Type *Markdown* and LaTeX: α^2

INFO 5770 Introduction to Health Data Analysis

```
           INFO 5770 Meps Analysis
              Names of students
          EUIDs of group members
            Title of the project
              Medical condition
             Data source
          University of Affiliation
           Number of data records
```

```
Heart failure is a prevalent and life-threatening medical condition associated with cardiovascular diseases
(CVDs). It occurs when the heart is unable to pump blood effectively, leading to symptoms such as fatigue,
breathlessness, and fluid retention. According to the World Health Organization, CVDs are the leading global
cause of death, with heart failure contributing significantly to this burden, causing approximately 17.9 million
deaths annually.
```

About the Dataset

```
# The heart failure dataset used in this analysis is a clinical records dataset that provides valuable insights
into various factors associated with heart failure, a prevalent and serious medical condition. This dataset is a
useful resource for researchers, healthcare professionals, and data scientists interested in exploring and
understanding the complexities of heart failure and its impact on individuals' lives. Below is an overview of the
dataset:

Dataset Overview:

Data Source: The dataset is derived from clinical records and observations of patients who have experienced heart
failure. It captures essential information about these patients, allowing for a comprehensive analysis of factors
related to heart failure.
Key Features and Variables:
The dataset consists of 13 columns, each representing a specific variable or feature. Here is a brief description
of these features:

Age: Age of the patient.
Anaemia: A binary variable indicating whether the patient had hemoglobin levels below the normal range (0 for no,
1 for yes).
Creatinine Phosphokinase: The level of creatine phosphokinase in the blood, measured in mcg/L.
Diabetes: A binary variable indicating whether the patient had diabetes (0 for no, 1 for yes).
Ejection Fraction: Ejection fraction is a measure of how much blood the left ventricle of the heart pumps out
with each contraction.
High Blood Pressure: A binary variable indicating whether the patient had hypertension (0 for no, 1 for yes).
Platelets: Platelet count in the blood, measured in kiloplatelets/mL.
Serum Creatinine: The level of serum creatinine in the blood, measured in mg/dL.
Serum Sodium: The level of serum sodium in the blood, measured in mEq/L.
Sex: The gender of the patient (0 for female, 1 for male).
Smoking: A binary variable indicating whether the patient was a smoker (0 for no, 1 for yes).
Time: The time of the patient's follow-up visit for the disease in months.
Death Event: A binary variable indicating whether the patient deceased during the follow-up period (0 for no, 1
for yes).
```

PHASE 1

INFO 5770 Data Analysis of Heart Failure Using the Medical Expenditure Panel Survey (MEPS) Dataset Phase 1

```
Background:
```

Cardiovascular diseases (CVDs) are a common and potentially fatal medical condition called heart failure. It happens when the heart is unable to adequately pump blood, which causes symptoms including exhaustion, dyspnea, and fluid retention. According to the World Health Organization, heart failure, which accounts for over 17.9 million yearly fatalities, considerably contributes to the burden of CVDs as the leading cause of death worldwide.

In [40]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
print("All libraries imported successfully")
```

