In [57]:

```
#2.3 Missing values
import pandas as pd
df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")
df.head(10)

df_dropped = df.dropna()
df_dropped


#filling with specific value
df_specific = df.fillna(0)
df_specific

df_mean = df.fillna(df.mean())
df_mean
```

Out[57]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	63.000000	1.000000	3	145.0	233	1	0	150	0	2.3	0
1	54.413333	0.684385	2	130.0	250	0	1	187	0	3.5	0
2	41.000000	0.000000	1	130.0	204	0	0	172	0	1.4	2
3	54.413333	1.000000	1	120.0	236	0	1	178	0	0.8	2
4	57.000000	0.684385	0	120.0	354	0	1	163	1	0.6	2
...	...	...	...	...	...	...	...	...	...	...	...
298	57.000000	0.000000	0	140.0	241	0	1	123	1	0.2	1
299	45.000000	1.000000	3	110.0	264	0	1	132	0	1.2	1
300	68.000000	1.000000	0	144.0	193	1	1	141	0	3.4	1
301	57.000000	1.000000	0	130.0	131	0	1	115	1	1.2	1
302	57.000000	0.000000	1	130.0	236	0	0	174	0	0.0	1

303 rows × 14 columns

In [134…

```
# 2.3 remove outliers
import pandas as pd
import numpy as np
from scipy.stats import zscore

df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")

z_score = zscore(df,nan_policy='omit')
abs_z_score = np.abs(z_score)
threshold = 2
abs_z_score.head(100)
```

Out[134...

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exan
0	0.948186	0.679091	1.973123	0.764565	0.256334	2.394438	1.005832	0.015443	0.69663
1	NaN	NaN	1.002577	0.091038	0.072199	0.417635	0.898962	1.633471	0.69663
2	1.481172	1.472556	0.032031	0.091038	0.816773	0.417635	1.005832	0.977514	0.69663
3	NaN	0.679091	0.032031	0.661440	0.198357	0.417635	0.898962	1.239897	0.69663
4	0.285634	NaN	0.938515	0.661440	2.082050	0.417635	0.898962	0.583939	1.43548
...	...	...	...	...	...	...	...	...	...
95	0.156068	0.679091	0.938515	0.593445	0.391612	0.417635	1.005832	1.690047	1.43548
96	0.837760	1.472556	0.938515	0.479364	2.855069	0.417635	1.005832	0.321556	0.69663
97	0.266493	0.679091	0.938515	1.345922	0.256334	2.394438	0.898962	0.115749	0.69663
98	1.260321	0.679091	1.002577	0.091038	1.328356	0.417635	0.898962	0.540209	0.69663
99	0.156068	0.679091	1.002577	0.091038	0.005102	2.394438	1.005832	1.021244	0.69663

100 rows × 14 columns

In [142...

```
df_cleaned = df.loc[(abs_z_score < threshold).all(axis=1)]

df_cleaned.head(100)
```

Out[142...

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
2	41.0	0.0	1	130.0	204	0	0	172	0	1.4	2	0	
7	44.0	1.0	1	120.0	263	0	1	173	0	0.0	2	0	
9	57.0	1.0	2	150.0	168	0	1	174	0	1.6	2	0	
10	54.0	1.0	0	140.0	239	0	1	160	0	1.2	2	0	
11	48.0	0.0	2	130.0	275	0	1	139	0	0.2	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
146	44.0	0.0	2	118.0	242	0	1	149	0	0.3	1	1	
147	60.0	0.0	3	150.0	240	0	1	171	0	0.9	2	0	
148	44.0	1.0	2	120.0	226	0	1	169	0	0.0	2	0	
149	42.0	1.0	2	130.0	180	0	1	150	0	0.0	2	0	
151	71.0	0.0	0	112.0	149	0	1	125	0	1.6	1	0	

100 rows × 14 columns

In [150…

```
#Api integration
import pandas as pd
import requests


url = "https://api.coingecko.com/api/v3/coins/markets"


params = {
    "vs_currency" : "usd",
    "order" : "market_cap_desc",
    "per_page" : 10,
    "page" : 1,
    "spakLine" : False
}
response = requests.get(url,params=params)

if response.status_code == 200 :
    data = response.json()
    df = pd.DataFrame(data)
    print("Data succesfully loaded into dataframe")
else :
    print(f"Failed to fetch the data : {response.status_code}")


df.head()
```

```
Data succesfully loaded into dataframe
```

Out[150…

	id	symbol	name	image	current_price	ma
0	bitcoin	btc	Bitcoin	https://coin-images.coingecko.com/coins/images...	92380.000000	182755
1	ethereum	eth	Ethereum	https://coin-images.coingecko.com/coins/images...	3113.340000	37480
2	tether	usdt	Tether	https://coin-images.coingecko.com/coins/images...	0.999951	1281
3	solana	sol	Solana	https://coin-images.coingecko.com/coins/images...	241.670000	11468
4	binancecoin	bnb	BNB	https://coin-images.coingecko.com/coins/images...	615.400000	899

5 rows × 26 columns

In [152…

Out[152…

```
Index(['id', 'symbol', 'name', 'image', 'current_price', 'market_cap',
       'market_cap_rank', 'fully_diluted_valuation', 'total_volume',
       'high_24h', 'low_24h', 'price_change_24h',
       'price_change_percentage_24h', 'market_cap_change_24h',
       'market_cap_change_percentage_24h', 'circulating_supply',
       'total_supply', 'max_supply', 'ath', 'ath_change_percentage',
       'ath_date', 'atl', 'atl_change_percentage', 'atl_date', 'roi',
       'last_updated'],
      dtype='object')
```

In [185…

```
print("Column names in DataFrame : ")
print(df.columns)

df_selected = df[['name','current_price','market_cap']]
print("\n Selectd Columns : ")
print(df_selected.head())

df_sorted = df_selected.sort_values(by='market_cap',ascending = False)
print("\nTop Cryptocurrencies of by market cap : ")
print(df_sorted)

df_selected['price_in_eur'] = df_selected['current_price'] * 0.85
print("\n Added 'price in eur' column : ")
print(df_selected.head())

df_filtered = df_selected[df_selected['market_cap'] > 1e9]
print("Cryptocurrencies with market cap greter than 1 billion")
print(df_filtered)
```

```
Column names in DataFrame :
Index(['id', 'symbol', 'name', 'image', 'current_price', 'market_cap',
       'market_cap_rank', 'fully_diluted_valuation', 'total_volume',
       'high_24h', 'low_24h', 'price_change_24h',
       'price_change_percentage_24h', 'market_cap_change_24h',
       'market_cap_change_percentage_24h', 'circulating_supply',
       'total_supply', 'max_supply', 'ath', 'ath_change_percentage',
       'ath_date', 'atl', 'atl_change_percentage', 'atl_date', 'roi',
       'last_updated'],
      dtype='object')

 Selectd Columns :
       name  current_price     market_cap
0   Bitcoin   92380.000000  1827556947075
1  Ethereum    3113.340000   374806136312
2    Tether       0.999951   128114926302
3    Solana     241.670000   114682944570
4       BNB     615.400000    89913267776

Top Cryptocurrencies of by market cap :
                name  current_price     market_cap
0            Bitcoin   92380.000000  1827556947075
1           Ethereum    3113.340000   374806136312
2             Tether       0.999951   128114926302
3             Solana     241.670000   114682944570
4                BNB     615.400000    89913267776
5                XRP       1.097000    62439424379
6           Dogecoin       0.396368    58184329728
7               USDC       0.999439    37239890081
8  Lido Staked Ether    3112.840000    30426780421
9            Cardano       0.733744    26307829353

 Added 'price in eur' column :
       name  current_price     market_cap  price_in_eur
0   Bitcoin   92380.000000  1827556947075  78523.000000
1  Ethereum    3113.340000   374806136312   2646.339000
2    Tether       0.999951   128114926302      0.849958
3    Solana     241.670000   114682944570    205.419500
4       BNB     615.400000    89913267776    523.090000
Cryptocurrencies with market cap greter than 1 billion
                name  current_price     market_cap  price_in_eur
0            Bitcoin   92380.000000  1827556947075  78523.000000
1           Ethereum    3113.340000   374806136312   2646.339000
2             Tether       0.999951   128114926302      0.849958
3             Solana     241.670000   114682944570    205.419500
4                BNB     615.400000    89913267776    523.090000
5                XRP       1.097000    62439424379      0.932450
6           Dogecoin       0.396368    58184329728      0.336913
7               USDC       0.999439    37239890081      0.849523
8  Lido Staked Ether    3112.840000    30426780421   2645.914000
9            Cardano       0.733744    26307829353      0.623682
```

```
C:\Users\vaibh\AppData\Local\Temp\ipykernel_11488\1469776383.py:12: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_selected['price_in_eur'] = df_selected['current_price'] * 0.85
```

In [201…

```python
import sqlite3

connection = sqlite3.connect("sample_database.db")
cursor = connection.cursor()

#create table
cursor.execute("""
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER,
    department TEXT,
    salary REAL

)
""")
cursor.executemany("""
INSERT INTO employees (name,age,department,salary)
VALUES(?,?,?,?)
""", [
    ("Allice",30,"HR",600000),
    ("Bob",25,"IT",70000),
     ("Charlie", 35, "Finance", 80000),
    ("Diana", 28, "IT", 65000),
    ("Eve", 40, "HR", 72000)

])
connection.commit()
connection.close()
print("Data and table setup complete")
```

```
Data and table setup complete
```

In [203…

```python
import pandas as pd
connection = sqlite3.connect("sample_database.db")

query = "SELECT * FROM employees"
df = pd.read_sql_query(query,connection)

print("Data loaded into Dataframe : ")
print(df)

connection.close()
```

```
Data loaded into Dataframe :
   id     name  age department    salary
0   1   Allice   30         HR  600000.0
1   2      Bob   25         IT   70000.0
2   3   Allice   30         HR  600000.0
3   4      Bob   25         IT   70000.0
4   5  Charlie   35    Finance   80000.0
5   6    Diana   28         IT   65000.0
6   7      Eve   40         HR   72000.0
```

In [215…

```
print("\n Summary Statistics")
print(df.describe())

df["tax"] = df["salary"]*0.10
print("\n DataFrame with Tax column : ")
print(df)

high_earners = df[df["salary"]>65000]
print("\n Employees with salary > 65000")
print(high_earners)

avg_sal_by_dept = df.groupby("department")["salary"].mean()
print("\nAverage salary by the department : ")
print(avg_sal_by_dept)

sorted_by_age = df.sort_values(by="age",ascending = False)
print("\nEmployees Sorted Age : ")
print(sorted_by_age)
```

```
 Summary Statistics
             id        age          salary           tax
count  7.000000   7.000000        7.000000      7.000000
mean   4.000000  30.428571   222428.571429  22242.857143
std    2.160247   5.442338   257968.898088  25796.889809
min    1.000000  25.000000    65000.000000   6500.000000
25%    2.500000  26.500000    70000.000000   7000.000000
50%    4.000000  30.000000    72000.000000   7200.000000
75%    5.500000  32.500000   340000.000000  34000.000000
max    7.000000  40.000000   600000.000000  60000.000000

 DataFrame with Tax column :
   id     name  age department    salary      tax
0   1   Allice   30         HR  600000.0  60000.0
1   2      Bob   25         IT   70000.0   7000.0
2   3   Allice   30         HR  600000.0  60000.0
3   4      Bob   25         IT   70000.0   7000.0
4   5  Charlie   35    Finance   80000.0   8000.0
5   6    Diana   28         IT   65000.0   6500.0
6   7      Eve   40         HR   72000.0   7200.0

 Employees with salary > 65000
   id     name  age department    salary      tax
0   1   Allice   30         HR  600000.0  60000.0
1   2      Bob   25         IT   70000.0   7000.0
2   3   Allice   30         HR  600000.0  60000.0
3   4      Bob   25         IT   70000.0   7000.0
4   5  Charlie   35    Finance   80000.0   8000.0
6   7      Eve   40         HR   72000.0   7200.0

Average salary by the department :
department
Finance     80000.000000
HR         424000.000000
IT          68333.333333
Name: salary, dtype: float64

Employees Sorted Age :
   id     name  age department    salary      tax
6   7      Eve   40         HR   72000.0   7200.0
4   5  Charlie   35    Finance   80000.0   8000.0
0   1   Allice   30         HR  600000.0  60000.0
2   3   Allice   30         HR  600000.0  60000.0
5   6    Diana   28         IT   65000.0   6500.0
1   2      Bob   25         IT   70000.0   7000.0
3   4      Bob   25         IT   70000.0   7000.0
```

