UCS2202 – Foundations of Data Science

Assignment – 2

Date : 09.06.24

Section : CSE B

By - Mokitha S (3122 23 5001 083), Ramlath Nisha A (3122 23 5001 106)

(1) Identify your own dataset to perform any one of the data modeling such as

1) Regression

2) Classification

3) Clustering

The main goal of this project is to develop Machine Learning models capable of accurately classifying patients based on the risk of a heart attack. In this notebook, we will explore a medical dataset containing patient information and whether they experienced the target disease or not.

The dataset is labeled, with the target variable representing the occurrence (target = 1) or absence (target = 0) of a heart attack in the patient. The dataset used in this project contains various medical characteristics that may be relevant for predicting the risk of a heart attack. The goal is to use this information to determine whether a patient is more likely to suffer a heart attack. Below are more details about the dataset:

age: Age of the patient in years

sex: Gender of the patient (0: female, 1: male)

cp: Type of chest pain (0: Typical angina, 1: Atypical angina, 2: Non-anginal pain, 3: Asymptomatic)

trestbps: Resting blood pressure in mmHg

chol: Serum cholesterol in mg/dl

fbs: Fasting blood sugar level categorized as above 120 mg/dl (0: false, 1: true)

restecg: Resting electrocardiographic results (0: Normal, 1: ST-T wave abnormality, 2: Showing probable or definite left ventricular hypertrophy)

thalach: Maximum heart rate achieved during a stress test

exng: Exercise induced angina (0: no, 1: yes)

oldpeak: ST depression induced by exercise relative to rest (unit -> depression)

slope: Slope of the peak exercise ST segment (0: Upsloping, 1: Flat, 2: Downsloping)

ca: Number of major vessels (0-4) colored by fluoroscopy

thall: Thalium stress test result (0: Normal, 1: Fixed defect, 2: Reversible defect, 3: Not described)

target: Heart disease status (0: no disease, 1: presence of disease

(2) Develop Python code to perform the following:

- **To read the dataset**

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('whitegrid')
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

```
global df
df=pd.read_csv("C:\\Users\\mokit\\OneDrive\\Documents\\SSN\\Data Science\\heart.csv")
df.rename(columns={
    "age":"Age",
    "sex":"Sex",
    "cp":"ChestPain",
    "trestbps":"RestingBloodPressure",
    "chol":"Cholesterol",
    "fbs":"FastingBloodSugar",
    "restecg":"RestingECG",
    "thalach":"MaxHeartRate",
    "exang":"ExcerciseAngina",
    "oldpeak":"OldPeak",
    "slope":"STSlope",
    "ca":"nMajorVessels",
    "thal":"Thalium",
    "target":"Status"
}, inplace=True)
df.head()
```

Here, the columns are renamed to make it more meaningful.

```
print("Dimensionality of original data:",df.shape)
print("Number of duplicate records:",df.duplicated().sum())
#There are 723 duplicate records and these must be removed
df = df.drop_duplicates()
print("Dimensionality of new data:",df.shape)
```

Duplicate rows are removed here.

```
df.describe()
```

Output:

	Age	Sex	ChestPain	RestingBloodPressure	Cholesterol	FastingBloodSugar	RestingECG	MaxHeartRate	ExcerciseAngina	C
0	52	1	0	125	212	0	1	168	0	
1	53	1	0	140	203	1	0	155	1	
2	70	1	0	145	174	0	1	125	1	
3	61	1	0	148	203	0	1	161	0	
4	62	0	0	138	294	1	1	106	0	

OldPeak	STSlope	nMajorVessels	Thalium	Status
1.0	2	2	3	0
3.1	0	0	3	0
2.6	0	0	3	0
0.0	2	1	3	0
1.9	1	3	2	0

```
Dimensionality of original data: (1025, 14)
Number of duplicate records: 723
Dimensionality of new data: (302, 14)
```

- **To check for null values**

Code:

```
def check_null(feature):
    if df[feature].isnull().any():
        print(f"Feature '{feature}' contains null values")
        df[feature] = df[feature].fillna(df[feature].mean())
    else:
        print("No null values found in",feature)
for feature in df.columns:
    check_null(feature)
```

Output:

```
No null values found in Age
No null values found in Sex
No null values found in ChestPain
No null values found in RestingBloodPressure
No null values found in Cholesterol
No null values found in FastingBloodSugar
No null values found in RestingECG
No null values found in MaxHeartRate
No null values found in ExcerciseAngina
No null values found in OldPeak
No null values found in STSlope
No null values found in nMajorVessels
No null values found in Thalium
No null values found in Status
```

- **To handle missing values**

The above function check_null also handles missing values by filling them with the mean of the feature.

(3) Examine a few approaches suitable for the identified dataset based on the exploratory data analysis.

