D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

DY Patil International University

TC- 2

Principle of Data Science & Engg.

Year - 3rd

Semester - 5th

Lab Manual

Name: Parth Deshpande

PRN: 20220802317

Dr. Anuj Kumar
(Main Faculty)

Mrs. Remya Praveen
(Teaching Assistant)

Index

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 1

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

Aim:
The aim of this project is to develop a Python program that performs various statistical operations on Comma Separated Value (CSV) data without relying on external libraries, using custom functions for each operation. The program will identify categorical and numerical variables, generate a contingency table for categorical variables, calculate statistical measures for numerical variables, and categorize categorical variables based on data type.

Objective:
The objectives of this project are as follows:
1.Load CSV data and convert it into a manageable format.
2. Identify categorical and numerical variables from the data.
3. Create a contingency table for at most 2 different categorical variables.
4. Calculate the mean, median, mode, variance, standard deviation, and quartile range for numerical variables.
5. Categorize categorical variables into binary, nominal, and ordinal types.
Theory:

1. **Loading CSV Data:** The CSV data is loaded into a list of dictionaries, where each row represents a dictionary with column names as keys and corresponding values.
2. **Identifying Variable Types:** By analyzing the first row of data, the program distinguishes between categorical and numerical variables. Categorical variables have non-numeric values, while numerical variables have numeric values.
3. **Contingency Table:** The contingency table is constructed by counting the occurrences of pairs of values from two categorical variables. The Counter class is utilized to achieve this.
4. **Statistical Measures:** For numerical variables, the program calculates mean, median, mode, variance, standard deviation, and quartile range. These statistics are computed using basic mathematical operations.
5. **Categorizing Categorical Variables:** The program categorizes categorical variables based on the number of unique values they have. Binary variables have 2 unique values, nominal variables have more than 2 unique values, and ordinal variables have 10 or fewer unique values.

Code

```
from collections import Counter
import math

data = [
    {'sepal_length': 5.1, 'sepal_width': 3.5, 'species': 'setosa', 'color': 'red'},
    {'sepal_length': 4.9, 'sepal_width': 3.0, 'species': 'setosa', 'color': 'blue'},
    {'sepal_length': 4.7, 'sepal_width': 3.2, 'species': 'setosa', 'color': 'red'},
    {'sepal_length': 4.6, 'sepal_width': 3.1, 'species': 'setosa', 'color': 'blue'},
    {'sepal_length': 5.0, 'sepal_width': 3.6, 'species': 'setosa', 'color': 'red'},
    {'sepal_length': 6.4, 'sepal_width': 3.2, 'species': 'versicolor', 'color': 'green'},
    {'sepal_length': 6.9, 'sepal_width': 3.1, 'species': 'versicolor', 'color': 'green'},
    {'sepal_length': 6.5, 'sepal_width': 3.0, 'species': 'virginica', 'color': 'yellow'},
    {'sepal_length': 6.3, 'sepal_width': 2.3, 'species': 'virginica', 'color': 'yellow'},
    {'sepal_length': 6.5, 'sepal_width': 3.0, 'species': 'virginica', 'color': 'green'}
]

def identify_variables(data):
    sample_row = data[0]
    categorical_vars = []
    numerical_vars = []

    for key, value in sample_row.items():
        if isinstance(value, (int, float)):
            numerical_vars.append(key)
        else:
            categorical_vars.append(key)

    return categorical_vars, numerical_vars

def create_contingency_table(data, cat_var1, cat_var2):
    table = {}
    for row in data:
        key = (row[cat_var1], row[cat_var2])
        if key not in table:
            table[key] = 1
        else:
            table[key] += 1
    return table

def calculate_statistics(data, numerical_vars):
    num_vars = numerical_vars
    for num_var in num_vars:
        values = [row[num_var] for row in data]
        mean = sum(values) / len(values)
        median = (sorted(values)[len(values) // 2] if len(values) % 2 != 0 else (sorted(values)[len(values) // 2 - 1] + sorted(values)[len(values) // 2]) / 2)
        mode = Counter(values).most_common(1)[0][0]
        variance = sum((x - mean) ** 2 for x in values) / len(values)
        std_dev = math.sqrt(variance)
        q1 = sorted(values)[len(values) // 4]
        q3 = sorted(values)[3 * len(values) // 4]
        iqr = q3 - q1
        print(f"{num_var}:")
        print(f"Mean: {mean}")
        print(f"Median: {median}")
        print(f"Mode: {mode}")
        print(f"Variance: {variance}")
        print(f"Standard Deviation: {std_dev}")
        print(f"Q1: {q1}")
        print(f"Q3: {q3}")
        print(f"IQR: {iqr}")

def categorize_categorical_vars(data, categorical_vars):
    binary_vars = []
    nominal_vars = []
    ordinal_vars = []

    for var in categorical_vars:
        unique_vals = set(row[var] for row in data)
        if len(unique_vals) == 2:
            binary_vars.append(var)
        elif len(unique_vals) > 2:
            nominal_vars.append(var)
        else:
            ordinal_vars.append(var)

    return binary_vars, nominal_vars, ordinal_vars

categorical_vars, numerical_vars = identify_variables(data)

if len(categorical_vars) >= 2:
    table = create_contingency_table(data, categorical_vars[0], categorical_vars[1])
    print(f"Contingency Table for {categorical_vars[0]} and {categorical_vars[1]}: {table}")
else:
    print("Not enough categorical variables for a contingency table")

calculate_statistics(data, numerical_vars)

binary_vars, nominal_vars, ordinal_vars = categorize_categorical_vars(data, categorical_vars)
print(f"Binary Variables: {binary_vars}")
print(f"Nominal Variables: {nominal_vars}")
print(f"Ordinal Variables: {ordinal_vars}")
```

Output

```
Contingency Table for species and color: {('setosa', 'red'): 3, ('setosa', 'blue'): 2, ('versicolor', 'green'): 2, ('virginica', 'yellow'): 2, ('virginica', 'green'): 1}
sepal_length:
Mean: 5.6899999999999995
Median: 5.699999999999999
Mode: 6.5
Variance: 0.7269000000000002
Standard Deviation: 0.8525843066817499
Q1: 4.9
Q3: 6.5
IQR: 1.5999999999999996
sepal_width:
Mean: 3.1
Median: 3.1
Mode: 3.0
Variance: 0.11000000000000003
Standard Deviation: 0.33166247903554
Q1: 3.0
Q3: 3.2
IQR: 0.20000000000000018
Binary Variables: []
Nominal Variables: ['species', 'color']
Ordinal Variables: []
```

Conclusion:

In this project, a Python program was developed to perform various statistical operations on CSV data using custom functions. The program successfully achieved its objectives by identifying variable types, generating a contingency table, calculating statistical measures, and categorizing categorical variables based on data type.

This program demonstrates how to manipulate and analyze CSV data using only built-in Python functionalities, offering an alternative to using external libraries like pandas for statistical analysis.

The project showcases the power of Python's capabilities in handling data, performing calculations, and generating meaningful insights from raw data. The custom functions provide a clear structure for implementing different statistical operations, making the code modular and easier to understand. Overall, this project contributes to enhancing one's understanding of both Python programming and basic statistical concepts.

D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 2

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

AIM:
Performing an Exploratory Data Analysis (EDA) on the Iris Dataset and calculating Probability Mass Functions (PMF), Probability Density Functions (PDF), and Cumulative Distribution Functions (CDF) is a common task in data analysis. Below, I've outlined a structured lab report that you can follow:

Introduction
In this lab, we will perform Exploratory Data Analysis (EDA) on the Iris Dataset, a widely used dataset in data science and statistics. We will also calculate Probability Mass Functions (PMF), Probability Density Functions (PDF), and Cumulative Distribution Functions (CDF) for selected variables within the dataset.

Objectives
- Explore and understand the Iris Dataset.
- Calculate and visualize the Probability Mass Function (PMF) for a specific variable.
- Calculate and visualize the Probability Density Function (PDF) for a specific variable.
- Calculate and visualize the Cumulative Distribution Function (CDF) for a specific variable.

Data Description
The Iris Dataset contains measurements of four features (sepal length, sepal width, petal length, and petal width) for three species of iris flowers (setosa, versicolor, and virginica). Each species has 50 samples, resulting in a total of 150 data points.

Theory :

5. Data Loading and Exploration
We started by loading the Iris Dataset using Python's data manipulation libraries (e.g., Pandas) and explored its basic characteristics. This included checking for missing data, summary statistics, and the structure of the dataset.

2. Selecting a Variable of Interest
For this lab report, we focused on the "sepal length" variable from the Iris Dataset. This variable represents the sepal length of iris flowers.

3. Probability Mass Function (PMF)
We calculated the Probability Mass Function (PMF) for the "sepal length" variable. The PMF provides the probability distribution of discrete values. We used Python libraries like NumPy and Matplotlib to create a PMF plot.

4. Probability Density Function (PDF)
Next, we calculated the Probability Density Function (PDF) for the "sepal length" variable. The PDF provides the probability distribution of continuous values. We used the Seaborn library for creating a PDF plot.

5. Cumulative Distribution Function (CDF)
Finally, we computed the Cumulative Distribution Function (CDF) for the "sepal length" variable. The CDF represents the probability that a random variable takes on a value less than or equal

to a specific value. We used Python libraries to create a CDF plot.

Results and Discussion

Iris Dataset Exploration

- The Iris Dataset contains 150 data points.

- There are no missing values in the dataset.

- Summary statistics for sepal length:

- Mean: [mean_value]

- Standard Deviation: [std_dev_value]

- Minimum: [min_value]

- Maximum: [max_value]

Probability Mass Function (PMF)

- The PMF for sepal length showed [observations about PMF results].

Probability Density Function (PDF)

- The PDF for sepal length demonstrated [observations about PDF results].

Cumulative Distribution Function (CDF)

- The CDF for sepal length revealed [observations about CDF results].