In [77]:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from io import StringIO

url = "https://en.wikipedia.org/wiki/Agriculture_in_India"

response = requests.get(url)
```

```
soup = BeautifulSoup(response.text,"html.parser")
table = soup.find("table",{"class" : "wikitable"})


df = pd.read_html(StringIO(str(table)))[0]


print("Extracted Data : ")
df.head()
```

```
Extracted Data :
```

Out[77]:

	Rank	Commodity	Value (US$, 2016)	Unit price (US$ / kilogram, 2009)	Average yield (tonnes per hectare, 2017)	Most productive country (tonnes per hectare, 2017)	Most productive country (tonnes per hectare, 2017).1
0	1	Rice	$70.18 billion	0.27	3.85	9.82	Australia
1	2	Buffalo milk	$43.09 billion	0.40	2.00[78]	2.00[78]	India
2	3	Cow milk	$32.55 billion	0.31	1.2[78]	10.3[78]	Israel
3	4	Wheat	$26.06 billion	0.15	2.8	8.9	Netherlands
4	5	Cotton (Lint + Seeds)	$23.30 billion	1.43	1.6	4.6	Israel

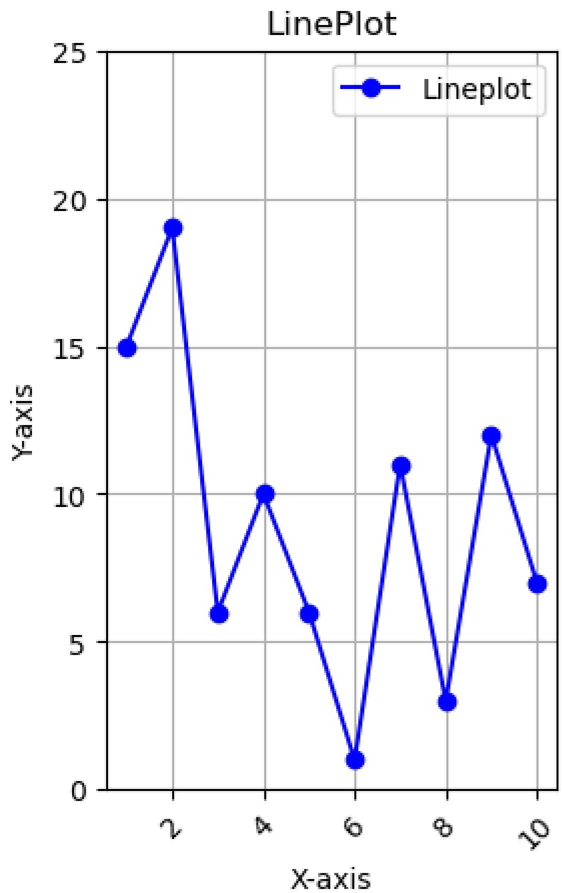
In [52]:

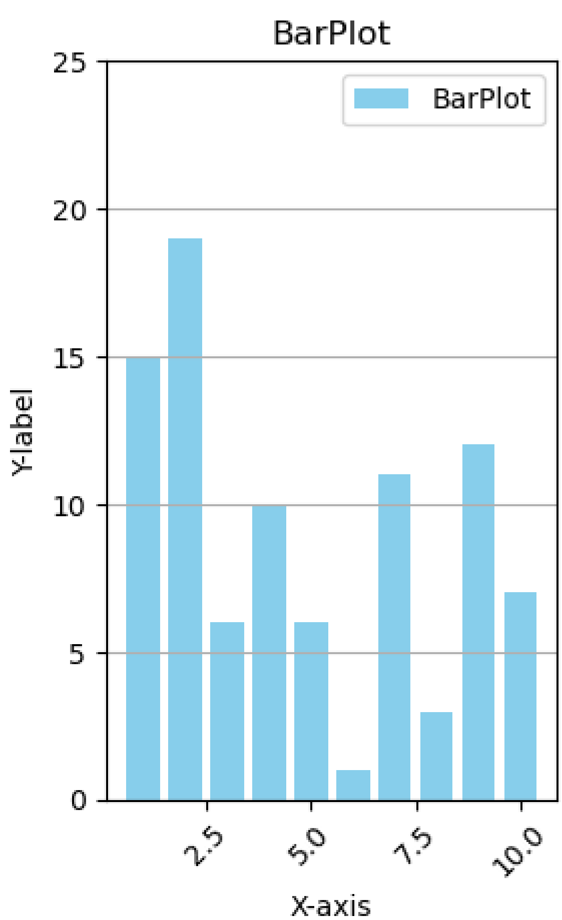
```
import pandas as pd

data = {
    'name' : ["Vaibhav","Riya","Taniya","Tanmay","Yashraj"],
    'Age' : [12,14,16,17,18],
    'rank' : [1,3,2,5,4],
    'score' : [99,67,87,12,34]
}
df = pd.DataFrame(data)
print(df)
print(df.columns)
df_selected = df[df['Age'] > 14]
print(df_selected)
df_sorted = df.sort_values(by="rank",ascending = False)
print(df_sorted)
df_grouping = df.groupby("name")["score"].mean()
print(df_grouping)
```

```
      name  Age  rank  score
0  Vaibhav   12     1     99
1     Riya   14     3     67
2   Taniya   16     2     87
3   Tanmay   17     5     12
4  Yashraj   18     4     34
Index(['name', 'Age', 'rank', 'score'], dtype='object')
      name  Age  rank  score
2   Taniya   16     2     87
3   Tanmay   17     5     12
4  Yashraj   18     4     34
      name  Age  rank  score
3   Tanmay   17     5     12
4  Yashraj   18     4     34
1     Riya   14     3     67
2   Taniya   16     2     87
0  Vaibhav   12     1     99
name
Riya       67.0
Taniya     87.0
Tanmay     12.0
Vaibhav    99.0
Yashraj    34.0
Name: score, dtype: float64
```

In [87]:

```
import numpy as np
import matplotlib.pyplot as plt
A = np.arange(1,11)
B = np.random.randint(1,20,size = 10)

plt.subplot(1, 2, 1)  # 1 row, 2 columns, first subplot
plt.plot(A,B,label="Lineplot",marker = "o",linestyle = "-",color = "blue")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.title("LinePlot")
plt.grid(True)
plt.ylim(0, 25)  # Set y-axis limits
plt.xticks(rotation=45)  # Rotate x-axis labels
plt.show()


plt.subplot(1, 2, 1)  # 1 row, 2 columns, first subplot
plt.bar(A,B,label = "BarPlot",color = "skyblue")
plt.xlabel("X-axis")
plt.ylabel("Y-label")
plt.legend()
plt.title("BarPlot")
plt.grid(axis="y")  # Grid only on the y-axis
plt.ylim(0, 25)  # Set y-axis limits
plt.xticks(rotation=45)  # Rotate x-axis labels
plt.show()
```


LinePlot
Lineplot
Y-axis
X-axis
25
20
15
10
5
0
2
4
6
8
10


BarPlot
BarPlot
Y-label
X-axis
25
20
15
10
5
0
2.5
5.0
7.5
10.0

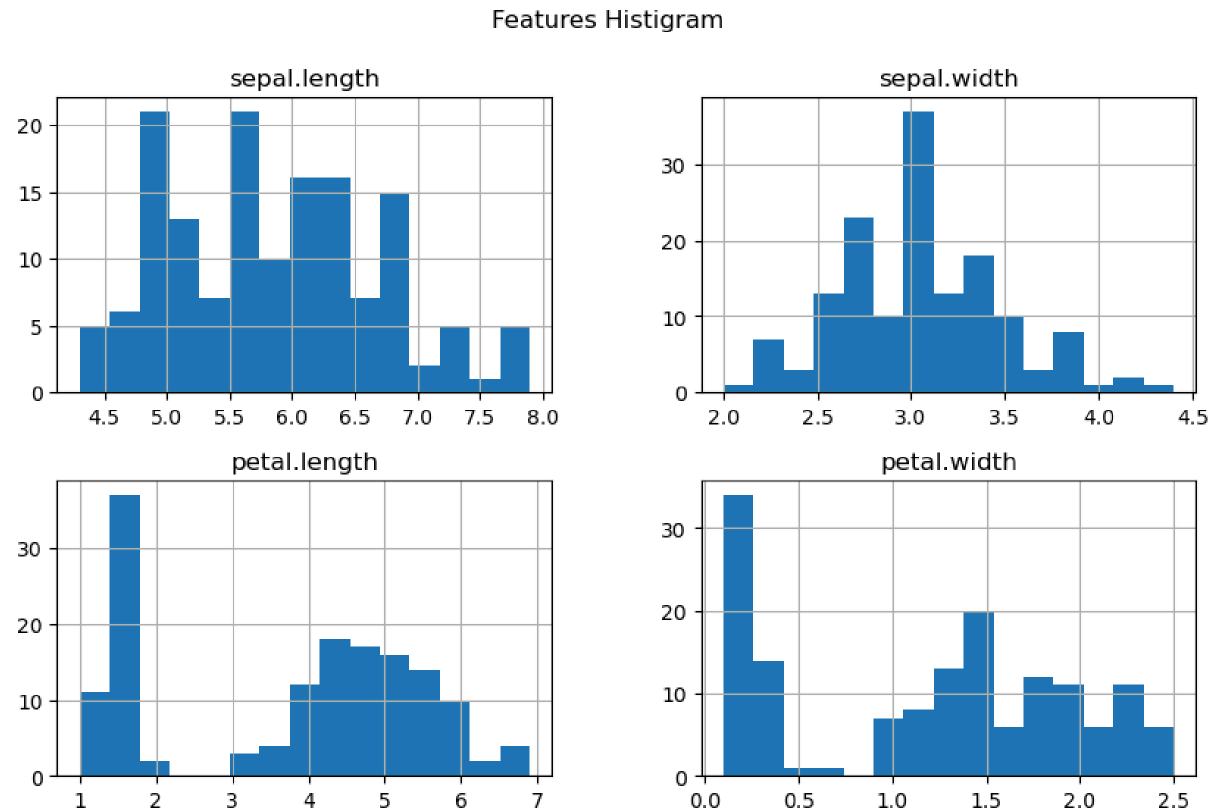
In [106…

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

df = pd.read_csv(r"C:\Users\vaibh\Downloads\iris.csv")
print(df.describe())
print()
print(df.info())
print(df.isnull().sum())
```

```
       sepal.length  sepal.width  petal.length  petal.width
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

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
species         0
dtype: int64
```

