HEART FAILURE PREDICTION

(Course Name: Introduction to Python Programming Lab)

(Course Code: 20CS3352)

A Python Project Report on Diabetes Prediction

Submitted by

C.S.V.S.SUBRAMANYAM (22501A0533)

G.GOWTHAM(22501A0549)

D.DELISHA(22501A0536)

B.KIRAN KUMAR(22501A0520)

I.NITHYANANDA REDDY(22501A0566)

II B.Tech I Sem

in

Computer Science and Engineering

Prasad V Potluri Siddhartha Institute of Technology

Accredited with A+ grade by NAAC, NBA
Accredited, and Autonomous ISO 9001:2015
Certified Institute
Permanently Affiliated to JNTUK-Kakinada and approved by
AICTE

Kanuru, Vijayawada 520007

Prasad V Potluri Siddhartha Institute of Technology

Accredited with A+ grade by NAAC, NBA

Accredited, and Autonomous ISO 9001:2015

Certified Institute

Permanently Affiliated to JNTUK-Kakinada and approved by AICTE

Kanuru, Vijayawada-520007

CERTIFICATE

This is to certify that the python project report titled "**HEART FAILURE Prediction"** of **Mr C.S.V.S.Subramanyam(22501A0533),Mr G.Gowtham(22501A0549), Miss D.Delisha (22501A0536) ,Mr B.Kiran Kumar (22501A0520), Mr I.Nithyanandha Reddy(22501A0566)** in Computer Science and Engineering during the academic year 2023-2024.

Signature of the Guide **Signature of the H.O.D**

TABLE OF CONTENTS

1. Introduction

- Background and Motivation

Among all fatal disease, heart attack diseases are considered as the most prevalent. Medical practitioners conduct different surveys on heart diseases and gather information of heart patients, their symptoms and disease progression. Increasingly are reported about patients with common diseases who have typical symptoms.

In this fast moving world people want to live a very luxurious life so they work like a machine in order to earn lot of money and live a comfortable life therefore in this race they forget to take care of themselves, because of this there food habits change their entire lifestyle change.In this type of lifestyle they are more tensed they have blood pressure, sugar at a very young age and they don't give enough rest for themselves and eat what they get and they even don't bother about the quality of the food if sick the go for their own medication as a result of all these small negligence it leads to a major threat that is the heart disease.

The term 'heart disease' includes the diverse diseases that affect heart. The number of people suffering from heart disease is on the rise (health topics, 2010). The report from world health organization shows us a large number of people that die every year due to the heart disease all over the world. Heart disease is also stated as one of the greatest killers in Africa.

- Problem Statement

With a plethora of medical data available and the rise of Data Science, a host of startups are taking up the challenge of attempting to create indicators for the for seen diseases that might be contracted! Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help. In this way, we try to solve automate another problem that occurs in the nature with a view to counter it and focus on to the next problem with the help of AI techniques!

Aim :

- To classify / predict whether a patient is prone to heart failure depending on multiple attributes.
- It is a binary classification with multiple numerical and categorical features.

- Relevant Previous Work

Relevant previous work related to heart failure prediction encompasses a range of studies and research efforts focused on identifying risk factors, developing predictive models, and improving the accuracy of stroke risk assessments.