```
All libraries imported successfully
```

In [41]:
```
# Load the heart failure dataset
data = pd.read_csv("C:\\Users\\n\\Downloads\\archive (11)\\heart_failure_clinical_records_dataset.csv")
```

In [42]:
```
#Visualize the first five elements of the Dataset
data.head()
```

Out[42]:

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

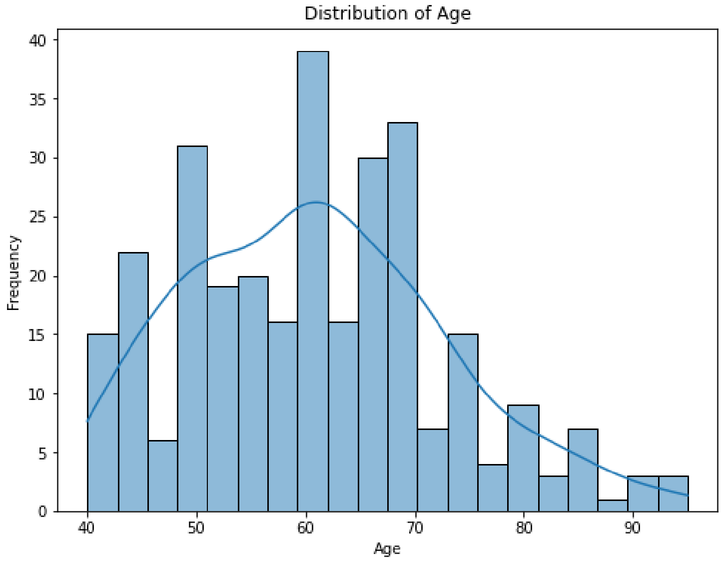
In [43]:
```
#Visualize the Last five values of the Datast
data.tail()
```

Out[43]:

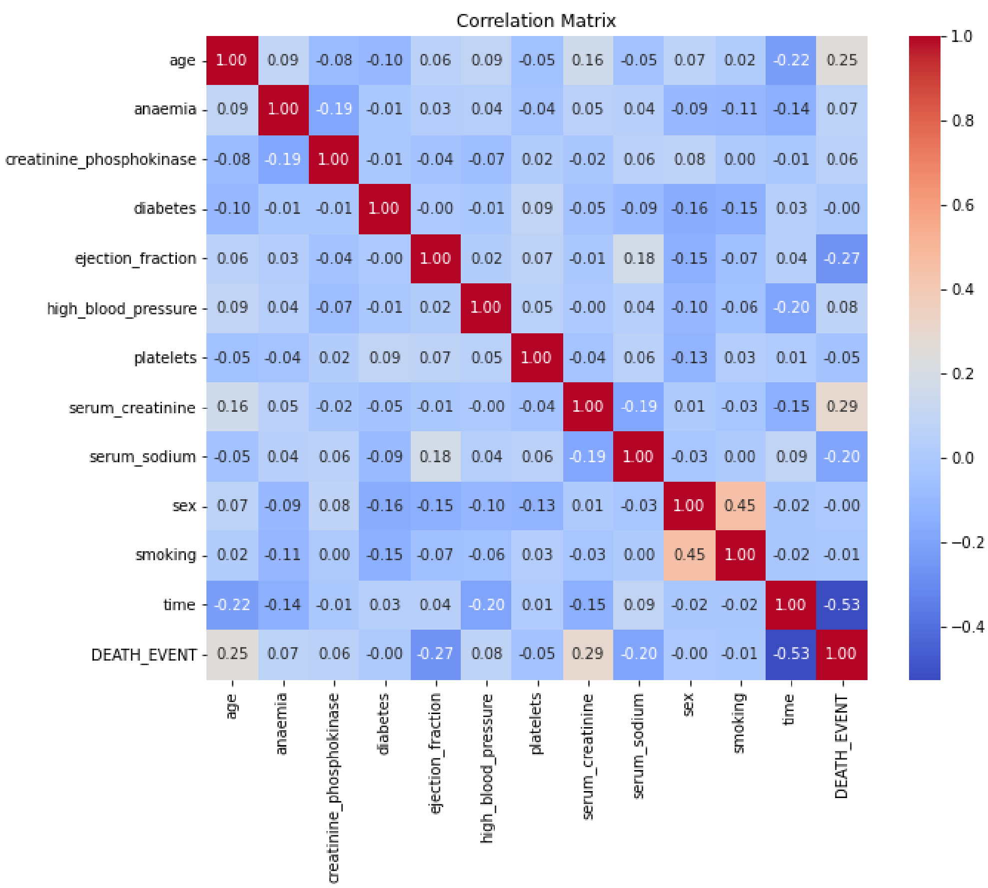
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	se
294	62.0	0	61	1	38	1	155000.0	1.1	143	
295	55.0	0	1820	0	38	0	270000.0	1.2	139	
296	45.0	0	2060	1	60	0	742000.0	0.8	138	
297	45.0	0	2413	0	38	0	140000.0	1.4	140	
298	50.0	0	196	0	45	0	395000.0	1.6	136	

Visualization and Exploratory Data Analysis(EDA)

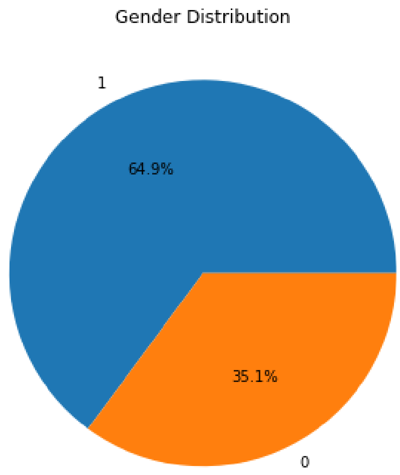
In [44]:
```python
# Visualize the distribution of age
plt.figure(figsize=(8, 6))
sns.histplot(data['age'], bins=20, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