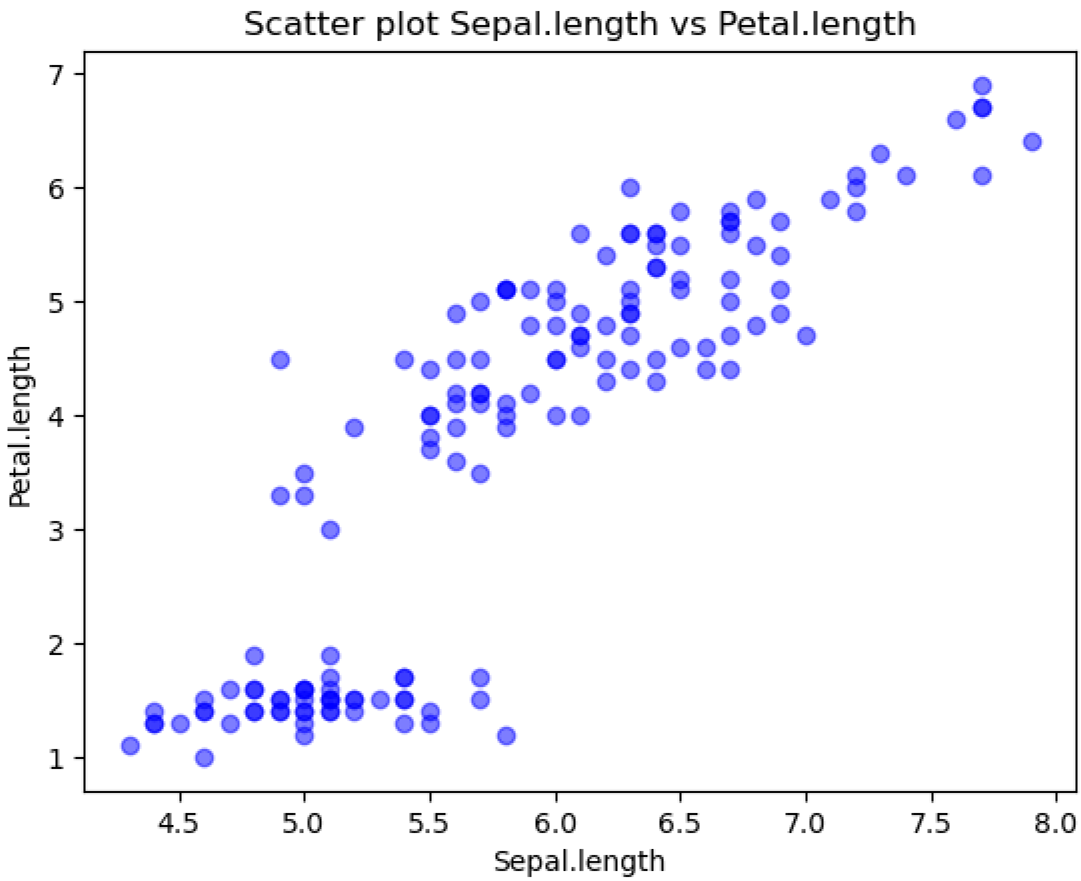
In [114…

```
df[['sepal.length','sepal.width','petal.length','petal.width']].hist(bins = 15,figs
plt.suptitle("Features Histigram")
plt.show()
```



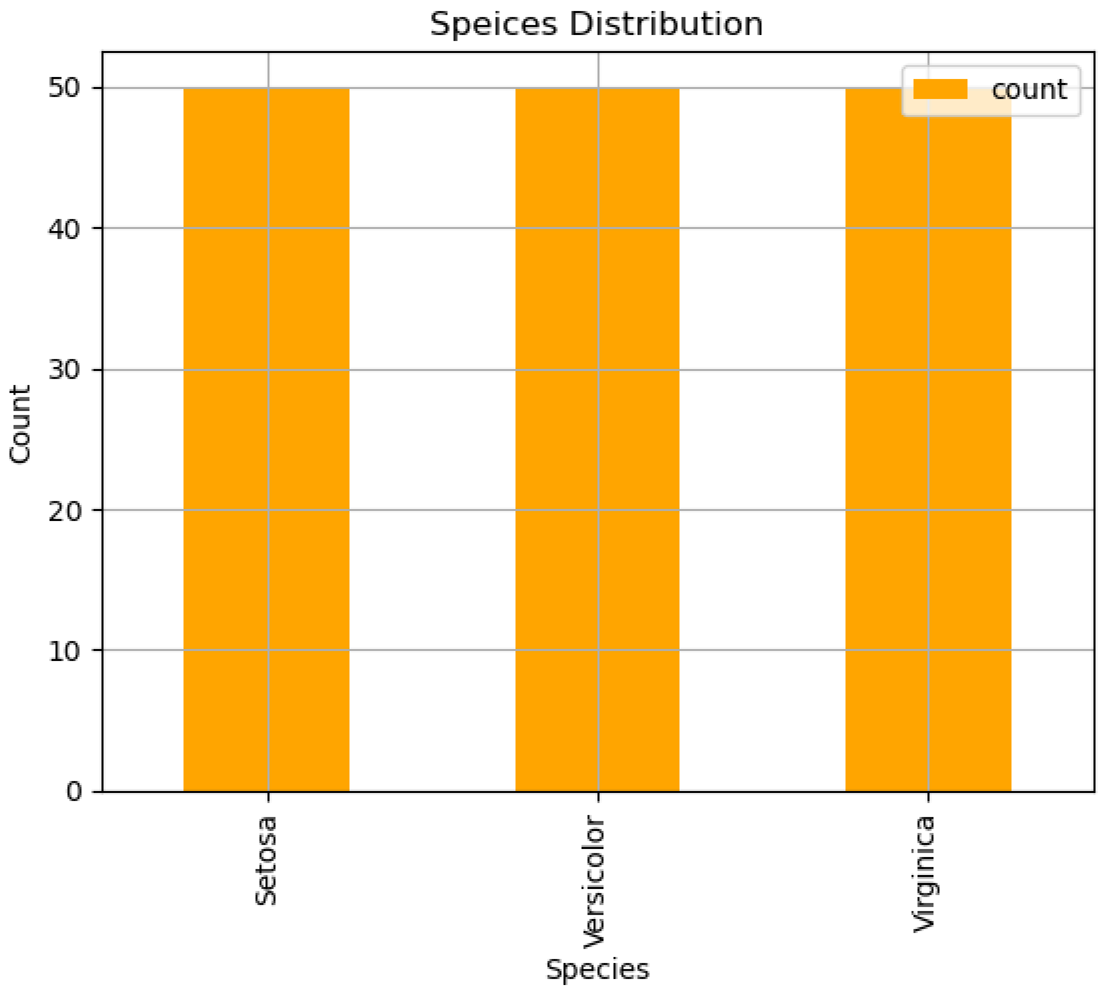
In [122…

```
plt.scatter(df['sepal.length'],df['petal.length'],color = 'blue',alpha = 0.5)
plt.title("Scatter plot Sepal.length vs Petal.length")
plt.xlabel("Sepal.length")
plt.ylabel("Petal.length")
plt.show()
```



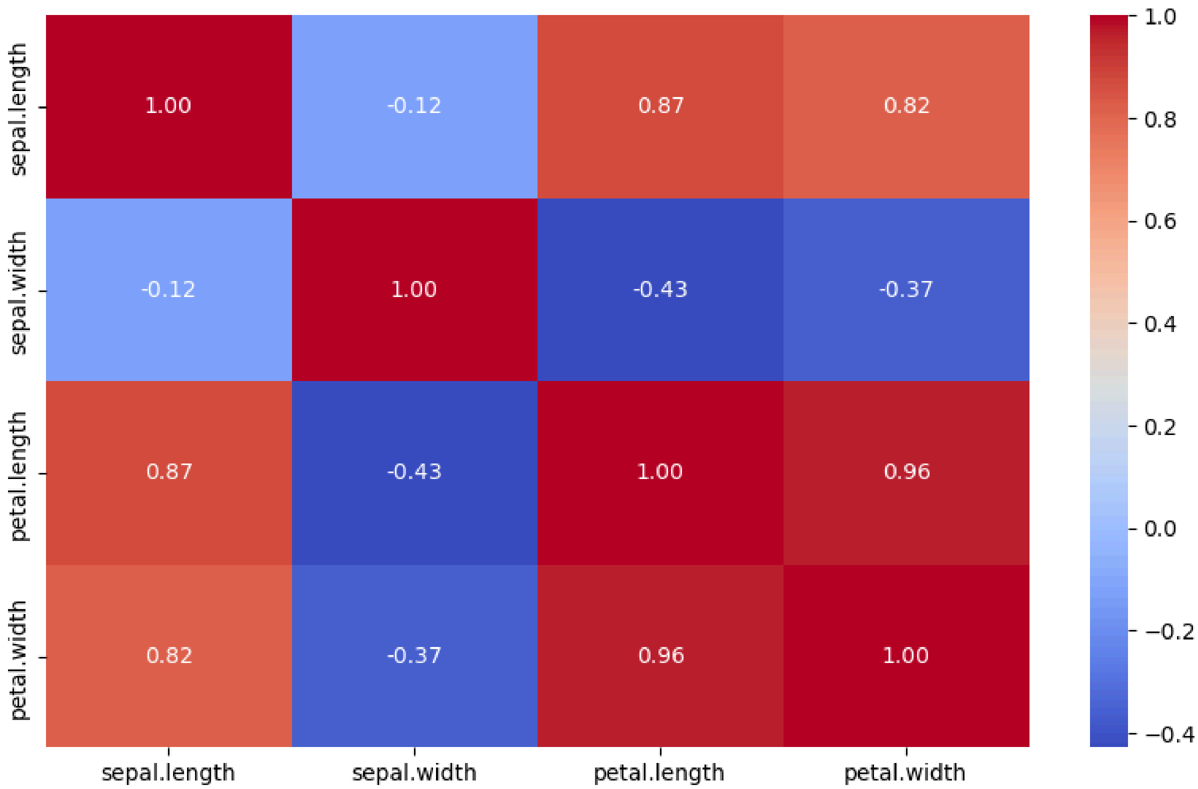
In [132…

```
df['species'].value_counts().plot(kind = 'bar',color = 'orange')
plt.title("Speices Distribution")
plt.xlabel("Species")
plt.ylabel("Count")
plt.grid(True)
plt.legend()
plt.show()
```



In [140…

```
import seaborn as sns
corelation_matrix = df.select_dtypes(include = 'number').corr()
plt.figure(figsize = (10,6))
sns.heatmap(corelation_matrix,annot = True,cmap = 'coolwarm',fmt='0.2f',cbar = True
plt.show()
```



