VIRGINIA COMMONWEALTH UNIVERSITY

Statistical Analysis & Modelling

A3 – Logistic Regression & Probit Regression

Using Python

Submitted by

MONIKA SARADHA

#V01068784

Date of Submission: 06/14/2023

Table of Contents

1. Introduction

Heart disease, also known as cardiovascular disease, is the world's leading cause of death and includes a variety of conditions that affect the heart and blood vessels. It includes ailments like arrhythmias, heart failure, and coronary artery disease. Age, gender, family history, high blood pressure, high cholesterol, smoking, inactivity, obesity, and diabetes are all risk factors for heart disease. Improving outcomes depends on early detection, management, and prevention. Techniques for predictive modelling assist in identifying people who are more vulnerable, allowing for targeted interventions and preventative measures. To combat heart disease, public health initiatives emphasize education and the promotion of a heart-healthy lifestyle.

1.1. About the Data

Heart Disease Dataset: The dataset offered includes data on various elements that may play a role in the development of a heart disease event. The columns in the dataset correspond to various characteristics, such as age, sex, blood pressure, cholesterol levels, and more, while each row in the dataset represents a patient. The 'output' target variable shows whether a patient had a heart disease event (1) or not (0). The dataset contains details on 14 different patient-related attributes.

Punjab NSSO Dataset: In Punjab, a sizeable portion of the population consumes animal products like meat, poultry, fish, and seafood, including non-vegetarians. Businesses can learn more about this consumer group's needs by examining their consumption patterns and preferences in the Punjab Food Consumption Dataset. This will help them create targeted marketing strategies, roll out cutting-edge products, and better serve this market. For businesses looking to thrive in Punjab's dynamic food industry and take advantage of market opportunities, understanding non-vegetarian behavior is essential.

1.2. Objective

Two goals serve as the foundation for this analysis. First, based on the heart disease dataset, create a logistic regression model and a decision tree model to forecast the occurrence of heart disease events. The decision tree analysis will offer additional insights and contrast the accuracy of both models in predicting the event, while the logistic regression model will assess assumptions, evaluate performance using a confusion matrix and ROC curve. Secondly, based on the NSSO dataset, to investigate the socioeconomic traits and elements affecting the non-vegetarian consumption patterns. To find significant variables and comprehend their effects on non-vegetarian consumption, this analysis will use descriptive statistics, hypothesis testing, and possibly regression analysis.

1.3. Business Significance

Heart Disease Dataset: For a variety of industries, accurate heart disease event prediction and knowledge of non-vegetarian consumption patterns are important. Healthcare professionals can identify people at higher risk and put timely interventions, individualized treatment plans, and lifestyle changes into place to stop serious cardiovascular incidents by creating trustworthy predictive models for heart disease. This may lower healthcare costs related to the management of heart disease and enhance patient outcomes and resource allocation.

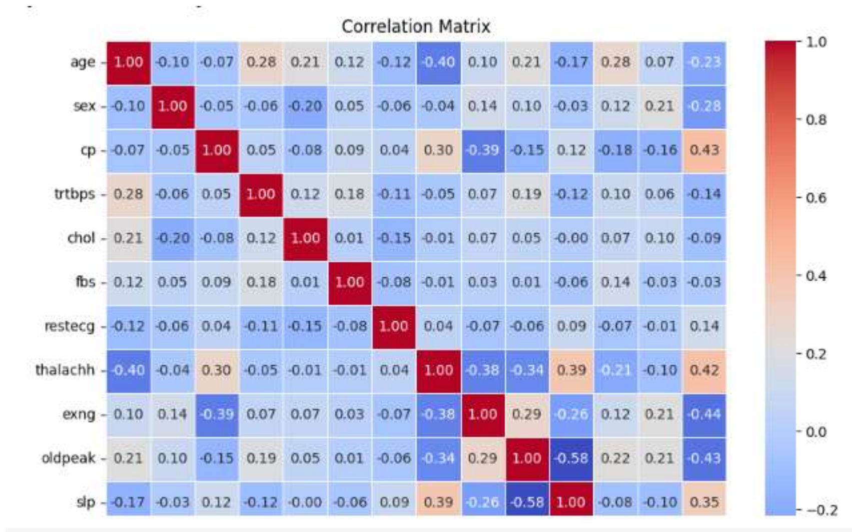
Punjab NSSO Dataset: Contrarily, comprehending the socioeconomic traits and elements affecting non-vegetarian consumption patterns can offer important insights for a number of industries, including the food industry, agriculture, and public health. This analysis can assist retailers and food producers in customizing their product lines to meet the needs of non-vegetarian customers. It can also support the development of targeted interventions and educational campaigns by policymakers and public health organizations to encourage healthier dietary choices and address ethical and environmental issues raised by non-vegetarian consumption.

Businesses and organizations can develop interventions that are tailored to the particular needs and preferences of the target population by utilizing the predictive models and insights obtained from both datasets. As a result, society and the environment may benefit, resource efficiency may increase, and health outcomes may be improved.