Checking for Outliers

Code:

```
def find_outliers(df,feature):
    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)

    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    print("Lower Bound:", lower_bound)
    print("Upper Bound:", upper_bound)

    outliers = df[(df[feature] < lower_bound) | (df[feature] >
upper_bound)]
    print("Outliers:", len(outliers))

    df = df[(df[feature] >= lower_bound) & (df[feature] <=
upper_bound)]
```

```
    return df

outliers={}

def plot(feature):
    fig, ax = plt.subplots(1, 1, figsize=(12, 3))
    plt.ylabel('Frequency')
    plt.xlabel(feature)
    plt.hist(np.array(df[feature]),
             bins = 10,
             density = True)
    ax.axvline(x = np.mean(df[feature]),
               color = [1, 0, 0])
    plt.tight_layout()
    plt.show()
```

Plot Value Counts for Categorical Features:

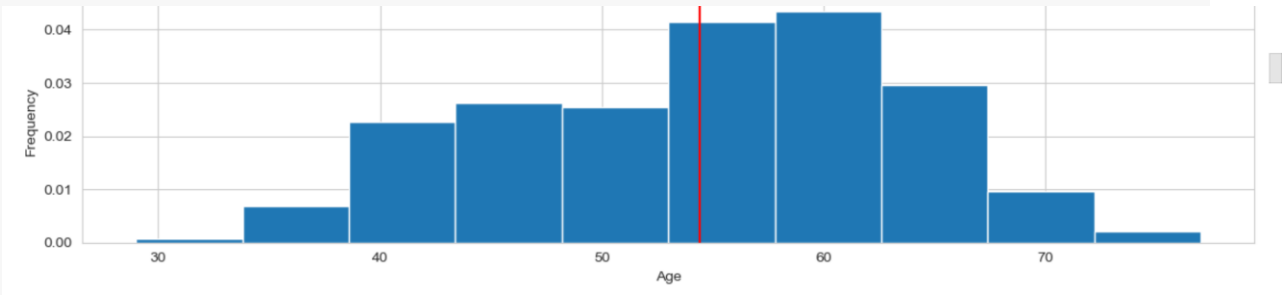
```
def plot_value_counts(feature):
    value_counts = df[feature].value_counts()

    fig, ax = plt.subplots(figsize=(12, 3))
    value_counts.plot(kind='bar', ax=ax)
    ax.set_xlabel(feature)
    ax.set_ylabel('Frequency')
    ax.set_title(f'Value Counts of {feature}')
    plt.xticks(rotation=90)
    plt.show()
```

- **Numerical feature: Age**

Code:

```
plot('Age')
df = find_outliers(df,'Age')
```

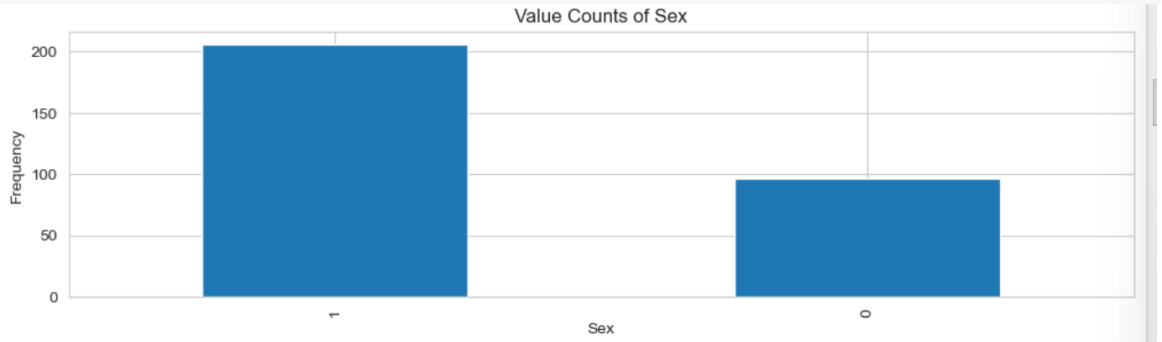
Output:




```
Lower Bound: 28.5
Upper Bound: 80.5
Outliers: 0
```

```
plot_value_counts('Sex')

print('''1: Male
0: Female''')
```

Output:



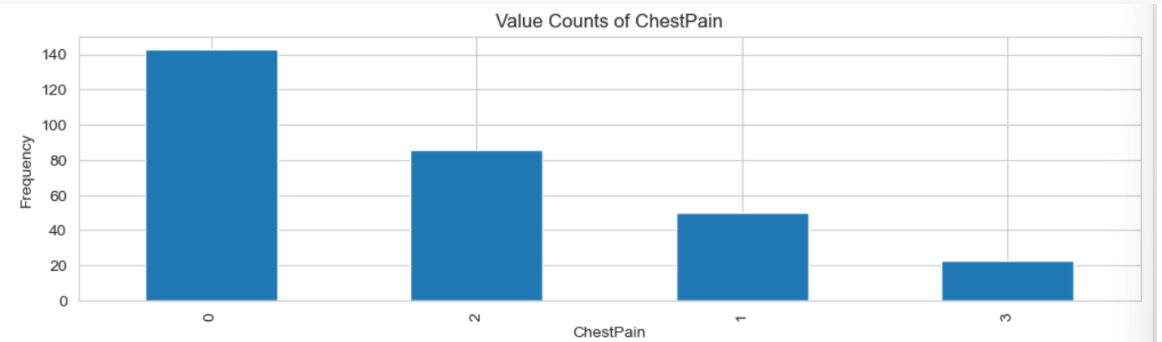

```
1: Male
0: Female
```

- **Categorical feature: Chest pain**

Code:

```
plot_value_counts('ChestPain')
print('''0: Typical angina
1: Atypical angina
2: Non-anginal pain
3: Asymptomatic''')
```