In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\vaibh\Downloads\Orange_Telecom_Churn_Data.csv")
df.head(5)
```

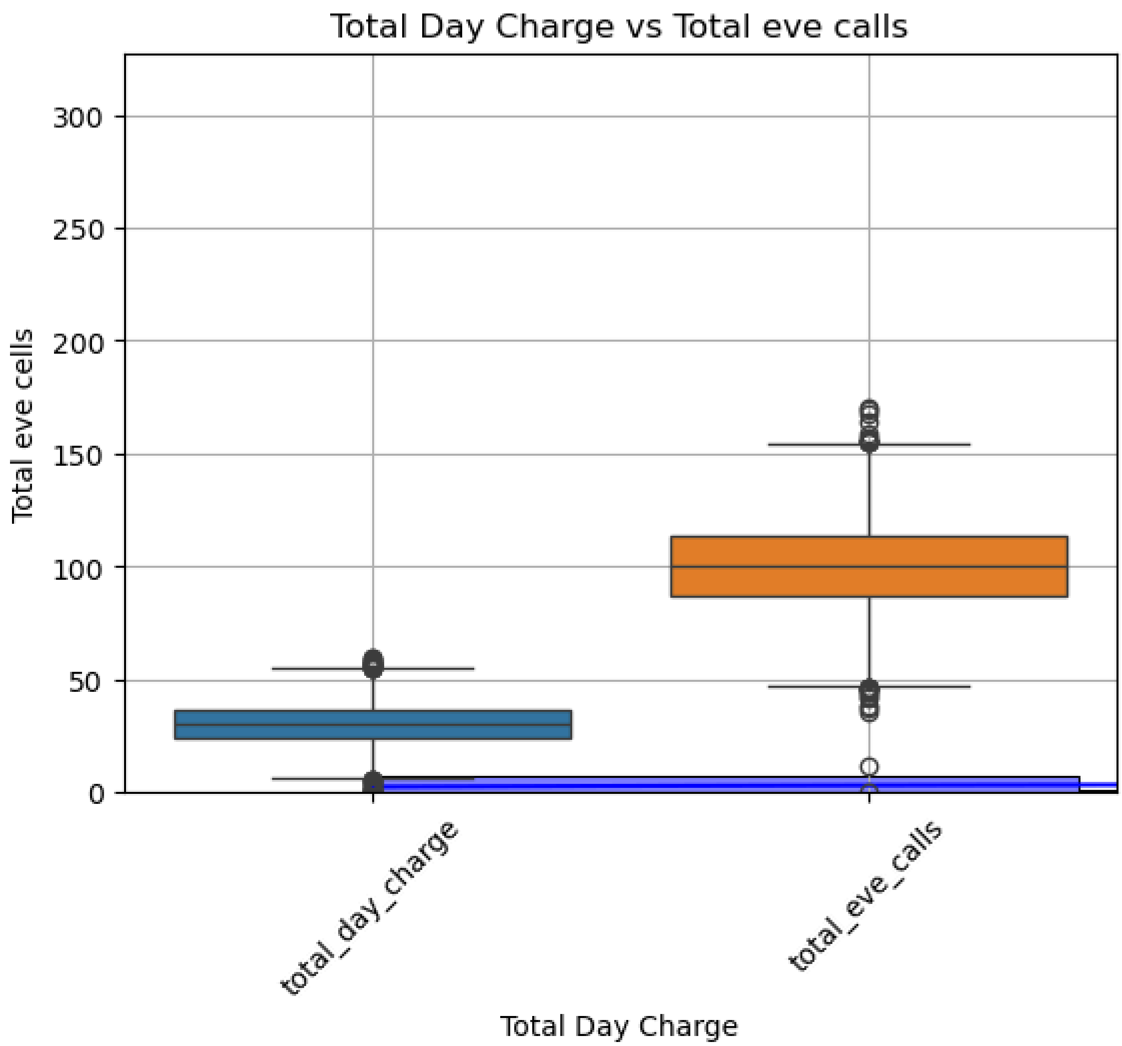
Out[3]:

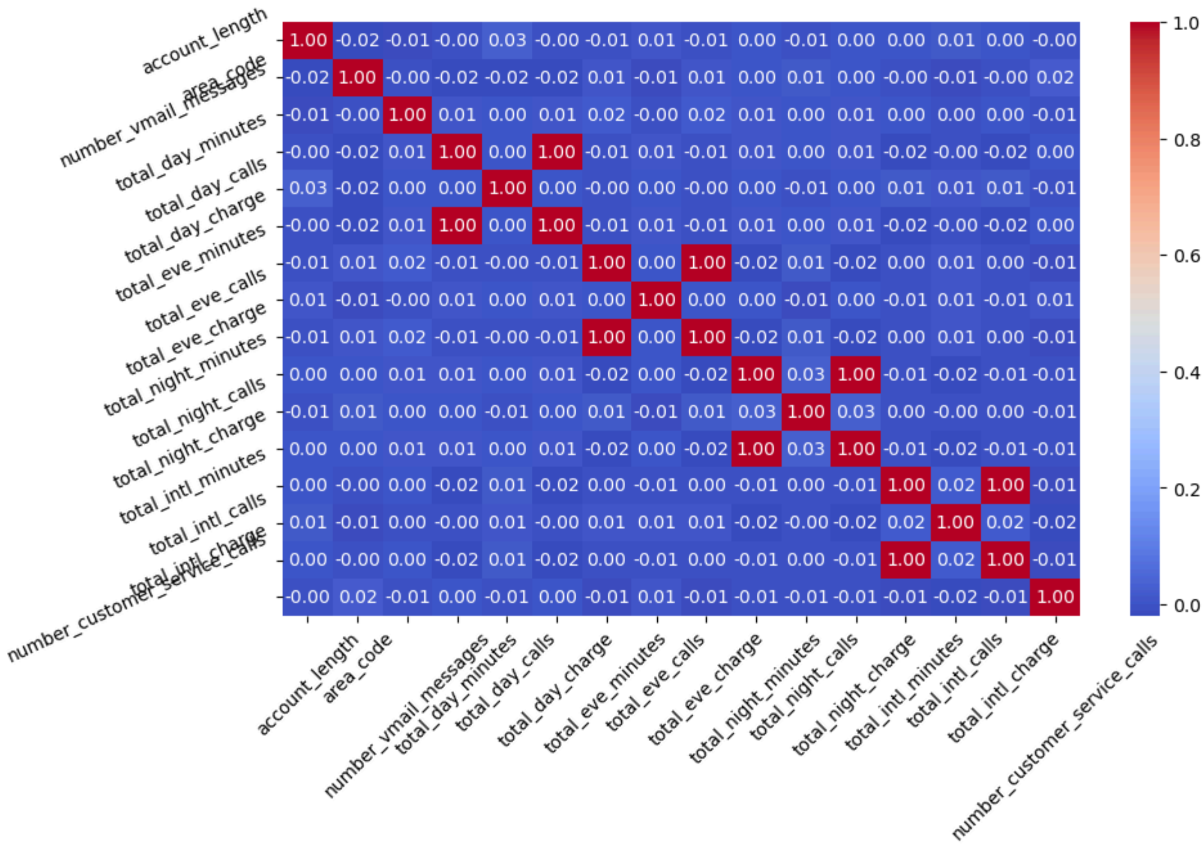
	state	account_length	area_code	phone_number	intl_plan	voice_mail_plan	number_vm
0	KS	128	415	382-4657	no	yes	
1	OH	107	415	371-7191	no	yes	
2	NJ	137	415	358-1921	no	no	
3	OH	84	408	375-9999	yes	no	
4	OK	75	415	330-6626	yes	no	

5 rows × 21 columns

In [9]:
```
sns.histplot(df['total_day_charge'],kde = True,color = 'blue')
plt.title("Histogram of total day charge")
plt.xlabel("Total Day Charge")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

sns.boxplot(data =  df[['total_day_charge','total_eve_calls']])
plt.title("Total Day Charge vs Total eve calls")
plt.xlabel("Total Day Charge")
```

```
plt.ylabel("Total eve cells")
plt.xticks(rotation = 45)
plt.grid(True)
plt.show()

correlation_matrix = df.select_dtypes(include = 'number').corr()
plt.figure(figsize = (10,6))
sns.heatmap(correlation_matrix,annot= True,cmap = 'coolwarm',fmt = '0.2f',cbar = Tr
plt.xticks(rotation = 45)
plt.yticks(rotation = 25)
plt.show()
```

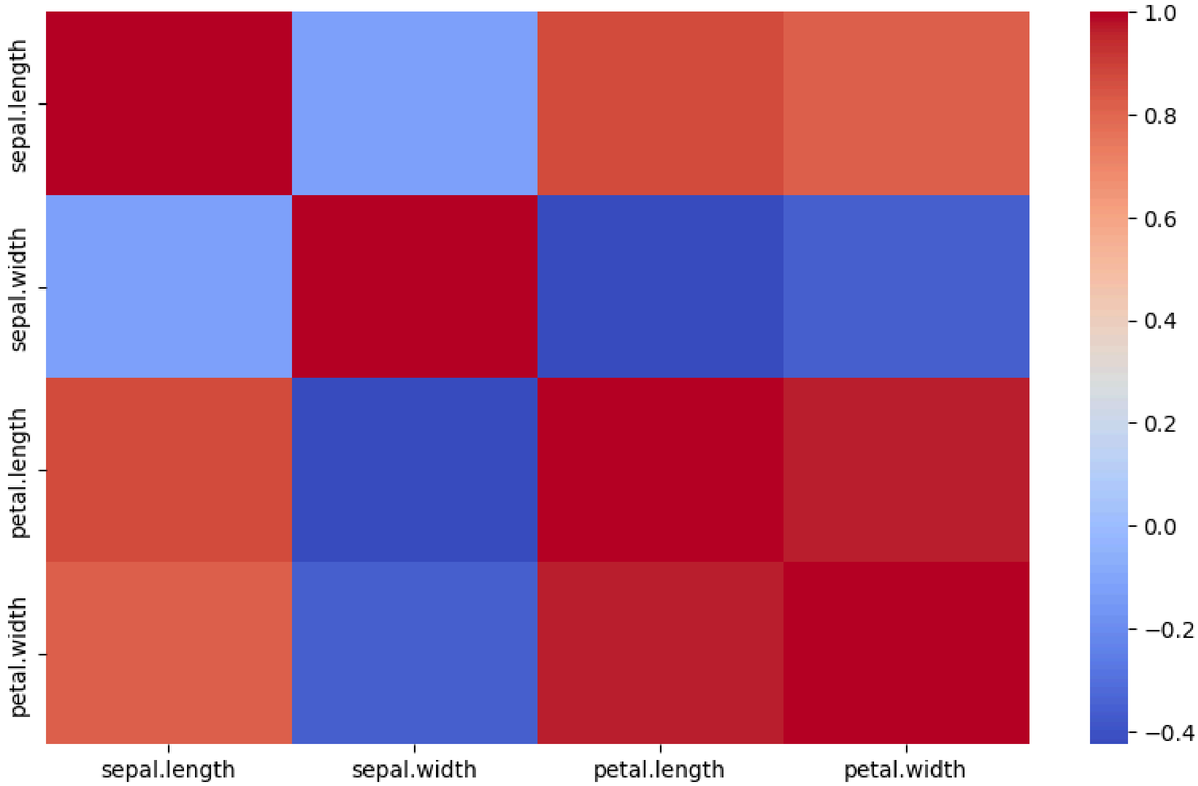




In [129…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\vaibh\Downloads\iris.csv")
print("Data frame with missing values : ")
print(df.isnull().sum())
df.dropna(inplace = True)
print("\n Number of Duplicates rows : ")
print(df.duplicated().sum())
df.drop_duplicates(inplace  = True)
correalation_matrix = df.select_dtypes(include = 'number').corr()
print("\n Data set after Cleaning ")
print(df.info())
plt.figure(figsize =(10,6))
sns.heatmap(correalation_matrix,annot =False,cmap='coolwarm',fmt ='0.2f',cbar = Tru
plt.show()
```

```
Data frame with missing values :
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
species         0
dtype: int64

 Number of Duplicates rows :
1

 Data set after Cleaning
<class 'pandas.core.frame.DataFrame'>
Index: 149 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal.length  149 non-null    float64
 1   sepal.width   149 non-null    float64
 2   petal.length  149 non-null    float64
 3   petal.width   149 non-null    float64
 4   species       149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 7.0+ KB
None
```



In [190…

```
import numpy as np
from scipy.stats import ttest_1samp

sample = [23, 27, 26, 30, 29, 22, 24, 28, 25, 31]
population_mean = 25
```

```
t_stat,p_value = ttest_1samp(sample,population_mean)
alpha = 0.05
print("T-statics : ",t_stat)
print("p_value : ",p_value)

if p_value < alpha :
    print("Reject the null hypothesis : The sample mean is completly difrent from p
else :
    print("Failed to reject the null hupothesis : there is no significant diffrence
```

