1. INTRODUCTION:

Heart attack is a life-threatening medical condition that occurs when the blood supply to the heart is blocked. It is a leading cause of death globally, accounting for millions of deaths each year. In many cases, heart attacks are preventable if individuals are aware of their risk factors and take steps to reduce them. Predictive modeling can be used to identify individuals at high risk of heart attack, allowing for early intervention and prevention of heart disease.

This project is based on a dataset that contains information on various risk factors associated with heart attack. The dataset includes information on age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar levels, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, the number of major vessels colored by fluoroscopy, and the thallium heart scan results.The goal of this project is to use this dataset to build a predictive model that can accurately predict the likelihood of a heart attack based on these risk factors. By accurately identifying individuals at high risk of heart attack, healthcare professionals can develop targeted prevention and treatment strategies to reduce the risk of heart disease and save lives.

In this report, we will explore the dataset and perform exploratory data analysis to gain insights into the distribution and relationship between different variables. We will then preprocess the data and select the most relevant features for the predictive model. We will train and evaluate different machine learning models using appropriate evaluation metrics and choose the best performing model. Finally, we will use the trained model to make predictions on new data and assess its performance.

This project has the potential to contribute to the development of personalized medicine, allowing for targeted prevention and treatment strategies for individuals at high risk of heart attack. It can also help raise awareness about the importance of monitoring and reducing risk factors associated with heart disease.

1.1 OBJECTIVES:

1. Load the dataset into Python using pandas library.
2. Explore the dataset using descriptive statistics and data visualization techniques to understand the relationship between the independent and dependent variables.
3. Pre-process the data by handling missing values, categorical variables, and feature scaling.
4. Split the data into training and testing sets.
5. Train a logistic regression model on the training set using the scikit-learn library.
6. Evaluate the performance of the model on the testing set using evaluation metrics such as accuracy, precision, recall, and F1-score.
7. Fine-tune the model by hyperparameter tuning and feature selection.
8. Test the final model on new data to assess its performance.

The dependent variable is the output column which indicates the presence or absence of a heart attack (1 for presence and 0 for absence). The independent variables are age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, ST depression induced by exercise relative to rest, slope of the peak exercise ST segment, number of major vessels colored by flourosopy , and thalassemia.

The logistic regression model will predict the probability of a person having a heart attack based on these independent variables. The model can be used by healthcare professionals to identify patients who are at high risk of having a heart attack and provide appropriate interventions to prevent the occurrence of heart attacks.

2.DATA VISUALIZATION:

2.1 Univariate:

Histogram:

Code:

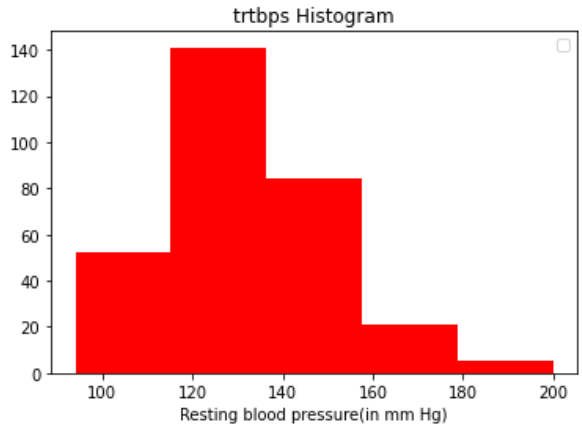


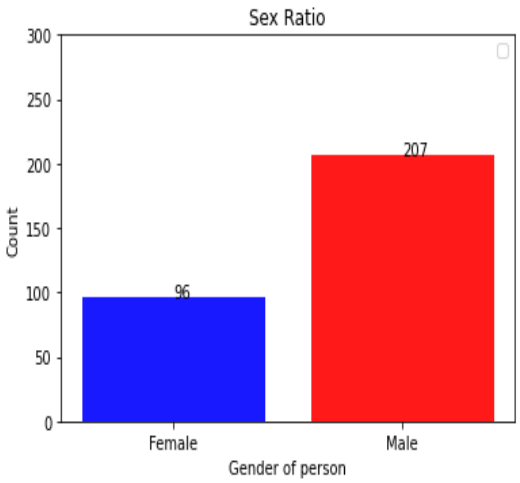
Fig.1 (Histogram for Age) **Fig.2 (Histogram for blood pressure)**

Interpretation:

From the above fig.1 we understand that patients above age 40 have higher chance to get heart attack.