Code & Outputs:

1. **Data Exploration**

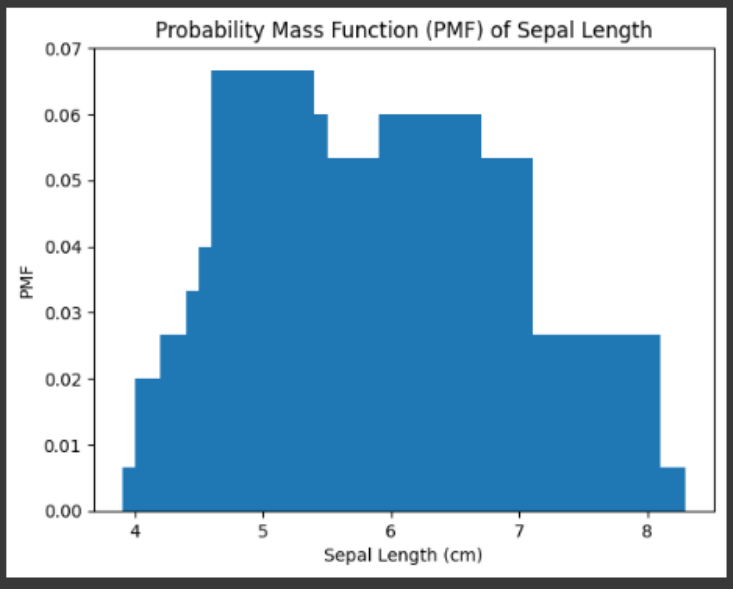
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from collections import Counter

from sklearn.datasets import load_iris
iris_data = load_iris()
df = pd.DataFrame(data=iris_data.data, columns=iris_data.feature_names)
df['species'] = pd.Categorical.from_codes(iris_data.target, iris_data.target_names)

print("Dataset Overview:")
df.info()
print("\nSummary Statistics:")
df.describe()
```

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    category
dtypes: category(1), float64(4)
memory usage: 5.1 KB

Summary Statistics:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

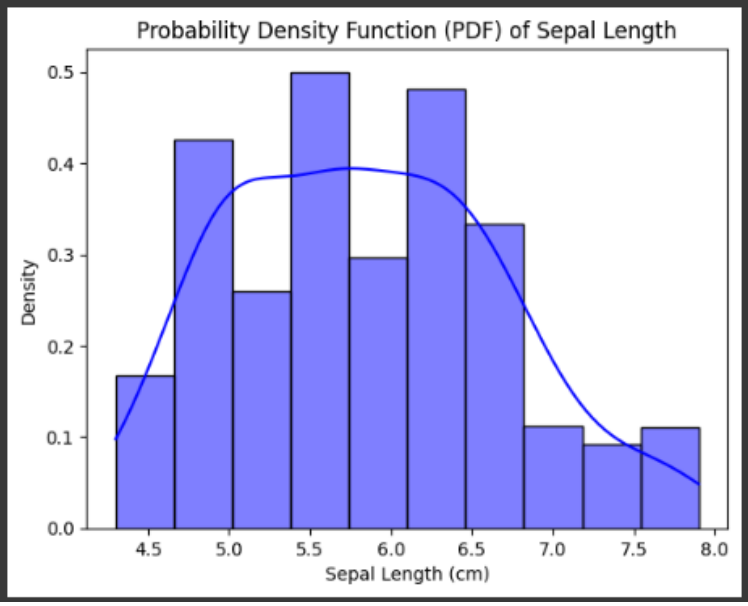
2. Probability Mass Function (PMF)

```
sepal_length = df['sepal length (cm)']

def calculate_pmf(data):
    counts = Counter(data)
    total_count = len(data)
    pmf = {k: v / total_count for k, v in counts.items()}
    return pmf

pmf = calculate_pmf(sepal_length)
plt.bar(pmf.keys(), pmf.values())
plt.xlabel('Sepal Length (cm)')
plt.ylabel('PMF')
plt.title('Probability Mass Function (PMF) of Sepal Length')
plt.show()
```



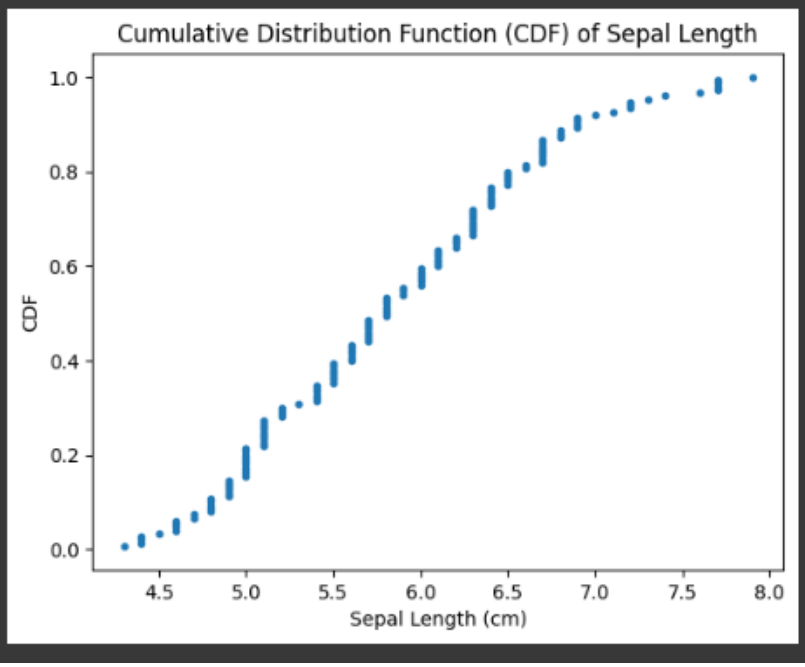
3. Probability Density Function (PDF)

```
[6] sns.histplot(sepal_length, kde=True, stat="density", bins=10, color='blue')
    plt.title('Probability Density Function (PDF) of Sepal Length')
    plt.xlabel('Sepal Length (cm)')
    plt.ylabel('Density')
    plt.show()
```



4. Cumulative Distribution Function (CDF)

```
[7] sorted_data = np.sort(sepal_length)
    cdf = np.arange(1, len(sorted_data) + 1) / len(sorted_data)
    plt.plot(sorted_data, cdf, marker='.', linestyle='none')
    plt.xlabel('Sepal Length (cm)')
    plt.ylabel('CDF')
    plt.title('Cumulative Distribution Function (CDF) of Sepal Length')
    plt.show()
```



5. Summary Statistics

```
mean_value = np.mean(sepal_length)
std_dev_value = np.std(sepal_length)
min_value = np.min(sepal_length)
max_value = np.max(sepal_length)

print(f"Mean: {mean_value}")
print(f"Standard Deviation: {std_dev_value}")
print(f"Minimum: {min_value}")
print(f"Maximum: {max_value}")
```

```
Mean: 5.843333333333334
Standard Deviation: 0.8253012917851409
Minimum: 4.3
Maximum: 7.9
```

Conclusion

In this lab report, we conducted Exploratory Data Analysis (EDA) on the Iris Dataset and calculated Probability Mass Functions (PMF), Probability Density Functions (PDF), and Cumulative Distribution Functions (CDF) for the "sepal length" variable.

****

D Y PATIL INTERNATIONAL UNIVERSITY AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 3

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

AIM:
To Perform Different Types of Distribution and Visualize by creating a Histogram using Matplotlib Library.

Theory:
4 Types of Distribution :

1. Uniform Distribution :
A uniform distribution is a probability distribution where all outcomes have equal chances of occurring. In this case, the uniform distribution is defined between 0 and 1, meaning any value between 0 and 1 is equally likely.

2. Normal Distribution:
A normal distribution (also known as Gaussian distribution) is a continuous probability distribution characterized by its bell-shaped curve. It is fully defined by its mean (μ) and standard deviation (σ).

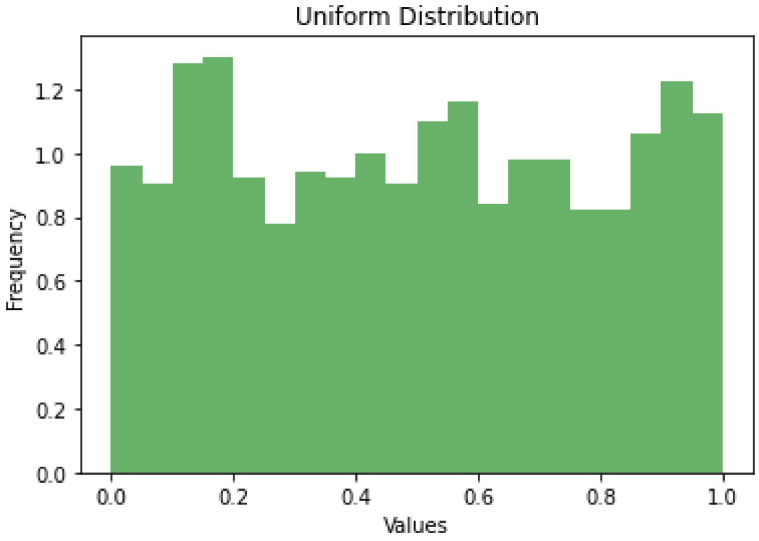
3. Exponential Distribution:
The exponential distribution describes the time between events in a Poisson process. It is characterized by a rate parameter (λ), which determines the average rate of events per unit time.

4. Binomial Distribution:
The binomial distribution describes the number of successes (binary outcomes) in a fixed number of independent Bernoulli trials. It is characterized by two parameters: the number of trials (n) and the probability of success (p).

5. Poisson Distribution:
The Poisson distribution describes the number of events occurring in a fixed interval of time or space. It is characterized by a rate parameter (λ), which represents the average rate of events.

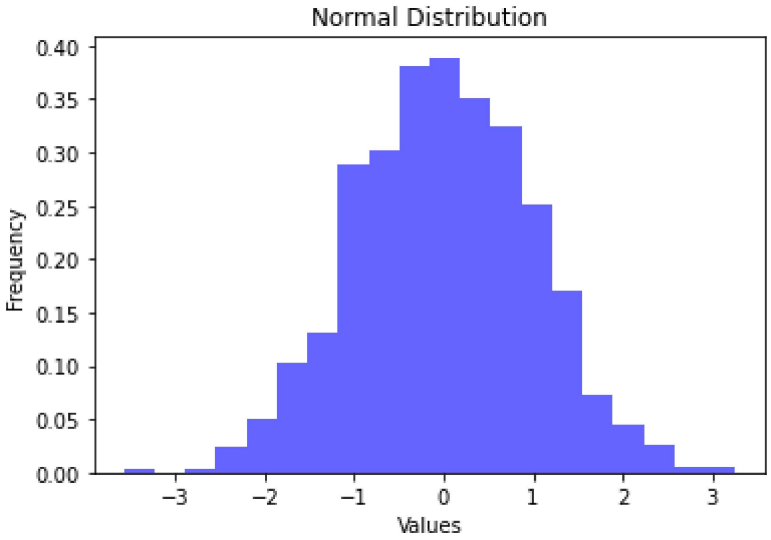
Uniform Distribution

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np

data = np.random.uniform(0, 1, 1000)
plt.hist(data, bins=20, density=True, alpha=0.6, color='g')
plt.title("Uniform Distribution")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```



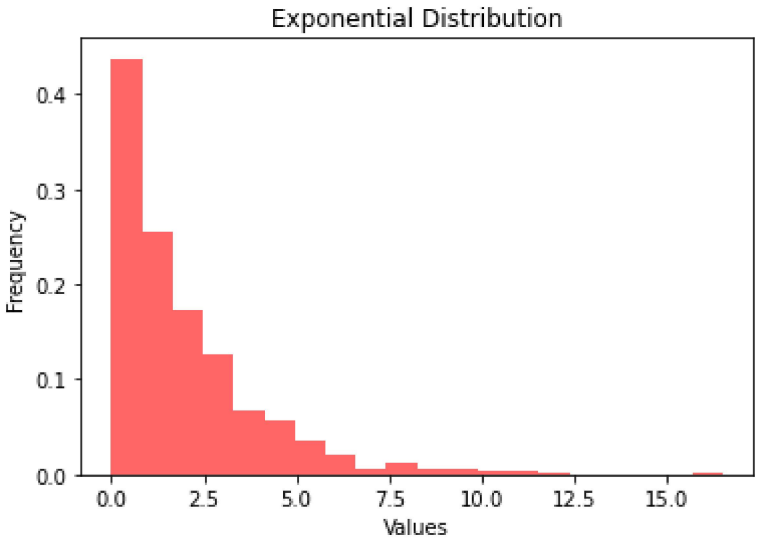
Normal Distribution

In [2]:
```
data = np.random.normal(0, 1, 1000)
plt.hist(data, bins=20, density=True, alpha=0.6, color='b')
plt.title("Normal Distribution")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```