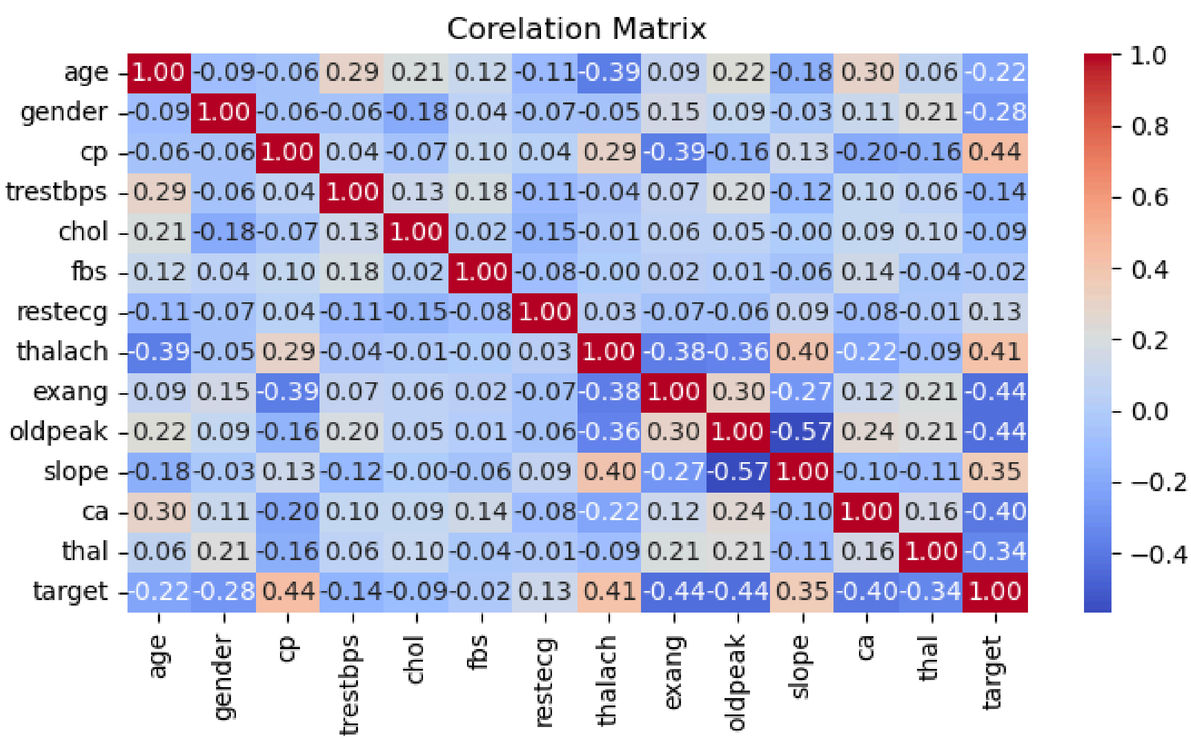
```
T-statics :  1.5666989036012806
p_value :  0.1516274744876827
Failed to reject the null hupothesis : there is no significant diffrence between sam
ple mean and population mean
```

In [232…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
import pandas as pd
df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")
print("Missing values : ")
print(df.isnull().sum())
df.dropna(inplace = True)
print("\n Duplicated items : ")
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)

X = df.iloc[:,0:13]
y = df['target']
print(X)
print(y)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state =
model = LogisticRegression(max_iter=1000)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

confusion = confusion_matrix(y_test,y_pred)
print("Confusion matrix ")
print(confusion)

acuuracy = accuracy_score(y_test,y_pred)
print("\nAccuracy Score : ")
print(acuuracy)

classification = classification_report(y_test,y_pred)
print("Classification Report : ")
print(classification)




plt.figure(figsize = (8,4))
sns.heatmap(df.corr(),annot = True,cmap = 'coolwarm',fmt = '0.2f',cbar = True)
plt.title("Corelation Matrix")
plt.show()
```

```
Missing values :
age         3
gender      2
cp          0
trestbps    1
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64

 Duplicated items :
1
      age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    63.0     1.0   3     145.0   233    1        0      150      0      2.3
2    41.0     0.0   1     130.0   204    0        0      172      0      1.4
5    57.0     1.0   0     140.0   192    0        1      148      0      0.4
7    44.0     1.0   1     120.0   263    0        1      173      0      0.0
8    52.0     1.0   2     172.0   199    1        1      162      0      0.5
..    ...     ...  ..       ...   ...  ...      ...      ...    ...      ...
298  57.0     0.0   0     140.0   241    0        1      123      1      0.2
299  45.0     1.0   3     110.0   264    0        1      132      0      1.2
300  68.0     1.0   0     144.0   193    1        1      141      0      3.4
301  57.0     1.0   0     130.0   131    0        1      115      1      1.2
302  57.0     0.0   1     130.0   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
2        2   0     2
5        1   0     1
7        2   0     3
8        2   0     3
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[298 rows x 13 columns]
0      1
2      1
5      1
7      1
8      1
      ..
298    0
299    0
300    0
301    0
```

```
302    0
Name: target, Length: 298, dtype: int64
Confusion matrix
[[27  9]
 [ 6 48]]

Accuracy Score :
0.8333333333333334
Classification Report :
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        36
           1       0.84      0.89      0.86        54

    accuracy                           0.83        90
   macro avg       0.83      0.82      0.82        90
weighted avg       0.83      0.83      0.83        90
```



In [288…

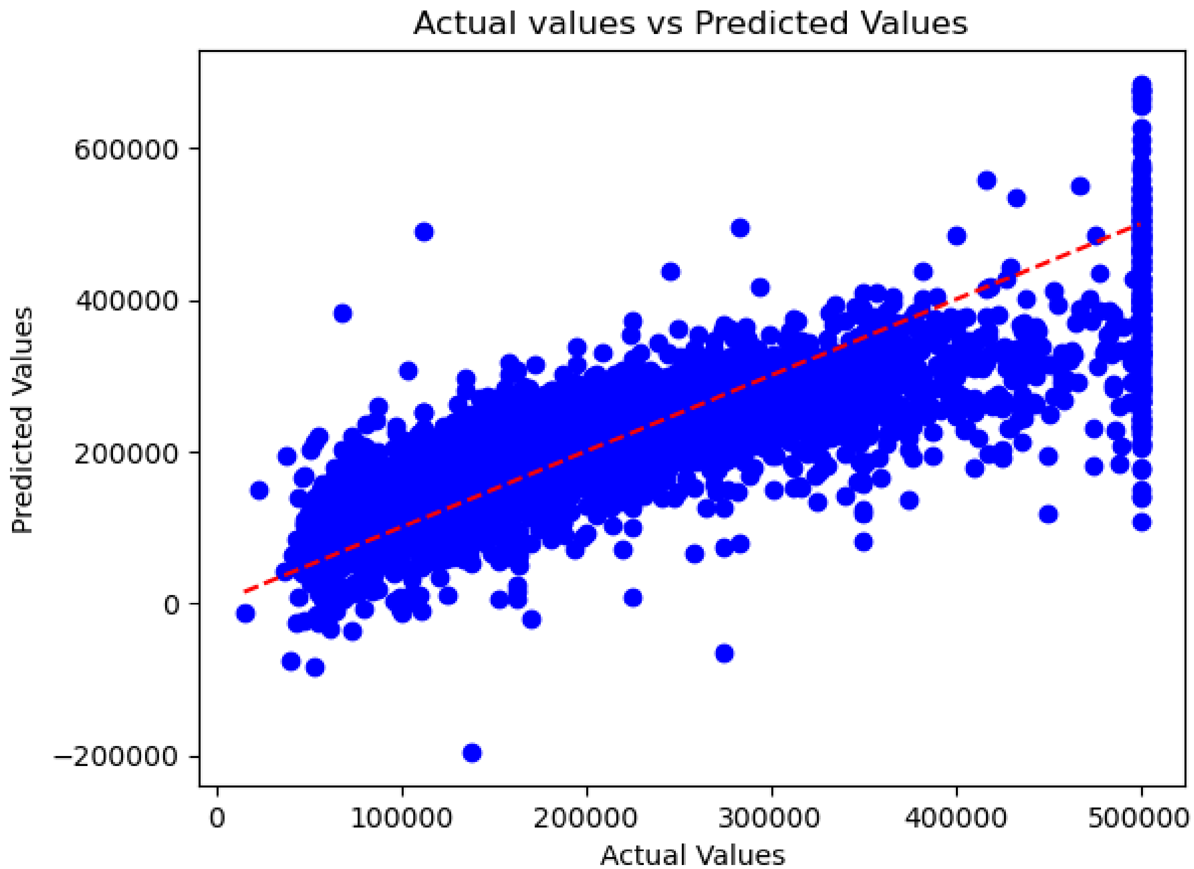
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
#from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
import pandas as pd

df = pd.read_csv(r"C:\Users\vaibh\Downloads\housing.csv")
print("Missing values : ")
print(df.isnull().sum())
df.dropna(inplace = True)
print("\n Duplicate Values : ")
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)
print(df.info())
df = df.select_dtypes(include = 'number')
```

```
X = df.drop(columns=['median_house_value'])
y = df['median_house_value']
print(y)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state =
#scaler = StandardScaler()
#X_train_scaled = scaler.fit_transform(X_train)
#X_test_scaled = scaler.fit_transform(X_test)
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test,y_pred)
print("\nMean Squared Error : ",mse)
mae = mean_absolute_error(y_test,y_pred)
print("Mean absolute square : ",mae)
r2 = r2_score(y_test,y_pred)
print("R2 score : ",r2)
```

```
Missing values :
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64

 Duplicate Values :
0
<class 'pandas.core.frame.DataFrame'>
Index: 20433 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20433 non-null  float64
 1   latitude            20433 non-null  float64
 2   housing_median_age  20433 non-null  float64
 3   total_rooms         20433 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20433 non-null  float64
 6   households          20433 non-null  float64
 7   median_income       20433 non-null  float64
 8   median_house_value  20433 non-null  float64
 9   ocean_proximity     20433 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.7+ MB
None
0        452600.0
1        358500.0
2        352100.0
3        341300.0
4        342200.0
           ...   
20635     78100.0
20636     77100.0
20637     92300.0
20638     84700.0
20639     89400.0
Name: median_house_value, Length: 20433, dtype: float64

Mean Squared Error :  4927778339.313827
Mean absolute square :  51434.3984174052
R2 score :  0.6396553423014997
```

In [290…

```
plt.scatter(y_test,y_pred,color = "blue")
plt.title("Actual values vs Predicted Values")
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.plot([y_test.min(),y_test.max()],[y_test.min(),y_test.max()],color = "red",line
plt.show()
```