Output:




```
0: Typical angina
1: Atypical angina
2: Non-anginal pain
3: Asymptomatic
```

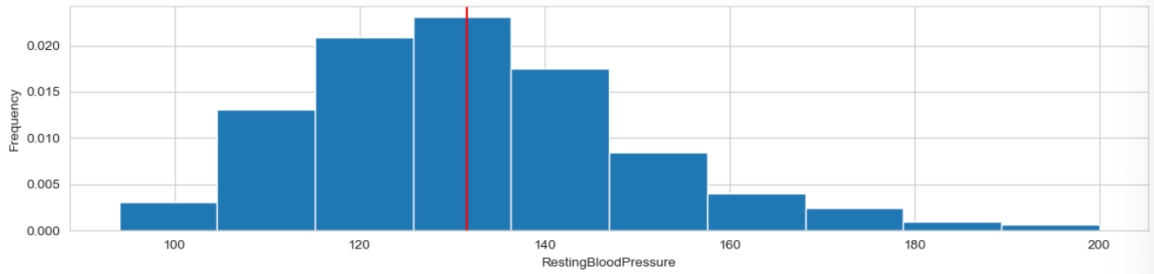
- **Numerical feature: RestingBloodPressure**

Code:

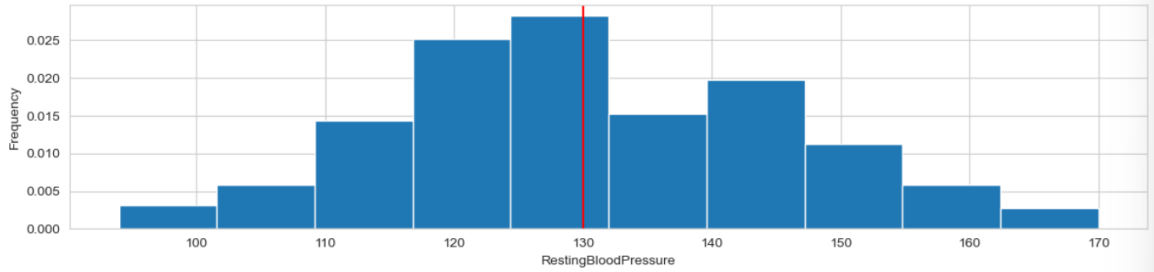
```
print("Distribution before removing outliers")
plot('RestingBloodPressure')
df = find_outliers(df,'RestingBloodPressure')

print("Distribution after removing outliers")
plot('RestingBloodPressure')
```

Output:

```
Distribution before removing outliers
```




```
Lower Bound: 90.0
Upper Bound: 170.0
Outliers: 9
Distribution after removing outliers
```



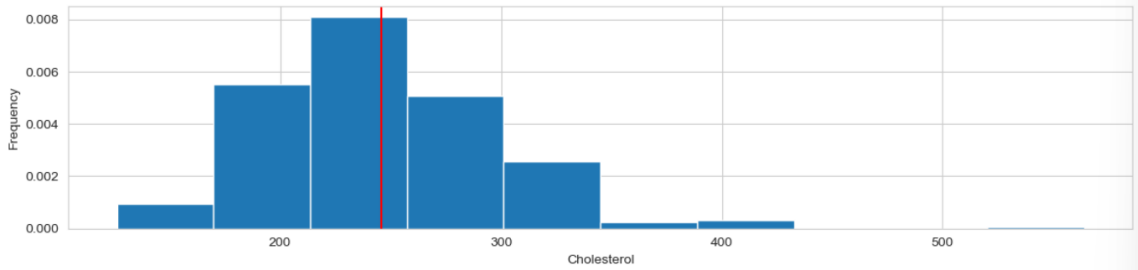
- **Numerical feature: Cholestrol**

Code:

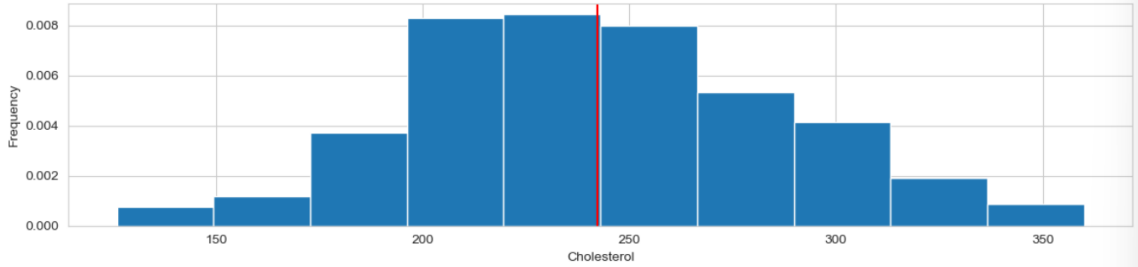
```
print("\nDistribution before removing outliers:")
plot('Cholesterol')
df = find_outliers(df,'Cholesterol')

print("Distribution after removing outliers:")
plot('Cholesterol')
```

Output:

```
Distribution before removing outliers:
```




```
Lower Bound: 116.5
Upper Bound: 368.5
Outliers: 5
Distribution after removing outliers:
```

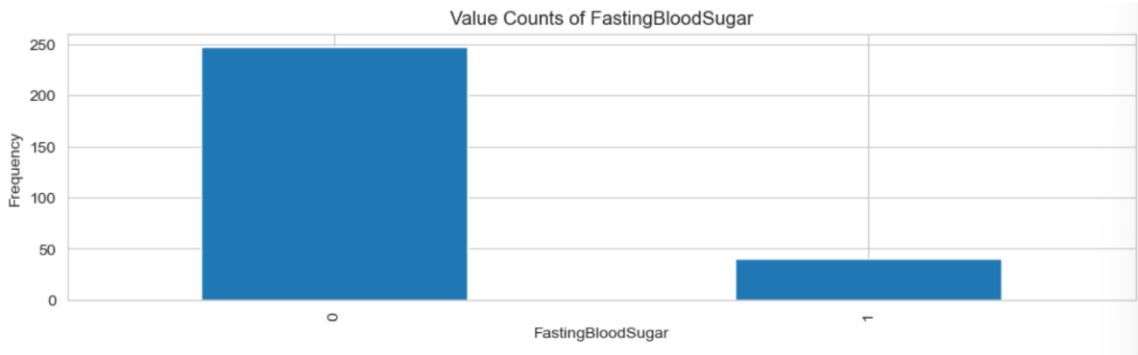


- **Categorical feature: Chest pain**

Code:

```
plot_value_counts('FastingBloodSugar')
print('''0: False
1: True''')
```

Output:



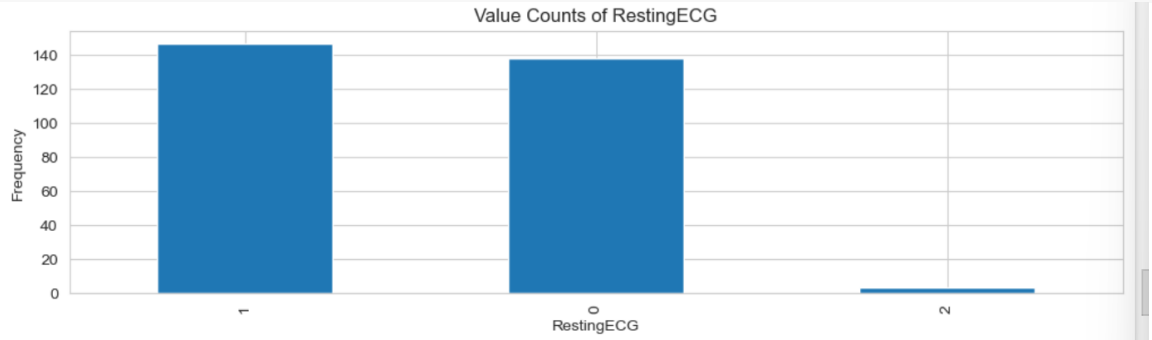

```
0: False
1: True
```

- **Categorical feature: Resting ECG**

Code:

```
plot_value_counts('RestingECG')
print('''0:Normal
1:Abnormality
2:Hypertrophy''')
```

Output:




```
0:Normal
1:Abnormality
2:Hypertrophy
```

- **Numerical feature: Cholestrol**

Code:

```
print("\nDistribution before removing outliers:")
plot('MaxHeartRate')
df = find_outliers(df,'MaxHeartRate')

print("Distribution after removing outliers:")
plot('MaxHeartRate')
```