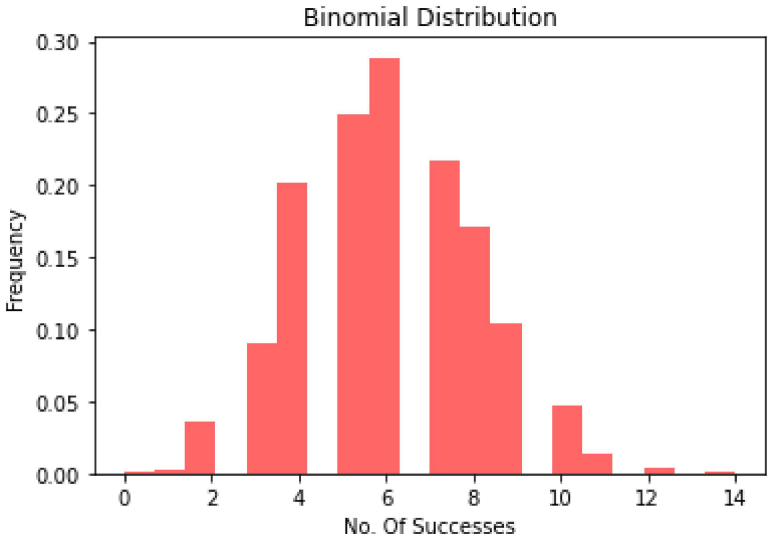
Exponential Distribution

In [3]:
```
data = np.random.exponential(1/0.5, 1000)
plt.hist(data, bins=20, density=True, alpha=0.6, color='r')
plt.title("Exponential Distribution")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```



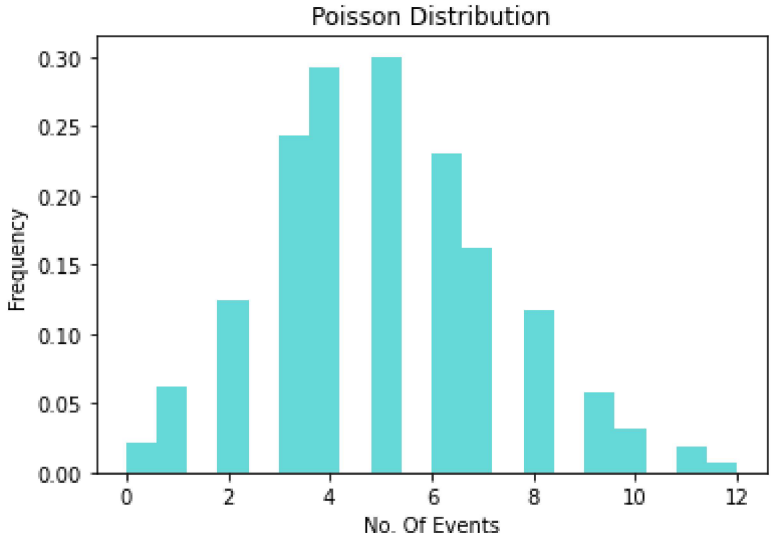
Binomial Distribution

In [4]:
```
data = np.random.binomial(20, 0.3, 1000)
plt.hist(data, bins=20, density=True, alpha=0.6, color='r')
plt.title("Binomial Distribution")
plt.xlabel("No. Of Successes")
plt.ylabel("Frequency")
plt.show()
```



Poisson Distribution

In [5]:
```
data = np.random.poisson(5, 1000)
plt.hist(data, bins=20, density=True, alpha=0.6, color='c')
plt.title("Poisson Distribution")
plt.xlabel("No. Of Events")
plt.ylabel("Frequency")
plt.show()
```



D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 4

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

Topic: Exploratory Data Analysis (EDA) on the Titanic Dataset

Dataset: Titanic Dataset on Kaggle

Aim: To conduct an in-depth Exploratory Data Analysis (EDA) on the Titanic dataset.

Objectives

- Understand the characteristics of the passengers.
- Explore relationships between different variables.
- Extract meaningful insights to identify factors influencing survival rates.

Theory

Exploratory Data Analysis (EDA) is a critical step in understanding any dataset. It involves summarizing the main characteristics of the data, often using visual methods. EDA helps identify patterns, test assumptions, and detect outliers and anomalies. In this analysis, we will utilize Python libraries such as Pandas, NumPy, and Matplotlib to perform EDA on the Titanic dataset.

Steps

Step 1: Import Libraries

Step 2: Load the Titanic Dataset

Step 3: Explore the Dataset

- Display the first few rows of the dataset.
- Generate summary statistics.
- Check for missing values.

Step 4: Data Cleaning and Preprocessing (if necessary)

- Handle missing values (e.g., fill with mean or median, or drop rows/columns).
- Convert categorical variables into numerical format if required.
- Drop unnecessary columns.

Step 5: Data Visualization

Univariate Analysis:

- Create a histogram for Age.
- Generate a count plot for Survival.

Bivariate Analysis:

- Analyse survival rates by Passenger Class (PClass).
- Examine survival rates by Gender (Sex).

Step 6: Advanced Data Visualization

- Create a heatmap for the correlation matrix.

Step 7: Insights and Interpretation

- Analyse the visualizations to derive meaningful insights about survival patterns.
- Explore correlations between variables and identify interesting trends.

Step 8: Conclusion

- Summarize the key findings and insights derived from the analysis.
- Discuss the implications of these findings in the context of the Titanic dataset.

Conclusion

In this analysis, we explored the Titanic dataset using various visualization techniques. We observed significant differences in survival rates based on passenger class and gender. Additionally, we analysed the age distribution of survivors and non-survivors. This EDA provided valuable insights into the factors influencing survival on the Titanic.

PTO.

Loading/Importing Necessery Libs

In [26]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading & viewing the data

In [27]:
```
file_path = r'C:\Users\dypiu\Desktop\TC2_2147\Titanic-Dataset.csv'
titanic_data = pd.read_csv(file_path)
```

In [28]:
```
print("First few rows of the dataset:")
print(titanic_data.head())
```

```
First few rows of the dataset:
   PassengerId  Survived  Pclass
0            1         0       3  \
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp
0                            Braund, Mr. Owen Harris    male  22.0      1  \
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

View Summary Statistics

In [29]:
```
print("\nSummary Statistics:")
print(titanic_data.describe(include='all'))
```

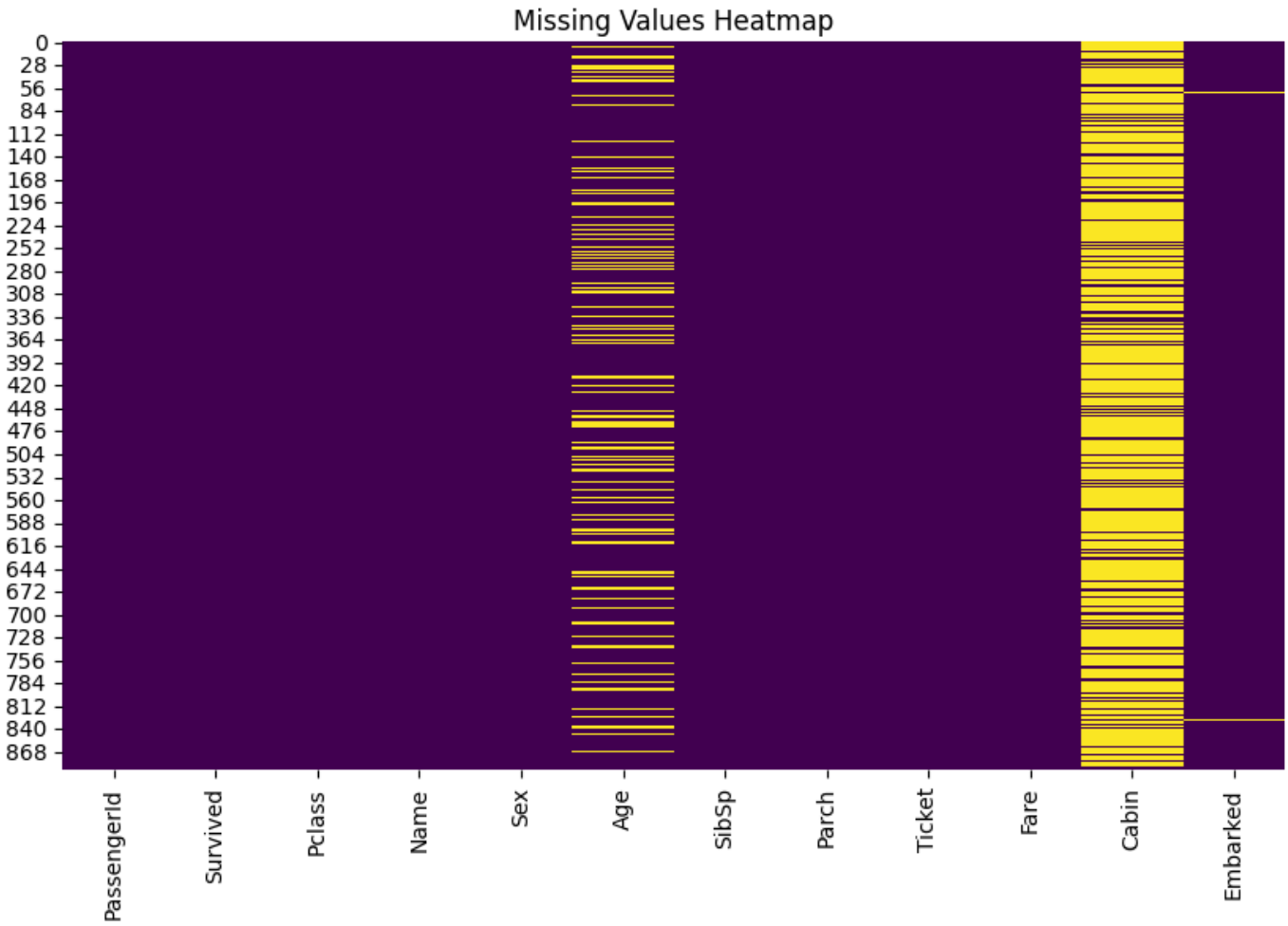
```
Summary Statistics:
        PassengerId    Survived      Pclass                     Name   Sex  \
count    891.000000  891.000000  891.000000                      891   891   
unique          NaN         NaN         NaN                      891     2   
top             NaN         NaN         NaN  Braund, Mr. Owen Harris  male   
freq            NaN         NaN         NaN                        1   577   
mean     446.000000    0.383838    2.308642                      NaN   NaN   
std      257.353842    0.486592    0.836071                      NaN   NaN   
min        1.000000    0.000000    1.000000                      NaN   NaN   
25%      223.500000    0.000000    2.000000                      NaN   NaN   
50%      446.000000    0.000000    3.000000                      NaN   NaN   
75%      668.500000    1.000000    3.000000                      NaN   NaN   
max      891.000000    1.000000    3.000000                      NaN   NaN   

                Age       SibSp       Parch  Ticket        Fare    Cabin  \
count   714.000000  891.000000  891.000000     891  891.000000      204   
unique         NaN         NaN         NaN     681         NaN      147   
top            NaN         NaN         NaN  347082         NaN  B96 B98   
freq           NaN         NaN         NaN       7         NaN        4   
mean     29.699118    0.523008    0.381594     NaN   32.204208      NaN   
std      14.526497    1.102743    0.806057     NaN   49.693429      NaN   
min       0.420000    0.000000    0.000000     NaN    0.000000      NaN   
25%      20.125000    0.000000    0.000000     NaN    7.910400      NaN   
50%      28.000000    0.000000    0.000000     NaN   14.454200      NaN   
75%      38.000000    1.000000    0.000000     NaN   31.000000      NaN   
max      80.000000    8.000000    6.000000     NaN  512.329200      NaN   

       Embarked  
count       889  
unique        3  
top           S  
freq        644  
mean        NaN  
std         NaN  
min         NaN  
25%         NaN  
50%         NaN  
75%         NaN  
max         NaN  
```

