```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
from sklearn.ensemble import RandomForestClassifier



st.write("""
# HEART FAILURE PREDICTION APP

This app predicts the likelihood of a person having an **Heart Attack** .

""")


st.sidebar.header('User Medical Records')
st.sidebar.write('please enter your credentials here')

st.header('**Upload a File, or Use the Sliders and Selecboxes by the
left.**')

st.header('**Notice!**')
st.write("""Uploaded File Must be in CSV format comprising of 12 Columns,
            with Column Names in the Following order.

            1. age
            2. anaemia
            3. creatinine_phosphokinase
            4. diabetes
            5. ejection_fraction
            6. high_blood_pressure
            7. platelets
            8. serum_creatinine
            9. serum_sodium
            10.sex
            11.smoking
            12.time""")

uploaded_file = st.sidebar.file_uploader("Upload your input health CSV
file", type = ['csv'])


try:
    input_df = pd.read_csv(uploaded_file)
    input_df['sex'] = np.where(input_df['sex'] == 1, 'Male','Female')

except BaseException:




    def user_input_features():
        age = st.sidebar.slider('What is your Age?',20,100,50)
        anaemia = st.sidebar.selectbox('Do you Have
Anaemia?',(True,False))
        creatinine_phosphokinase = st.sidebar.slider('What is the level
of Creatinine_Phosphokinase(CP) in your body?',20,8000,3000)
```

```
        diabetes = st.sidebar.selectbox('Do you have
Diabetes?',(True,False))
        ejection_fraction = st.sidebar.slider('What is your Ejection_
Fraction?',0,150,75)
        high_blood_pressure = st.sidebar.selectbox('Are you
Hypertensive?',(True,False))
        platelets = st.sidebar.slider('What is your Blood Platelets
count?',15000,900000,15000)
        serum_creatinine = st.sidebar.slider('What is the amount of
Serum_creatinine in your bloodstream?',0.5,10.0,0.5)
        serum_sodium = st.sidebar.slider('What is the level of
Serum_Sodium in your Body?',50,200,50)
        sex = st.sidebar.selectbox('What is your Sex?',('Male','Female'))
        smoking = st.sidebar.selectbox('Do you Smoke?',(True,False))
        time = st.sidebar.slider('How many times have you gone for an
appointment at the Hospital?',0,400,20)
        data = {'age':
age,'anaemia':anaemia,'creatinine_phosphokinase':creatinine_phosphokinase
,

'diabetes':diabetes,'ejection_fraction':ejection_fraction,

'high_blood_pressure':high_blood_pressure,'platelets':platelets,

'serum_creatinine':serum_creatinine,'serum_sodium':serum_sodium,
                'sex':sex,'smoking':smoking,'time':time}

        features = pd.DataFrame(data,index=[0])
        return features
    input_df = user_input_features()


heart_raw = pd.read_csv('heart_failure_clinical_records_dataset.csv')

heart_raw['sex'] = np.where(heart_raw['sex'] == 1, 'Male','Female')

heart = heart_raw.drop(columns = ['DEATH_EVENT'])
data = pd.concat([input_df,heart], axis = 0)
df = data.copy()
df1 = data.copy()



def set_cpk(row):
    if row["creatinine_phosphokinase"] >=10 and
row["creatinine_phosphokinase"] <= 120:
        return "Normal"
    else:
        return "High"

df = df.assign(cp_desc = df.apply(set_cpk, axis = 1))


def set_eject_fract(row):
    if row["ejection_fraction"] <= 35:
        return "Low"
    elif row["ejection_fraction"] > 35 and row["ejection_fraction"] <=
49:
```

```
        return "Below_Normal"
    elif row["ejection_fraction"] > 50 and row["ejection_fraction"] <=
75:
        return "Normal"
    else:
        return "High"
df['ejection_fraction_desc'] =  df.apply(set_eject_fract, axis =1)



def set_platelets(row):
    if row["sex"] == 'Female':  #females
        if row["platelets"] < 157000:
            return "Low"
        elif row["platelets"] >=157000 and row["platelets"] <= 371000:
            return "Normal"
        else:
            return "High"

    elif row["sex"] == 'Male':  #males
        if row["platelets"] < 135000:
            return "Low"
        if row["platelets"] >= 135000 and row["platelets"] <= 317000:
            return "Normal"
        else:
            return "High"
df['platelets_desc'] = df.apply(set_platelets, axis = 1)


def set_sodium(row):
    if row["serum_sodium"] < 135:
        return "Low"
    elif row["serum_sodium"] >=135 and row["serum_sodium"] <= 145:
        return "Normal"
    else:
        return "High"
df['sodium_desc'] = df.apply(set_sodium, axis =1)

def set_creatinine(row):
    if row["sex"] == 'Female':  #females
        if row["serum_creatinine"] >=0.5 and row["serum_creatinine"] <=
1.1:
            return "Normal"
        else:
            return "High"

    elif row["sex"] == 'Male':  #males
        if row["serum_creatinine"] >=0.6 and row["serum_creatinine"] <=
1.2:
            return "Normal"
        else:
            return "High"
df['serum_creatinine_desc'] = df.apply(set_creatinine, axis = 1)

df2 = df1.copy()
df1 = pd.get_dummies(df1,columns = ['sex'], drop_first = True)

df2 = pd.get_dummies(df2,columns = ['sex'], drop_first = True)
```

```
st.subheader('User Medical Profile')

if df is not None:
    st.write(input_df[:len(input_df)])
else:
    st.write('This is the raw input df')
    st.write(input_df[:len(input_df)])

col = ['age','creatinine_phosphokinase','ejection_fraction',
       'platelets','serum_creatinine','serum_sodium','time',
       'anaemia','diabetes','high_blood_pressure','smoking',
       'sex_Male']

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import MinMaxScaler

col_trans = ColumnTransformer(remainder='passthrough',
                              transformers = [('scaler',MinMaxScaler(),
                              [0,2,4,6,7,8,10])])
trans = col_trans.fit_transform(df1)
trans = col_trans.transform(df2)
try:
    trans = pd.DataFrame(trans,columns = col)

except ValueError:
    st.header('**The data you entered is invalid!**')
    st.header("""It either contains wrongly spelt and/or arranged column
headers,"""
              """or more than seven columns.""")


df_ = trans[:len(input_df)]

st.subheader('Medical Profile Description')

if uploaded_file is not None:
    st.write(df.iloc[:len(input_df),12:])
else:
    st.write('These are the scaled input features of the user')
    st.write(df.iloc[:len(input_df),12:])


load_clf = pickle.load(open('model.pkl', 'rb'))

try:
    prediction = load_clf.predict(df_)
    prediction_proba = load_clf.predict_proba(df_)


    st.subheader('DIAGNOSIS')
    for i in range(len(prediction)):
        if prediction[i] > 0:
            st.write(prediction[i],'-->This Patient is at Risk of
Suffering a Heart Attack')
        else:
```

```
            st.write(prediction[i],'-->This Patient is in a stable Health
Condition')

    st.subheader('Probability')
    st.write(prediction_proba)
except ValueError:
    st.header("Invalid data was supplied to the predictor")
```