2.2 Bivariate:

Bar chart:



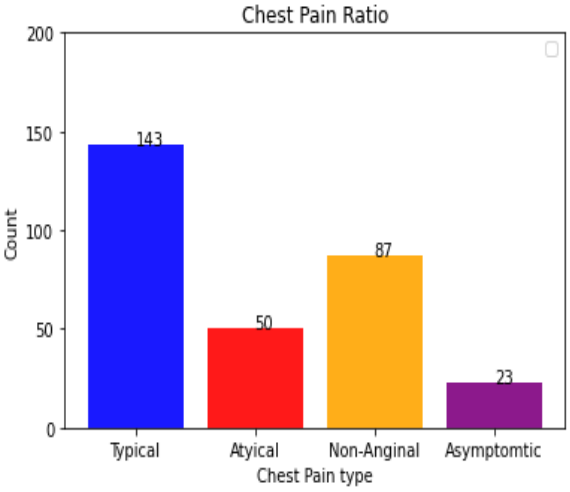


Fig .3 (Bar chart for sex) **Fig.4 (Bar chart for chest pain)**

Interpretation:

From the above Fig.3 we understand that the male patients have higher chance to get heart attack.

2.3 OUTLIER REDUCTION:

Boxplot:

Code:

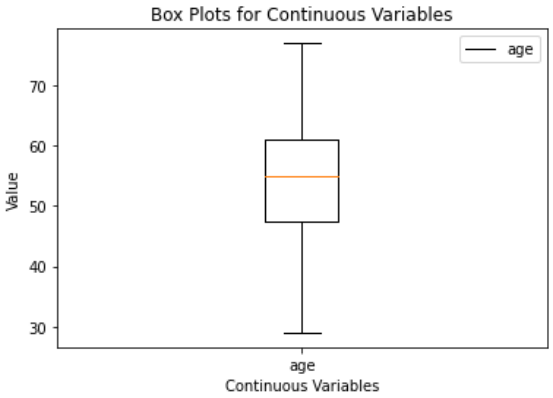


Fig. 5 (Box plot for age)

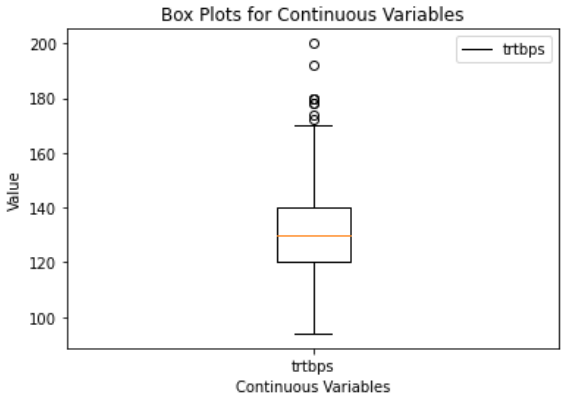


Fig.6(Box plot for chest pain)

Interpretation:

From the above Fig.5 we conclude that there is no ouliers found in the variable age.

3. METHODOLOGY:

3.1 LOGISTIC REGRESSION.

Logistic regression is a statistical method used to model binary outcomes (i.e., outcomes that take on only two values, such as "yes" or "no"). The goal of logistic regression is to find a function that can predict the probability of a particular outcome, given a set of input variables.

The basic logistic regression model is represented by the following equation:

p = 1 / (1 + e^(-z))

where p is the predicted probability of the binary outcome, z is the linear combination of the input variables, and e is the base of the natural logarithm (approximately 2.71828).

The linear combination of the input variables is represented by the following equation:

z = b0 + b1x1 + b2x2 + ... + bk*xk

where b0 is the intercept term and b1, b2, ..., bk are the coefficients associated with the input variables x1, x2, ..., xk.

The coefficients in the logistic regression model are estimated using maximum likelihood estimation. The goal of maximum likelihood estimation is to find the values of the coefficients that maximize the likelihood of the observed data, given the model. Once the coefficients are estimated, the logistic regression model can be used to make predictions by plugging in values for the input variables and using the logistic function to calculate the predicted probability of the binary outcome.