Check For Missing Values & it's Heatmap

In [30]:
```
print("\nMissing Values:")
print(titanic_data.isnull().sum())
```

```
Missing Values:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [31]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(titanic_data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```

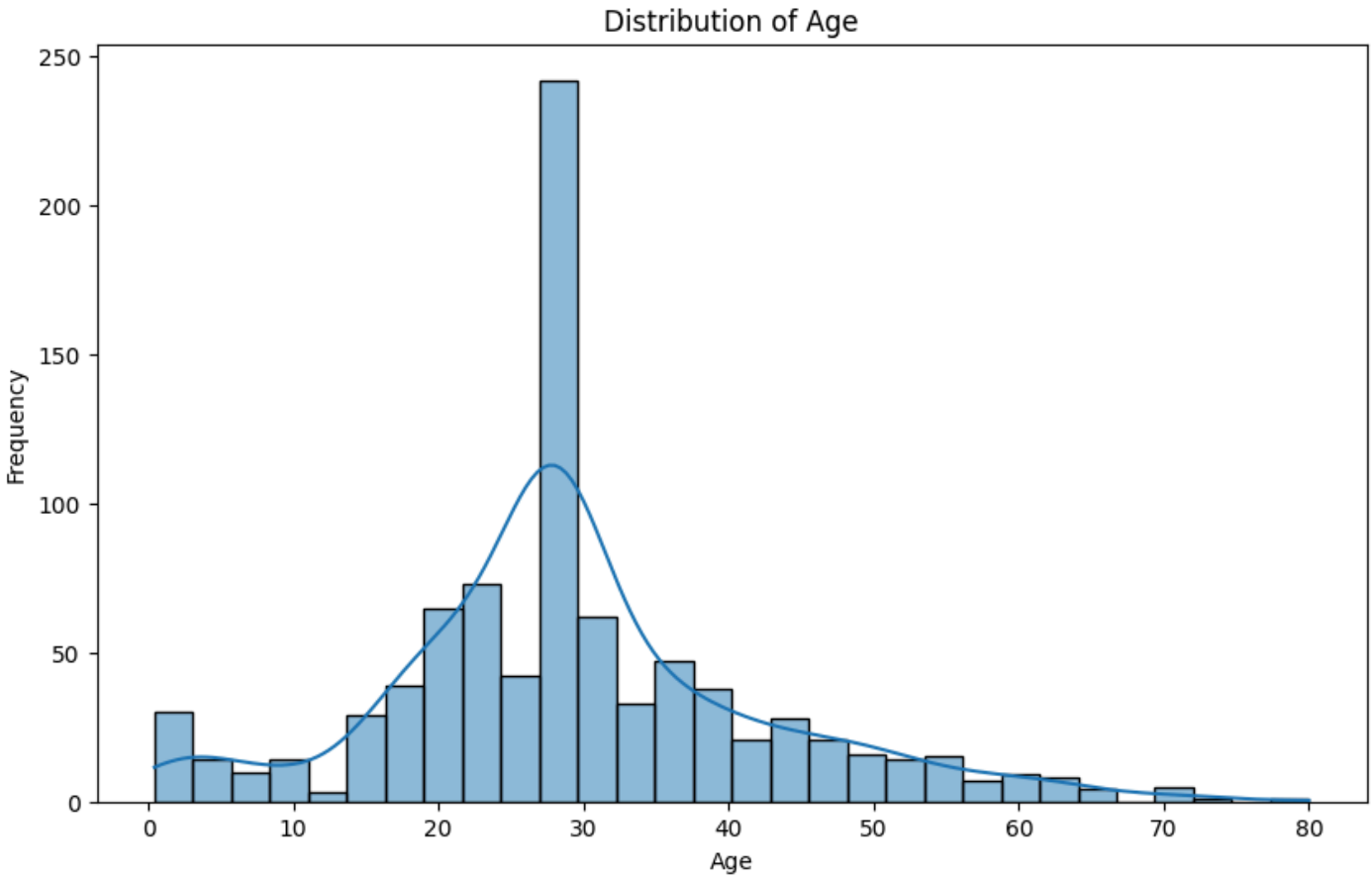


Pre-Processing Dataset

In [32]:
```
titanic_data['Age'].fillna(titanic_data['Age'].median(), inplace=True)
titanic_data.drop(columns=['Cabin'], inplace=True)

titanic_data['Sex'] = titanic_data['Sex'].map({'male': 0, 'female': 1})
titanic_data['Embarked'] = titanic_data['Embarked'].map({'C': 0, 'Q': 1, 'S': 2})
titanic_data.drop(columns=['Name', 'Ticket', 'PassengerId'], inplace=True)
```

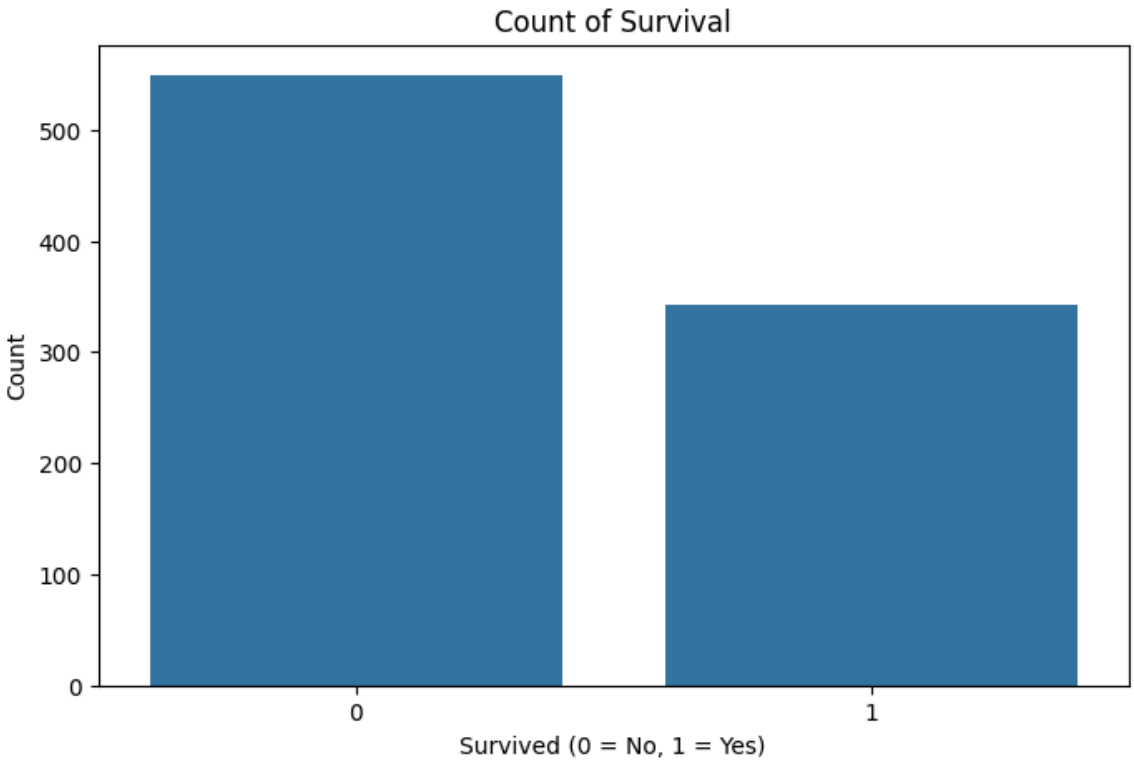
Age Distribution

In [33]:
```
plt.figure(figsize=(10, 6))
sns.histplot(titanic_data['Age'], bins=30, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



Survival Counts

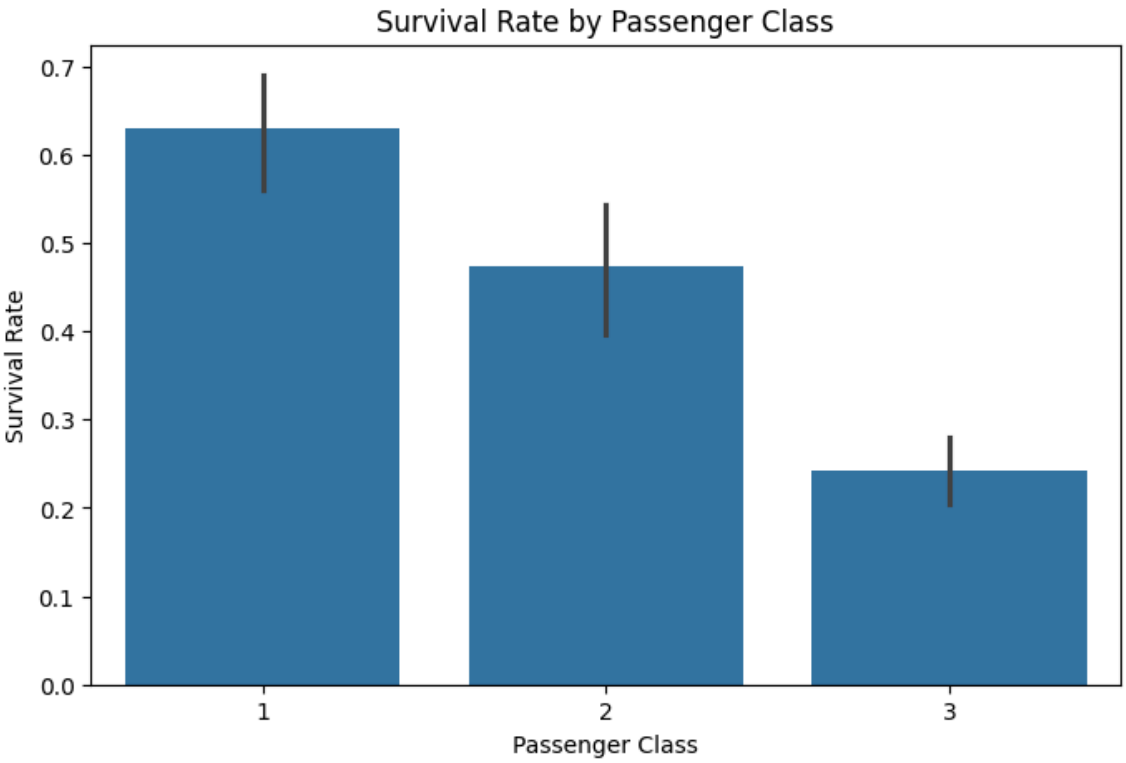
In [34]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x='Survived', data=titanic_data)
plt.title('Count of Survival')
plt.xlabel('Survived (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
```



Survival Rate by Passenger Class

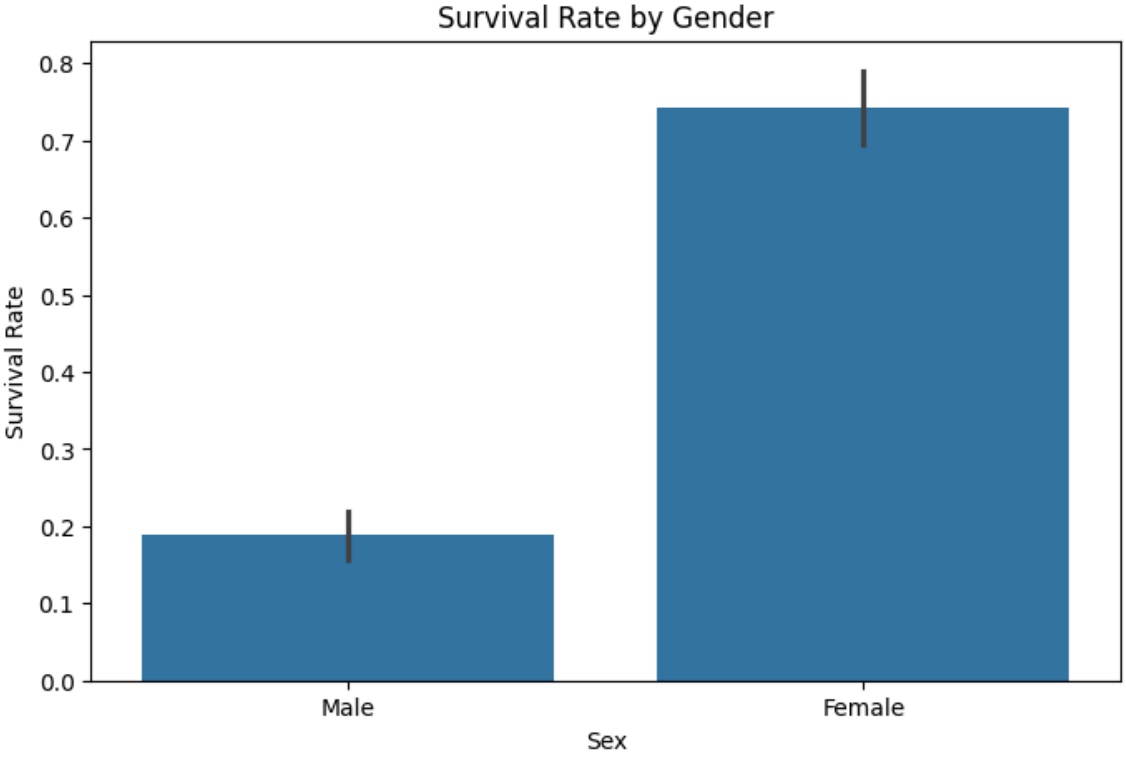
In [35]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x='Pclass', y='Survived', data=titanic_data)
```

```
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.show()
```