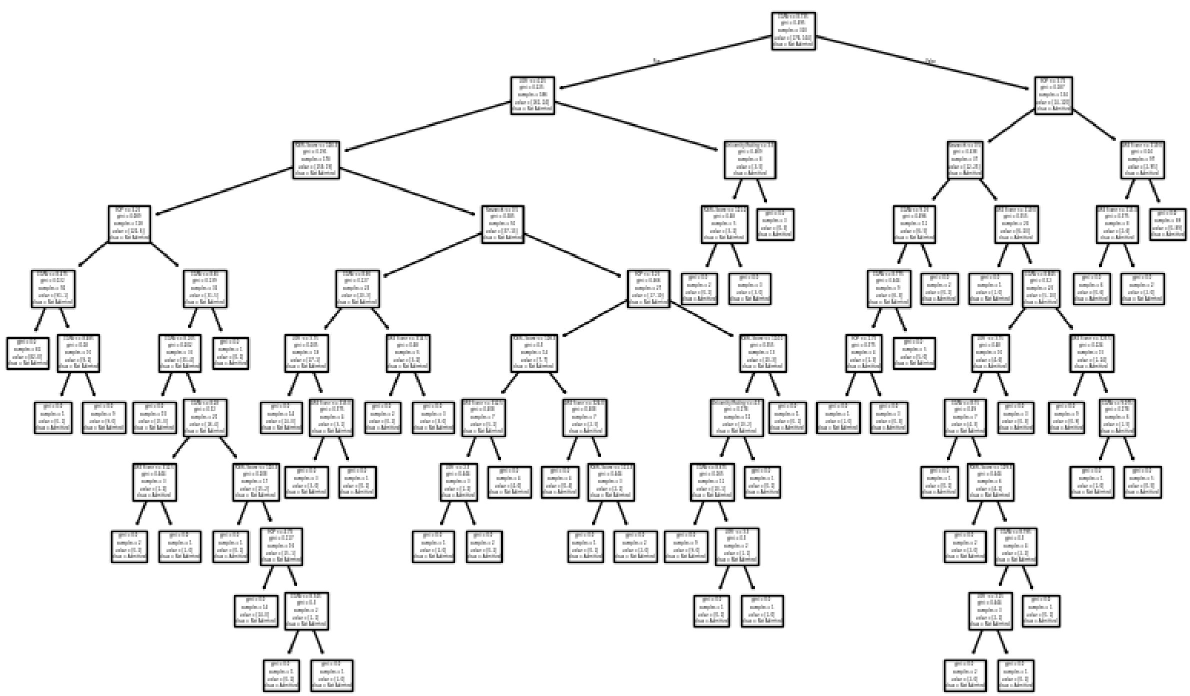
In [63]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.metrics import accuracy_score,classification_report

df = pd.read_csv(r"C:\Users\vaibh\Downloads\Admission_Predict.csv")
df.head()


df = df.drop(columns = ['Serial No.'])

df['Admit'] = (df['Chance of Admit ']>= 0.75).astype(int)

df = df.drop(columns = ['Chance of Admit '])
X = df.drop(columns = 'Admit')
y= df['Admit']


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state

clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
print("Acucuracy  score : ",end = " ")
print(accuracy_score(y_test,y_pred))
```

```
print("Classification Report : ")
print(classification_report(y_test,y_pred))
```

```
Acucuracy  score :  0.8375
Classification Report :
              precision    recall  f1-score   support

           0       0.79      0.95      0.87        44
           1       0.93      0.69      0.79        36

    accuracy                           0.84        80
   macro avg       0.86      0.82      0.83        80
weighted avg       0.85      0.84      0.83        80
```

In [85]:
```
plt.figure(figsize = (10,6))
plot_tree(clf,feature_names=X.columns,class_names= ["Not Adimted","Admitted"])
plt.show()
```



In [ ]:
```
#sample code
import sqlite3
import pandas as pd
connection = sqlite3.connect("sample_database.db")
cursor = connection.cursor()

#create table
cursor.execute("""
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER,
    department TEXT,
    salary REAL
```

```
)
""")
cursor.executemany("""
INSERT INTO employees (name,age,department,salary)
VALUES(?,?,?,?)
""", [
    ("Allice",30,"HR",600000),
    ("Bob",25,"IT",70000),
     ("Charlie", 35, "Finance", 80000),
    ("Diana", 28, "IT", 65000),
    ("Eve", 40, "HR", 72000)

])
connection.commit()
connection.close()
print("Data and table setup complete")

connection = sqlite3.connect("sample_database.db")

query = "SELECT * FROM employees"
df = pd.read_sql_query(query,connection)

print("Data loaded into Dataframe : ")
print(df)

connection.close()


print("\n Summary Statistics")
print(df.describe())

df["tax"] = df["salary"]*0.10
print("\n DataFrame with Tax column : ")
print(df)

high_earners = df[df["salary"]>65000]
print("\n Employees with salary > 65000")
print(high_earners)

avg_sal_by_dept = df.groupby("department")["salary"].mean()
print("\nAverage salary by the department : ")
print(avg_sal_by_dept)

sorted_by_age = df.sort_values(by="age",ascending = False)
print("\nEmployees Sorted Age : ")
print(sorted_by_age)
```

In [45]:

```
import sqlite3
connection = sqlite3.connect("samp.db")
cursor   = connection.cursor()

cursor.execute ("""
CREATE TABLE IF NOT EXISTS employees(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER,
```

```
    department TEXT,
    salary REAL

)
""")

cursor.executemany("""
INSERT INTO employees(name,age,department,salary)
VALUES(?,?,?,?)
""",[
    ("Allice",23,"HR",60000),
    ("BOB",34,"IT",70000),
    ("Charlie",25,"Finance",80000),
    ("Dianna",45,"HR",56000),
    ("EVE",40,"HR",72000)
]
)

connection.commit()
connection.close()
print("Data loaded succesfully")
```

```
Data loaded succesfully
```

In [47]:

```
import pandas as pd
connection = sqlite3.connect("samp.db")
query ="SELECT * FROM  employees"
df = pd.read_sql_query(query,connection)
print("data loaded into dataframe")
df.head()
```

```
data loaded into dataframe
```

Out[47]:

	id	name	age	department	salary
0	1	Allice	23	HR	60000.0
1	2	BOB	34	IT	70000.0
2	3	Charlie	25	Finance	80000.0
3	4	Dianna	45	HR	56000.0
4	5	EVE	40	HR	72000.0

In [59]:

```
df['tax'] = (df['salary']*0.05)
print(df.head())
high_earners = df[df['salary']>65000]
print("\n Empoyees with salry greter than 65000")
print(high_earners)
avg_sal_by_dept = df.groupby("department").salary.mean()
print("\n Average salary by department")
print(avg_sal_by_dept)
sort_by_age = df.sort_values(by="age",ascending = False)
print("\nsort by age")
print(sort_by_age)
```

```
   id     name  age department   salary     tax
0   1   Allice   23         HR  60000.0  3000.0
1   2      BOB   34         IT  70000.0  3500.0
2   3  Charlie   25    Finance  80000.0  4000.0
3   4   Dianna   45         HR  56000.0  2800.0
4   5      EVE   40         HR  72000.0  3600.0

 Empoyees with salry greter than 65000
   id     name  age department   salary     tax
1   2      BOB   34         IT  70000.0  3500.0
2   3  Charlie   25    Finance  80000.0  4000.0
4   5      EVE   40         HR  72000.0  3600.0

 Average salary by department
department
Finance    80000.000000
HR         62666.666667
IT         70000.000000
Name: salary, dtype: float64

sort by age
   id     name  age department   salary     tax
3   4   Dianna   45         HR  56000.0  2800.0
4   5      EVE   40         HR  72000.0  3600.0
1   2      BOB   34         IT  70000.0  3500.0
2   3  Charlie   25    Finance  80000.0  4000.0
0   1   Allice   23         HR  60000.0  3000.0
```

In [67]:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Fetch the webpage content
url = "https://en.wikipedia.org/wiki/Agriculture_in_India"
response = requests.get(url)

# Parse the HTML using BeautifulSoup
soup = BeautifulSoup(response.text, "html.parser")

# Extract all tables using pandas
tables = pd.read_html(response.text)

# Print all tables
print(f"Number of tables found: {len(tables)}")
for i, table in enumerate(tables):
    print(f"\nTable {i + 1}:")
    print(table.head())  # Display the first few rows of each table
```

```
  Country or Territory Population(1 July 2022) Population(1 July 2023)  \
0                World           8,021,407,192           8,091,734,930
1                India           1,425,423,212           1,438,069,596
2             China[a]           1,425,179,569           1,422,584,933
3        United States             341,534,046             343,477,335
4            Indonesia             278,830,529             281,190,067

   Change UN Continental Region UN Statistical Subregion
0  +0.88%                     -                        -
1  +0.89%                  Asia            Southern Asia
2  -0.18%                  Asia             Eastern Asia
3  +0.57%              Americas         Northern America
4  +0.85%                  Asia       South-eastern Asia
```

In [153…

```python
import pandas as pd
import requests
from bs4 import BeautifulSoup
from io import StringIO

url = "https://en.wikipedia.org/wiki/Agriculture_in_India"

response = requests.get(url)

soup = BeautifulSoup(response.text,"html.parser")
table = soup.find("table",{"class" : "wikitable"})


df = pd.read_html(StringIO(str(table)))[0]


print("Extracted Data : ")
df.head()
```

```
Extracted Data :
```

Out[153…

	Rank	Commodity	Value (US$, 2016)	Unit price (US$ / kilogram, 2009)	Average yield (tonnes per hectare, 2017)	Most productive country (tonnes per hectare, 2017)	Most productive country (tonnes per hectare, 2017).1
0	1	Rice	$70.18 billion	0.27	3.85	9.82	Australia
1	2	Buffalo milk	$43.09 billion	0.40	2.00[78]	2.00[78]	India
2	3	Cow milk	$32.55 billion	0.31	1.2[78]	10.3[78]	Israel
3	4	Wheat	$26.06 billion	0.15	2.8	8.9	Netherlands
4	5	Cotton (Lint + Seeds)	$23.30 billion	1.43	1.6	4.6	Israel

In [123…
```
# import pandas as pd
# from bs4 import BeautifulSoup
# import requests
# from io import StringIO

# url = "https://en.wikipedia.org/wiki/Agriculture_in_India"
# response = requests.get(url)

# soup = BeautifulSoup(response.text,"html.parser")
# table =soup.find("table",{"class" : "wikitable"})

# df = pd.read_html(StringIO(str(table)))[0]

# rows = table.find_all("tr")
# data = []
# for row in rows:
#     cells = row.find_all(["th", "td"])  # Include both headers and data cells
#     data.append([cell.text.strip() for cell in cells])

# # Convert to DataFrame
# df = pd.DataFrame(data)
# print("Extracted Table Data:")
# print(df.head())
# tables = pd.read_html(response.text)
# for i, table in enumerate(tables):
#     print(f"\nTable {i + 1}:\n", table.head())
```

In [107…
```
df.describe()
```

Out[107…

	0	1	2	3	4	5	6
count	21	21	21	21	21	21	20
unique	21	21	21	19	20	21	16
top	Rank	Commodity	Value (US$, 2016)	0.4	1.1	Most productive country(tonnes per hectare, 2017)	Israel
freq	1	1	1	2	2	1	3

In [117…
```
#Api integration
import pandas as pd
import requests


url = "https://api.coingecko.com/api/v3/coins/markets"


params = {
    "vs_currency" : "usd",
    "order" : "market_cap_desc",
    "per_page" : 10,
    "page" : 1,
    "spakLine" : False
}
```

```python
response = requests.get(url,params=params)

if response.status_code == 200 :
    data = response.json()
    df = pd.DataFrame(data)
    print("Data succesfully loaded into dataframe")
else :
    print(f"Failed to fetch the data : {response.status_code}")


df.head()
```

```
Data succesfully loaded into dataframe
```

Out[117…

	id	symbol	name	image	current_price	ma
0	bitcoin	btc	Bitcoin	https://coin-images.coingecko.com/coins/images...	97041.000000	191756
1	ethereum	eth	Ethereum	https://coin-images.coingecko.com/coins/images...	3107.370000	37371
2	tether	usdt	Tether	https://coin-images.coingecko.com/coins/images...	0.998863	12986
3	solana	sol	Solana	https://coin-images.coingecko.com/coins/images...	239.550000	11355
4	binancecoin	bnb	BNB	https://coin-images.coingecko.com/coins/images...	609.310000	8873

