**Codes Used for Implementing Dashboard**

External CSS Files and JavaScript files are linked from github.

**Python Codes.**

#  app.py

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import dash\_bootstrap\_components as dbc

import plotly.graph\_objects as go

import plotly.express as px

import pandas as pd

from dash.dependencies import Input, Output, State

# java script

external\_scripts = ["https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/app.js", "https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/bootstrap.min.js", "https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/jquery-3.2.1.slim.min.js", "https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/popper.min.js"

                    ]

# external CSS stylesheets

external\_stylesheets = ["https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/app.css", "https://gitcdn.link/repo/DksRukshan/ST3011-ASSIGNMENT/master/assets/bootstrap.min.css"

                        ]

app = dash.Dash(

    \_\_name\_\_, external\_scripts=external\_scripts,

    external\_stylesheets=external\_stylesheets)

server = app.server

# dataset heart\_failure\_records

df\_HF = pd.read\_csv("heart\_failure\_clinical\_records\_dataset.csv")

# Create Count field in df\_HF

df\_HF["Count"] = [1]\*len(df\_HF)

# turn anamemia in to a series of catagorical data.

S\_HF = pd.Series(df\_HF["anaemia"].astype('category'))

# Change the catagories into negative and positive.

S\_HF = S\_HF.cat.rename\_categories(["Negative", "Positive"])

# insert it into df\_HF.

df\_HF["anaemia"] = S\_HF

# turn sex in to a series of catagorical data.

S\_HF = pd.Series(df\_HF["sex"].astype('category'))

# Change the catagories into negative and positive.

S\_HF = S\_HF.cat.rename\_categories(["Female", "Male"])

# insert it into df\_HF.

df\_HF["sex"] = S\_HF

# turn diabetes in to a series of catagorical data.

S\_HF = pd.Series(df\_HF["diabetes"].astype('category'))

# Change the catagories into negative and positive

S\_HF = S\_HF.cat.rename\_categories(["Negative", "Positive"])

# Change the order of the categories.

S\_HF = S\_HF.cat.reorder\_categories(["Positive", "Negative"])

# insert it into df\_HF

df\_HF["diabetes"] = S\_HF

# turn death in to a series of catagorical data to make the bar chart.

S\_HF = pd.Series(df\_HF["DEATH\_EVENT"].astype('category'))

# Change the catagories into negative and positive

S\_HF = S\_HF.cat.rename\_categories(["Alive", "Dead"])

# insert it into df\_HF

df\_HF["DEATH\_EVENT"] = S\_HF

# turn diabetes in to a series of catagorical data.

S\_HF = pd.Series(df\_HF["smoking"].astype('category'))

# Change the catagories into negative and positive

S\_HF = S\_HF.cat.rename\_categories(["Non-smoking", "Smoking"])

# Change the order of the categories.

S\_HF = S\_HF.cat.reorder\_categories(["Smoking", "Non-smoking"])

# insert it into df\_HF

df\_HF["smoking"] = S\_HF

# turn age into a integer value

S\_HF = pd.Series(df\_HF["age"].astype('int64'))

df\_HF["age"] = S\_HF

# remame columns

df\_HF = df\_HF.rename(columns={"age": "Age", "anaemia": "Anaemia", "diabetes": "Diabetes", "platelets": "Platelets", "serum\_creatinine": "Serum Creatinine",

                              "serum\_sodium": "Serum Sodium", "sex": "Sex", "smoking": "Smoking", "DEATH\_EVENT": "Death Event", "Count": "Count"})

# Navigation Bar for title.

navbar = html.Div(children=[html.Nav(className="nav justify-content-center bg-dark text-light", children=[

    html.H2(children=["Welcome to the dashboard"])

]), html.Nav(className="nav justify-content-center bg-dark text-light", children=[

    html.H3(children=["Heart Failure Clinical Records"])

])

])

# Navigation Bar for drop down menue.

Navigation\_bar = dbc.Navbar(

    [

        dbc.Row(

            [

                dbc.Col(dbc.NavbarBrand(

                    "Variable:", className="ml-2")),

            ],

            align="center",

            no\_gutters=True,

        ),

        dbc.NavbarToggler(

            id="navbar-toggler"),

        dbc.Collapse(

            dbc.InputGroup(

                [

                    dbc.Select(

                        options=[

                            {'label': 'Sex',

                             'value': 'Sex'},

                            {'label': 'Diabetes',

                             'value': 'Diabetes'},

                            {'label': 'Age',

                             'value': 'Age'},

                            {'label': 'Anaemia',

                             'value': 'Anaemia'},

                            {'label': 'Platelets',

                             'value': 'Platelets'},

                            {'label': 'Serum Creatinine',

                             'value': 'Serum Creatinine'},

                            {'label': 'Serum Sodium',

                             'value': 'Serum Sodium'},

                            {'label': 'Smoking',

                             'value': 'Smoking'},

                            {'label': 'Death Event (Response Variable)',

                             'value': 'Death Event'}

                        ],

                        value='Age', id='select',

                    ),

                ]