3.2 MODEL DESCRIPTION:

- Importing dataset
- Data pre-processing
- Data visualization
- Split the data into training and testing sets.
- Selection of our model
- Model

3.3 SOFTWARE SPECIFICATION:

Python is a high-level, interpreted programming language that is widely used for various applications such as web development, data analysis, machine learning, scientific computing, and automation. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world due to its ease of use, simplicity, and large community of developers.

Python is known for its readability, which makes it easy to learn for beginners. Its syntax is straightforward, and its use of indentation instead of brackets for code blocks makes it easy to read and understand. Python also has a vast collection of libraries and frameworks that make it suitable for a wide range of applications.

Here are some of the popular uses of Python:

1. Web Development.
2. Data Analysis.
3. Machine Learning .
4. Scientific Computing.
5. Automation.

3.4 RECEIVER OPERATING CHARACTERISTICS (ROC):

In statistics, ROC stands for Receiver Operating Characteristic. It is a graphical representation of the performance of a binary classifier system. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

A binary classifier system is a system that separates objects into two groups based on some criteria, such as disease vs. non-disease, positive vs. negative, or spam vs. not spam. The ROC curve is commonly used in medical diagnosis, biometrics, information retrieval, and many other fields.

The true positive rate (TPR) is the proportion of positive cases that are correctly identified as positive by the classifier. It is also called sensitivity or recall. It is calculated as follows:

TPR = TP / (TP + FN)

where TP is the number of true positives and FN is the number of false negatives. The false positive rate (FPR) is the proportion of negative cases that are incorrectly identified as positive by the classifier. It is calculated as follows:

FPR = FP / (FP + TN)

where FP is the number of false positives and TN is the number of true negatives.

The ROC curve is created by plotting the TPR against the FPR at different threshold settings. The threshold is the minimum value that a classifier needs to assign to a positive case in order to classify it as positive. By varying the threshold, we can trade-off the TPR and FPR.

The area under the ROC curve (AUC) is a measure of the classifier's performance. The AUC ranges from 0 to 1, where 0 represents a classifier that makes random guesses, and 1 represents a perfect classifier. AUC values between 0.5 and 1 indicate that the classifier is performing better than random guessing.

The formula for calculating the AUC is as follows:

AUC = ∫ TPR(FPR) dFPR

where TPR(FPR) is the TPR as a function of the FPR. The integral is taken over the entire range of FPR values

3.5 DECISION TREE :

A decision tree is a popular machine learning algorithm used in statistics for both classification and regression problems. It is a model that maps observations about an item to conclusions about the item's target value.

Here's how it works:

Start with the entire dataset.

Determine which attribute best splits the data into smaller subsets.

Create a decision node based on that attribute and split the data into smaller subsets.

Repeat steps 2 and 3 for each subset until all subsets are pure or the algorithm reaches a predefined stopping criteria.

Here are some key formulas used in decision trees:

Entropy: Entropy is a measure of the impurity of a node in a decision tree. It is calculated using the following formula:

$H(S) = - \Sigma [p(x) * \log2(p(x))]$

where H(S) is the entropy of a node, p(x) is the proportion of observations of class x in the node, and log2 is the logarithm to base 2.

Information Gain: Information gain is the measure of the effectiveness of a splitting attribute in a decision tree. It is calculated using the following formula:

$IG(S, A) = H(S) - \Sigma [(|Sv| / |S|) * H(Sv)]$

where IG(S, A) is the information gain of attribute A on dataset S, Sv is the subset of S for which attribute A has a specific value, and |S| and |Sv| are the number of observations in S and Sv, respectively.

Gini Impurity: Gini impurity is another measure of the impurity of a node in a decision tree. It is calculated using the following formula:

$Gini(S) = 1 - \Sigma [p(x)^2]$

where Gini(S) is the Gini impurity of a node, p(x) is the proportion of observations of class x in the node.

Decision trees are a powerful tool in statistics, as they allow for both prediction and interpretation. However, they can suffer from overfitting if not properly optimized, and they may not perform as well on data that is highly unbalanced or noisy.

3.6 RANDOM FOREST :

Random forest is a popular machine learning algorithm that uses an ensemble of decision trees to make predictions. It is used for both classification and regression problems.