3. Methodology:

- Data Collection

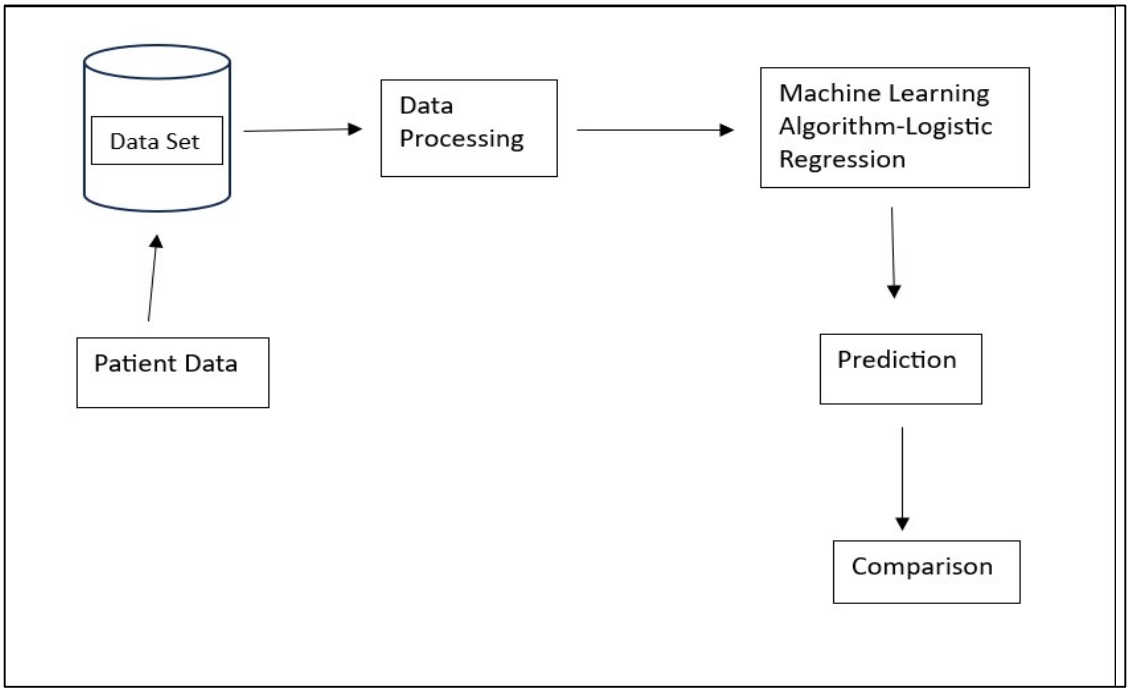
The dataset for stroke prediction is from Kaggle. This particular dataset has 918 rows and 12 columns. The columns have 'age', 'sex', 'chest pain type', 'Resting BP', 'Cholesterol', 'Fasting BS', 'Resting ECG', 'Max HR', 'Exercise Angina', 'old peak', 'ST_ slope' and 'heart disease' as the main attributes. The output column 'Heartdisease' has the value as either '1' or '0'. The value '0' indicates no heart failure risk detected, whereas the value '1' indicates a possible risk of heart failure. **Oldpeak's** data distribution is rightly skewed. **Cholestrol** has a bid modal data distribution.

- Data Pre-processing

Data Preprocessing is required before model building to remove the unwanted noise and outliers from the dataset, resulting in a deviation from proper training. Anything that interrupts the model from performing with less efficiency is taken care of in this stage. After collecting the appropriate dataset, the next step lies in cleaning the data and making sure that it is ready for model building. The dataset taken has 12 attributes. Firstly, the column 'age' is dropped because its existence does not make much difference in model building. Then the dataset is checked for null values and filled if any found. In this case, no column has null values. The data is unbalanced. The dataset chosen for the task of heart failure prediction should be balanced. The entire dataset has 918 rows, of which 12 rows are suggesting 518 occurrence of a heart failure and 410 rows having the possibility of no failure. Training a machine-level model with such data might give accuracy, but other accuracy metrics like precision and recall are shallow. If such imbalanced data is not handled, the results are not accurate, and the prediction is inefficient. Therefore, to get an efficient model, this imbalanced data is to be first handled. For this purpose, the method of under sampling is used. Under sampling balances the data wherein the majority class is under sampled to match the minority class. In this case, the class resulting dataset will have 410 rows with value '0' and 508 rows with value '1'.

4. Project Design:

- Data Flow Diagram



Implementation:

- Code Development

Data Preprocessing

Step 4: Analysing the dataset.

1. shape of the graph
2. columns of the graphs.
3. checking null values
4. checking the presence of duplicates
5. checking mean,min,etc. using describe function.

```
[ ]  #shape of the graph
     heart.shape
```

```
(918, 12)
```

```
[ ]  #columns of the graphs.
     heart.columns
```

```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

```
#checking null values
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

```
heart.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	918.0	53.510893	9.432617	28.0	47.00	54.0	60.0	77.0
RestingBP	918.0	132.396514	18.514154	0.0	120.00	130.0	140.0	200.0
Cholesterol	918.0	198.799564	109.384145	0.0	173.25	223.0	267.0	603.0
FastingBS	918.0	0.233115	0.423046	0.0	0.00	0.0	0.0	1.0
MaxHR	918.0	136.809368	25.460334	60.0	120.00	138.0	156.0	202.0
Oldpeak	918.0	0.887364	1.066570	-2.6	0.00	0.6	1.5	6.2
HeartDisease	918.0	0.553377	0.497414	0.0	0.00	1.0	1.0	1.0

Step 5: Dividing numerical and categorial data

1. Here the value which are unique in the categorical data are atmost 4.
2. So if we calculate the unique value count and get more than 6(let's consider) then we can consider it as numerical data else categorical.