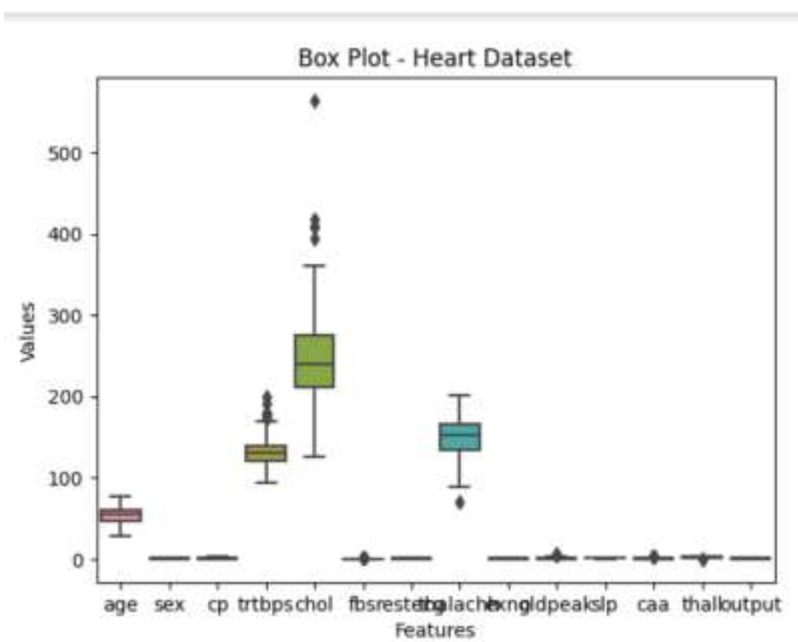
2. Results

2.1. Data Preprocessing

Correlation Matrix:



Boxplot for Outliers:

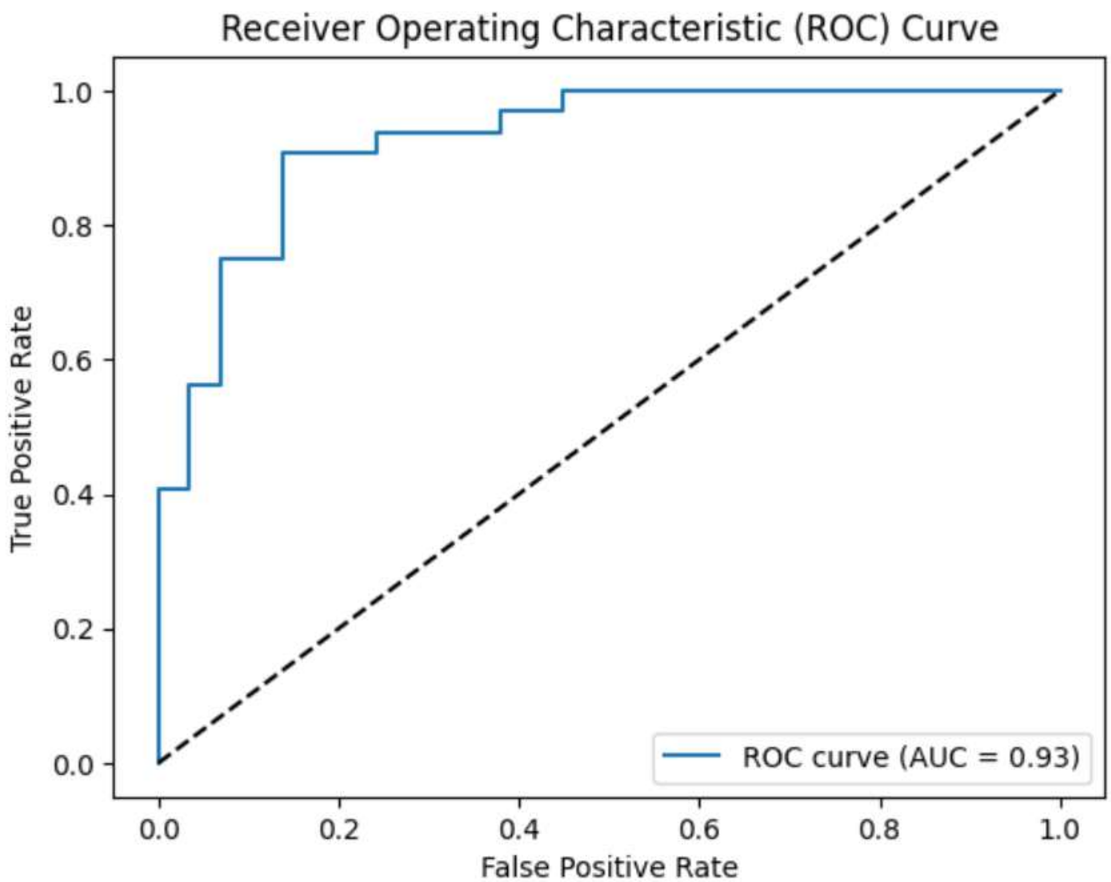


Inference:

The box plot shows the distribution of continuous variables, whereas the correlation matrix details the relationships between the variables in the heart data. Together, they shed light on the relationships and trends in the data. Outliers were identified and accurately removed so as to not affect the modelling of the data.

Missing values were also checked for and since the data was clear of it, no other processing was done.

2.2. Logistic Regression




```
Confusion Matrix:
[[25  4]
 [ 5 27]]
```

Inference:

Confusion Matrix:

The counts of true positive, true negative, false positive, and false negative predictions are displayed in the confusion matrix, which illustrates how well a classification model performs.

- 25 instances were correctly predicted as positive, or true positives (TP).
- 27 instances were correctly identified as true negatives (TNs).
- There have been four instances where a positive outcome has been incorrectly predicted.
- There are 5 instances where the outcome was incorrectly predicted as negative (FN).