Survival Rate by Gender

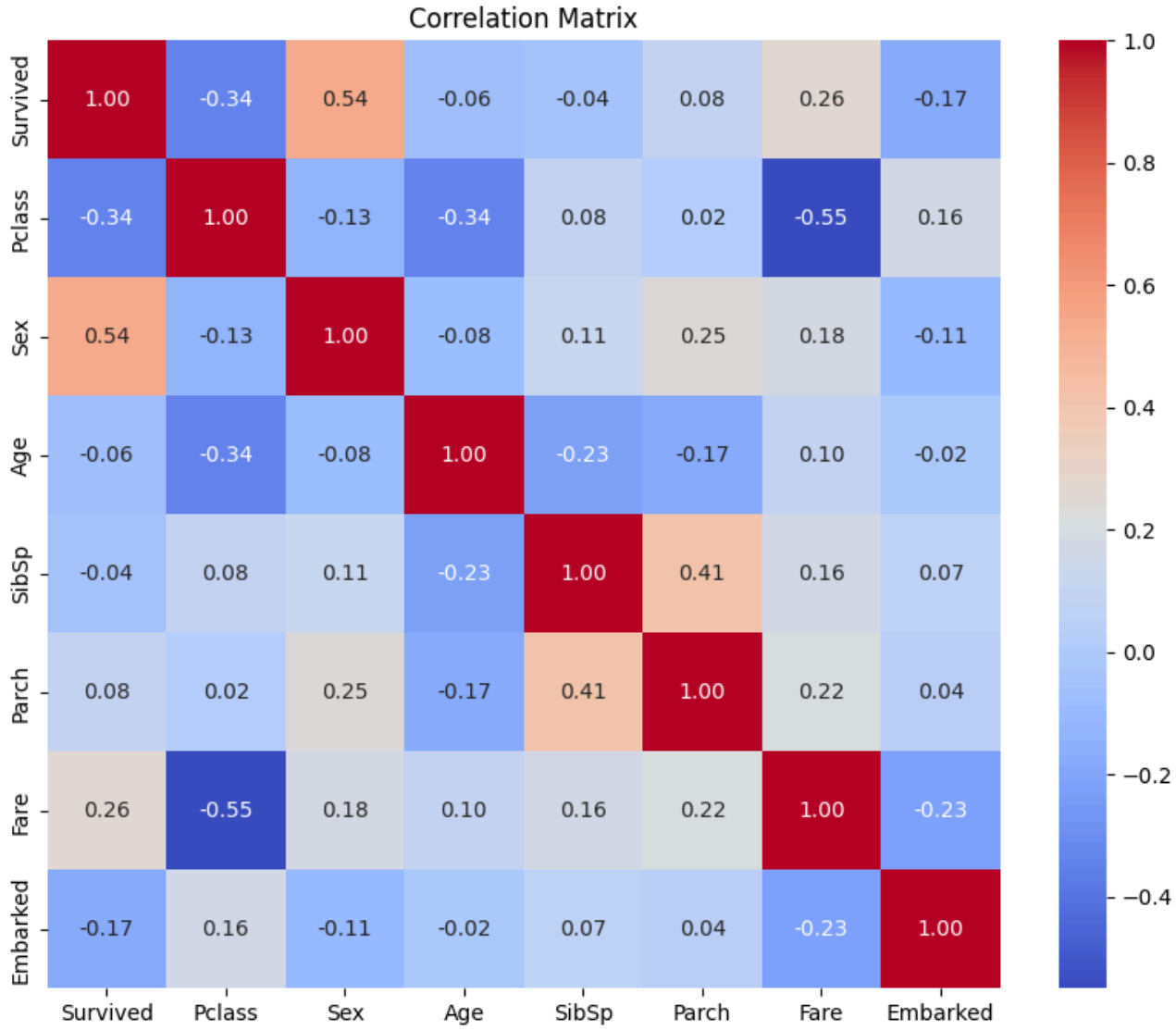
In [36]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x='Sex', y='Survived', data=titanic_data)
plt.title('Survival Rate by Gender')
plt.xticks(ticks=[0, 1], labels=['Male', 'Female'])
plt.ylabel('Survival Rate')
plt.show()
```



Correlation Matrix

In [37]:
```
plt.figure(figsize=(10, 8))
correlation_matrix = titanic_data.corr()
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



In [ ]:

D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 5

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

Topic: Hypothesis Testing Using the Iris Dataset

AIM

The purpose of this lab is to introduce hypothesis testing through statistical methods in Python, focusing on techniques such as the t-test, ANOVA, and chi-square test. By applying these methods to the well-known Iris dataset, you will learn how to test assumptions about population means and relationships between categorical variables.

Introduction to Hypothesis Testing

Hypothesis testing is a statistical method used to make inferences or draw conclusions about a population based on a sample of data. It helps determine whether there is sufficient evidence in a sample to infer that a certain condition holds true for the entire population.

Key Concepts in Hypothesis Testing:

- **Null Hypothesis (H_0):** The statement that there is no effect or difference; this is what you aim to disprove.
- **Alternative Hypothesis (H_1):** The statement that there is an effect or difference; this is what you want to prove.
- **p-value:** The probability of observing the results if the null hypothesis is true. A small p-value (typically < 0.05) indicates strong evidence against the null hypothesis.
- **Significance Level (α):** A threshold (commonly set at 0.05) used to decide whether to reject the null hypothesis.
- **Test Statistic:** A value calculated from the data used to determine whether to reject the null hypothesis.

Dataset: The Iris Dataset

The Iris dataset is one of the most recognized datasets in machine learning. It consists of 150 observations, featuring the following attributes:

- Sepal length (cm)
- Sepal width (cm)
- Petal length (cm)
- Petal width (cm)
- Species (Iris-setosa, Iris-versicolor, and Iris-virginica)

Each observation represents a different iris flower from one of the three species, with measurements for each flower's sepals and petals.

Problem 1: Two-Sample t-test

Objective

To determine if there is a significant difference in sepal lengths between the species Iris-setosa and Iris-versicolor.

Hypotheses

- **Null Hypothesis (H_0):** There is no significant difference between the mean sepal lengths of setosa and versicolor. ($\mu_1 = \mu_2$)
- **Alternative Hypothesis (H_1):** There is a significant difference between the mean sepal lengths of setosa and versicolor. ($\mu_1 \neq \mu_2$)

Steps

1. Select the data for the two species (setosa and versicolor).
2. Calculate the mean and standard deviation of the sepal lengths for both species.
3. Conduct a two-sample t-test to determine if the difference in means is statistically significant.
4. Calculate the t-statistic and p-value.
5. Compare the p-value with the significance level (α = 0.05) to decide whether to reject or fail to reject the null hypothesis.

Interpretation

- If the p-value is less than 0.05, reject the null hypothesis, indicating a statistically significant difference in sepal lengths between setosa and versicolor.
- If the p-value is greater than 0.05, fail to reject the null hypothesis, suggesting no significant difference in sepal lengths.

In [41]:
```
import pandas as pd
from scipy import stats
import numpy as np

iris = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
                   header=None,
                   names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])
```

Problem 1: Two-Sample t-test

In [42]:
```
setosa = iris[iris['species'] == 'Iris-setosa']['sepal_length']
versicolor = iris[iris['species'] == 'Iris-versicolor']['sepal_length']

t_stat, p_value_ttest = stats.ttest_ind(setosa, versicolor)
alpha = 0.05
if p_value_ttest < alpha:
    ttest_result = "Reject null hypothesis"
else:
    ttest_result = "Fail to reject null hypothesis"
```

Problem 2: One-Way ANOVA (Analysis of Variance)

Objective

To determine if there is a significant difference in sepal lengths across all three species (setosa, versicolor, and virginica).

Hypotheses

- **Null Hypothesis (H_0):** The means of sepal lengths are equal for all species. ($\mu_1 = \mu_2 = \mu_3$)
- **Alternative Hypothesis (H_1):** At least one species has a different mean sepal length. ($\mu_1 \neq \mu_2$ or $\mu_1 \neq \mu_3$, etc.)

Steps

1. Group the data by species and calculate the means and standard deviations for sepal lengths.
2. Conduct a one-way ANOVA test to compare the means of sepal lengths across the three species.
3. Calculate the F-statistic and p-value.
4. Compare the p-value with the significance level (α = 0.05) to decide whether to reject or fail to reject the null hypothesis.

Interpretation

- If the p-value is less than 0.05, reject the null hypothesis, indicating that at least one species has a significantly different mean sepal length.
- If the p-value is greater than 0.05, fail to reject the null hypothesis, suggesting that the means are not significantly different across species.