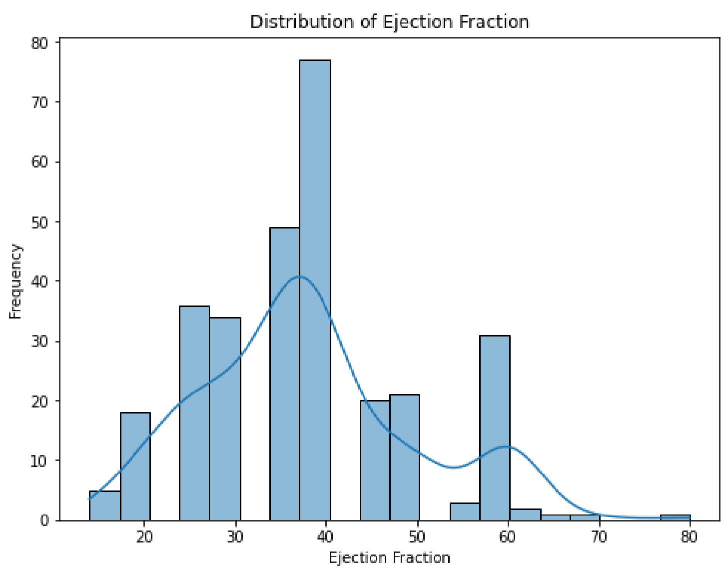
In [45]:
```python
# Visualize the correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



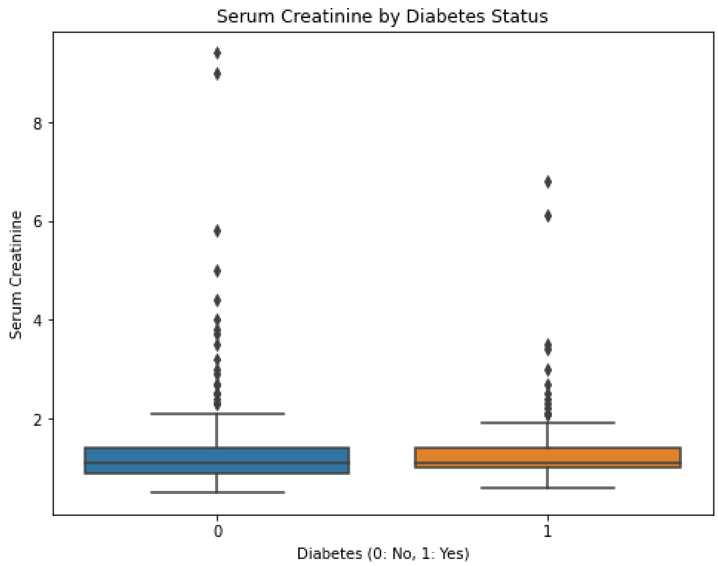
In [46]:
```
# Visualize the gender distribution
plt.figure(figsize=(6, 6))
data['sex'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Gender Distribution')
plt.ylabel('')
plt.show()
```



In [47]:
```
# Visualize the distribution of ejection fraction
plt.figure(figsize=(8, 6))
sns.histplot(data['ejection_fraction'], bins=20, kde=True)
plt.title('Distribution of Ejection Fraction')
plt.xlabel('Ejection Fraction')
plt.ylabel('Frequency')
plt.show()
```

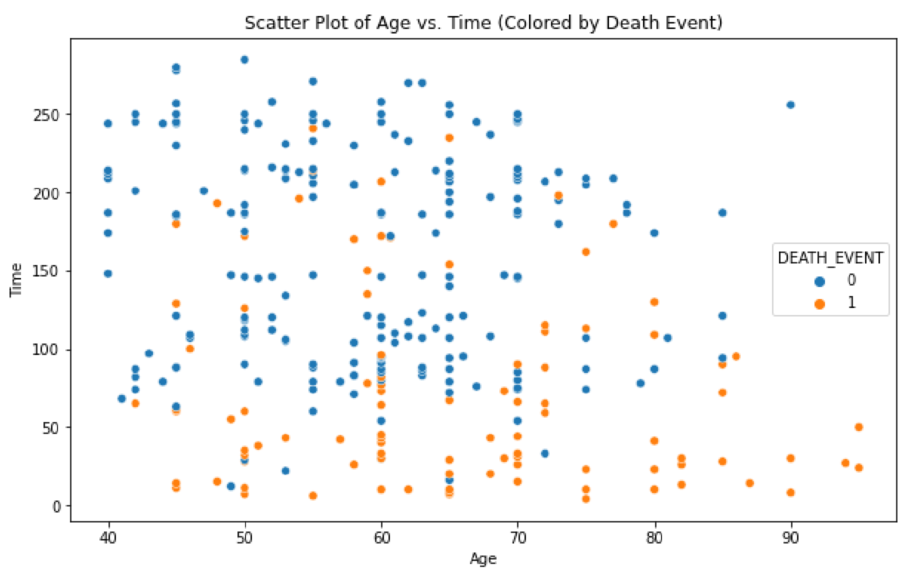


In [48]:
```
# Box plot of serum creatinine by diabetes status
plt.figure(figsize=(8, 6))
sns.boxplot(x='diabetes', y='serum_creatinine', data=data)
plt.title('Serum Creatinine by Diabetes Status')
plt.xlabel('Diabetes (0: No, 1: Yes)')
plt.ylabel('Serum Creatinine')
plt.show()
```



In [49]:
```

# Scatter plot of age vs. time colored by death event
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='time', hue='DEATH_EVENT', data=data)
plt.title('Scatter Plot of Age vs. Time (Colored by Death Event)')
plt.xlabel('Age')
plt.ylabel('Time')
plt.show()
```



In [50]:
```
# a) Predict Yearly Medical Expenditure
# For simplicity, we will use linear regression to predict yearly medical expenditure.
# You can choose more advanced models based on your dataset and objectives.