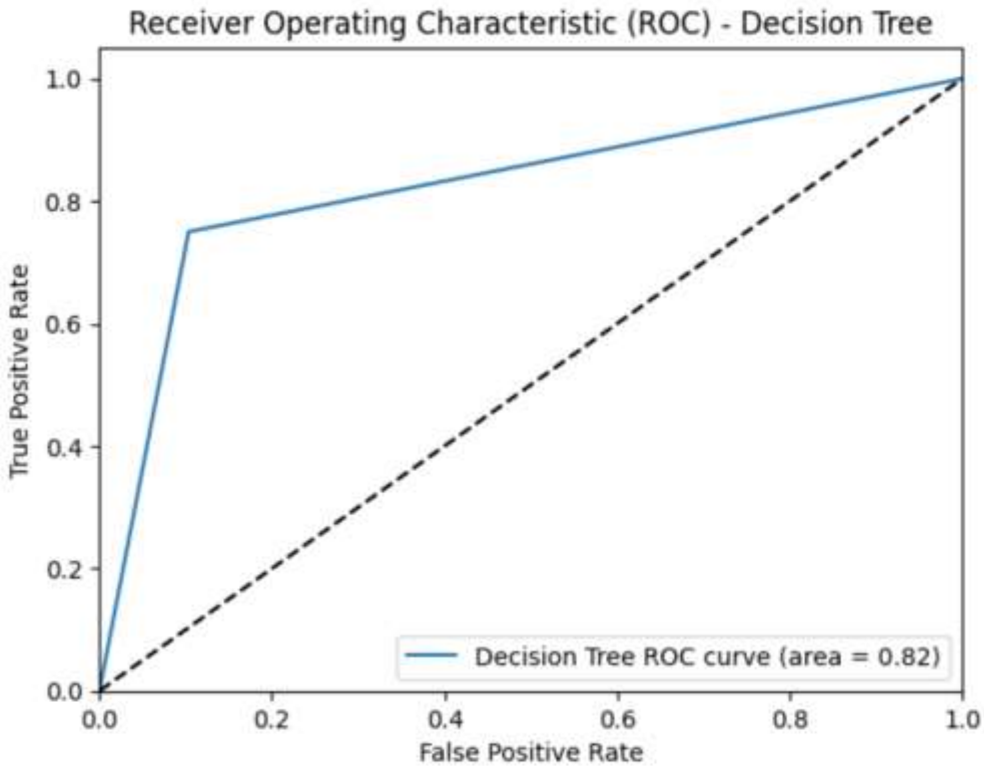
We can use this data to assess the model's effectiveness and determine various metrics, including accuracy, precision, recall, and F1 score. In general, the model appears to have a respectable performance, with more correct predictions than incorrect ones.

ROC Curve:

A classification model's performance at different thresholds is graphically represented by the receiver operating characteristic (ROC) curve. The model's capacity to distinguish between classes is measured by the area under the ROC curve (AUC), with a higher AUC indicating better performance. In this instance, the AUC of 0.93 indicates that the classification model has a strong ability to discriminate between the positive and negative classes.

The model has a strong ability to correctly classify instances across a range of threshold values, according to an AUC of 0.93. This implies that the model is successful in capturing the true positive rate while reducing the false positive rate. Overall, the model performs well in terms of classification accuracy and can be regarded as a reliable predictor for the given problem, according to the high AUC score.

2.3. Decision Tree Analysis




```
Decision Tree Accuracy: 0.819672131147541
Confusion Matrix (Decision Tree):
[[26  3]
 [ 8 24]]
AUC-ROC Score (Decision Tree): 0.8232758620689656
```

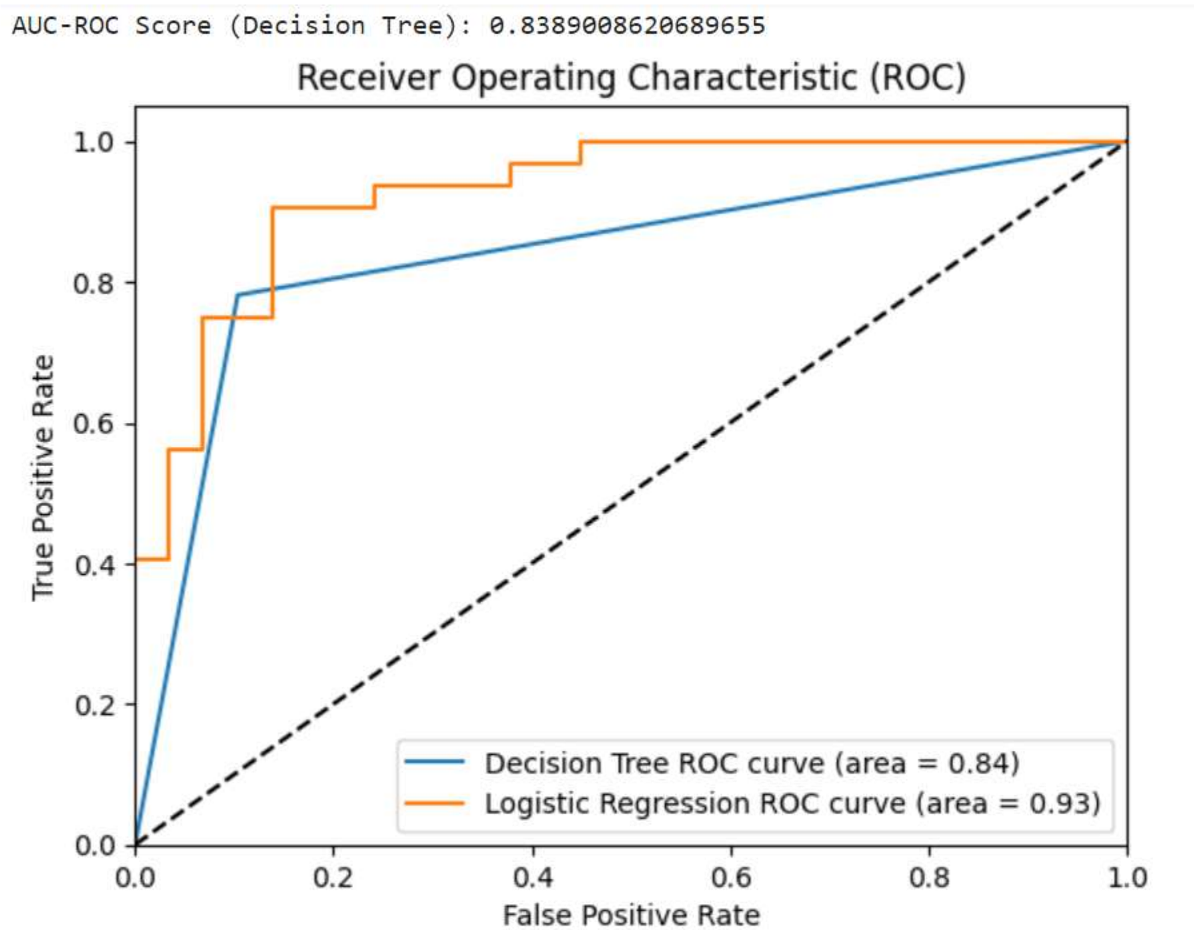
Inference:

The decision tree model had an accuracy of 0.8197, which means that 81.97% of the instances were correctly classified. According to the confusion matrix, of the 61 instances, 26 were correctly identified as belonging to the positive class, 24 as belonging to the negative class, 3 as false positives, and 8 as false negatives.

The decision tree model's AUC-ROC score is 0.8233. The trade-off between the true positive rate and the false positive rate at different threshold values is represented by the ROC curve. An improved ability to distinguish between the positive and negative classes is indicated by a higher AUC score. In this instance, the decision tree model performs reasonably well in terms of classification accuracy and has good discriminatory power, according to the AUC-ROC score of 0.8233.

The decision tree model performs admirably overall, displaying a respectable AUC-ROC score, a relatively high accuracy, and an effective performance in classification tasks.

Result Comparison Logistic Regression and Decision tree:



Inference:

Accuracy: The decision tree model correctly predicts 81.97% of the cases, with an accuracy of 0.8197. However, we lack the logistic regression model's accuracy.

Confusion Matrix: The confusion matrices for the two models are dissimilar. 26 true negatives, 3 false positives, 8 false negatives, and 24 true positives make up the decision tree model. 25 true negatives, 4 false positives, 5 false negatives, and 27 true positives make up the logistic regression model.

AUC-ROC Score: The decision tree model has a good discriminatory power according to its AUC-ROC score of 0.8233. The logistic regression model, on the other hand, has a higher AUC-ROC score of 0.93, indicating better performance in separating the positive and negative classes.

In conclusion, despite having a lower AUC-ROC score and accuracy than the logistic regression model, the decision tree model still offers important insights into the classification performance. The logistic regression model exhibits better discrimination ability and has a higher AUC-ROC score.

2.4. Probit Regression

```
Optimization terminated successfully.
         Current function value: 0.405128
         Iterations 6
                          Probit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                 3118
Model:                         Probit   Df Residuals:                     3109
Method:                           MLE   Df Model:                            8
Date:                Fri, 16 Jun 2023   Pseudo R-squ.:                  0.1313
Time:                        05:30:21   Log-Likelihood:                -1263.2
converged:                       True   LL-Null:                       -1454.2
Covariance Type:            nonrobust   LLR p-value:                 1.358e-77
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
const          -0.9488      0.106     -8.987      0.000      -1.156      -0.742
Age            -0.0008      0.002     -0.376      0.707      -0.005       0.003
Religion       -0.0601      0.019     -3.118      0.002      -0.098      -0.022
fishprawn_q    -0.3920      0.509     -0.771      0.441      -1.389       0.605
goatmeat_q      1.7474      0.368      4.748      0.000       1.026       2.469
beef_q          1.9261      5.231      0.368      0.713      -8.326      12.178
pork_q          0.7647      2.314      0.331      0.741      -3.770       5.299
chicken_q       2.1617      0.135     16.005      0.000       1.897       2.426
othrbirds_q     4.5271      3.117      1.453      0.146      -1.581      10.636
===============================================================================
```