5 rows × 26 columns

In [121…

```python
import requests
import pandas as pd

url = "https://api.coingecko.com/api/v3/coins/markets"
params = {
    "vs_currency" : "usd",
    "order" : "market_cap_desc",
    "per_page" : 10,
    "page" : 1,
    "spakLine" : False
}

response = requests.get(url,params = params)
if response.status_code == 200 :
    data = response.json()
    df = pd.DataFrame(data)
    print("Data succesfully loaded")
else :
    print(f"Failed to load the data {response.status_code}")
```

```
df.head()
```

```
Data succesfully loaded
```

Out[121…

	id	symbol	name	image	current_price	ma
0	bitcoin	btc	Bitcoin	https://coin-images.coingecko.com/coins/images...	97041.000000	191756
1	ethereum	eth	Ethereum	https://coin-images.coingecko.com/coins/images...	3107.370000	37371
2	tether	usdt	Tether	https://coin-images.coingecko.com/coins/images...	0.998863	12986
3	solana	sol	Solana	https://coin-images.coingecko.com/coins/images...	239.550000	11355
4	binancecoin	bnb	BNB	https://coin-images.coingecko.com/coins/images...	609.310000	8873

5 rows × 26 columns

In [139…

```
import pandas as pd

df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")
print(df)
print("\nMissing values : ")
print(df.isnull().sum())

df_dropped = df.dropna()
print("\n Data after droping Missing values : ")
print(df_dropped)

df_specific_value = df.fillna(0)
print("\nFilling with Specific Value(0) ")
print(df_specific_value)

df_fill_mean = df.fillna(df.mean())
print("Filling missing values with mean: ")
print(df_fill_mean)
```

```
      age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    63.0     1.0   3     145.0   233    1        0      150      0      2.3   
1     NaN     NaN   2     130.0   250    0        1      187      0      3.5   
2    41.0     0.0   1     130.0   204    0        0      172      0      1.4   
3     NaN     1.0   1     120.0   236    0        1      178      0      0.8   
4    57.0     NaN   0     120.0   354    0        1      163      1      0.6   
..    ...     ...  ..       ...   ...  ...      ...      ...    ...      ...   
298  57.0     0.0   0     140.0   241    0        1      123      1      0.2   
299  45.0     1.0   3     110.0   264    0        1      132      0      1.2   
300  68.0     1.0   0     144.0   193    1        1      141      0      3.4   
301  57.0     1.0   0     130.0   131    0        1      115      1      1.2   
302  57.0     0.0   1     130.0   236    0        0      174      0      0.0   

     slope  ca  thal  target  
0        0   0     1       1  
1        0   0     2       1  
2        2   0     2       1  
3        2   0     2       1  
4        2   0     2       1  
..     ...  ..   ...     ...  
298      1   0     3       0  
299      1   0     3       0  
300      1   2     3       0  
301      1   1     3       0  
302      1   1     2       0  

[303 rows x 14 columns]

Missing values :
age         3
gender      2
cp          0
trestbps    1
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64

 Data after droping Missing values :
      age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    63.0     1.0   3     145.0   233    1        0      150      0      2.3   
2    41.0     0.0   1     130.0   204    0        0      172      0      1.4   
5    57.0     1.0   0     140.0   192    0        1      148      0      0.4   
7    44.0     1.0   1     120.0   263    0        1      173      0      0.0   
8    52.0     1.0   2     172.0   199    1        1      162      0      0.5   
..    ...     ...  ..       ...   ...  ...      ...      ...    ...      ...   
298  57.0     0.0   0     140.0   241    0        1      123      1      0.2   
299  45.0     1.0   3     110.0   264    0        1      132      0      1.2   
300  68.0     1.0   0     144.0   193    1        1      141      0      3.4   
```

```
301  57.0     1.0   0     130.0   131    0        1      115      1      1.2
302  57.0     0.0   1     130.0   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
2        2   0     2       1
5        1   0     1       1
7        2   0     3       1
8        2   0     3       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[299 rows x 14 columns]

Filling with Specific Value(0)
      age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    63.0     1.0   3     145.0   233    1        0      150      0      2.3
1     0.0     0.0   2     130.0   250    0        1      187      0      3.5
2    41.0     0.0   1     130.0   204    0        0      172      0      1.4
3     0.0     1.0   1     120.0   236    0        1      178      0      0.8
4    57.0     0.0   0     120.0   354    0        1      163      1      0.6
..    ...     ...  ..       ...   ...  ...      ...      ...    ...      ...
298  57.0     0.0   0     140.0   241    0        1      123      1      0.2
299  45.0     1.0   3     110.0   264    0        1      132      0      1.2
300  68.0     1.0   0     144.0   193    1        1      141      0      3.4
301  57.0     1.0   0     130.0   131    0        1      115      1      1.2
302  57.0     0.0   1     130.0   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
Filling missing values with mean:
           age    gender  cp  trestbps  chol  fbs  restecg  thalach  exang  \
0    63.000000  1.000000   3     145.0   233    1        0      150      0
1    54.413333  0.684385   2     130.0   250    0        1      187      0
2    41.000000  0.000000   1     130.0   204    0        0      172      0
3    54.413333  1.000000   1     120.0   236    0        1      178      0
4    57.000000  0.684385   0     120.0   354    0        1      163      1
..         ...       ...  ..       ...   ...  ...      ...      ...    ...
298  57.000000  0.000000   0     140.0   241    0        1      123      1
299  45.000000  1.000000   3     110.0   264    0        1      132      0
```

```
300  68.000000  1.000000   0     144.0   193    1         1       141      0
301  57.000000  1.000000   0     130.0   131    0         1       115      1
302  57.000000  0.000000   1     130.0   236    0         0       174      0

     oldpeak  slope  ca  thal  target
0        2.3      0   0     1       1
1        3.5      0   0     2       1
2        1.4      2   0     2       1
3        0.8      2   0     2       1
4        0.6      2   0     2       1
..       ...    ...  ..   ...     ...
298      0.2      1   0     3       0
299      1.2      1   0     3       0
300      3.4      1   2     3       0
301      1.2      1   1     3       0
302      0.0      1   1     2       0

[303 rows x 14 columns]
```

In [151…

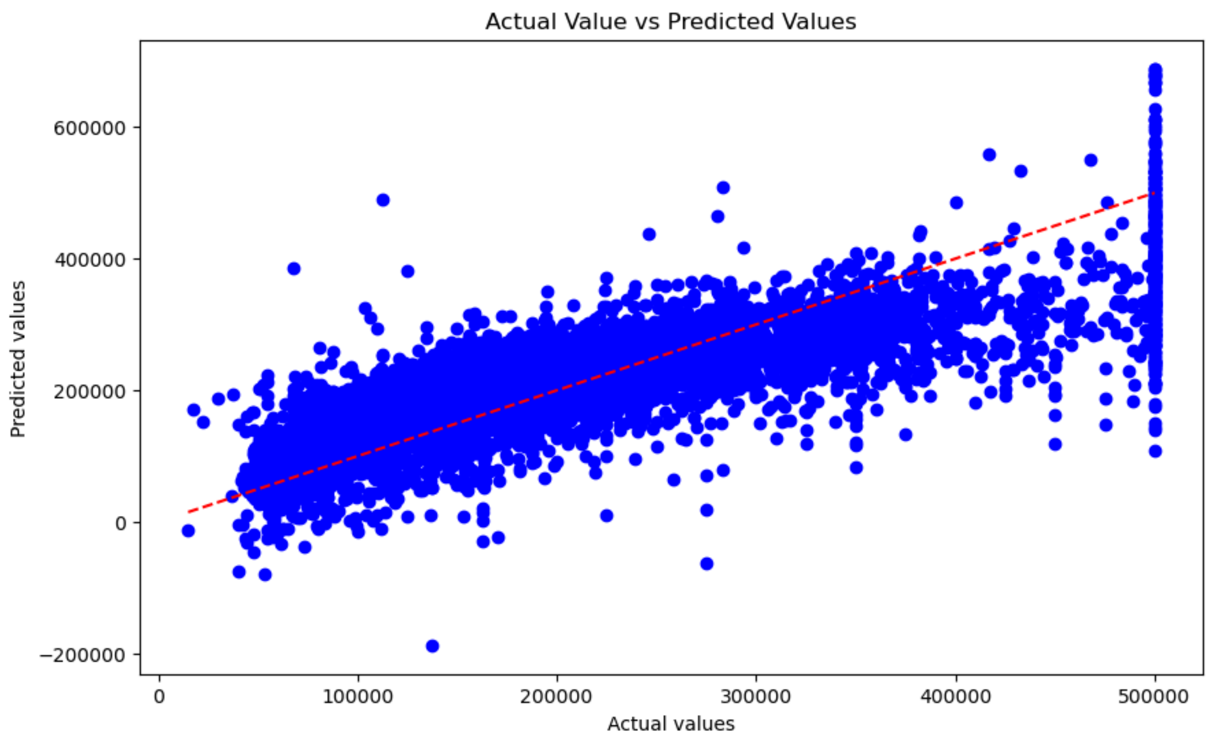
```
import pandas as pd
from scipy.stats import zscore
import numpy as np
df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")
z_score = zscore(df,nan_policy = "omit")
print(abs_z_score)
abs_z_score = np.abs(z_score)
threshold  = 2

df_cleaned = df.loc[(abs_z_score < threshold).all(axis = 1)]
print("\nData cleaned : ")
print(df_cleaned)
```

```
          age    gender        cp  trestbps      chol       fbs   restecg  \
0    0.948186  0.679091  1.973123  0.764565  0.256334  2.394438  1.005832
1         NaN       NaN  1.002577  0.091038  0.072199  0.417635  0.898962
2    1.481172  1.472556  0.032031  0.091038  0.816773  0.417635  1.005832
3         NaN  0.679091  0.032031  0.661440  0.198357  0.417635  0.898962
4    0.285634       NaN  0.938515  0.661440  2.082050  0.417635  0.898962
..        ...       ...       ...       ...       ...       ...       ...
298  0.285634  1.472556  0.938515  0.479364  0.101730  0.417635  0.898962
299  1.039471  0.679091  1.973123  1.231842  0.342756  0.417635  0.898962
300  1.500313  0.679091  0.938515  0.707525  1.029353  2.394438  0.898962
301  0.285634  0.679091  0.938515  0.091038  2.227533  0.417635  0.898962
302  0.285634  1.472556  0.032031  0.091038  0.198357  0.417635  1.005832

      thalach     exang   oldpeak     slope        ca      thal    target
0    0.015443  0.696631  1.087338  2.274579  0.714429  2.148873  0.914529
1    1.633471  0.696631  2.122573  2.274579  0.714429  0.512922  0.914529
2    0.977514  0.696631  0.310912  0.976352  0.714429  0.512922  0.914529
3    1.239897  0.696631  0.206705  0.976352  0.714429  0.512922  0.914529
4    0.583939  1.435481  0.379244  0.976352  0.714429  0.512922  0.914529
..        ...       ...       ...       ...       ...       ...       ...
298  1.165281  1.435481  0.724323  0.649113  0.714429  1.123029  1.093459
299  0.771706  0.696631  0.138373  0.649113  0.714429  1.123029  1.093459
300  0.378132  0.696631  2.036303  0.649113  1.244593  1.123029  1.093459
301  1.515125  1.435481  0.138373  0.649113  0.265082  1.123029  1.093459
302  1.064975  0.696631  0.896862  0.649113  0.265082  0.512922  1.093459

[303 rows x 14 columns]

Data cleaned :
      age  gender  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
2    41.0     0.0   1     130.0   204    0        0      172      0      1.4
7    44.0     1.0   1     120.0   263    0        1      173      0      0.0
9    57.0     1.0   2     150.0   168    0        1      174      0      1.6
10   54.0     1.0   0     140.0   239    0        1      160      0      1.2
11   48.0     0.0   2     130.0   275    0        1      139      0      0.2
..    ...     ...  ..       ...   ...  ...      ...      ...    ...      ...
293  67.0     1.0   2     152.0   212    0        0      150      0      0.8
296  63.0     0.0   0     124.0   197    0        1      136      1      0.0
298  57.0     0.0   0     140.0   241    0        1      123      1      0.2
299  45.0     1.0   3     110.0   264    0        1      132      0      1.2
302  57.0     0.0   1     130.0   236    0        0      174      0      0.0

     slope  ca  thal  target
2        2   0     2       1
7        2   0     3       1
9        2   0     2       1
10       2   0     2       1
11       2   0     2       1
..     ...  ..   ...     ...
293      1   0     3       0
296      1   0     2       0
298      1   0     3       0
299      1   0     3       0
302      1   1     2       0

[176 rows x 14 columns]
```