```
col = list(heart.columns)
categorical_features = []
numerical_features = []
for i in col:
    if len(heart[i].unique()) > 6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)

print('Categorical Features :',*categorical_features)
print('Numerical Features :',*numerical_features)
```

```
Categorical Features : Sex ChestPainType FastingBS RestingECG ExerciseAngina ST_Slope HeartDisease
Numerical Features : Age RestingBP Cholesterol MaxHR Oldpeak
```

Step 6: creating a deep copy of our data set and converting all the columns to numerical values

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
deep_copy_heart = heart.copy(deep = True)

deep_copy_heart['Sex'] = le.fit_transform(deep_copy_heart['Sex'])
deep_copy_heart['ChestPainType'] = le.fit_transform(deep_copy_heart['ChestPainType'])
deep_copy_heart['RestingECG'] = le.fit_transform(deep_copy_heart['RestingECG'])
deep_copy_heart['ExerciseAngina'] = le.fit_transform(deep_copy_heart['ExerciseAngina'])
deep_copy_heart['ST_Slope'] = le.fit_transform(deep_copy_heart['ST_Slope'])
```

```
deep_copy_heart
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	1	1	140	289	0	1	172	0	0.0	2	0
1	49	0	2	160	180	0	1	156	0	1.0	1	1
2	37	1	1	130	283	0	2	98	0	0.0	2	0
3	48	0	0	138	214	0	1	108	1	1.5	1	1
4	54	1	2	150	195	0	1	122	0	0.0	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	1	3	110	264	0	1	132	0	1.2	1	1
914	68	1	0	144	193	1	1	141	0	3.4	1	1
915	57	1	0	130	131	0	1	115	1	1.2	1	1
916	57	0	1	130	236	0	0	174	0	0.0	1	1
917	38	1	2	138	175	0	1	173	0	0.0	2	0

918 rows × 12 columns

Data Visualization

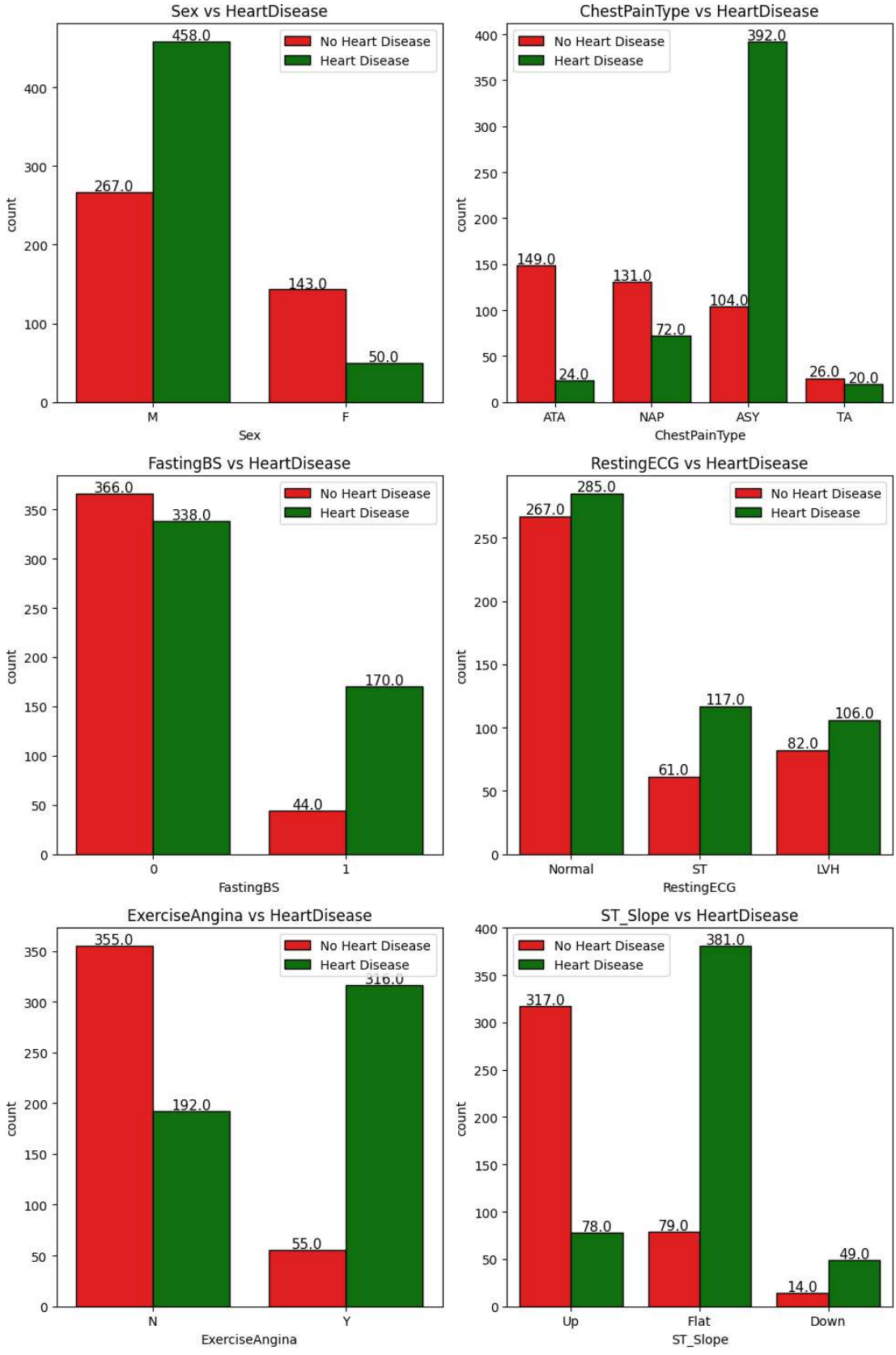
Step 7: visualizing the count of the total no. of heart diseased people along with people without heart disease