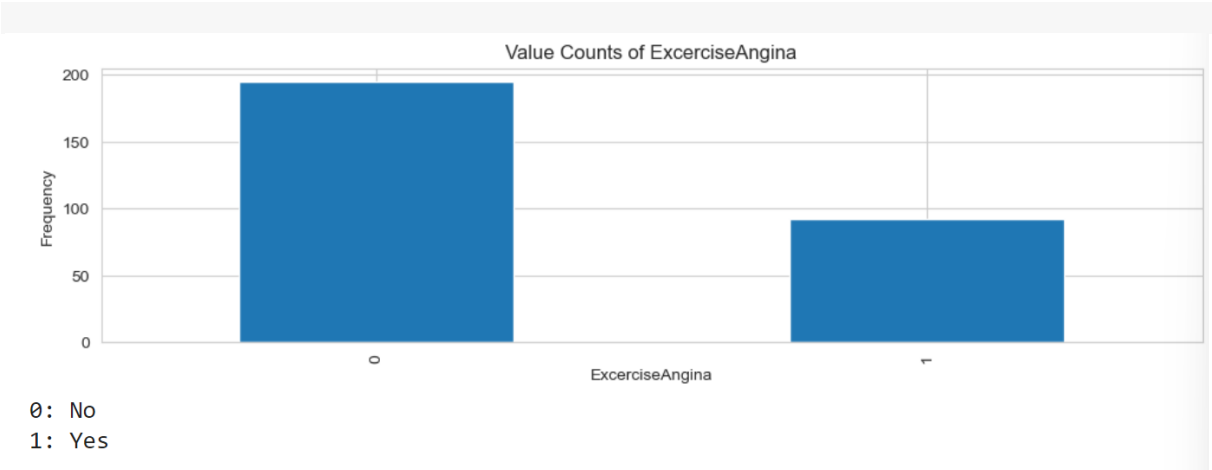
Output:

- **Categorical feature: ExcerciseAngina**

Code:

```
plot_value_counts('ExcerciseAngina')
print('''0: No
1: Yes''')
```

Output:



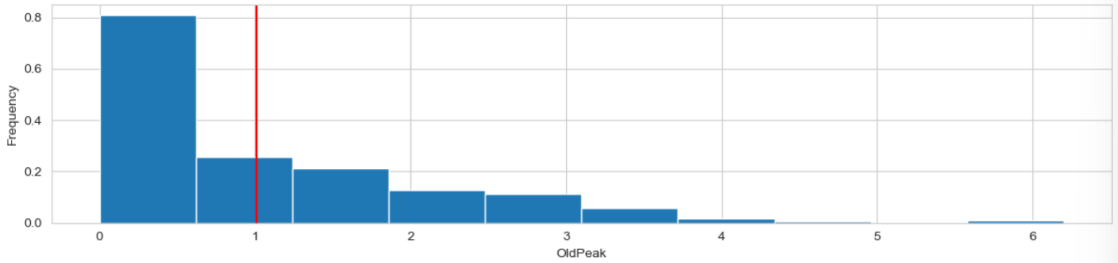
- **Numerical feature: OldPeak**

Code:

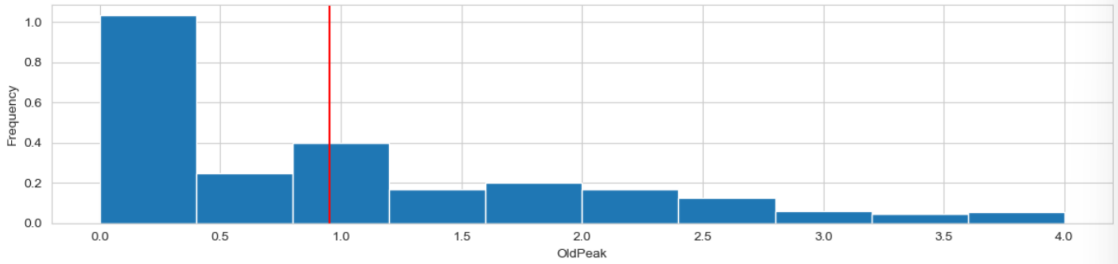
```
print("\nDistribution before removing outliers:")
plot('OldPeak')
df = find_outliers(df,'OldPeak')

print("Distribution after removing outliers:")
plot('OldPeak')
```

Output:

```
Distribution before removing outliers:
```




```
Lower Bound: -2.4000000000000004
Upper Bound: 4.0
Outliers: 4
Distribution after removing outliers:
```

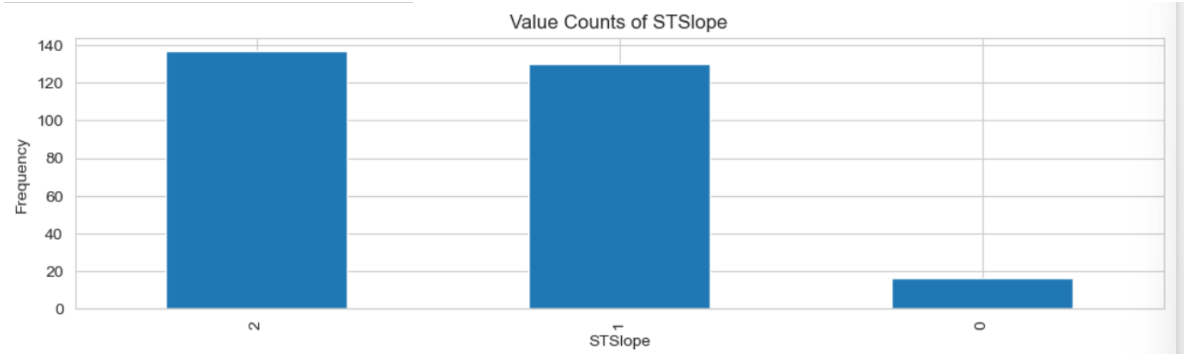


- **Categorical feature: STSlope**

Code:

```
plot_value_counts('STSlope')
print('''0:Upsloping
1:Flat
2:Downsloping''')
```