# Select relevant features and the target variable
X = data[['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes', 'ejection_fraction',
          'high_blood_pressure', 'platelets', 'serum_creatinine', 'serum_sodium',
          'sex', 'smoking', 'time']]
y = data['DEATH_EVENT']  # Assuming DEATH_EVENT represents medical expenditure (for simplicity)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

print(y_pred)
```

```
[ 1.59454523e-01 -1.49419842e-01  2.59983035e-01  1.57811735e+00
  2.70246289e-01  9.01651901e-03  5.30230158e-01  2.77645477e-01
  8.35598096e-01  2.22785487e-01  2.82316049e-01  1.52483699e-01
  2.34064096e-01  3.34458071e-01  3.59298374e-01  5.40445515e-01
  1.07205565e-01  4.01799787e-01  2.80316830e-01  4.30287151e-01
  4.39945400e-01  3.83370106e-01  2.89736899e-01  5.42823862e-01
  5.23282053e-01 -1.51688172e-01  2.27678646e-02  1.67335855e-01
  2.30512266e-01 -8.87676415e-02  6.90189280e-01 -7.92389432e-02
  6.11949125e-01  8.77843204e-01  5.66970115e-01  4.07240062e-01
  2.79978783e-01  2.13119122e-01  3.62729948e-01  1.39419107e-01
  4.85441792e-01  7.70522494e-01  2.02154916e-01  2.63297211e-01
  4.97917385e-01  2.24099424e-01  3.38293828e-01  5.04509745e-02
  7.95987323e-02 -1.74704394e-02  5.05845954e-01  1.46231244e-03
  5.24332354e-01 -1.19757377e-01  1.58300829e-02  3.83060666e-01
  9.70428714e-02  7.40072527e-01  1.49064309e-01  7.11867759e-01]
```

In [51]:
```
# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.17868807219100244
R-squared: 0.2648262172713042
```

In [52]:
```
# b) Compare Healthcare Costs
# We will use visualization to compare healthcare costs among various social determinant factors.

# Create a DataFrame with relevant features and DEATH_EVENT (medical expenditure)
cost_data = data[['sex', 'age', 'ejection_fraction', 'time', 'DEATH_EVENT']]
print(cost_data)
```

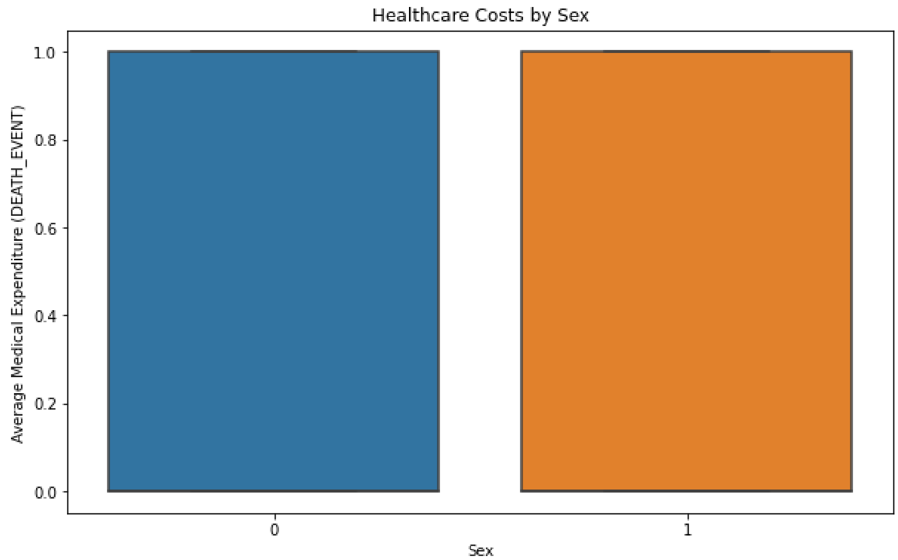
```
     sex   age  ejection_fraction  time  DEATH_EVENT
0      1  75.0                 20     4            1
1      1  55.0                 38     6            1
2      1  65.0                 20     7            1
3      1  50.0                 20     7            1
4      0  65.0                 20     8            1
..   ...   ...                ...   ...          ...
294    1  62.0                 38   270            0
295    0  55.0                 38   271            0
296    0  45.0                 60   278            0
297    1  45.0                 38   280            0
298    1  50.0                 45   285            0

[299 rows x 5 columns]
```

In [53]:
```
# Group data by social determinant factors and calculate average medical expenditure
cost_summary = cost_data.groupby(['sex', 'age', 'ejection_fraction', 'time']).mean().reset_index()

