NAME: ROSHAN GURUNG

BATCH CODE: LISUM32

SUBMISSION DATE: 5TH MAY 2024

SUBMITTED TO: DATA GLACIER

STEPS:

1. **MODEL CREATION**

DATASET USED:

AIM: TO PREDICT THE WHETHER THE PATIENT WILL SURVIVE OR NOT

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

Python

```
df=pd.read_csv('heart_failure_clinical_records_dataset.csv')
df.head()
```

Python

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	D
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	

Figure 1importing library and dataset

cleaning the data

```
df.isna().sum()
```

```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

Figure 2 cleaning the dataset 1

There are no null values

Let's check for the duplicate data

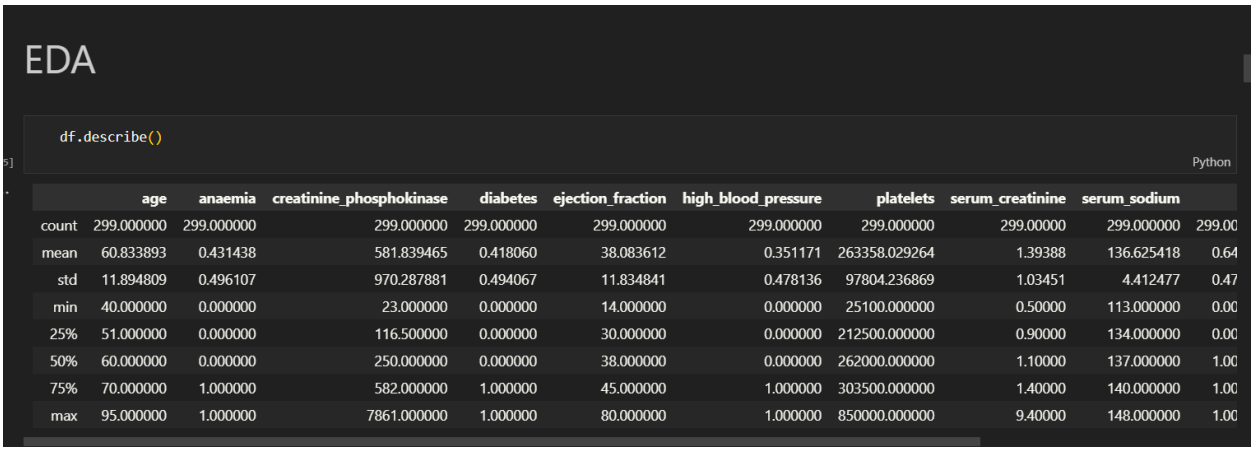
Empty markdown cell, double-click or press enter to edit.

```
[4] df.duplicated().sum()
```

```
0
```

There are no duplicate data too

Figure 3 cleaning the dataset 2



	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.00
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418	0.64
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477	0.47
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000	0.00
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000	0.00
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000	1.00
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	1.40000	140.000000	1.00
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000	1.00

Figure 4 EDA

```
# creating a RF classifier
clf=RandomForestClassifier(n_estimators=100)

# training the model on the training set
clf.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
y_pred=clf.predict(X_test_scaled)
```

```
accuracy=metrics.accuracy_score(y_test, y_pred)
```

Figure 5 Random forest classifier model training and testing

```
import pickle
file_name='heart_failure.pkl'
pickle.dump(clf, open(file_name, 'wb'))
```

Figure 6 Saving the model using the pickle

2. **Deploying the model in flask**

```
from flask import Flask, render_template, request
from sklearn.ensemble import RandomForestClassifier
import pickle


# create a flask application
app=Flask(__name__)

# Load the trained model
try:
    with open('heart_failure.pkl', 'rb') as model_file:
        model = pickle.load(model_file)
except FileNotFoundError:
    print("Model file not found! Make sure 'heart_failure.pkl' exists in the current directory.")



@app.route('/')
def index():
    """
    Renders the index.html template when the user accesses the root URL.
    """
    return render_template('index.html')
```

Figure 7 Flask application creation and defining route

In this step I have created a flaks application by importing the flask library and defined a route to render the index.html where I have created a simple form which takes input from the users.

```python
@app.route('/predict', methods=['POST'])
def predict():
    """
    Handles POST requests sent to the /predict route, makes predictions based on user input,
    and renders the result.html template with the prediction result.
    """
    if request.method == 'POST':
        # Get user input from the form
        user_input = request.form
        features = [float(user_input['age']), int(user_input['anaemia']), int(user_input['creatinine_phosphokinase']),
                    int(user_input['diabetes']), int(user_input['ejection_fraction']), int(user_input['high_blood_pressure']),
                    float(user_input['platelets']), float(user_input['serum_creatinine']), int(user_input['serum_sodium']),
                    int(user_input['sex']), int(user_input['smoking']), int(user_input['time'])]

        # Make prediction
        prediction = model.predict([features])[0]

        # Display prediction result
        result = "Survived" if prediction == 0 else "Deceased"
        return render_template('result.html', prediction=result)

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 8 defining route to show result using the post function

In this step I have created a function which shows the output for the user input. It renders the result.html file which shows the final prediction of the patient.

t Failure prediction > templates > index.html > html > body

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Death Event Predictor</title>
</head>

<body>
    <h2>Enter Patient Information</h2>
    <form action="/predict" method="POST">
        <label for="age">Age:</label><br>
        <input type="text" id="age" name="age" required><br><br>

        <label for="anaemia">Anaemia (0 for No, 1 for Yes):</label><br>
        <input type="text" id="anaemia" name="anaemia" required><br><br>

        <label for="creatinine_phosphokinase">Creatinine Phosphokinase:</label><br>
        <input type="text" id="creatinine_phosphokinase" name="creatinine_phosphokinase" required><br><br>

        <label for="diabetes">Diabetes (0 for No, 1 for Yes):</label><br>
        <input type="text" id="diabetes" name="diabetes" required><br><br>

        <label for="ejection_fraction">Ejection Fraction:</label><br>
        <input type="text" id="ejection_fraction" name="ejection_fraction" required><br><br>

        <label for="high_blood_pressure">High Blood Pressure (0 for No, 1 for Yes):</label><br>
        <input type="text" id="high_blood_pressure" name="high_blood_pressure" required><br><br>

        <label for="platelets">Platelets:</label><br>
        <input type="text" id="platelets" name="platelets" required><br><br>

        <label for="serum_creatinine">Serum Creatinine:</label><br>
        <input type="text" id="serum_creatinine" name="serum_creatinine" required><br><br>
```

Figure 9 Index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Death Event Predictor</title>
</head>

<body>
    <h2>Prediction Result</h2>
    <p>The patient is predicted to be: {{ prediction }}</p>
    <!-- Display the prediction result -->
</body>

</html>
```

Figure 10 result.html