```
[ ] heart_disease_counts = heart['HeartDisease'].value_counts()

    # Define the colors. '1' for heart disease is red, '0' for healthy heart is any other color (e.g., green)
    colors = ['red' if label == 1 else 'green' for label in heart_disease_counts.index]

    fig, axs = plt.subplots(1, 2, figsize=(12, 6))  # 1 row, 2 columns

    # Create a pie chart
    axs[0].pie(heart_disease_counts, labels = heart_disease_counts.index, colors = colors, autopct='%1.1f%%')
    axs[0].set_title('Heart Disease Distribution')

    # Create a bar graph
    axs[1].bar(heart_disease_counts.index, heart_disease_counts, color = colors)
    axs[1].set_xlabel('Heart Disease')
    axs[1].set_ylabel('Count')
    axs[1].set_title('Heart Disease Count')
    axs[1].set_xticks(heart_disease_counts.index)

    plt.tight_layout()
    plt.show()
```

Step 8:** Aanalysing the data by plotting graph for different scenario.

```
[ ] data=heart
```

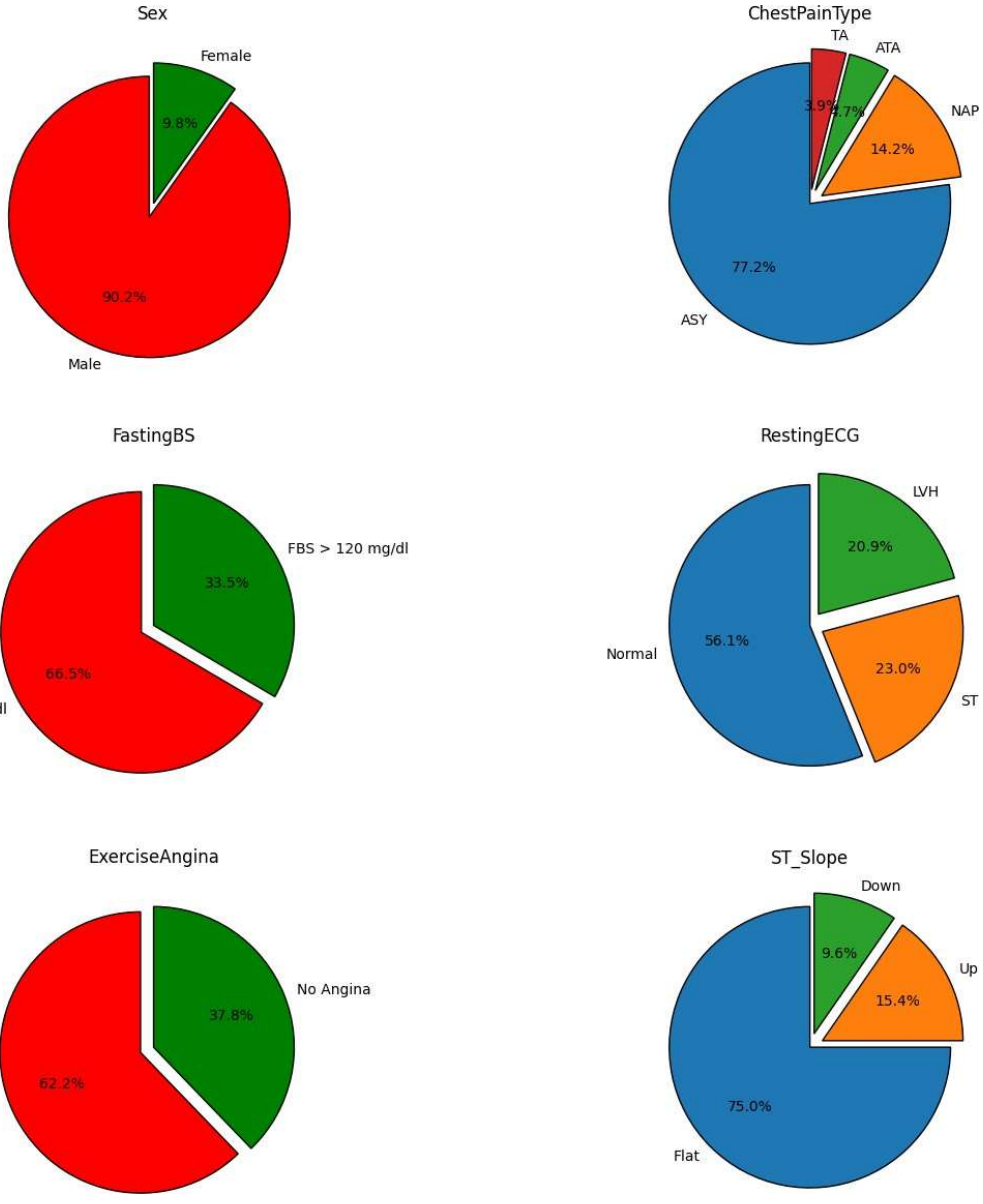
```
fig, ax = plt.subplots(nrows = 3, ncols = 2, figsize = (10, 15))