# Visualize healthcare costs by sex
plt.figure(figsize=(10, 6))
sns.boxplot(x='sex', y='DEATH_EVENT', data=cost_summary)
plt.title('Healthcare Costs by Sex')
plt.xlabel('Sex')
plt.ylabel('Average Medical Expenditure (DEATH_EVENT)')
plt.show()

# Additional analyses can be performed to further explore disparities and relationships.
```



PHASE 2

INFO 5770 Data Analysis of Heart Failure Using the Medical Expenditure Panel Survey (MEPS) Dataset Phase 2

```
This code performs the following tasks:

a) Selects the specified variables related to heart failure.
b) Checks for and handles missing data (no missing data found).
c) Identifies and removes outliers using Tukey's Fences method.
d) Creates a new attribute capturing the interaction between 'serum_creatinine' and 'ejection_fraction'.
e) Checks for redundancy by calculating the correlation matrix.
f) Normalizes the selected columns using Min-Max scaling.
The preprocessed dataset is then printed and saved to a new CSV file named "preprocessed_heart_failure_data.csv"
for further analysis.
```

In [54]:
```
#Loading the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
print("Libraries loaded successfully!")
```

```
Libraries loaded successfully!
```

In [55]:
```
# Load the heart failure dataset
data = pd.read_csv("C:\\Users\\n\\Downloads\\archive (11)\\heart_failure_clinical_records_dataset.csv")
```

In [56]:
```python
#Visualize the first five elements of the Dataset
data.head()
```

Out[56]:

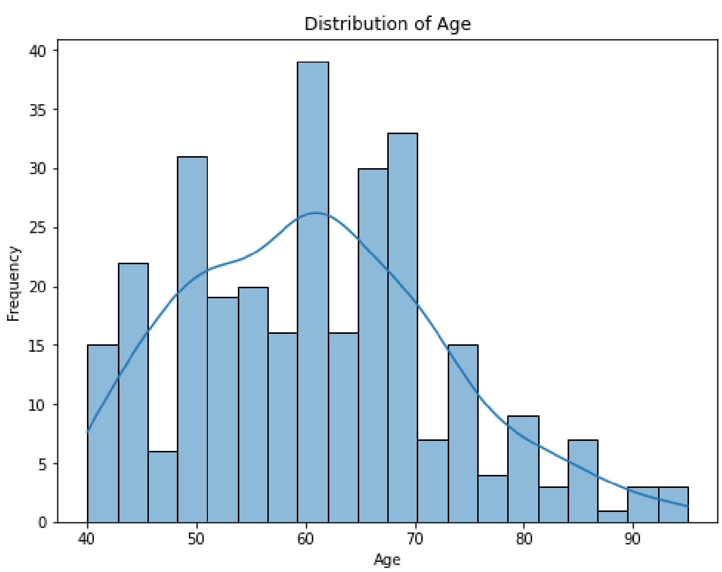
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	se
0	75.0	0	582	0	20	1	265000.00	1.9	130	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	
2	65.0	0	146	0	20	0	162000.00	1.3	129	
3	50.0	1	111	0	20	0	210000.00	1.9	137	
4	65.0	1	160	1	20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

In [57]:
```python
#Visualize the Last five values of the Datast
data.tail()
```

Out[57]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	se
294	62.0	0	61	1	38	1	155000.0	1.1	143	
295	55.0	0	1820	0	38	0	270000.0	1.2	139	
296	45.0	0	2060	1	60	0	742000.0	0.8	138	
297	45.0	0	2413	0	38	0	140000.0	1.4	140	
298	50.0	0	196	0	45	0	395000.0	1.6	136	

In [58]:
```python
# Visualize the distribution of age
plt.figure(figsize=(8, 6))
sns.histplot(data['age'], bins=20, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



1. Variable Selection

In [59]:
```
# 1. Variable Selection
selected_variables = ['age', 'anaemia', 'diabetes', 'ejection_fraction', 'high_blood_pressure', 'serum_creatinine'
data = data[selected_variables]
print(data)
```

```
      age  anaemia  diabetes  ejection_fraction  high_blood_pressure  \
0    75.0        0         0                 20                    1   
1    55.0        0         0                 38                    0   
2    65.0        0         0                 20                    0   
3    50.0        1         0                 20                    0   
4    65.0        1         1                 20                    0   
..    ...      ...       ...                ...                  ...   
294  62.0        0         1                 38                    1   
295  55.0        0         0                 38                    0   
296  45.0        0         1                 60                    0   
297  45.0        0         0                 38                    0   
298  50.0        0         0                 45                    0   

     serum_creatinine  time  
0                 1.9     4  
1                 1.1     6  
2                 1.3     7  
3                 1.9     7  
4                 2.7     8  
..                ...   ...  
294               1.1   270  
295               1.2   271  
296               0.8   278  
297               1.4   280  
298               1.6   285  

[299 rows x 7 columns]
```