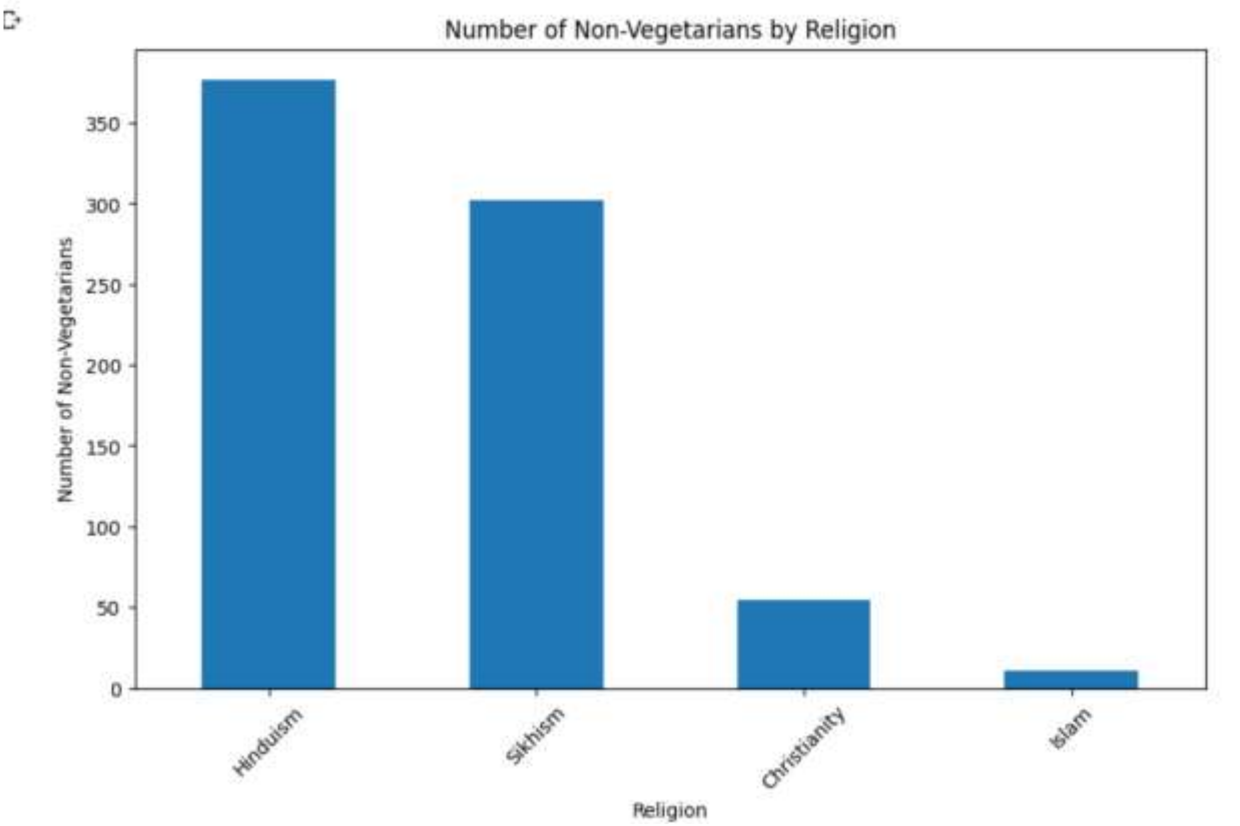
Inference:

Variables: Consuming "goatmeat_q," "chicken_q," and "otherbirds_q" had a favourable effect on the likelihood of not being a vegetarian. But "Religion" had a negative effect, indicating that it was a deterrent.

The "Religion" variable and other variables were statistically significant, which means they significantly affected the likelihood of being a non-vegetarian.

Model Fit: According to the pseudo-R-squared value, the model explained roughly 13.13 percent of the variation in non-vegetarian status.

In conclusion, the probit regression model showed that variables like specific meat consumption and religion affected whether someone was a non-vegetarian.



Inference:

Based on the obtained bar graph it could be visible that Hindus consume the most non-vegetarian food. Although the graph values are in correlation with population of each religious community in Punjab.

3. Recommendation

3.1. Business Implications

- Marketing strategies should be targeted at particular consumer groups based on their non-vegetarian eating habits, consumption patterns, and religious affiliation.
- Product development: Create new products or alter existing ones to satisfy the needs and preferences of non-vegetarian customers.
- Menu planning: Enhance the selection of meat dishes on restaurant and food service menus to appeal to customers who aren't vegetarians.
- Targeted Marketing: Create specialized marketing plans to advertise heart disease prevention services and goods. For targeted marketing campaigns, identify high-risk individuals based on their demographic and lifestyle traits.
- Product Development: Use the heart dataset's insights to develop new, cutting-edge medical equipment, digital health solutions, and dietary supplements that promote heart health.
- Optimize healthcare services by personalizing patient care and putting preventative measures in place based on the main risk factors for heart disease.

3.2. Business Recommendations

- Targeted Advertising: To draw non-vegetarian customers, develop targeted advertising campaigns that emphasize dishes and products that contain meat.
- Product diversification: Increase the variety of meat-based products available to appeal to the preferences of non-vegetarian people
- Customizable menu options: Provide customers with the option to customise their meals by choosing particular meat options and combinations.

- Partnerships with Suppliers: Work with meat suppliers to guarantee a consistent and varied supply of premium meat products that appeal to consumers who aren't vegetarians.
- Customer education: Through educational content, cooking demonstrations, and partnerships with nutritionists or chefs, educate customers about the nutritional value and advantages of meat-based products.
- Market research: To stay current and adjust business strategies appropriately, continuously track consumer trends and preferences regarding non-vegetarian food options.
- Create targeted marketing campaigns to educate high-risk individuals about preventing heart disease and to advertise particular healthcare services and products.
- Research and development: Make an investment in this area to develop practical solutions that address known risk factors and meet the requirements of people who are at risk for heart disease.
- Collaboration with Healthcare Providers: Work together to promote preventive measures and treatment options while integrating developed products and services into the current healthcare system.
- Conduct educational initiatives and awareness campaigns to educate people about the risk factors for heart disease and the value of leading a healthy lifestyle.
- Data analytics and personalization: To efficiently manage and prevent heart disease, use data analytics to identify high-risk individuals and personalize healthcare services.

```
import pandas as pd
import numpy as np
from scipy import stats
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files ASSG1.xlsx

- **ASSG1.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 5293035 bytes, last modified: 6/16/2023 - 100% done