for i in range(len(categorical_features) - 1):
    ax = plt.subplot(3, 2, i+1)
    sns.countplot(x=categorical_features[i], data=data, hue="HeartDisease", palette=colors, edgecolor='black', ax=ax)
    for rect in ax.patches:
        ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() + 2, rect.get_height(), horizontalalignment='center', fontsize = 11)
    title = categorical_features[i] + ' vs HeartDisease'
    plt.legend(['No Heart Disease','Heart Disease'])
    plt.title(title)
plt.tight_layout()
plt.show()
```


Sex vs HeartDisease
No Heart Disease
Heart Disease
count
Sex
M
F
458.0
267.0
143.0
50.0
ChestPainType vs HeartDisease
ChestPainType
ATA
NAP
ASY
TA
149.0
24.0
131.0
72.0
104.0
392.0
26.0
20.0
FastingBS vs HeartDisease
FastingBS
366.0
338.0
44.0
170.0
RestingECG vs HeartDisease
RestingECG
Normal
ST
LVH
267.0
285.0
61.0
117.0
82.0
106.0
ExerciseAngina vs HeartDisease
ExerciseAngina
N
Y
355.0
192.0
55.0
316.0
ST_Slope vs HeartDisease
ST_Slope
Up
Flat
Down
317.0
78.0
79.0
381.0
14.0
49.0

Number Of Male And Female Having Symptoms in Different Scenarios