The basic idea of random forest is to build multiple decision trees and combine their predictions to produce a final output. Each decision tree is built using a random subset of the original data and a random subset of the features. The randomness helps to reduce overfitting and improve the generalization performance of the model.

The formula for building a random forest can be broken down into several steps:

1. Randomly sample N data points from the original dataset
2. Randomly select a subset of M features
3. Build a decision tree using the selected data points and features

4. Repeat steps 1-3 B times to create B decision trees
5. To make a prediction for a new data point, pass it through each of the B decision trees and take the average (for regression) or the majority vote (for classification) of the individual tree predictions.

The main hyperparameters of a random forest are the number of trees B and the maximum depth of each tree. Increasing the number of trees generally improves performance, but can also increase the computational cost. The maximum depth controls the complexity of each tree and can help to prevent overfitting.

The algorithm for building a random forest can be summarized as follows:

1. Input: training data X, labels y, number of trees B, maximum depth of each tree D
2. For b = 1 to B:
- Sample N data points from X with replacement
- Randomly select M features from X
- Build a decision tree with maximum depth D using the sampled data and features
3. Output: a random forest consisting of B decision trees

The formula for predicting the output for a new data point x using a random forest can be expressed as

y = 1/B * sum_i=1^B f_i(x)

where y is the predicted output, B is the number of decision trees in the forest, and fi(x) is the prediction of the i-th tree for the input x.

4. DATA ANALYSIS:

4.1 VARIABLE DESCRIPTION:

Age – Age of the patient.

Sex – Sex of the patient.

0 - Female.

1 - Male.

Cp – Chest pain type.

0 - Typical Angina.
1 - Atypical Angina.
2 - Non - Anginal pain.
3 -Asymptomatic.

Trtbps – Resting blood pressure(in mm Hg)

Chol – Cholestrol in mg/dl fetched via BMI sensor.

Fbs - Fasting blood sugar > 120 mg/dl

1 - True

0 - False

restecg - Resting electrocardiographic results

0 - Normal

1 - ST-T wave normality

2 - Left ventricular hypertrophy

thalachh - Maximum heart rate achieved

oldpeak - Previous peak

slp - Slope

caa - Number of major vessels

thall - Thalium Stress Test result ~ (0,3)

exng - Exercise induced angina

1 - Yes

0 - No

output - Target variable

0 – Patients not prone to heart attack

1 – Patients prone to heart attack

4.2 Importing CSV file in python:

```
In [15]: cd C:/Users/umesh d/Desktop/project
C:\Users\umesh d\Desktop\project

In [16]: import pandas as pd

In [17]: import numpy as np

In [18]: import seaborn as sns

In [19]: import matplotlib.pyplot as plt

In [20]: from sklearn.linear_model import LogisticRegression

In [21]: from sklearn.model_selection import train_test_split

In [22]: from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


In [24]: data = pd.read_csv('heart.csv')

In [25]: data
Out[25]:
     age  sex  cp  trtbps  chol  fbs  ...  exng  oldpeak  slp  caa  thall  output
0     63    1   3     145   233    1  ...     0      2.3    0    0      1       1
1     37    1   2     130   250    0  ...     0      3.5    0    0      2       1
2     41    0   1     130   204    0  ...     0      1.4    2    0      2       1
3     56    1   1     120   236    0  ...     0      0.8    2    0      2       1
4     57    0   0     120   354    0  ...     1      0.6    2    0      2       1
..   ...  ...  ..     ...   ...  ...  ...   ...      ...  ...  ...    ...     ...
298   57    0   0     140   241    0  ...     1      0.2    1    0      3       0
299   45    1   3     110   264    0  ...     0      1.2    1    0      3       0
300   68    1   0     144   193    1  ...     0      3.4    1    2      3       0
301   57    1   0     130   131    0  ...     1      1.2    1    1      3       0
302   57    0   1     130   236    0  ...     0      0.0    1    1      2       0

[303 rows x 14 columns]
```

4.3 Summary:

```
Optimization terminated successfully.
         Current function value: 0.407079
         Iterations 7
```

Logit Regression Results

Dep. Variable:	output	No. Observations:	303
Model:	Logit	Df Residuals:	294
Method:	MLE	Df Model:	8
Date:	Thu, 30 Mar 2023	Pseudo R-squ.:	0.4093
Time:	17:02:11	Log-Likelihood:	-123.35
converged:	True	LL-Null:	-208.82
Covariance Type:	nonrobust	LLR p-value:	8.162e-33