```
Saving ASSG1.xlsx to ASSG1 (1).xlsx
```

```
punjab_ds = pd.read_excel('ASSG1.xlsx')
```

```
subset = punjab_ds[['Age', 'Religion', 'eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']]
```

```
print(subset.describe())
```

```
               Age     Religion     eggsno_q  fishprawn_q   goatmeat_q  \
count  3118.000000  3118.000000  3118.000000  3118.000000  3118.000000
mean     47.778384     2.570879     0.000047     0.003974     0.010788
std      14.126518     1.481989     0.000155     0.061446     0.092871
min       8.000000     1.000000     0.000000     0.000000     0.000000
25%      38.000000     1.000000     0.000000     0.000000     0.000000
50%      46.000000     4.000000     0.000000     0.000000     0.000000
75%      58.000000     4.000000     0.000000     0.000000     0.000000
max      95.000000     9.000000     0.003300     2.500000     2.000000

            beef_q       pork_q    chicken_q  othrbirds_q
count  3118.000000  3118.000000  3118.000000  3118.000000
mean      0.000107     0.000371     0.061360     0.000187
std       0.004719     0.009895     0.212437     0.007461
min       0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     0.000000     0.000000
max       0.250000     0.333333     3.000000     0.333333
```

```
print(subset.shape)
```

```
(3118, 9)
```

```
subset.isnull().sum()
```

```
Age            0
Religion       0
eggsno_q       0
fishprawn_q    0
goatmeat_q     0
beef_q         0
pork_q         0
chicken_q      0
othrbirds_q    0
dtype: int64
```

```
print(subset.columns)
```

```
Index(['Age', 'Religion', 'eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q',
       'pork_q', 'chicken_q', 'othrbirds_q'],
      dtype='object')
```

```
subset.dtypes
```

```
Age              int64
Religion         int64
eggsno_q       float64
fishprawn_q    float64
goatmeat_q     float64
beef_q         float64
pork_q         float64
chicken_q      float64
othrbirds_q    float64
dtype: object
```

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

z_scores = (subset - subset.mean()) /subset.std()
outliers = (z_scores > 3) | (z_scores < -3)

print("Outliers:")
print(outliers)
```

```
Outliers:
        Age  Religion  eggsno_q  fishprawn_q  goatmeat_q  beef_q  pork_q  \
0     False     False     False        False       False   False   False
1     False     False     False        False       False   False   False
2     False     False     False        False       False   False   False
3     False     False     False        False       False   False   False
4     False     False     False        False       False   False   False
...     ...       ...       ...          ...         ...     ...     ...
3113  False     False     False        False       False   False   False
3114  False     False     False        False       False   False   False
3115  False     False     False        False       False   False   False
3116  False     False     False        False       False   False   False
3117  False     False     False        False       False   False   False

      chicken_q  othrbirds_q
0         False        False
1         False        False
2         False        False
3         False        False
4         False        False
...         ...          ...
3113      False        False
3114      False        False
3115      False        False
3116      False        False
3117      False        False

[3118 rows x 9 columns]
```

```
z_scores = (subset - subset.mean()) / subset.std()

outliers = (z_scores > 3) | (z_scores < -3)

subset_no_outliers = subset[~outliers.any(axis=1)]

subset_no_outliers.reset_index(drop=True, inplace=True)

print("punjab without Outliers:")
print(subset_no_outliers)
```

```
punjab without Outliers:
      Age  Religion  eggsno_q  fishprawn_q  goatmeat_q  beef_q  pork_q  \
0      75         4       0.0          0.0         0.0     0.0     0.0
1      60         4       0.0          0.0         0.0     0.0     0.0
2      33         4       0.0          0.0         0.0     0.0     0.0
3      42         4       0.0          0.0         0.0     0.0     0.0
4      50         4       0.0          0.0         0.0     0.0     0.0
...   ...       ...       ...          ...         ...     ...     ...
2943   30         4       0.0          0.0         0.0     0.0     0.0
2944   36         4       0.0          0.0         0.0     0.0     0.0
2945   50         3       0.0          0.0         0.0     0.0     0.0
2946   22         3       0.0          0.0         0.0     0.0     0.0
2947   30         4       0.0          0.0         0.0     0.0     0.0

      chicken_q  othrbirds_q
0      0.000000          0.0
1      0.000000          0.0
2      0.000000          0.0
3      0.000000          0.0
4      0.000000          0.0
...         ...          ...
2943   0.000000          0.0
2944   0.000000          0.0
2945   0.666667          0.0
2946   0.000000          0.0
2947   0.000000          0.0
```

```
[2948 rows x 9 columns]
```

```
import pandas as pd
import numpy as np


Q1 = subset.quantile(0.25)
Q3 = subset.quantile(0.75)
IQR = Q3 - Q1
subset_outliers_removed = subset[~((punjab_ds < (Q1 - 1.5 * IQR)) | (subset > (Q3 + 1.5 * IQR))).any(axis=1)]


subset_no_missing_values = subset_outliers_removed.dropna()


subset_cleaned = subset_no_missing_values.reset_index(drop=True)
```

```
<ipython-input-45-6c0bd5b6bbd6>:8: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will raise \
  subset_outliers_removed = subset[~((punjab_ds < (Q1 - 1.5 * IQR)) | (subset > (Q3 + 1.5 * IQR))).any(axis=1)]
```

b) Fit a probit regression to identify non-vegetarians in your sample. Discuss your results and the characteristics of a probit model

```
religion_mapping = {1: 'Hinduism', 2: 'Christianity', 3: 'Islam', 4: 'Sikhism', 5: 'Jainism', 7: 'Buddhism'}
punjab_ds['Religion'] = punjab_ds['Religion'].replace(religion_mapping)
print(punjab_ds['Religion'].value_counts())
```

```
Sikhism         1574
Hinduism        1421
Christianity      90
Islam             26
Jainism            5
Buddhism           1
9                  1
Name: Religion, dtype: int64
```