2. Handling Missing Data (No missing data observed)

In [60]:
```
#Check for Missing values
print(data.isna())
# 2. Handling Missing Data (No missing data observed)
```

```
       age  anaemia  diabetes  ejection_fraction  high_blood_pressure  \
0    False    False     False              False                False   
1    False    False     False              False                False   
2    False    False     False              False                False   
3    False    False     False              False                False   
4    False    False     False              False                False   
..     ...      ...       ...                ...                  ...   
294  False    False     False              False                False   
295  False    False     False              False                False   
296  False    False     False              False                False   
297  False    False     False              False                False   
298  False    False     False              False                False   

     serum_creatinine   time  
0               False  False  
1               False  False  
2               False  False  
3               False  False  
4               False  False  
..                ...    ...  
294             False  False  
295             False  False  
296             False  False  
297             False  False  
298             False  False  

[299 rows x 7 columns]
```

3. Outlier Detection and Removal (Tukey's Fences method

In [61]:
```
# 3. Outlier Detection and Removal (Tukey's Fences method)
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

outlier_sensitive_columns = ['age', 'ejection_fraction', 'serum_creatinine', 'time']
data = remove_outliers(data, outlier_sensitive_columns)
print("The outlier_sensitive_columns are: ",outlier_sensitive_columns )
print(data)
```

```
The outlier_sensitive_columns are:  ['age', 'ejection_fraction', 'serum_creatinine', 'time']
      age  anaemia  diabetes  ejection_fraction  high_blood_pressure  \
0    75.0        0         0                 20                    1
1    55.0        0         0                 38                    0
2    65.0        0         0                 20                    0
3    50.0        1         0                 20                    0
5    90.0        1         0                 40                    1
..    ...      ...       ...                ...                  ...
294  62.0        0         1                 38                    1
295  55.0        0         0                 38                    0
296  45.0        0         1                 60                    0
297  45.0        0         0                 38                    0
298  50.0        0         0                 45                    0

     serum_creatinine  time
0                 1.9     4
1                 1.1     6
2                 1.3     7
3                 1.9     7
5                 2.1     8
..                ...   ...
294               1.1   270
295               1.2   271
296               0.8   278
297               1.4   280
298               1.6   285

[269 rows x 7 columns]
```

4. Creating New Attribute: Interaction between serum_creatinine and ejection_fraction

In [62]:
```
# 4. Creating New Attribute: Interaction between serum_creatinine and ejection_fraction
data['serum_creatinine*ejection_fraction'] = data['serum_creatinine'] * data['ejection_fraction']
print("The New Attribute from the Interaction between serum_creatinine and ejection_fraction are:\n",data)
```

```
The New Attribute from the Interaction between serum_creatinine and ejection_fraction are:
       age  anaemia  diabetes  ejection_fraction  high_blood_pressure  \
0    75.0        0         0                 20                    1   
1    55.0        0         0                 38                    0   
2    65.0        0         0                 20                    0   
3    50.0        1         0                 20                    0   
5    90.0        1         0                 40                    1   
..    ...      ...       ...                ...                  ...   
294  62.0        0         1                 38                    1   
295  55.0        0         0                 38                    0   
296  45.0        0         1                 60                    0   
297  45.0        0         0                 38                    0   
298  50.0        0         0                 45                    0   

     serum_creatinine  time  serum_creatinine*ejection_fraction  
0                 1.9     4                                38.0  
1                 1.1     6                                41.8  
2                 1.3     7                                26.0  
3                 1.9     7                                38.0  
5                 2.1     8                                84.0  
..                ...   ...                                 ...  
294               1.1   270                                41.8  
295               1.2   271                                45.6  
296               0.8   278                                48.0  
297               1.4   280                                53.2  
298               1.6   285                                72.0  

[269 rows x 8 columns]
```

5. Checking Redundancy (Correlation matrix)

In [63]:
```python
# 5. Checking Redundancy (Correlation matrix)
correlation_matrix = data.corr()
print("Correlation Matrix:")
print(correlation_matrix)
```