Output:



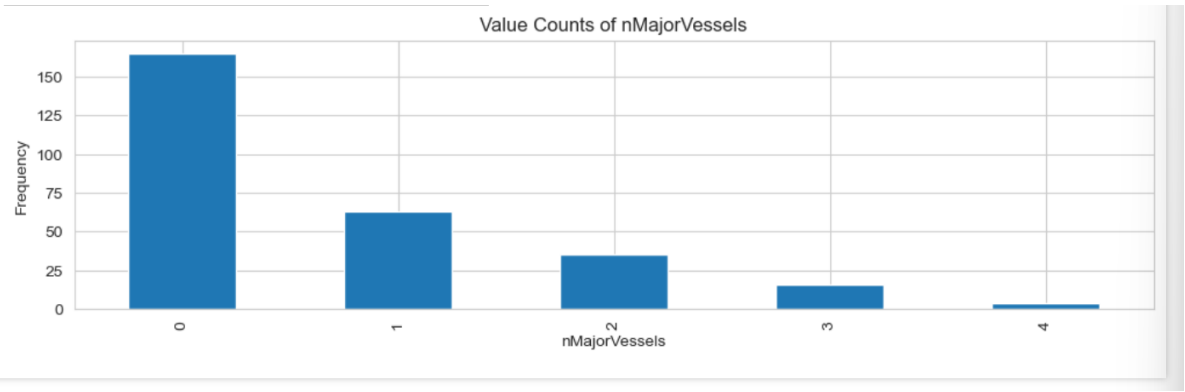

```
0:Upsloping
1:Flat
2:Downsloping
```

- **Categorical feature: nMajorVessels**

Code:

```
plot_value_counts('nMajorVessels')
```

Output:



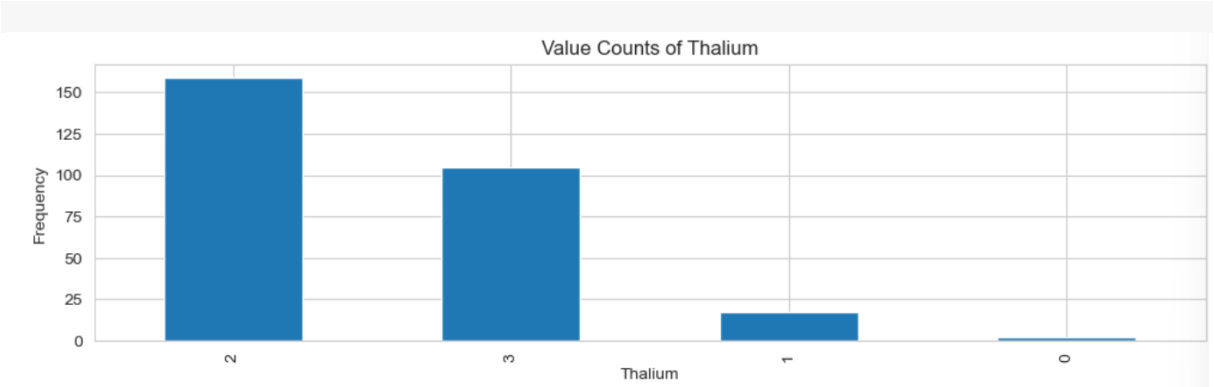
- **Categorical feature: Thalium**

Code:

```
plot_value_counts('Thalium')
print('''0:Normal
1:Fixed defect
```

```
2:Reversible defect
3:Not described''')
```

Output:



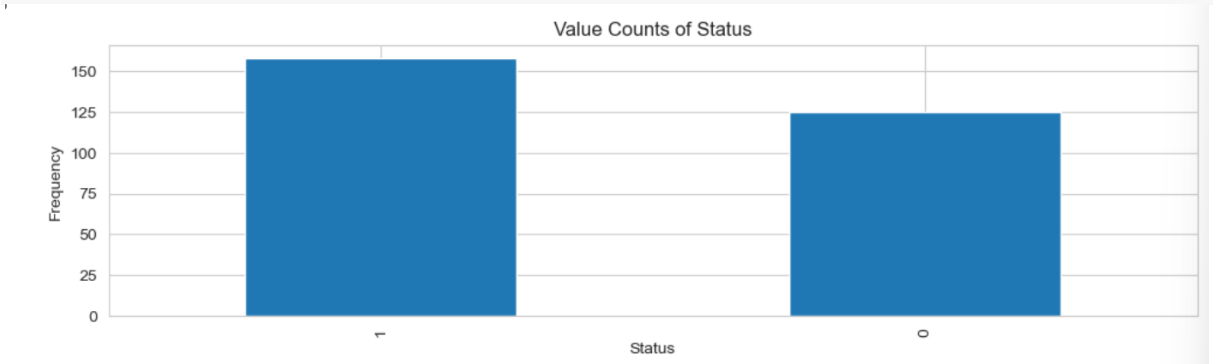

```
0:Normal
1:Fixed defect
2:Reversible defect
3:Not described
```