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



subset_cleaned.dtypes
```

```
Age              int64
Religion        object
eggsno_q       float64
fishprawn_q    float64
goatmeat_q     float64
beef_q         float64
pork_q         float64
chicken_q      float64
othrbirds_q    float64
dtype: object
```

```
subset_cleaned.tail()
```

	Age	Religion	eggsno_q	fishprawn_q	goatmeat_q	beef_q	pork_q	chicken_q	othrbirds_q
2361	45	Sikhism	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2362	30	Hinduism	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2363	36	Hinduism	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2364	22	Islam	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2365	30	Hinduism	0.0	0.0	0.0	0.0	0.0	0.0	0.0

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

```
columns = ['Age', 'Religion', 'eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']
data = punjab_ds[columns].copy()


data['target'] = np.where(data['eggsno_q'] > 0, 1, 0)

x = data.drop(['eggsno_q', 'target'], axis=1)

x = sm.add_constant(x)

y = data['target']


model = sm.Probit(y, x).fit()

print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.405128
         Iterations 6
                          Probit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                 3118
Model:                         Probit   Df Residuals:                     3109
Method:                           MLE   Df Model:                            8
Date:                Fri, 16 Jun 2023   Pseudo R-squ.:                  0.1313
Time:                        05:30:21   Log-Likelihood:                -1263.2
converged:                       True   LL-Null:                       -1454.2
Covariance Type:            nonrobust   LLR p-value:                 1.358e-77
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
const          -0.9488      0.106     -8.987      0.000      -1.156      -0.742
Age            -0.0008      0.002     -0.376      0.707      -0.005       0.003
Religion       -0.0601      0.019     -3.118      0.002      -0.098      -0.022
fishprawn_q    -0.3920      0.509     -0.771      0.441      -1.389       0.605
goatmeat_q      1.7474      0.368      4.748      0.000       1.026       2.469
beef_q          1.9261      5.231      0.368      0.713      -8.326      12.178
pork_q          0.7647      2.314      0.331      0.741      -3.770       5.299
chicken_q       2.1617      0.135     16.005      0.000       1.897       2.426
othrbirds_q     4.5271      3.117      1.453      0.146      -1.581      10.636
===============================================================================
```

```
import pandas as pd

avg_eggs_by_religion = punjab_ds.groupby('Religion')['eggsno_q'].mean()

max_eggs_religion = avg_eggs_by_religion.idxmax()


print("The religion with the highest average egg consumption is:", max_eggs_religion)
```

```
The religion with the highest average egg consumption is: Christianity
```

```
import pandas as pd


avg_chicken_by_religion = punjab_ds.groupby('Religion')['chicken_q'].mean()

max_chicken_religion = avg_chicken_by_religion.idxmax()

print("The religion with the highest average chicken consumption is:", max_chicken_religion)
```

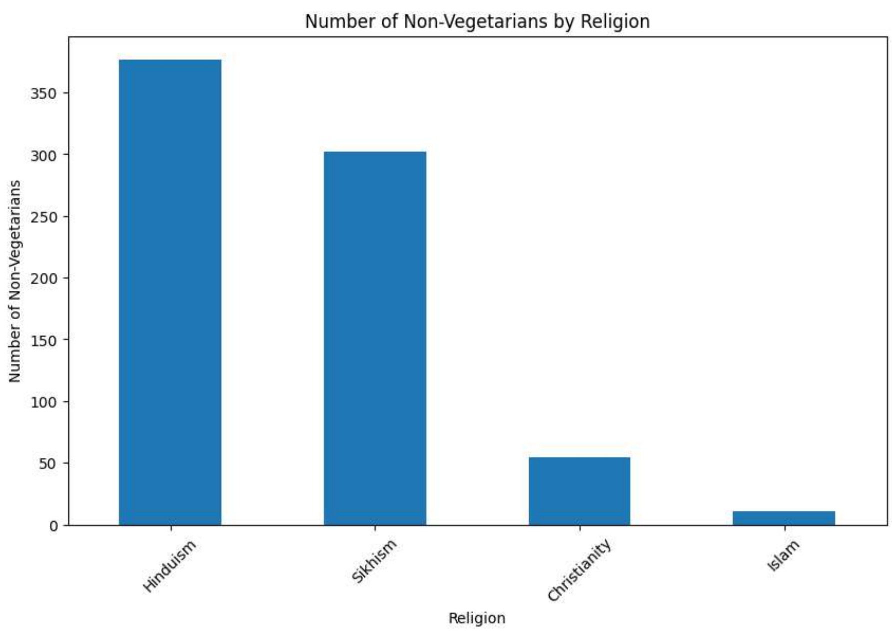
```
The religion with the highest average chicken consumption is: Christianity
```