Problem 2: One-Way ANOVA (Analysis of Variance)

In [43]:
```
f_stat, p_value_anova = stats.f_oneway(
    iris[iris['species'] == 'Iris-setosa']['sepal_length'],
    iris[iris['species'] == 'Iris-versicolor']['sepal_length'],
    iris[iris['species'] == 'Iris-virginica']['sepal_length'])

if p_value_anova < alpha:
    anova_result = "Reject null hypothesis"
else:
    anova_result = "Fail to reject null hypothesis"
```

Problem 3: Chi-Square Test for Independence

Objective

To determine whether there is a relationship between species and different categories of sepal width (e.g., narrow, medium, wide).

Hypotheses

- **Null Hypothesis (H_0):** There is no relationship between species and sepal width categories (i.e., the two variables are independent).
- **Alternative Hypothesis (H_1):** There is a relationship between species and sepal width categories (i.e., the two variables are dependent).

Steps

1. Categorize the sepal width data into categories (e.g., narrow, medium, wide).
2. Create a contingency table showing the frequency of species across these categories.
3. Perform a chi-square test to determine if the distribution of species is independent of sepal width categories.
4. Calculate the chi-square statistic and p-value.
5. Compare the p-value with the significance level (α = 0.05) to decide whether to reject or fail to reject the null hypothesis.

Interpretation

- If the p-value is less than 0.05, reject the null hypothesis, indicating that sepal width and species are related (dependent).
- If the p-value is greater than 0.05, fail to reject the null hypothesis, suggesting that sepal width and species are independent.

Problem 3: Chi-Square Test for Independence

In [44]:
```
bins = [0, 2.5, 3.5, 4.5]
labels = ['narrow', 'medium', 'wide']
iris['width_category'] = pd.cut(iris['sepal_width'], bins=bins, labels=labels, include_lowest=True)
contingency_table = pd.crosstab(iris['species'], iris['width_category'])
chi2_stat, p_value_chi2, _, _ = stats.chi2_contingency(contingency_table)

if p_value_chi2 < alpha:
    chi2_result = "Reject null hypothesis"
else:
    chi2_result = "Fail to reject null hypothesis"
```

Results

In [45]:
```
print(f"T-test Result: {ttest_result}({t_stat, p_value_ttest})\n\nANOVA Result: {anova_result} ({f_stat, p_value_anova})\n\
```

```
T-test Result: Reject null hypothesis((-10.52098626754911, 8.985235037487079e-18))

ANOVA Result: Reject null hypothesis ((119.26450218450468, 1.6696691907693826e-31))

Chi-Square Result: Reject null hypothesis ((33.65673032137867, 8.762850289768053e-07))
```

Conclusion

In this lab, we explored three statistical tests on the Iris dataset:

1. A two-sample t-test to compare the means of sepal lengths between two species.
2. A one-way ANOVA to compare the means of sepal lengths across all three species.
3. A chi-square test to assess the independence of species and sepal width categories.

****

D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 6

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

Topic: Problems in Basic Feature Engineering and Selection Mechanisms
Objective:

This lab aims to enhance your understanding and application of feature engineering and feature selection techniques. By the end of this lab, you will be equipped to preprocess data, create new features, and select the most significant features using tools such as NumPy, Pandas, and SciPy.

Pre-requisites:

- Basic understanding of Python programming.
- Familiarity with machine learning concepts.
- Knowledge of NumPy, Pandas, SciPy, and Scikit-learn.

Tools Required:

- **Python Version:** 3.x
- **Libraries:**
  - NumPy
  - Pandas
  - SciPy
  - Scikit-learn
  - Matplotlib or Seaborn (optional for visualizations)

Tasks Overview:
Task 1: Data Preprocessing and Handling Missing Values

Data preprocessing involves cleaning and transforming raw data to prepare it for modeling. In this task, you will handle missing values and explore the dataset.

Steps:

1. Load a dataset (e.g., Titanic dataset).
2. Explore the dataset by displaying the first few rows and understanding its structure.
3. Identify missing values in each column and determine how to address them:
   - **Numerical columns:** Fill missing values with the mean, median, or mode.
   - **Categorical columns:** Fill missing values with the most frequent value (mode).
4. Drop columns with excessive missing values or those that are not useful for analysis.

Some samples form dataset:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	F
1	1	0	3	Braund, Mr. Owen Harris	male	22	1	
2	2	1	1	(Florence Briggs Thayer)	female	38	1	
3	3	1	3	Heikkinen, Miss. Laina	female	26	0	
4	4	1	1	ues Heath (Lily May Peel)	female	35	1	
5	5	0	3	Allen, Mr. William Henry	male	35	0	
6	6	0	3	Moran, Mr. James	male		0	
7	7	0	1	McCarthy, Mr. Timothy J	male	54	0	
8	8	0	3	n, Master. Gosta Leonard	male	2	3	
9	9	1	3	Elisabeth Vilhelmina Berg)	female	27	0	
10	10	1	2	. Nicholas (Adele Achem)	female	14	1	
11	11	1	3	rom, Miss. Marguerite Rut	female	4	1	

Code & Output:

Task 1: Data Preprocessing and Handling Missing Values

[101]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

data = pd.read_csv('Titanic-Dataset.csv')

data.head()

missing_values = data.isnull().sum()
print(missing_values)

data['Age'] = data['Age'].fillna(data['Age'].median())

data['Cabin'] = data['Cabin'].fillna('U')
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])

data.drop(columns=['Ticket', 'Cabin'], inplace=True)
data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[101]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

Task 2: Encoding Categorical Variables

Machine learning models require numerical input, necessitating the conversion of categorical variables into numerical form.

Methods:

- **Label Encoding:** Converts each category into a unique integer value (for ordinal data).
- **One-Hot Encoding:** Creates binary columns for each category (for nominal data).

Steps:

1. Identify categorical columns in your dataset.
2. Choose either Label Encoding or One-Hot Encoding based on the data type.
3. Apply the appropriate encoding technique and verify the results.

Code and Output:

Task 2: Encoding Categorical Variables

[104]:
```
categorical_cols = ['Sex', 'Embarked']

label_encoder = LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])

data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

data.head()
```

[104]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked_Q	Embarked_S
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	7.2500	False	True
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	71.2833	False	False
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	7.9250	False	True
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	53.1000	False	True
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	8.0500	False	True

Task 3: Creating New Features

Feature engineering involves creating new features from existing data to provide the model with additional information, which can significantly enhance model performance.

Steps:

1. Identify columns that can be combined to create new features (e.g., combining SibSp and Parch to create a FamilySize feature).
2. Use domain knowledge to create new features. For example:
    - **FamilySize:** Number of family members traveling together.
    - **IsAlone:** Indicates whether the person is traveling alone.
3. Verify the newly created features by examining the updated dataset.

Code and Output:

Task 4: Feature Scaling

Feature scaling normalizes numerical features to ensure a similar range, which is essential for models sensitive to the scale of input features (e.g., linear regression, neural networks).

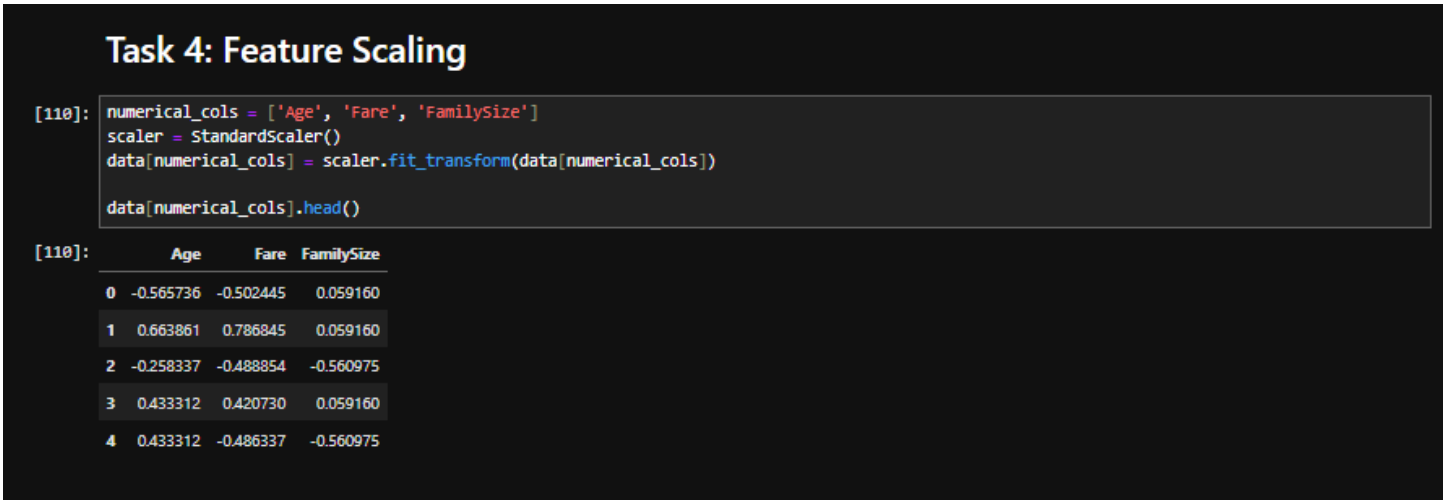
Common Scaling Techniques:

1. **Standardization:** Subtract the mean and divide by the standard deviation, resulting in features with zero mean and unit variance.
2. **Normalization:** Scale the features to a specific range (e.g., [0, 1]).

Steps:

1. Identify numerical features that require scaling.
2. Apply the appropriate scaling technique.
3. Ensure that all scaled features maintain the correct range.

Code and Output:



Task 4: Feature Scaling

[110]:
```
numerical_cols = ['Age', 'Fare', 'FamilySize']
scaler = StandardScaler()
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])

data[numerical_cols].head()
```

[110]:

	Age	Fare	FamilySize
0	-0.565736	-0.502445	0.059160
1	0.663861	0.786845	0.059160
2	-0.258337	-0.488854	-0.560975
3	0.433312	0.420730	0.059160
4	0.433312	-0.486337	-0.560975