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	0.0359	1.712	0.021	0.983	-3.319	3.391
C(sex)[T.1]	-1.7668	0.389	-4.538	0.000	-2.530	-1.004
C(cp)[T.1]	1.4361	0.489	2.940	0.003	0.479	2.394
C(cp)[T.2]	1.9740	0.410	4.817	0.000	1.171	2.777
C(cp)[T.3]	2.2627	0.601	3.767	0.000	1.086	3.440
C(exng)[T.1]	-0.8029	0.373	-2.151	0.031	-1.534	-0.071
trtbps	-0.0209	0.009	-2.202	0.028	-0.040	-0.002
thalachh	0.0275	0.008	3.271	0.001	0.011	0.044
oldpeak	-0.7036	0.175	-4.032	0.000	-1.046	-0.362

Table.1(Summary for logistic regression)

```
0      0.502761
1      0.552308
2      0.924402
3      0.822486
4      0.687600
         ...
298    0.389639
299    0.735222
300    0.037237
301    0.050669
302    0.971910
Length: 303, dtype: float64
```

4.4 Interpretation:

Sex:

```
In [2]: import numpy as np

In [3]: np.exp(1.7668)
Out[3]: 5.8520966573583335
```

The p-value(0.000) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

Cp:

```
In [4]: import numpy as np

In [5]: np.exp(1.4361)
Out[5]: 4.204267160014698

In [6]: np.exp(1.974)
Out[6]: 7.199416636275671

In [7]: np.exp(2.2627)
Out[7]: 9.608998469333212
```

The p-value(0.003) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

Exng:

```
In [8]: import numpy as np

In [9]: np.exp(0.8029)
Out[9]: 2.2320043646377155
```

The p-value(0.031) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

Trtbps:

```
In [10]: import numpy as np

In [11]: np.exp(0.0209)
Out[11]: 1.0211199345383049
```

The p-value(0.028) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

Thalachh:

```
In [12]: import numpy as np

In [13]: np.exp(0.0275)
Out[13]: 1.0278816151072527
```

The p-value(0.001) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

Old peak:

```
In [14]: import numpy as np

In [15]: np.exp(0.7036)
Out[15]: 2.021015282007959
```

The p-value(0.000) of the variable is lesser than 0.05(p<0.05).so the variable is significant to the target variable.

4.5 Logistic regression

```
cd C:/Users/umesh d/Desktop/project
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
ls
data = pd.read_csv('heart.csv')
data
print(data.head())

X_train, X_test, y_train, y_test = train_test_split(data.drop('output', axis=1),
                                                    data['output'], test_size=0.2, random_state=42)

logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)

y_pred = logistic_reg.predict(X_test)

print('Accuracy Score:', accuracy_score(y_test, y_pred))
print('Classification Report:', classification_report(y_test, y_pred))
print('Confusion Matrix:', confusion_matrix(y_test, y_pred))
```

```
Accuracy Score: 0.8852459016393442
```

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

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61

Confusion Matrix: [[25  4]
 [ 3 29]]
```

Table.2(Confusion matrix)

Interpretation:

From the table.2 we found Accuracy, macro average and Weighted average.

4.6 Creating a logistic regression model:

```
# create a logistic regression model
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logistic = LogisticRegression()
logistic.fit(X_train, y_train)
y_pred = logistic.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logistic.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.89
```

4.7 Confusion matrix and classification report:

```
Confusion Matrix:
 [[25  4]
 [ 3 29]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61
```

4.8 ROC Plot:

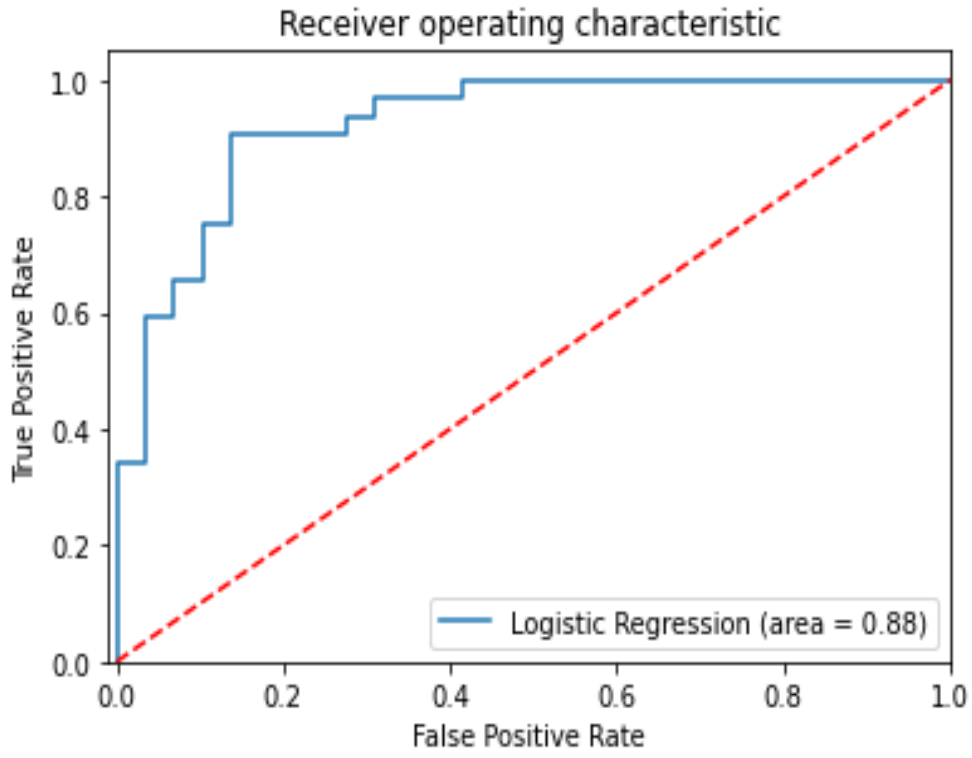


Fig.7(ROC plot)

Interpretation:

From the fig.7 the logistic area = 0.88 in ROC plot.

4.9 Decision Tree:

Plot:

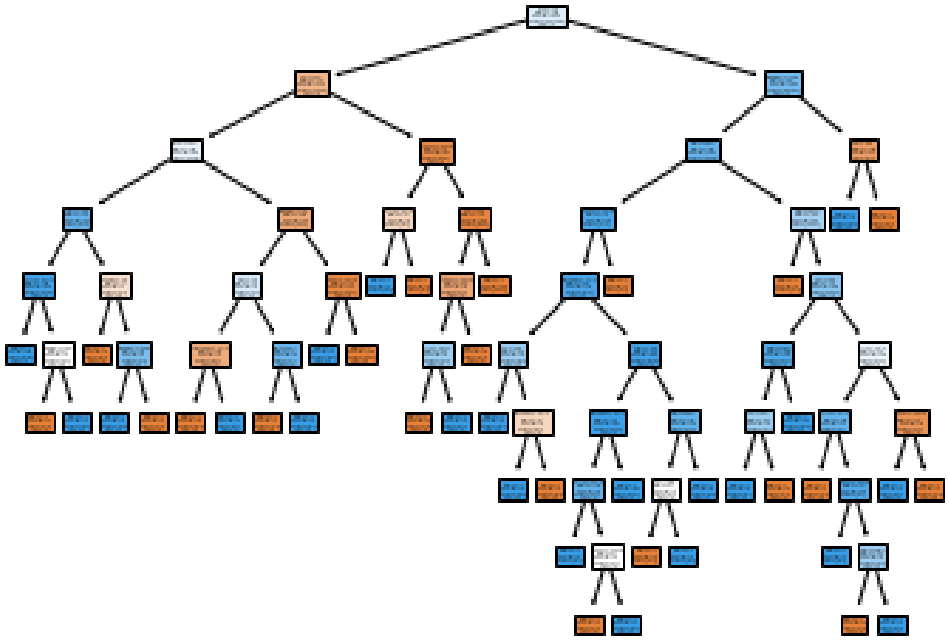


Fig.8(Decision tree)

```
Accuracy: 0.8032786885245902
```

Interpretation:

From fig.8 we found the accuracy of the data using decision tree.

4.10 Random Forest:

```
from sklearn.ensemble import RandomForestClassifier

rt=RandomForestClassifier(n_estimators=100)

rt.fit(X_train,y_train)
RandomForestClassifier()

y_pred=rt.predict(X_test)

from sklearn import metrics

Accuracy: 0.8688524590163934
```

Interpretation:

From the above we concluded that accuracy of the data is 0.8688 using Random forest method.

5. CONCLUSION:

Logistic regression accuracy = 0.89

Receiver operating characteristics accuracy = 0.88

Random forest accuracy = 0.87

Decision tree accuracy = 0.75

Based on the given accuracies, it is difficult to definitively conclude which method is the best as accuracy alone is not always the best measure of performance for classification models. However, we can make some observations based on the provided accuracy values:

1. The logistic regression model has the highest accuracy of 0.89.
2. The receiver operating characteristics (ROC) model has an accuracy of 0.88, which is slightly lower than that of the logistic regression model.
3. The random forest model has an accuracy of 0.87, which is also slightly lower than the logistic regression model.
4. The decision tree model has the lowest accuracy of 0.75.

Based on the above observations, we can tentatively conclude that the logistic regression model is performing the best, followed by the ROC and random forest models. However, it is important to note that accuracy alone may not provide a complete picture of the performance of these models. Other performance measures such as precision, recall, F1-score, and area under the ROC curve (AUC) may provide additional insights into the performance of these models. It is also important to evaluate the models on a test set that was not used during model training to ensure the generalization of the models to new data.

6. REFERANCE:

1.Data source (https://www.kaggle.com/)

2.logistic regression (https://link.springer.com/10.1007%2F978-0-387-30164-8_493)

3.Random forest (https://www.ibm.com/topics/random-forest)

4.ROC (https://www.sciencedirect.com/topics/engineering/random-forest)

5.Decision tree (https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_204)

7.APPENDIX:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf

model = smf.logit("output ~ C(sex) + C(cp) + trtbps + thalachh + C(exng) + oldpeak", data=df1).fit()

print(model.summary())

pred_prob = model.predict(df1)
pred_prob

np.exp(1.8632)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#sex
x=['Female','Male']
y=[96,207]
col=('b','r')
y_pos = np.arange(len(x))
plt.bar(y_pos, y, align='center',color=col,alpha=0.9)
for a,b in zip(y_pos, y):
    plt.text(a, b, str(b))
plt.xticks(y_pos, x)
plt.yticks(np.arange(0,350,50))
plt.ylabel('Count')
plt.xlabel('Gender of person')
plt.title('Sex Ratio')
plt.show()

# Histogram age
import numpy as np
import matplotlib.pyplot as plt

plt.hist(df1.age,bins=5,color='r')
plt.yticks(np.arange(0,140,20))
plt.ylabel('')
plt.xlabel('Age of the person in years')
plt.title('Age Histogram')
plt.show()
```

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

# Load data from CSV file
data = pd.read_csv('heart.csv')

# Get list of continuous variables
continuous_vars = ["age"]

# Create figure and axis objects
fig, ax = plt.subplots()

# Create box plots for each variable
bp = ax.boxplot(data[continuous_vars], labels=continuous_vars)

# Set axis labels and title
ax.set_xlabel('Continuous Variables')
ax.set_ylabel('Value')
ax.set_title('Box Plots for Continuous Variables')

# Add legend
legend_labels = ["age"]

ax.legend(bp['boxes'], legend_labels, loc='upper right')

# Show plot
plt.show()
```

```
# ROC
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logistic.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logistic.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
# decision tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="output", max_depth=3)
dt = dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
# diagram
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
from IPython.display import Image
import pydotplus


dot_data = StringIO()
export_graphviz(dt, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,
                class_names=['0','1'], precision=1)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

```
# packages
pip install pydotplus
pip install six
pip install -U scikit-learn
from six import StringIO

# random forest
import pandas as pd
data = pd.read_csv('heart.csv')
from sklearn.ensemble import RandomForestClassifier
rt=RandomForestClassifier(n_estimators=100)
rt.fit(X_train,y_train)
y_pred=rt.predict(X_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))



# Import necessary libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset into a pandas dataframe
df = pd.read_csv("heart.csv")

# Separate the target variable from the features
X = df.drop('output', axis=1)
y = df['output']

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

# Create a decision tree classifier
dt = DecisionTreeClassifier()

# Fit the classifier to the training data
dt.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = dt.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```