and 0.0 being the worst fit. This value is often used as it provides a balance of both the precision and the recall.
Support is defined as the number of actual occurrences in the dataset. This doesn't change between models but changes with the changes in test data.

Logistic Regression

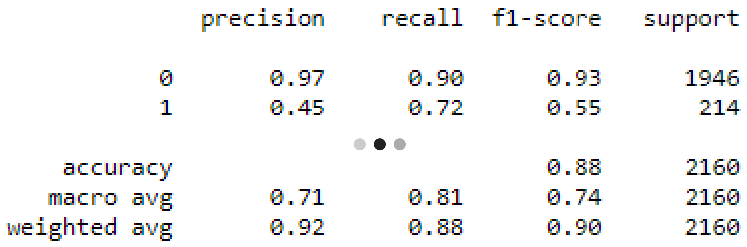


```
              precision    recall  f1-score   support

           0       0.97      0.88      0.93      1997
           1       0.33      0.69      0.44       163

    accuracy                           0.87      2160
   macro avg       0.65      0.79      0.68      2160
weighted avg       0.92      0.87      0.89      2160
```

Random Forest Classifier

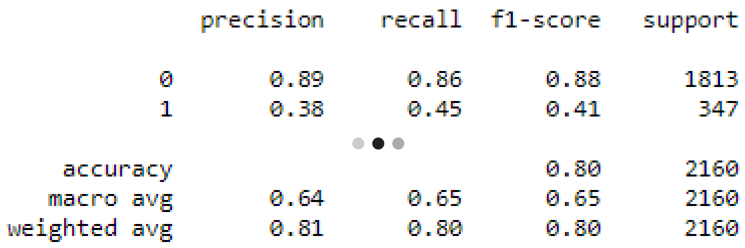


```
              precision    recall  f1-score   support

           0       0.97      0.90      0.93      1946
           1       0.45      0.72      0.55       214

    accuracy                           0.88      2160
   macro avg       0.71      0.81      0.74      2160
weighted avg       0.92      0.88      0.90      2160
```

Decision Tree Classifier



```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88      1813
           1       0.38      0.45      0.41       347

    accuracy                           0.80      2160
   macro avg       0.64      0.65      0.65      2160
weighted avg       0.81      0.80      0.80      2160
```

As we can see, the Random Forest Classifier is giving an overall better result with the highest overall accuracy of 0.88 while the Logistic Regression gives an overall accuracy of 0.87. Hence the Random Forest Classifier has been chosen as the model for the prediction of Heart failure.

Flowchart

START

↓

ENTER AVERAGE HEART BEATS PER MINUTE

↓

ENTER PALPITATIONS PER DAY

↓

ENTER CHOLESTROL LEVEL

↓

ENTER SEX

↓

ENTER FAMILY HISTORY

↓

ENTER BMI

↓

SMOKELAST5YEARS

↓

AGE

↓

EXERCISEMINPERWEEK

↓

DISPLAY OF PREDICTED RESULT

Result

The Random Forest Classifier has been chosen as the model for the prediction of Heart failure and the accuracy has been determined as 0.88. ML models using Logistic regression, Random Forest and Decision Tree algorithm were made and Random Forest is found to be the most accurate one. Also, data visualization was made with the dataset.

Advantages & Disadvantages

Advantages

The main advantage of the proposed approach for the researcher is the possibility of being quickly operative, focusing only on the experiment, without taking care of hardware requirements (high computational power is needed for these analyses) or machine learning algorithms development.

From a medical point of view, the results of this study can be interpreted as the possibility to perform a preliminary and early diagnosis of HF, based solely on the analysis of alterable and unalterable risk factors.

Disadvantages

These findings are not meant to replace the diagnostic procedures for an exhaustive diagnosis (accepting a certain level of uncertainty, as shown by the accuracy values) but can be very helpful in many scenarios such as home telemonitoring for the daily monitoring of patient status. The model also needs inputs such as HEARTBEATs PENMIN which has to be measured using another external device.