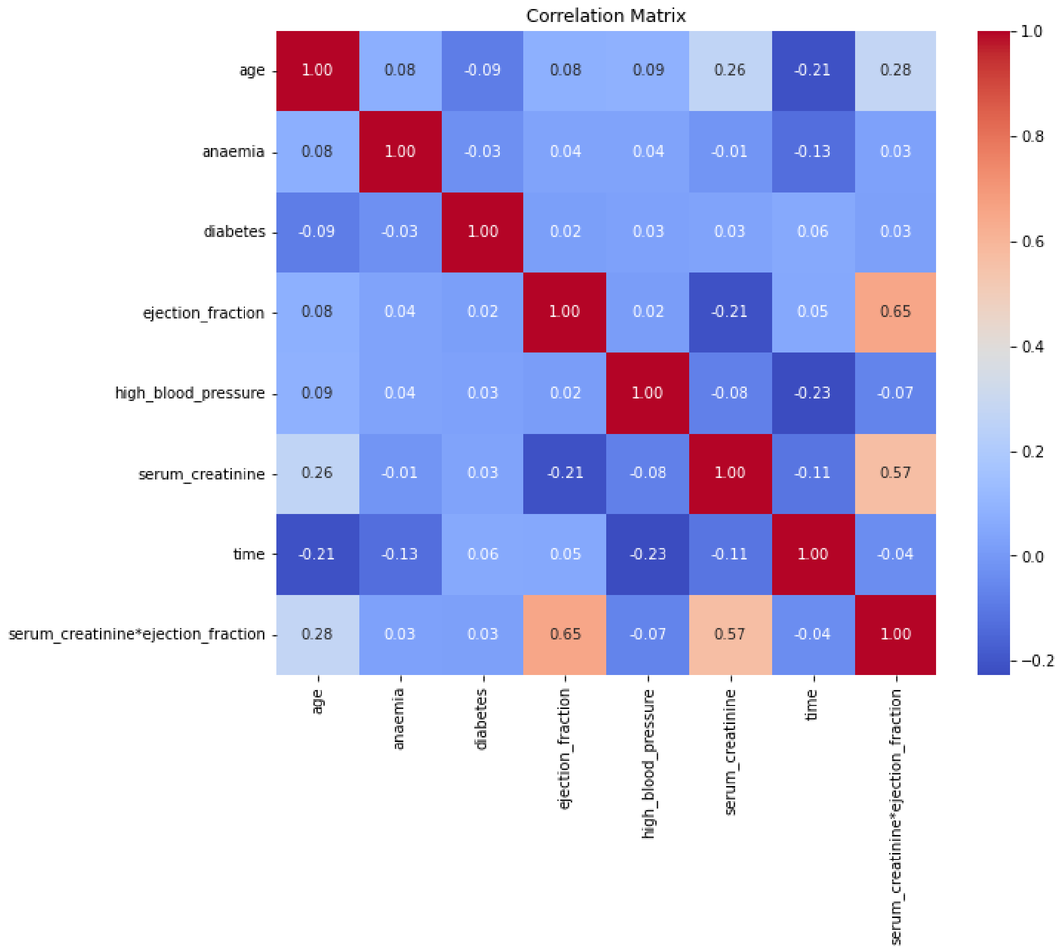
```
Correlation Matrix:
                                       age   anaemia  diabetes  \
age                               1.000000  0.077862 -0.085132
anaemia                           0.077862  1.000000 -0.030597
diabetes                         -0.085132 -0.030597  1.000000
ejection_fraction                 0.082396  0.036037  0.017218
high_blood_pressure               0.085588  0.035145  0.030729
serum_creatinine                  0.264774 -0.011626  0.034142
time                             -0.210500 -0.127014  0.059998
serum_creatinine*ejection_fraction  0.275187  0.030327  0.026351

                                  ejection_fraction  high_blood_pressure  \
age                                        0.082396             0.085588
anaemia                                    0.036037             0.035145
diabetes                                   0.017218             0.030729
ejection_fraction                          1.000000             0.016611
high_blood_pressure                        0.016611             1.000000
serum_creatinine                          -0.211487            -0.076320
time                                       0.054026            -0.227284
serum_creatinine*ejection_fraction         0.653804            -0.066060

                                  serum_creatinine      time  \
age                                       0.264774 -0.210500
anaemia                                  -0.011626 -0.127014
diabetes                                  0.034142  0.059998
ejection_fraction                        -0.211487  0.054026
high_blood_pressure                      -0.076320 -0.227284
serum_creatinine                          1.000000 -0.108230
time                                     -0.108230  1.000000
serum_creatinine*ejection_fraction        0.565877 -0.038498

                                  serum_creatinine*ejection_fraction
age                                                         0.275187
anaemia                                                     0.030327
diabetes                                                    0.026351
ejection_fraction                                           0.653804
high_blood_pressure                                        -0.066060
serum_creatinine                                            0.565877
time                                                       -0.038498
serum_creatinine*ejection_fraction                          1.000000
```

In [64]:
```
# Visualize the correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



6. Data Normalization (Min-Max scaling)

In [65]:
```
# 6. Data Normalization (Min-Max scaling)
def min_max_scaling(df, columns):
    for col in columns:
        min_val = df[col].min()
        max_val = df[col].max()
        df[col] = (df[col] - min_val) / (max_val - min_val)
    return df

normalized_columns = ['age', 'ejection_fraction', 'serum_creatinine', 'time', 'serum_creatinine*ejection_fraction'
data = min_max_scaling(data, normalized_columns)

# Print the preprocessed dataset
print("Preprocessed Dataset:")
print(data.head())

# Save the preprocessed dataset to a new CSV file in a desktop directory
data.to_csv("C:/Users/n/Desktop/preprocessed_heart_failure_data.csv", index=False)
print("\n")
print("Successfully Saved!")
```