In [49]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv(r"C:\Users\vaibh\Downloads\housing.csv")
print("Missing values : ")
print(df.isnull().sum())
df.dropna(inplace = True)
print("Duplicate values : ")
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)

df = df.select_dtypes(include = "number")
X  = df.drop(columns = ['median_house_value'])
y = df['median_house_value']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state =
model = LinearRegression()
model.fit(X_train,y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test,y_pred)
print("\nMean squared Error",end = " ")
print(mse)
print("Mean absolute square : ",mean_absolute_error(y_test,y_pred))
print("r2 score : ",r2_score(y_test,y_pred))



plt.figure(figsize = (10,6))
plt.scatter(y_test,y_pred,color = "blue")
plt.title("Actual Value vs Predicted Values")
plt.xlabel("Actual values")
plt.ylabel("Predicted values")
plt.plot([y_test.min(),y_test.max()],[y_test.min(),y_test.max()],color = "red",line
plt.show()
```

```
Missing values :
longitude              0
latitude               0
housing_median_age     0
total_rooms            0
total_bedrooms       207
population             0
households             0
median_income          0
median_house_value     0
ocean_proximity        0
dtype: int64
Duplicate values :
0

Mean squared Error 4738972791.400478
Mean absolute square :  50704.919692847856
r2 score :  0.6445130291082346
```



In [45]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import pandas as pd

df = pd.read_csv(r"C:\Users\vaibh\Downloads\heart (1).csv")
print("Missing values : ")
print(df.isnull().sum())
df.dropna(inplace = True)
print("\nDuplicate values : ")
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)

X = df.iloc[:,0:13]
y = df['target']
```

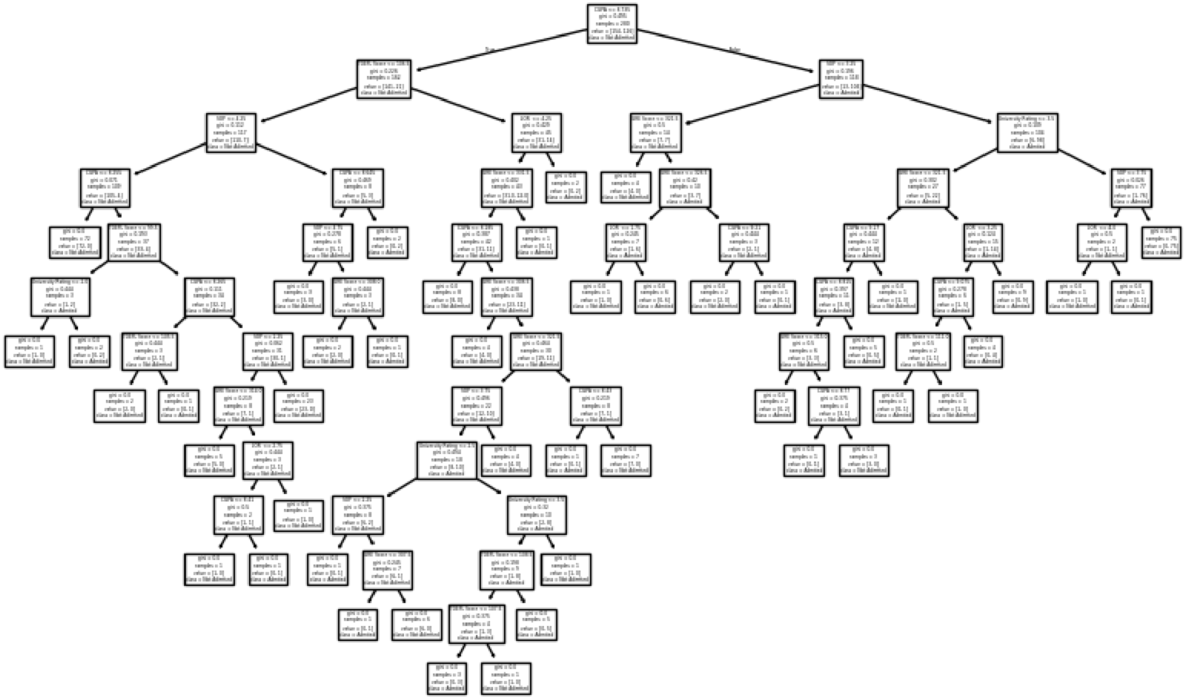
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state =
model = LogisticRegression(max_iter = 1000)
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
classification = classification_report(y_test,y_pred)
confusion = confusion_matrix(y_test,y_pred)
accuracy = accuracy_score(y_test,y_pred)

print("\nAccuracy Score : ")
print(accuracy)
print("\nClassification Report : ")
print(classification)
print("\nconfusion Matrix : ")
print(confusion)
```

```
Missing values :
age         3
gender      2
cp          0
trestbps    1
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64

Duplicate values :
1

Accuracy Score :
0.8333333333333334

Classification Report :
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        36
           1       0.84      0.89      0.86        54

    accuracy                           0.83        90
   macro avg       0.83      0.82      0.82        90
weighted avg       0.83      0.83      0.83        90


confusion Matrix :
[[27  9]
 [ 6 48]]
```

In [92]:
```
import pandas as pd
from sklearn.tree import  DecisionTreeClassifier,plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
import matplotlib.pyplot as plt

df = pd.read_csv(r"C:\Users\vaibh\Downloads\Admission_Predict.csv")
df.dropna(inplace = True)
df.drop_duplicates(inplace = True)
df = df.select_dtypes(include = "number")
df = df.drop(columns = ['Serial No.'])
df['Admit'] = (df['Chance of Admit '] >= 0.75).astype(int)
df = df.drop(columns = ['Chance of Admit '])
X = df.drop(columns = ['Admit'])
y = df['Admit']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state =
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
classification = classification_report(y_test,y_pred)
confusion = confusion_matrix(y_test,y_pred)
print("\nConfusion Matrix")
print(confusion)
print("\nClssification Report")
print(classification)
```

```
Confusion Matrix
[[59  7]
 [13 41]]

Clssification Report
              precision    recall  f1-score   support

           0       0.82      0.89      0.86        66
           1       0.85      0.76      0.80        54

    accuracy                           0.83       120
   macro avg       0.84      0.83      0.83       120
weighted avg       0.84      0.83      0.83       120
```

In [98]:
```
plt.figure(figsize = (10,6))
plot_tree(clf,feature_names = X.columns,class_names = ['Not Adimited','Admited'])
plt.show()
```



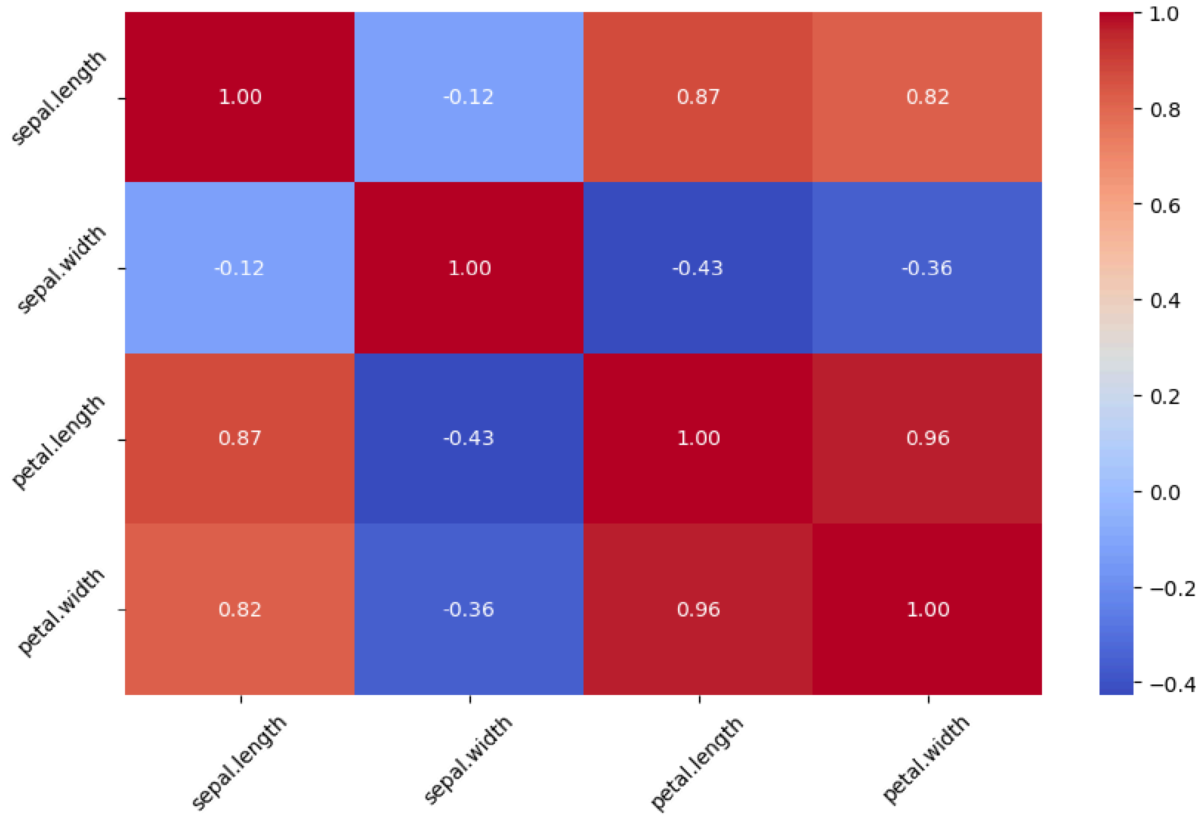
In [7]:
```
from scipy.stats import ttest_1samp

sample = [23,24,25,26,27,28]
population_mean = 25

t_stat,p_value = ttest_1samp(sample,population_mean)
print(t_stat)
print(p_value)
alpha = 0.05
if p_value < alpha :
    print("Reject the nulll hypothes : there is significant diiffrence bettween pop
else :
    print("Fail to reject the null hypothesis : There is no ignificaant diffrence b
```

```
0.6546536707079772
0.5416045607931204
Fail to reject the null hypothesis : There is nos ignificaant diffrence between samp
le mean and population mean
```

In [25]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\vaibh\Downloads\iris.csv")
print("missing values : ")
print(df.isnull().sum())
print("Duplicates : ")
print(df.duplicated().sum())
df.dropna(inplace = True)
df.drop_duplicates(inplace =True)
corelation_matrix = df.select_dtypes(include = "number").corr()
plt.figure(figsize=(10,6))
```

```
sns.heatmap(corelation_matrix,annot = True,cmap = 'coolwarm',fmt = '0.2f',cbar = Tr
plt.xticks(rotation = 45)
plt.yticks(rotation = 45)
plt.show()
```

```
missing values :
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
species         0
dtype: int64
Duplicates :
1
```




```
In [ ]:
```