Applications

The model exactly predicted the class Y/N representing the possibility of Heart Failure/No Heart Failure. Here, without any coding, automatically built IBM services are utilised and a Binary classification AutoAI model is executed successfully. The proposed model can also be used to create an awareness for heart failures. So instead of detecting failures, an application can be created to be a preventive measure for failures.

Conclusion

The prediction of heart failure is one of the mostly needed requirement in the medical field. Here, without any coding, automatically built IBM services are utilised and a Binary classification AutoAI model is executed successfully. The best performance measure which the model resulted is accuracy with the Random Forest Algorithm with a best result of 0.88. The build time 00:00:23 seconds, which is saved as model and deployed. The model is further tested and integrated with Flask App for interfacing and creating a web service. It is seen that the model exactly predicted the class Y/N representing the possibility of Heart Failure/No Heart Failure. Thus, the IBM Watson studio, AutoAI, Flask App and Cloud services are utilised and a model is automatically deployed successfully without any implementation of coding.

After running and comparing two ML model i.e. Decision Tree and Random Forest, we observe that the accuracy of Random Forest (88.%) is better than that of Decision Tree (80%) and hence we finalize our model with the Random Forest Algorithm to make our predictions. However even with the high accuracy, the decision tree does make errors during classification which can be extremely harmful, especially in cases where it was predicated that there is no heart failure for a person who has it.

Future Scope

Several other services in IBM can also be utilised where a real time mobile application can be developed. The Chat bots, Natural Language disambiguations, Sensor based cloud applications can also be developed which is automatic and fast.

Bibliography

1). G, K. P., T, S. S., G, V., & Madian, N. (2021). An approach for predicting heart failure rate using IBM auto AI service. *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. https://doi.org/10.1109/iccike51210.2021.9410783

2). Guidi, G., Miniati, R., Mazzola, M., & Iadanza, E. (2016). Case study: IBM Watson analytics cloud platform as analytics-as-a-Service system for heart failure early detection. Future Internet, 8(3), 32. https://doi.org/10.3390/fi8030032

3). https://github.com/IBM/predictive-model-on-watson-ml/blob/master/data/patientdataV6.csv

4). *Using AI and science to predict heart failure*. (2019, February 8). IBM Research Blog. https://www.ibm.com/blogs/research/2017/04/using-ai-to-predict-heart-failure/

5). *Create and deploy a scoring model to predict heartrate failure*. (n.d.). IBM Developer. https://developer.ibm.com/technologies/analytics/patterns/create-and-deploy-a-scoring-model-to-predict-heartrate-failure/

6). Almustafa, K. M. (2020). Prediction of heart disease and classifiers' sensitivity analysis. *BMC Bioinformatics*, *21*(1). https://doi.org/10.1186/s12859-020-03626-y

Appendix

Flask code

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import requests

import json
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "RyRxnYkTnQcStWr68PJjuDfpl0vp2u8f5VQaBqZc5ifA"
token_response = requests.post('https://iam.eu-gb.bluemix.net/identity/token', data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
```

```
print("mltoken",mltoken)

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
#payload_scoring = {"input_data": [{"fields": [array_of_input_fields], "values": [array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}



app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():

    AVGHEARTBEATSPERMIN = request.form["AVGHEARTBEATSPERMIN"]
    PALPITATIONSPERDAY = request.form["PALPITATIONSPERDAY"]
    CHOLESTEROL = request.form["CHOLESTEROL"]
    BMI = request.form["BMI"]
    SEX = request.form["SEX"]

    FAMILYHISTORY = request.form["FAMILYHISTORY"]
    SMOKERLAST5YRS = request.form["SMOKERLAST5YRS"]
    Age = request.form["Age"]
    EXERCISEMINPERWEEK = request.form["EXERCISEMINPERWEEK"]