            ), id="navbar-collapse", navbar=True, className="col-4"),

    ],

    color="dark",

    dark=True,

)

# App layout for the web page

app.layout = html.Div(className="container-fluid",

                      children=[navbar, Navigation\_bar, html.Br(), html.Div(className="row",

                                                                            children=[html.Div(className="col-sm-6",

                                                                                               children=[html.Div(className="card text-white bg-dark mb-3", children=[html.Div(className="card-body", children=[html.H3(className="card-title text-center", children=["Univariate Analysis"]), dcc.Graph(

                                                                                                   id='example-graph1',

                                                                                               )])])]),

                                                                                      html.Div(className="col-sm-6", id="hide",

                                                                                               children=[html.Div(className="card text-white bg-dark mb-3", children=[html.H3(className="card-title text-center", children=["Bivariate Analysis"]), html.Div(className="card-body", children=[dcc.Graph(

                                                                                                   id='example-graph2',

                                                                                               )])])])

                                                                                      ]

                                                                            )

                                ]

                      )

# For the drop down menu

@ app.callback(

    Output('example-graph1', 'figure'),

    Output('example-graph2', 'figure'),

    Input('select', 'value'))

def update\_figure(Val):

    graph\_colors = ['#1fe074', '#ff7f50']

    uni = px.bar()

    bi = px.bar()

    if Val == "Death Event":

        uni = px.pie(data\_frame=df\_HF.groupby(

            [Val]).count().reset\_index(), names=Val, values="Count", color\_discrete\_sequence=graph\_colors, title=f"Pie chart of {Val}")

    elif type(df\_HF[Val][0]) == type("str"):

        uni, bi = px.pie(data\_frame=df\_HF.groupby(

            [Val]).count().reset\_index(), names=Val, values="Count", color\_discrete\_sequence=graph\_colors, title=f"Pie chart of {Val}"), px.bar(df\_HF.groupby([Val, "Death Event"]).count(

            ).reset\_index(), x=Val, y="Count", color="Death Event", color\_discrete\_sequence=graph\_colors, title=f"Stacked Bar chart of {Val} vs Death Event")

    else:

        uni, bi = px.histogram(

            df\_HF, x=Val, color\_discrete\_sequence=graph\_colors, title=f"Histogram of {Val}"), px.box(

            df\_HF, color="Death Event", x="Death Event", y=Val, color\_discrete\_sequence=['#ff7f50', '#1fe074'], title=f"Box plot of {Val} vs Death Event")

    uni.update\_layout(

        font\_color="rgb(255, 255, 255)", paper\_bgcolor='rgb(43, 63, 82)', plot\_bgcolor='rgb(43, 63, 82)', legend\_title\_text="", legend=dict(x=1, y=1, font=dict(size=17), orientation='h'))

    bi.update\_layout(

        font\_color="rgb(255, 255, 255)", paper\_bgcolor='rgb(43, 63, 82)', plot\_bgcolor='rgb(43, 63, 82)', legend\_title\_text="", legend=dict(x=1, y=1, font=dict(size=17), orientation='h'))

    return uni, bi

# For the Toggler

@ app.callback(

    Output("navbar-collapse", "is\_open"),

    [Input("navbar-toggler", "n\_clicks")],

    [State("navbar-collapse", "is\_open")],

)

def toggle\_navbar\_collapse(n, is\_open):

    if n:

        return not is\_open

    return is\_open

if \_\_name\_\_ == '\_\_main\_\_':

    app.run\_server(debug=True)

Java script code

# app.js

window.onload = setTimeout(function() {

    console.log("loaded");

    var selector = document.querySelector('#select');

selector.onchange = function() {

    if(document.getElementById("select").value =="Death Event"){

        document.getElementById("hide").style.display = "none";

    }

    else{

    document.getElementById("hide").style.display = "table-column";

    }

}

  },6000)

Code for checking missing Values.

**import** pandas **as** pd

​

*#dataset heart\_failure\_records*

df\_HF **=** pd.read\_csv("heart\_failure\_clinical\_records\_dataset.csv")

​

*#check for duplicates*

print("duplicated values: ",df\_HF.duplicated().sum())

​

*#check for missing values*

print("Missing Values: \n",df\_HF.isnull().sum())

Output:

duplicated values: 0

Missing Values:

age 0

anaemia 0

diabetes 0

platelets 0

serum\_creatinine 0

serum\_sodium 0

sex 0

smoking 0

DEATH\_EVENT 0