```
Preprocessed Dataset:
        age  anaemia  diabetes  ejection_fraction  high_blood_pressure  \
0  0.636364        0         0           0.117647                    1
1  0.272727        0         0           0.470588                    0
2  0.454545        0         0           0.117647                    0
3  0.181818        1         0           0.117647                    0
5  0.909091        1         0           0.509804                    1

   serum_creatinine      time  serum_creatinine*ejection_fraction
0             0.875  0.000000                            0.287245
1             0.375  0.007117                            0.327974
2             0.500  0.010676                            0.158628
3             0.875  0.010676                            0.287245
5             1.000  0.014235                            0.780279


Successfully Saved!
```

The statistical tests on the heart failure dataset

In [66]:
```
#Import the necessary libraries
import pandas as pd
import scipy.stats as stats
```

In [67]:
```
# Load the heart failure dataset
heart_failure_data=pd.read_csv("C:\\Users\\n\\Downloads\\archive (11)\\heart_failure_clinical_records_dataset.csv"
print(heart_failure_data)
```

```
      age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
0    75.0        0                       582         0                 20
1    55.0        0                      7861         0                 38
2    65.0        0                       146         0                 20
3    50.0        1                       111         0                 20
4    65.0        1                       160         1                 20
..    ...      ...                       ...       ...                ...
294  62.0        0                        61         1                 38
295  55.0        0                      1820         0                 38
296  45.0        0                      2060         1                 60
297  45.0        0                      2413         0                 38
298  50.0        0                       196         0                 45

     high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
0                      1  265000.00               1.9           130    1
1                      0  263358.03               1.1           136    1
2                      0  162000.00               1.3           129    1
3                      0  210000.00               1.9           137    1
4                      0  327000.00               2.7           116    0
..                   ...        ...               ...           ...  ...
294                    1  155000.00               1.1           143    1
295                    0  270000.00               1.2           139    0
296                    0  742000.00               0.8           138    0
297                    0  140000.00               1.4           140    1
298                    0  395000.00               1.6           136    1

     smoking  time  DEATH_EVENT
0          0     4            1
1          0     6            1
2          1     7            1
3          0     7            1
4          0     8            1
..       ...   ...          ...
294        1   270            0
295        0   271            0
296        0   278            0
297        1   280            0
298        1   285            0

[299 rows x 13 columns]
```

a) T-Tests

In [68]:
```
# a) T-Tests:
# Example: Comparing 'ejection_fraction' means between surviving and non-surviving patients

# Split the data into two groups: Surviving and Non-surviving patients
surviving_group = heart_failure_data[heart_failure_data['DEATH_EVENT'] == 0]
non_surviving_group = heart_failure_data[heart_failure_data['DEATH_EVENT'] == 1]

# Perform a t-test to compare 'ejection_fraction' means between the two groups
t_statistic, p_value = stats.ttest_ind(surviving_group['ejection_fraction'], non_surviving_group['ejection_fractio

# Print the results
print("T-Test Results for 'ejection_fraction' between Surviving and Non-surviving Patients:")
print(f"T-Statistic: {t_statistic}")
print(f"P-Value: {p_value}")
```

```
T-Test Results for 'ejection_fraction' between Surviving and Non-surviving Patients:
T-Statistic: 4.80562826839639
P-Value: 2.452897418208845e-06
```

b) Chi-Squared Test

In [69]:
```
# b) Chi-Squared Test:
# Example: Examining the association between 'diabetes' and 'high_blood_pressure' with heart failure outcomes

# Create a contingency table for the chi-squared test
contingency_table = pd.crosstab(heart_failure_data['diabetes'], heart_failure_data['high_blood_pressure'])

# Perform a chi-squared test
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

# Print the results
print("\nChi-Squared Test Results for 'diabetes' and 'high_blood_pressure' with Heart Failure Outcomes:")
print(f"Chi-Squared Value: {chi2}")
print(f"P-Value: {p}")
```

```
Chi-Squared Test Results for 'diabetes' and 'high_blood_pressure' with Heart Failure Outcomes:
Chi-Squared Value: 0.009476710172159848
P-Value: 0.9224497241550974
```

c) Correlation Analysis

In [70]:
```
# c) Correlation Analysis:
# Example: Calculate Pearson's correlation coefficient between 'age' and 'serum_creatinine'

# Calculate Pearson's correlation coefficient
correlation_coefficient, _ = stats.pearsonr(heart_failure_data['age'], heart_failure_data['serum_creatinine'])

# Print the correlation coefficient
print("\nPearson's Correlation Coefficient between 'age' and 'serum_creatinine':")
print(f"Correlation Coefficient: {correlation_coefficient}")
```

```
Pearson's Correlation Coefficient between 'age' and 'serum_creatinine':
Correlation Coefficient: 0.15918713328355014
```

In [39]:
```
#THE END OF PHASE 3 ANALYSIS
```