- **Categorical feature: Status (Target variable)**

Code:

```
plot_value_counts('Status')
print('''0: No disease
1: Heart disease''')
```

Output:



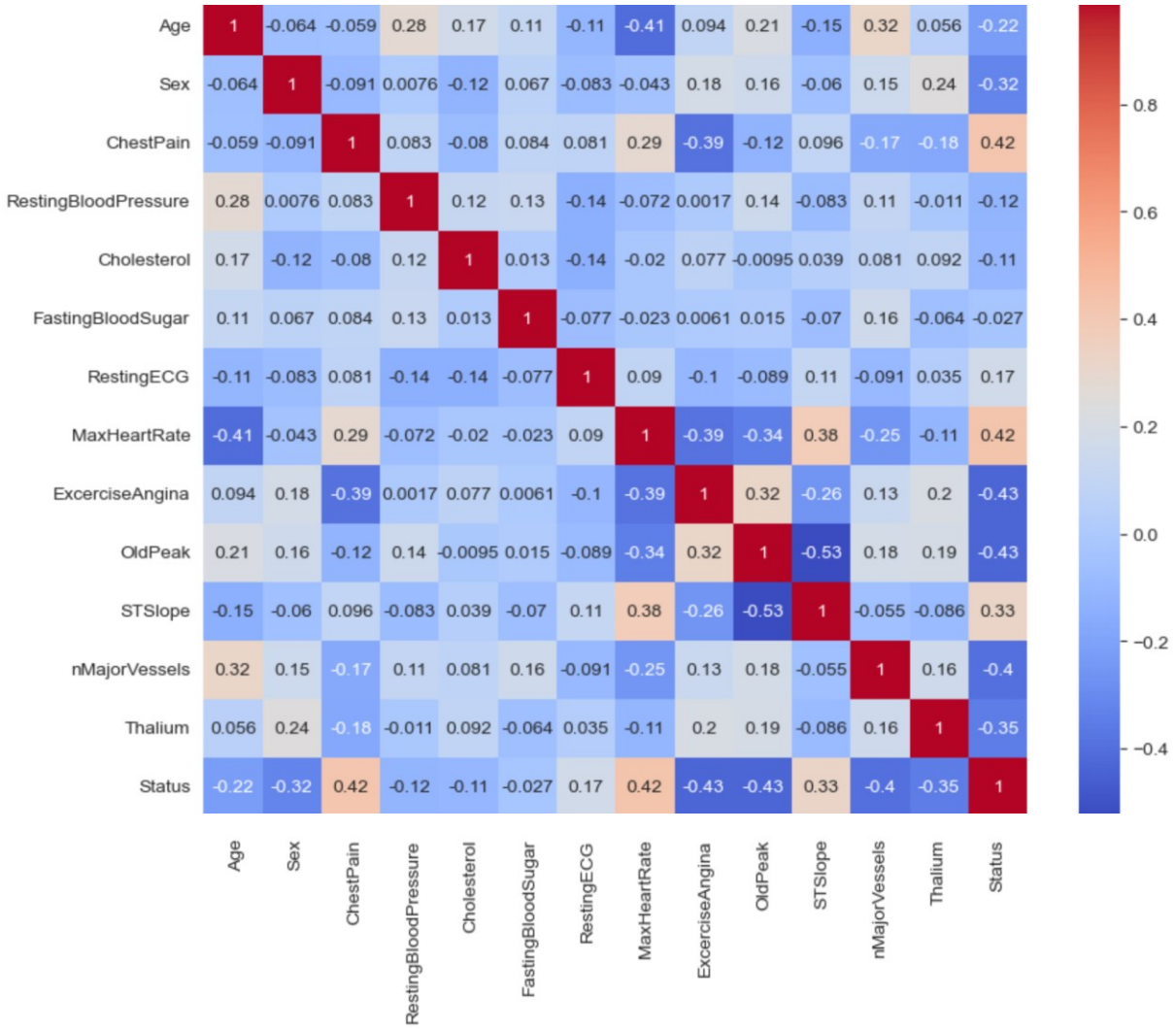

```
0: No disease
1: Heart disease
```

Correlation Analysis:

Code:

```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

Output:



(4) Choose an appropriate approach that can be used for model building.

After exploratory data analysis, a classification model like Logistic Regression or Random Forest is suitable for this dataset.

Logistic regression is suitable for the heart disease dataset because:

Binary Classification: The dependent variable; Status is dichotomous, and this model addresses exactly such a variable – the binary variable.

Interpretability: Dutifully gives a direct chance estimate of how each feature causes the risk probability of heart diseases.

Baseline Model: This is an easy and effective way for the start of Model from Classification problem.

Probabilistic Outputs: Provides probability measure for a prediction which can be used in medical diagnostics.

Handling Mixed Features: Can work with features which have to be encoded before are categorical features or the ones which have to be normalized or standardized such as continuous features.

In conclusion, the usage of logistic regression in this case is rational and well-founded because it is easy to interpret, relatively simple, and capable of providing a satisfactory level of accuracy.

(5) Develop Python code to build the model.

Logistic Regression Model

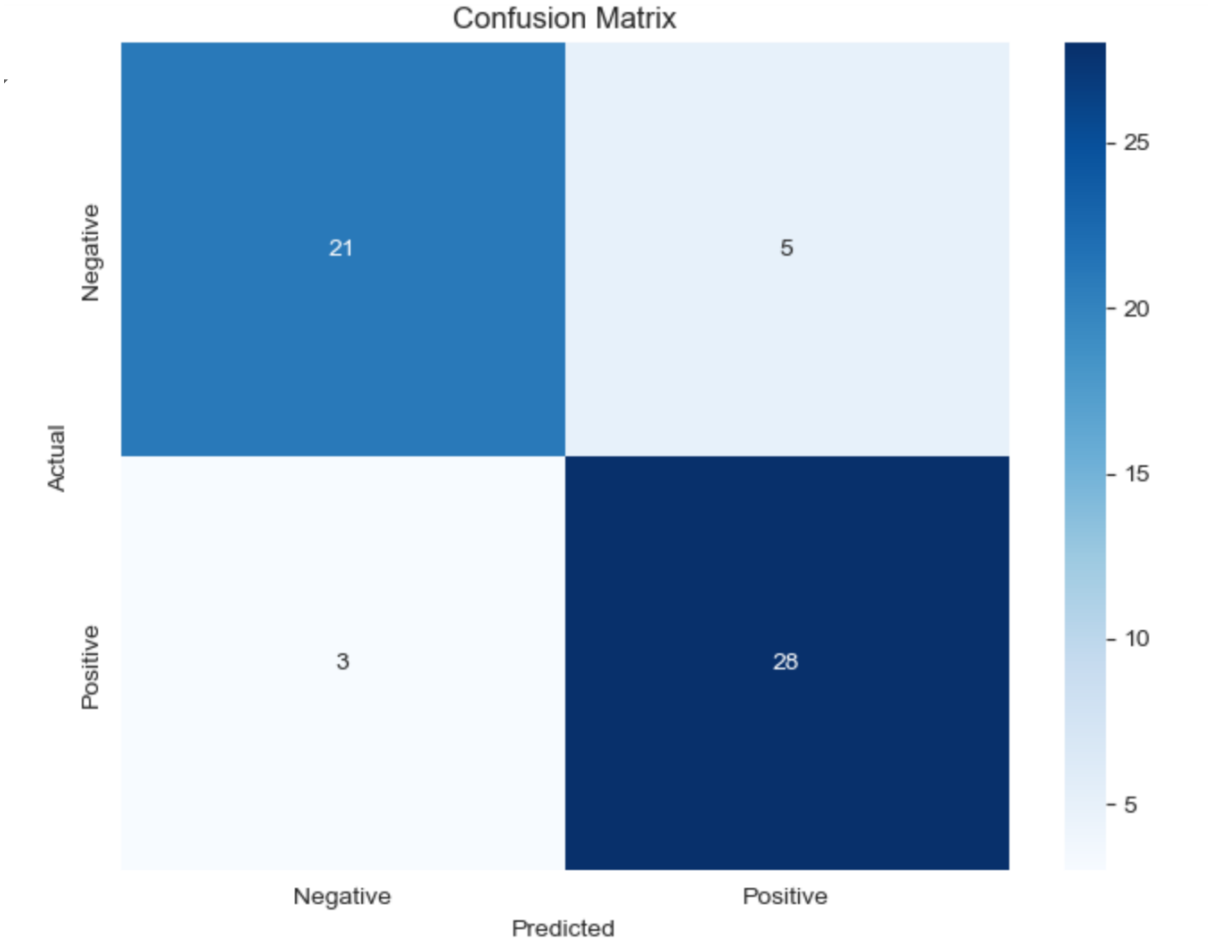
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
X = df.drop('Status', axis = 1)
y = df['Status']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=42)
model = LogisticRegression(random_state=42, max_iter=1000)
model.fit(X_train, y_train)
```

(6) Evaluate the built model using appropriate evaluation metrics.

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues',
            xticklabels=['Negative', 'Positive'],
            yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output:

```
Accuracy: 0.8596491228070176
Precision: 0.8484848484848485
Recall: 0.9032258064516129
F1 Score: 0.875
Confusion Matrix:
 [[21  5]
 [ 3 28]]
```


Confusion Matrix
Actual
Negative
Positive
21
5
3
28
Negative
Positive
Predicted
25
20
15
10
5