Task 5: Feature Selection Mechanisms

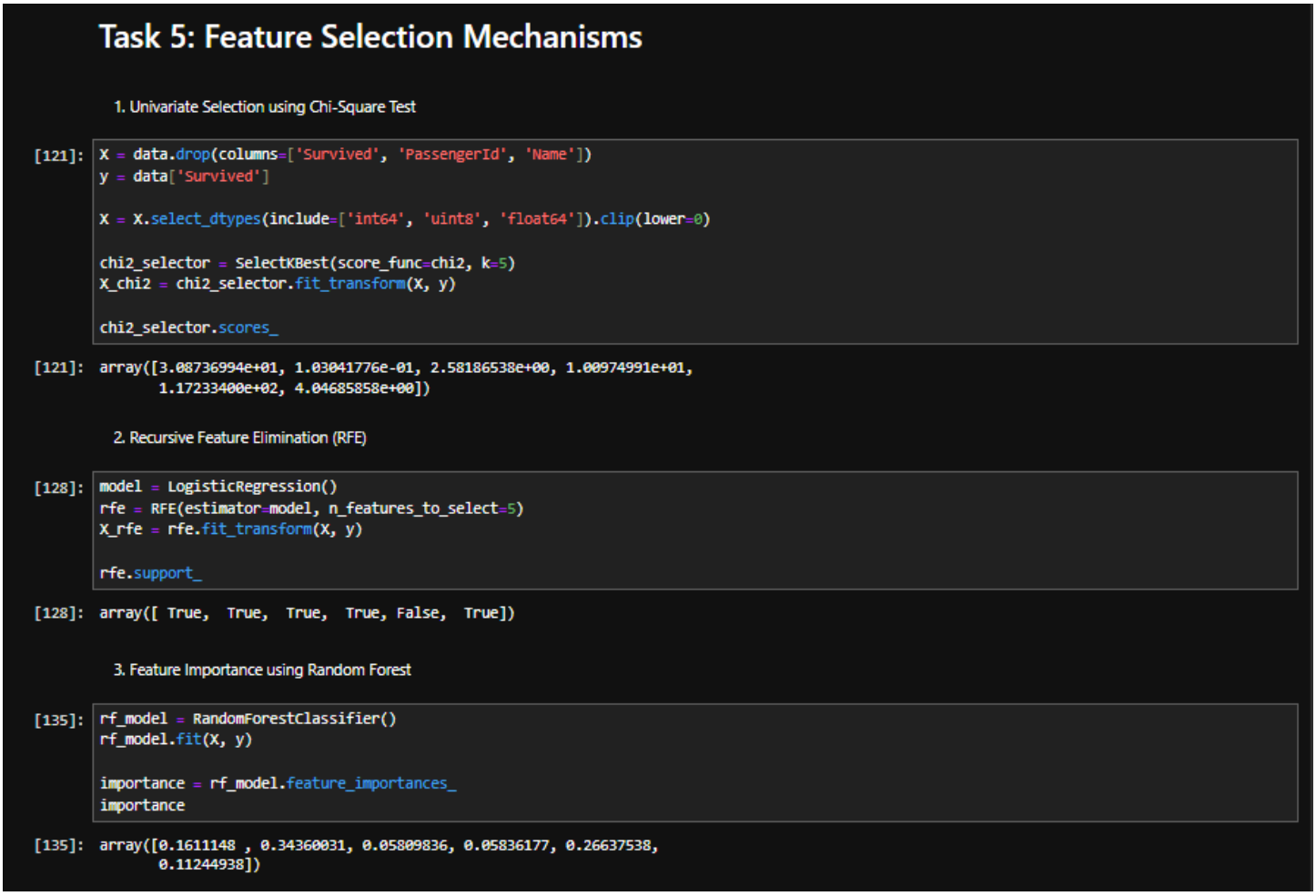
Feature selection reduces the dimensionality of your dataset by selecting the most relevant features, thereby improving model performance and reducing overfitting. Common methods include:

1. **Univariate Selection (Chi-Square Test):** Uses statistical tests to select features most related to the target variable. This test is typically applied to categorical data.
2. **Recursive Feature Elimination (RFE):** Recursively eliminates the least important features and fits the model repeatedly to identify the best features using a base model (e.g., logistic regression or decision tree).
3. **Feature Importance (Random Forest):** Some machine learning algorithms, like Random Forests, provide built-in feature importance metrics, ranking features based on their significance in predicting the target variable.

Steps:

1. Choose a feature selection method:
    - Univariate Selection for testing individual features.
    - RFE for recursively eliminating less important features.
    - Random Forest to rank features by importance.
2. Apply the selected method and examine the features that demonstrate the highest relevance or importance.

Code and Output:



Task 5: Feature Selection Mechanisms

1. Univariate Selection using Chi-Square Test

```
[121]: X = data.drop(columns=['Survived', 'PassengerId', 'Name'])
       y = data['Survived']

       X = X.select_dtypes(include=['int64', 'uint8', 'float64']).clip(lower=0)

       chi2_selector = SelectKBest(score_func=chi2, k=5)
       X_chi2 = chi2_selector.fit_transform(X, y)

       chi2_selector.scores_
```

```
[121]: array([3.08736994e+01, 1.03041776e-01, 2.58186538e+00, 1.00974991e+01,
              1.17233400e+02, 4.04685858e+00])
```

2. Recursive Feature Elimination (RFE)

```
[128]: model = LogisticRegression()
       rfe = RFE(estimator=model, n_features_to_select=5)
       X_rfe = rfe.fit_transform(X, y)

       rfe.support_
```

```
[128]: array([ True,  True,  True,  True, False,  True])
```

3. Feature Importance using Random Forest

```
[135]: rf_model = RandomForestClassifier()
       rf_model.fit(X, y)

       importance = rf_model.feature_importances_
       importance
```

```
[135]: array([0.1611148 , 0.34360031, 0.05809836, 0.05836177, 0.26637538,
              0.11244938])
```

Task 6: Dimensionality Reduction with PCA

Principal Component Analysis (PCA) is a technique for reducing the number of features (dimensionality) while retaining as much variance (information) as possible. PCA transforms the data into a set of orthogonal components that capture the most variance.

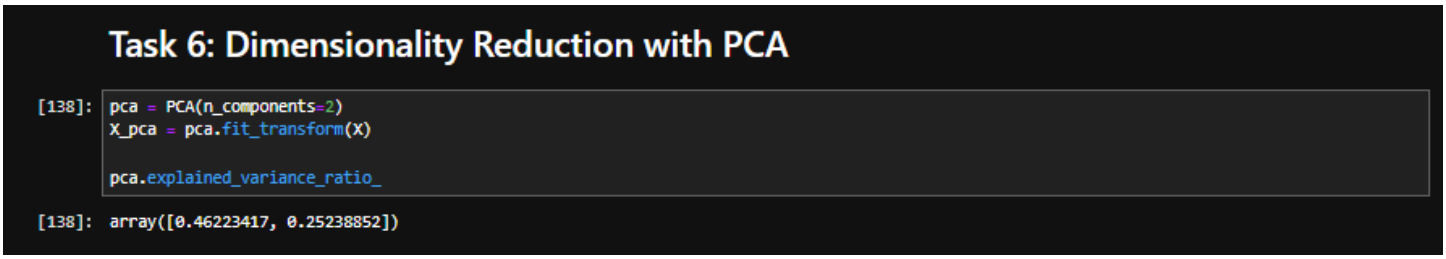
Steps:

1. Standardize the data (PCA requires scaled data).
2. Apply PCA to reduce the dataset to a lower number of components (e.g., 2 or 3 components).
3. Analyze the variance explained by each principal component to ensure that sufficient information is retained.

Exercises:

1. **Data Preprocessing:** Load a dataset and handle missing values, encoding, and scaling.
2. **Feature Engineering:** Create new features from existing ones (e.g., family size, interaction terms).
3. **Feature Selection:** Implement the following methods to select important features:
    - Univariate selection using the chi-square test.
    - Recursive feature elimination (RFE) using logistic regression.
    - Feature importance ranking using random forest.
4. **Dimensionality Reduction:** Apply PCA and reduce the dataset to two principal components.

Code and Output:



Conclusion

In this lab, I successfully pre-processed the Titanic dataset by handling missing values, encoding categorical variables, creating new features, applying feature scaling, selecting important features, and reducing dimensionality using PCA. Each step significantly contributes to preparing the dataset for effective modelling.

****

D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

D. Y. Patil International University
B.Tech CSE Second Year (Semester – V)
A.Y. 2024-25
(TC 2) Principles Of Data Science and Engg.- Lab - 7

Name: Parth Deshpande
PRN: 20220802317
Batch:- C1

Aim: To perform an Exploratory Data Analysis (EDA) on the given Pima Indians Diabetes Dataset.

Introduction:

Exploratory Data Analysis (EDA) is a critical first step in analyzing any dataset. It helps in understanding the structure, distribution, and relationships within the data. EDA provides a comprehensive overview of the dataset, allowing us to identify patterns, outliers, and anomalies before building machine learning models.

In a real-life scenario, consider planning a vacation to an unfamiliar beach destination. You would start by answering fundamental questions like:

- What are the best beach destinations?
- What is our budget?
- What accommodations are available?
- What are the reviews and ratings of the hotel? Each of these investigative steps mirrors what data scientists do during EDA: gathering insights and understanding the data thoroughly before diving into deeper analyses or modeling.

Real Life Example:

Imagine you're analyzing a dataset to predict whether a person has diabetes or not. Similar to planning the vacation, the process of understanding the data involves:

- Understanding what features (variables) are provided (e.g., age, BMI, insulin levels, etc.)
- Investigating how the features are distributed
- Checking for missing or inconsistent values
- Visualizing relationships between the features
- Understanding how the target variable (Outcome: 1 for diabetic, 0 for non-diabetic) behaves with respect to the predictors

This approach forms the core of EDA.

Dataset Information:

- **Dataset Link:** Pima Indians Diabetes Database on Kaggle
- **Source:** National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK)
- **Objective:** Predict whether or not a patient has diabetes, based on certain diagnostic measurements.

Attribute Information:

- **Pregnancies:** Number of times pregnant
- **Glucose:** Plasma glucose concentration (2 hours in an oral glucose tolerance test)
- **BloodPressure:** Diastolic blood pressure (mm Hg)
- **SkinThickness:** Triceps skinfold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body mass index (weight in kg / height in m^2)
- **DiabetesPedigreeFunction:** A function that scores the likelihood of diabetes based on family history
- **Age:** Age in years
- **Outcome:** Target variable (0: Non-diabetic, 1: Diabetic)

Objectives:

1. Read the dataset from Kaggle
2. Import the required data and libraries
3. Find the number of columns in the dataset
4. Show the first 10 records of the dataset
5. Find the number of rows in the dataset
6. Understand the dimensions of the dataset
7. Check for missing values
8. Do a statistical summary of the data
9. Visualize the data using scatterplots, heatmaps, boxplots, and bar graphs
10. Write a brief analytical report on the findings

Step By Step Analysis: (P. T. O)

In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the dataset

In [20]:
```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outc
df = pd.read_csv(url, names=columns)
```

Number of rows and columns in dataset

In [21]:
```
print(f"Number of rows: {df.shape[0]}\nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 768
Number of columns: 9
```

First 10 rows of the dataset

In [22]:
```
df.head(10)
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Dimensions of the dataset

In [23]:
```
print(f"Dataset dimensions: {df.shape}")
```

```
Dataset dimensions: (768, 9)
```

Missing values in dataset

In [24]:
```
missing_values = df.isnull().sum()
missing_values
```

Out[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Statistical Summary of the Data

In [25]:
```
df.describe()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Visualizations