                                                          t =
[[int(AVGHEARTBEATSPERMIN),int(PALPITATIONSPERDAY),int(CHOLESTEROL),int(BMI),str(SEX),str(FAMILYHISTORY),str(SMOKERLAST5YRS),int(Age),int(EXERCISEMINPERWEEK)]]
    print(t)
                                payload_scoring = {"input_data": [ {"field":
[["AVGHEARTBEATSPERMIN","PALPITATIONSPERDAY","CHOLESTEROL","BMI","SEX","FAMILYHISTORY","SMOKERLAST5YRS","Age","EXERCISEMINPERWEEK"]],
                   "values": t}]}
response_scoring=
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/e5e8ad95-8348-441a-8acd-374bb5a33c78/predictions?version=2021-06-27', json=payload_scoring, headers={'Authorization': 'Bearer ' +
mltoken})
    print("Scoring response")
    predictions = response_scoring.json()
    print(predictions)
    pred = predictions['predictions'][0]['values'][0][0]

    return render_template('index.html', prediction_text= pred)



if __name__ == "__main__":
    app.run(debug=False)
```

Data Pre-processing and Model Building codes

```
#!/usr/bi
 # # Prediction of Heart Failure using Machine Learning Algorithms and IBM Auto AI Service

 # # Data Preprocessing
 import pandas as pd #used for data manipulation
 import numpy as np #used for numerical analysis
 from collections import Counter as c #return counts
 import seaborn as sns # used for data visualization
 import matplotlib.pyplot as plt
 from sklearn.model_selection import train_test_split #splits data in random train and test array
 from sklearn.metrics import accuracy_score,mean_squared_error,mean_absolute_error#model performance
 import pickle #python object hierarchy is converted into a byte stream
 from sklearn.linear_model import LinearRegression #Regression ML algorithm

data=pd.read_csv("C:\\Users\\joyal\\OneDrive\\Desktop\\Smart_internz\\patientdataV6.csv")#loading the csv
data

 #Data Analysis
data.columns#returns all the column names
data.columns=['Avg_heartbeats_per_min','Palpitations_per_day','Cholestrol','BMI','Heart_failure','Age','Sex','F
amily_history','Smoker_last_5years','Exercise_mins_per_week']#manually giving the names of the columns
data.columns#returns all the column names
data.head()#returns the first 5 row values
Understanding the Data Type and Summary of features
data.info() #to get the summary of dataset

data.describe()#returns important values for continuous column data
# Observing Target, Numerical and Categorical columns
```

```
np.unique(data.dtypes,return_counts=True)#data.dtypes will return us all the different types of data present in
our data and return_countswill givw us the count
cat=data.dtypes[data.dtypes=='O'].index.values#for fetching all the object or categorical type of columns from
our data and we are sorting it as set in variable catcols
cat

 for i in cat:
    print("Column:",i)
    print('count of classes : ',data[i].nunique())
    print(c(data[i]))
    print('*'*120) #We are looping with each categorical column and printing the classes of each categorical
column using the counter function so that we can detect which columns are categorical and which are not

data.dtypes[data.dtypes!='O'].index.values#finding the numerical columns

#Handling null values
data.isnull().sum()#used for finding null values
#Handling categorical data
#Labeling the categorical columns
from sklearn.preprocessing import LabelEncoder#Importig the LabelEncoding from sklearn
 x='*'
 for i in cat:
    print("LABEL ENCODING OF: ",i)
    LE = LabelEncoder()
    print(c(data[i]))
    data[i]=LE.fit_transform(data[i])
    print(c(data[i]))
    print(x*100)

 data.head()
 # # Data Visualization

 #grouping the SEX and HEARTFAILURE columns

 data['Heart_failure'].value_counts()
 # In[49]:
 sns.countplot(data['Heart_failure'])
 # In[50]:
sns.countplot(x='Age', hue='Heart_failure', data=data, palette='colorblind',
edgecolor=sns.color_palette('dark',n_colors=1))
 # In[29]:
HF1=data.groupby(['BMI'])['Heart_failure'].mean().sort_values()[:15].reset_index()
 plt.figure(figsize=(25,6))
 sns.barplot(x = "BMI",y="Heart_failure",data=HF1)


HF2=data.groupby(['Palpitations_per_day'])['Heart_failure'].mean().sort_values()[:15].reset_index()
 plt.figure(figsize=(25,6))
 sns.barplot(x = "Palpitations_per_day",y="Heart_failure",data=HF2)