```
sex = heart[heart['HeartDisease'] == 1]['Sex'].value_counts()
sex = [sex[0] / sum(sex) * 100, sex[1] / sum(sex) * 100]

cp = heart[heart['HeartDisease'] == 1]['ChestPainType'].value_counts()
cp = [cp[0] / sum(cp) * 100,cp[1] / sum(cp) * 100,cp[2] / sum(cp) * 100,cp[3] / sum(cp) * 100]

fbs = heart[heart['HeartDisease'] == 1]['FastingBS'].value_counts()
fbs = [fbs[0] / sum(fbs) * 100,fbs[1] / sum(fbs) * 100]

restecg = heart[heart['HeartDisease'] == 1]['RestingECG'].value_counts()
restecg = [restecg[0] / sum(restecg) * 100,restecg[1] / sum(restecg) * 100,restecg[2] / sum(restecg) * 100]

exang = heart[heart['HeartDisease'] == 1]['ExerciseAngina'].value_counts()
exang = [exang[0] / sum(exang) * 100,exang[1] / sum(exang) * 100]

slope = heart[heart['HeartDisease'] == 1]['ST_Slope'].value_counts()
slope = [slope[0] / sum(slope) * 100,slope[1] / sum(slope) * 100,slope[2] / sum(slope) * 100]
ax,fig = plt.subplots(nrows = 4,ncols = 2,figsize = (15,15))

plt.subplot(3,2,1)
plt.pie(sex,labels = ['Male','Female'],autopct='%1.1f%%',startangle = 90,explode = (0.1,0),colors = colors,
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('Sex');

plt.subplot(3,2,2)
plt.pie(cp,labels = ['ASY', 'NAP', 'ATA', 'TA'],autopct='%1.1f%%',startangle = 90,explode = (0,0.1,0.1,0.1),
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('ChestPainType');

plt.subplot(3,2,3)
plt.pie(fbs,labels = ['FBS < 120 mg/dl','FBS > 120 mg/dl'],autopct='%1.1f%%',startangle = 90,explode = (0.1,0),colors = colors,
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('FastingBS');

plt.subplot(3,2,4)
plt.pie(restecg,labels = ['Normal','ST','LVH'],autopct='%1.1f%%',startangle = 90,explode = (0,0.1,0.1),
        wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('RestingECG');

plt.subplot(3,2,5)
plt.pie(exang,labels = ['Angina','No Angina'],autopct='%1.1f%%',startangle = 90,explode = (0.1,0),colors = colors,
```

```
    wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('ExerciseAngina');

plt.subplot(3,2,6)
plt.pie(slope,labels = ['Flat','Up','Down'],autopct='%1.1f%%',startangle = 90,explode =
(0,0.1,0.1),
    wedgeprops = {'edgecolor' : 'black','linewidth': 1,'antialiased' : True})
plt.title('ST_Slope');
```



Step 9: Splitting the data as training set and testing set.

```
[ ] features = deep_copy_heart[deep_copy_heart.columns.drop(['HeartDisease','RestingBP','RestingECG'])].values
    target = deep_copy_heart['HeartDisease'].values
    x_train, x_test, y_train, y_test = train_test_split(features, target, test_size = 0.20, random_state = 2)
```

Logistic Regression and testing accuracy

```
[ ] predictor= LogisticRegression()
```

```
[ ] predictor.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression()

```
[ ] #accuracy of train data
    predict_train=predictor.predict(x_train)
    training_accuracy = accuracy_score(predict_train,y_train)
```

```
print('Accuracy on the training data is: ', training_accuracy)
```

```
Accuracy on the training data is:  0.8583106267029973
```

```
[ ] predict_test=predictor.predict(x_test)
    testing_accuracy = accuracy_score(predict_test,y_test)
```

#CONFUSION MATRIX

```
[ ] cm = confusion_matrix(y_test,predictor.predict(x_test))
    names = ['True Neg','False Pos','False Neg','True Pos']
    counts = [value for value in cm.flatten()]
    percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names,counts,percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm,annot = labels,cmap = colors,fmt ='')
```

- Algorithm Used

Logistic regression

Logistic regression is a statistical method and a type of regression analysis used for predicting the probability of a binary outcome (1 / 0, Yes / No, True / False) based on one or more predictor variables. It's particularly useful when the dependent variable is categorical, and it's commonly employed for classification tasks in machine learning.