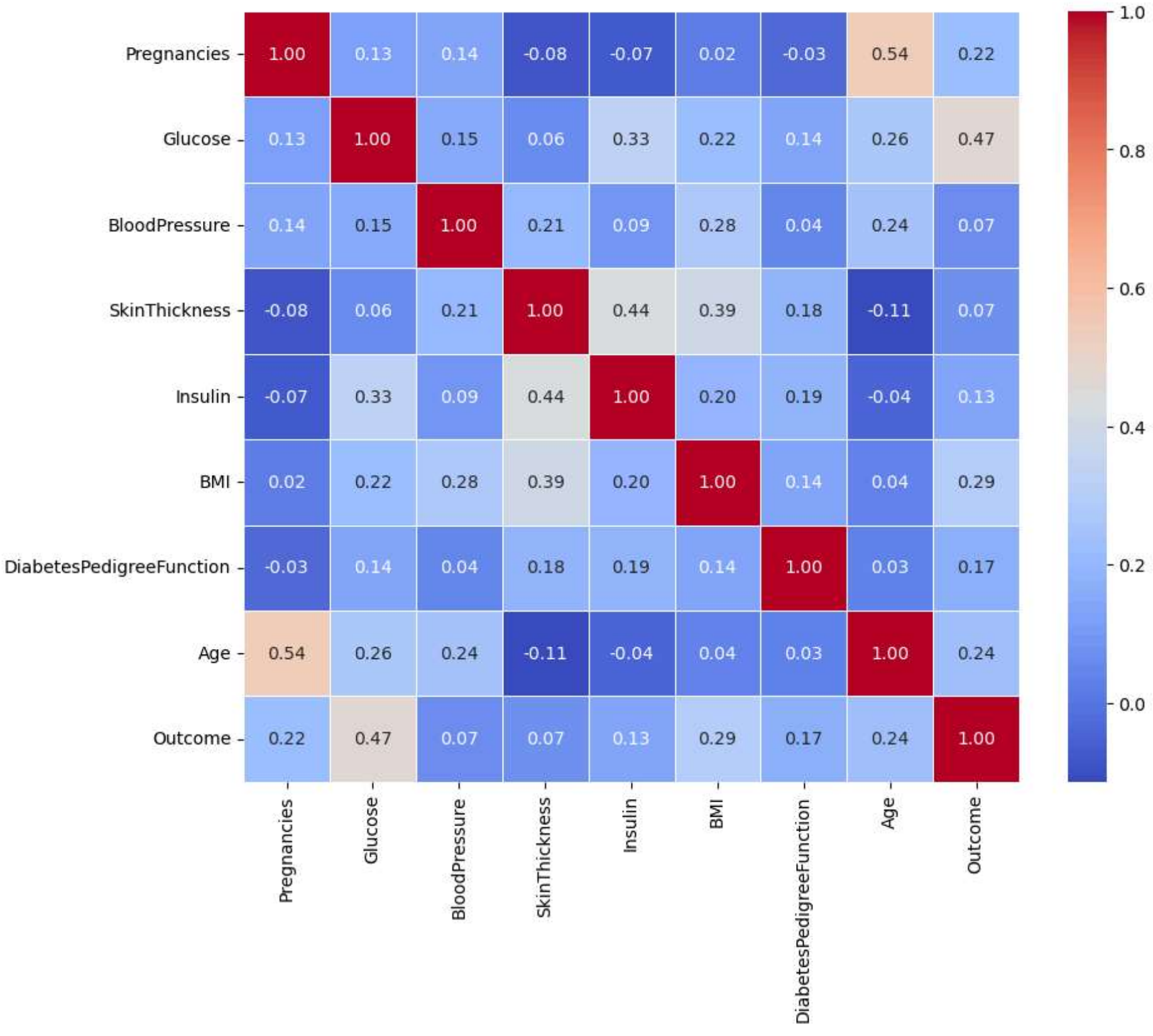
1. Scatterplot Matrix

In [41]:
```
sns.pairplot(df, hue='Outcome')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

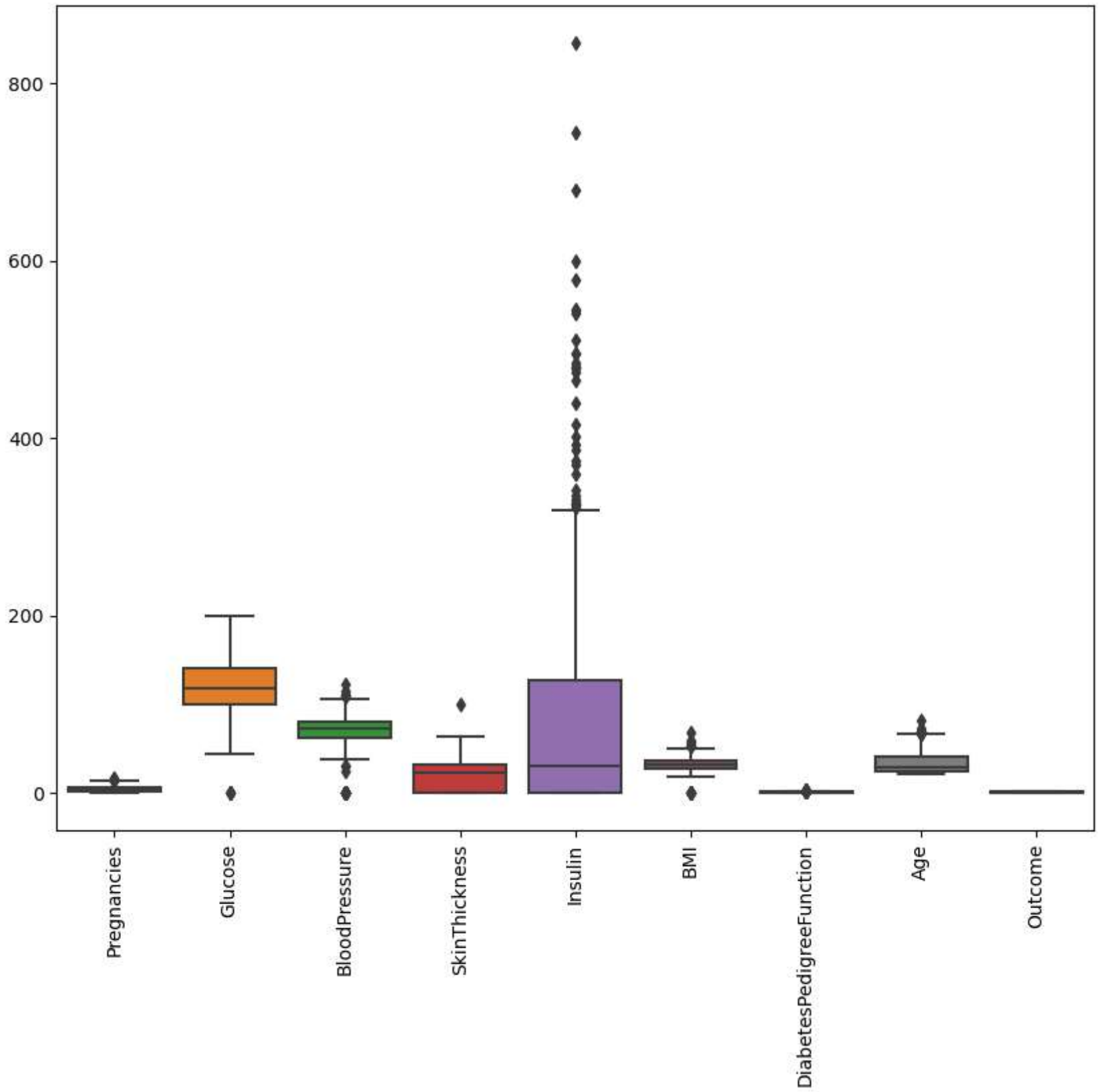
2. Heatmap of Correlations

In [42]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt='.2f', linewidths=0.5)
plt.show()
```



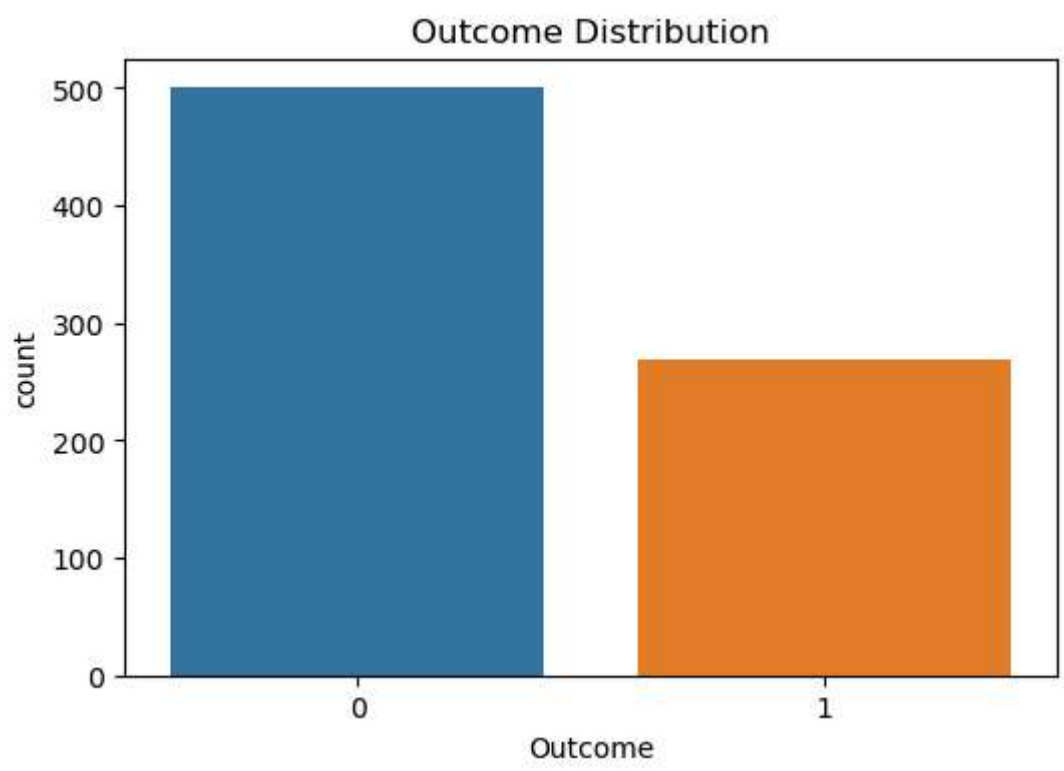
3. Box plot

In [45]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(data=df)
plt.xticks(rotation=90)
plt.show()
```



4. Bar plot

In [31]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x='Outcome', data=df)
plt.title('Outcome Distribution')
plt.show()
```



In [ ]:
```

```

Analytical Report:

1. Data Overview:

- The dataset consists of 768 rows and 9 columns. Each row represents a patient, and each column represents a specific medical attribute.
- The target variable, **Outcome**, indicates whether the patient is diabetic (1) or non-diabetic (0).

2. Missing Values:

- The dataset does not have any missing values, which is crucial for ensuring the quality of analysis. However, some values (such as 0 for **Glucose**, **BloodPressure**, **SkinThickness**, **Insulin**, and **BMI**) could represent missing or invalid data. These need to be handled in the data preprocessing stage.

3. Key Findings:

- **Statistical Summary:**
  - Variables like **BMI**, **Glucose**, and **Insulin** have values spread across a wide range, with certain features (e.g., **BMI** and **Insulin**) showing potential outliers.
- **Correlations:**
  - There is a strong positive correlation between **Glucose** and **Insulin**. This suggests that higher glucose levels are associated with higher insulin levels.
  - **Age** and **BMI** show a moderate correlation with **Outcome**, indicating that older individuals and those with higher BMI are more likely to be diabetic.

4. Visualizations Insights:

- **Scatterplots** reveal relationships between various features, particularly that **Glucose** and **Insulin** show a strong correlation.
- **Heatmaps** confirm the presence of high correlations between features such as **Glucose**, **Insulin**, and **BMI**.
- **Boxplots** show that **Age** and **BMI** have a greater variance in the diabetic population, while **BloodPressure** and **SkinThickness** tend to be more uniform.

5. Outcome Distribution:

- The dataset is relatively balanced with respect to the **Outcome** variable, but there are more non-diabetic individuals (0) than diabetic individuals (1). This is important for model training, as the class imbalance may affect the model's performance.

Conclusion:

EDA has provided valuable insights into the Pima Indians Diabetes dataset. Key findings include correlations between glucose levels, insulin levels, and BMI. Visualizations highlighted important patterns that will help guide future predictive modeling efforts. Understanding the structure and distribution of the data helps in making decisions about data cleaning, feature engineering, and model selection.