HF3=data.groupby(['Age'])['Heart_failure'].mean().sort_values()[:20].reset_index()
 plt.figure(figsize=(25,6))
 sns.barplot(x = "Age",y="Heart_failure",data=HF3)

HF4=data.groupby(['Exercise_mins_per_week'])['Heart_failure'].mean().sort_values()[:30].reset_index()
 plt.figure(figsize=(25,6))
 sns.barplot(x = "Exercise_mins_per_week",y="Heart_failure",data=HF4)

HF5=data.groupby(['Avg_heartbeats_per_min'])['Heart_failure'].mean().sort_values()[:30].reset_index()
 plt.figure(figsize=(25,6))
 sns.barplot(x = "Avg_heartbeats_per_min",y="Heart_failure",data=HF5)

 #get correlation of each features in a dataset
 corr = data.corr()#perform correlation between all continuous features
 plt.subplots(figsize=(20,20));
 sns.heatmap(corr, annot=True, square=True)#plotting heatmap of correlations
 plt.title("Correlation matrix of numerical features")
 plt.tight_layout()
 plt.show()
```

Feature selection

```
 corr = data.corr()#perform correlation between all continuous features
 top_corr_features = corr.index
 plt.figure(figsize=(20,20))
 #plot heat map
 g=sns.heatmap(corr, annot=True,cmap="RdYlGn")

 #Correlation with target column
 plt.figure(figsize=(16,5))
 corr["Heart_failure"].sort_values(ascending=True)[:-1].plot(kind="barh")

 # Creating the dependent and independent variable
 x = data.drop(['Heart_failure'],axis=1)#independent feature
 x=pd.DataFrame(x)
 y = data['Heart_failure']#dependent feature
```

```
y=pd.DataFrame(y)
y

#Splitting the dataset into train and test
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)


# # Building the ML model

# 1). Logistic Regression

from sklearn.linear_model import LogisticRegression
logr=LogisticRegression()
logr.fit(x_train,y_train)

y_test
y_pred=logr.predict(x_test)
y_pred[:30]

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
#accuracy_score(y_pred,y_test)
confusion_matrix(y_pred,y_test)

# accuracy score we will use only if the classes are balanced in nature

print(classification_report(y_pred,y_test))

# Use score method to get accuracy of model
score = logr.score(x_test, y_test)
print(score)

from sklearn import metrics #importing the metrics library
print("MAE:",metrics.mean_absolute_error(y_test,y_pred))#Mean Absolute Error
print("MSE:",metrics.mean_squared_error(y_test,y_pred))#Mean Squared Error
print("RMSE:",np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))#Root Mean Square Error


# 2). Random Forest Classifier
```

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_test

y_pred=rfc.predict(x_test)
y_pred[:30]

from sklearn.metrics import accuracy_score, confusion_matrix
accuracy_score(y_test,y_pred)

confusion_matrix(y_test,y_pred)

print(classification_report(y_pred,y_test))
# Use score method to get accuracy of model
score = rfc.score(x_test, y_test)
print(score)

from sklearn import metrics #importing the metrics library
print("MAE:",metrics.mean_absolute_error(y_test,y_pred))#Mean Absolute Error
print("MSE:",metrics.mean_squared_error(y_test,y_pred))#Mean Squared Error
print("RMSE:",np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))#Root Mean Square Error

# 3). Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)

y_test
y_pred=dtc.predict(x_test)
y_pred[:30]

from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
accuracy_score(y_test,y_pred)

print(classification_report(y_test,y_pred))

confusion_matrix(y_test,y_pred)

from sklearn import metrics #importing the metrics library
print("MAE:",metrics.mean_absolute_error(y_test,y_pred))#Mean Absolute Error
print("MSE:",metrics.mean_squared_error(y_test,y_pred))#Mean Squared Error
print("RMSE:",np.sqrt(metrics.mean_absolute_error(y_test,y_pred)))#Root Mean Square Error
```