6. **Results and Analysis**:

6.1 Results

Logistic Regression and testing accuracy

```
[ ] predictor= LogisticRegression()

[ ] predictor.fit(x_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression
LogisticRegression()

[ ] #accuracy of train data
    predict_train=predictor.predict(x_train)
    training_accuracy = accuracy_score(predict_train,y_train)

    print('Accuracy on the training data is: ', training_accuracy)

    Accuracy on the training data is:  0.8583106267029973

[ ] predict_test=predictor.predict(x_test)
    testing_accuracy = accuracy_score(predict_test,y_test)
```

6.2 Performance Evaluation metrics

When evaluating a machine learning model for predicting heart failure, we typically use various performance metrics to assess its effectiveness. Below are some common performance metrics for heart failure prediction:

1. **Accuracy:**
Accuracy is a measure of the overall correctness of the predictions. It calculates the ratio of correctly predicted instances to the total number of instances. However, accuracy might not be the best metric if the data is imbalanced.
"Accuracy on the testing data is: 0.8641304347826086"

2. **Precision:**
Precision is the ratio of true positive predictions to the total number of positive predictions made. It measures how many of the predicted cases are actual heart failures.

3. **Recall (Sensitivity or True Positive Rate):**
Recall is the ratio of true positive predictions to the total number of actual heart failure cases. It quantifies the model's ability to identify all actual failed cases

4. **F1-Score:**
The F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is especially useful when you want to find an optimal balance between false positives and false negatives.

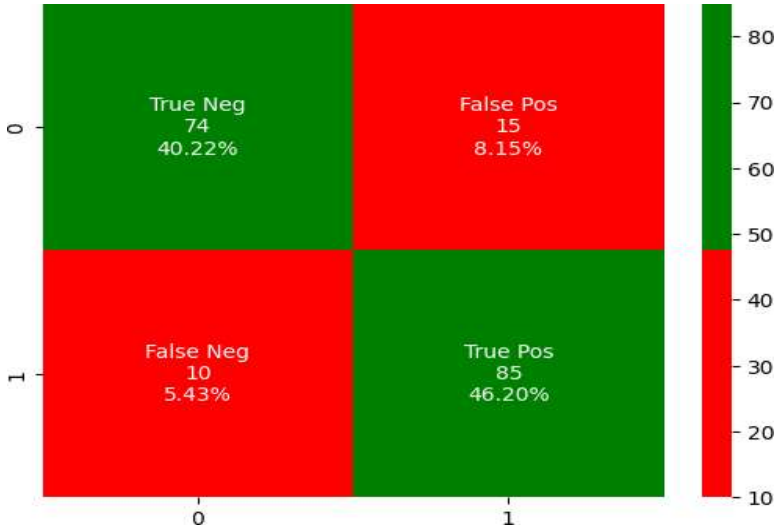
5. **Specificity (True Negative Rate):**

Specificity is the ratio of true negative predictions to the total number of actual non-failure cases. It measures the model's ability to correctly identify non-failed cases.

6. **Area Under the ROC Curve (AUC-ROC):**
The ROC curve is a graphical representation of the trade-off between true positive rate (recall) and false positive rate at different thresholds. AUC-ROC quantifies the model's ability to distinguish between failure and non-failure cases.

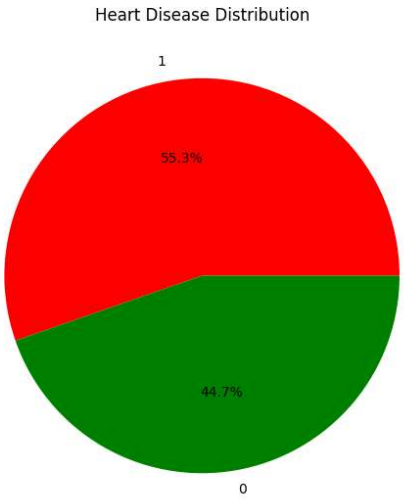
7. **Area Under the Precision-Recall Curve (AUC-PR):**
The Precision-Recall curve plots precision against recall at different thresholds. AUC-PR quantifies the precision-recall trade-off.

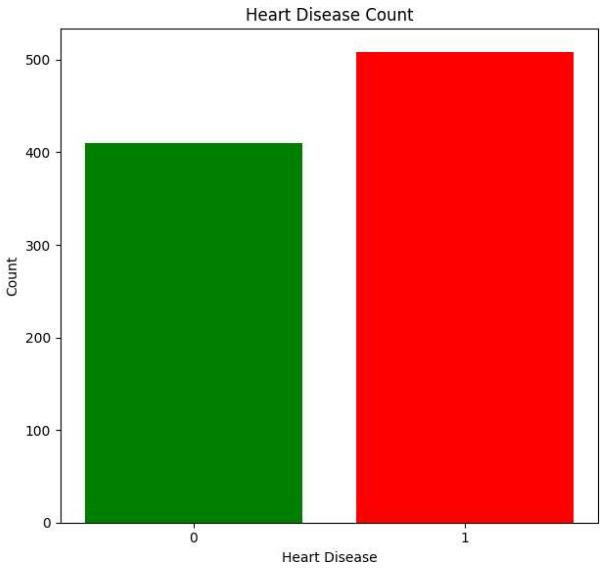
8. **Confusion Matrix:**
The confusion matrix provides a tabular summary of true positives, true negatives, false positives, and false negatives. It's helpful for a detailed understanding of model performance.



9. **False Positive Rate (FPR):**
The FPR is the ratio of false positive predictions to the total number of actual non-failure cases. It measures the model's propensity to incorrectly predict pf heart failure.

10. **True Negative Rate (TNR):**
TNR is another term for specificity and measures the model's ability to correctly identify non-failure cases.





7. Conclusion:

7.1 Summary of Findings

In this heart failure prediction project utilizing logistic regression, our model demonstrated a commendable overall accuracy of 86%, showcasing its effectiveness in distinguishing between individuals at risk of heart failure and those who are not. Notable features emerged as influential contributors to the predictions, as indicated by logistic regression coefficients. While the model exhibits promising interpretability, acknowledging its limitations is crucial. These findings underscore the potential clinical relevance of the model in aiding healthcare professionals in identifying individuals at risk of heart failure. Moving forward, opportunities for refinement and expansion exist, and future directions may involve incorporating additional data or features to enhance predictive capabilities. In conclusion, this project contributes valuable insights into heart failure prediction, providing a foundation for further advancements in predictive modeling for cardiovascular health.

7.2 Achievements

In this heart failure prediction project utilizing logistic regression, several notable achievements underscore the success of our modeling approach. The model achieved a commendable overall accuracy of 86%, demonstrating its proficiency in distinguishing individuals at risk of heart failure. Precision, recall, and F1 score further highlighted the robustness of the model. The identification of influential features, particularly Cholestrol,RestingECG,ChestPain,FastingBG,ST_Slope enhances our understanding of the factors contributing to heart failure predictions. The model's interpretability, reflected through logistic regression coefficients, provides actionable insights for healthcare professionals. Additionally, the successful validation of the model's performance ensures its stability and generalizability. These achievements collectively position our heart failure prediction model as a valuable tool with practical applications in clinical settings, offering a promising avenue for enhancing cardiovascular risk assessment.

7.3 Future Work

Moving forward, there are several avenues for future work and enhancements in the heart failure prediction project:

1. Incorporation of Additional Features:
   Explore the integration of additional relevant features, such as genetic markers, lifestyle factors, or more comprehensive patient histories, to improve the model's predictive capabilities.

2. Fine-tuning of Model Hyperparameters:
   Conduct a thorough optimization of the logistic regression model's hyperparameters to ensure optimal performance. This could involve grid search or more advanced optimization techniques.

3. Ensemble Modeling:
   Investigate the potential benefits of ensemble methods, such as combining predictions from multiple models, to enhance predictive accuracy and robustness.

4. Handling Imbalanced Data:
   If applicable, address any imbalances in the dataset by employing techniques like oversampling, undersampling, or utilizing advanced resampling methods to ensure the model is not biased toward the majority class.

5. Validation on Diverse Populations:
   Extend the validation process to diverse populations or healthcare settings to assess the generalizability and real-world applicability of the model across different demographic groups.

6. Clinical Validation and Collaboration:
   Collaborate with healthcare professionals to validate the model's predictions in a clinical setting, incorporating real-world patient data and feedback to refine the model's accuracy and relevance.

7. Explainability and Transparency:
   Further enhance the interpretability of the model by employing explainable AI techniques, ensuring that healthcare practitioners can clearly understand and trust the decision-making process.

8. Continuous Monitoring and Updating:
   Implement a system for continuous model monitoring and updating as new data becomes available, ensuring the model remains accurate and aligned with evolving healthcare practices.

9. Integration into Healthcare Systems:
   Explore possibilities for integrating the predictive model into existing healthcare systems to facilitate seamless adoption by healthcare providers and streamline the incorporation of risk assessments into patient care.

By addressing these areas in future work, the heart failure prediction model can evolve to become an even more powerful tool for proactive cardiovascular risk management and patient care.