```
import pandas as pd
```

```
avg_beef_by_religion = punjab_ds.groupby('Religion')['beef_q'].mean()

max_beef_religion = avg_beef_by_religion.idxmax()


print("The religion with the highest average beef consumption is:", max_beef_religion)
```

```
The religion with the highest average beef consumption is: Hinduism
```

```
import matplotlib.pyplot as plt

def is_non_veg(row):
    return any(row > 0)

punjab_ds['non_veg'] = punjab_ds[['eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']].apply(is_non_veg,

non_veg_df = punjab_ds[punjab_ds['non_veg']]

non_veg_counts = non_veg_df['Religion'].value_counts()

plt.figure(figsize=(10, 6))
non_veg_counts.plot(kind='bar')
plt.xlabel('Religion')
plt.ylabel('Number of Non-Vegetarians')
plt.title('Number of Non-Vegetarians by Religion')
plt.xticks(rotation=45)
plt.show()
```



✓ 0s completed at 11:23 AM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files heart (1).csv

- **heart (1).csv**(text/csv) - 11321 bytes, last modified: 6/9/2023 - 100% done

```
Saving heart (1).csv to heart (1).csv
```

```
heart = pd.read_csv('heart (1).csv')
```

```
heart
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

```
print(heart.describe())
```

```
              age         sex          cp      trtbps        chol         fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg    thalachh        exng     oldpeak         slp         caa  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

            thall      output
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

```
heart.isnull().sum()
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

```
heart.shape
```

```
(303, 14)
```

```
heart.columns
```

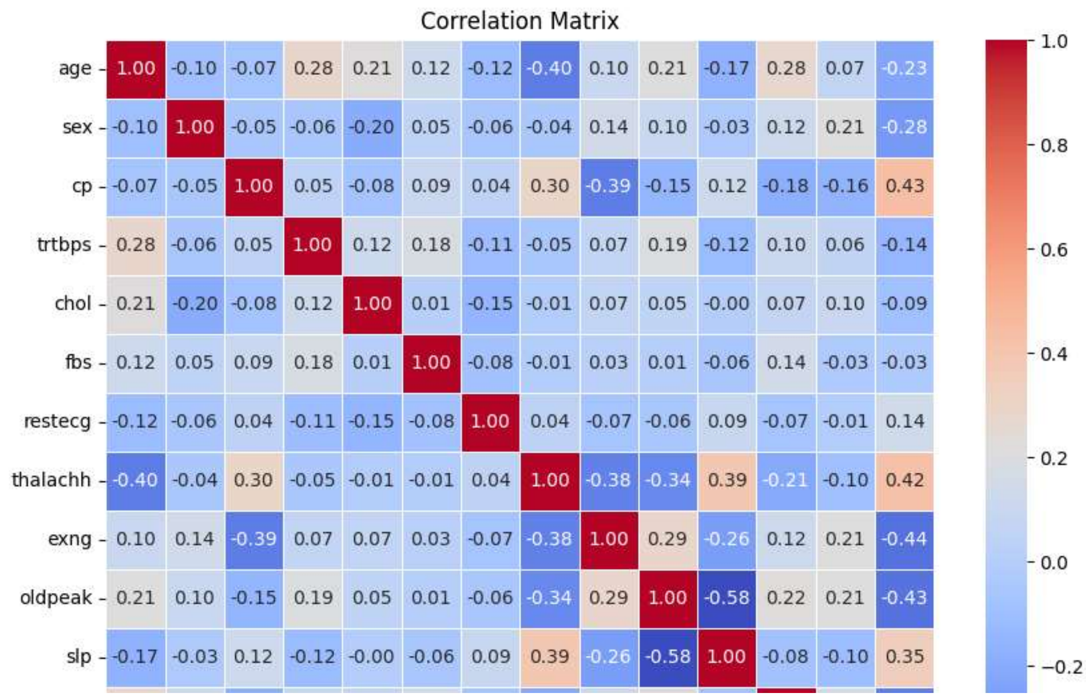
```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')
```

```
heart.dtypes
```

```
age           int64
sex           int64
cp            int64
trtbps        int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exng          int64
oldpeak     float64
slp           int64
caa           int64
thall         int64
output        int64
dtype: object
```

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

z_scores = (heart - heart.mean()) / heart.std()
outliers = (z_scores > 3) | (z_scores < -3)

print("Outliers:")
print(outliers)

correlation_matrix = heart.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

```
Outliers:
       age    sex     cp  trtbps   chol    fbs  restecg  thalachh   exng  \
0    False  False  False   False  False  False    False     False  False
1    False  False  False   False  False  False    False     False  False
2    False  False  False   False  False  False    False     False  False
3    False  False  False   False  False  False    False     False  False
4    False  False  False   False  False  False    False     False  False
..     ...    ...    ...     ...    ...    ...      ...       ...    ...
298  False  False  False   False  False  False    False     False  False
299  False  False  False   False  False  False    False     False  False
300  False  False  False   False  False  False    False     False  False
301  False  False  False   False  False  False    False     False  False
302  False  False  False   False  False  False    False     False  False

     oldpeak    slp    caa  thall  output
0      False  False  False  False   False
1      False  False  False  False   False
2      False  False  False  False   False
3      False  False  False  False   False
4      False  False  False  False   False
..       ...    ...    ...    ...     ...
298    False  False  False  False   False
299    False  False  False  False   False
300    False  False  False  False   False
301    False  False  False  False   False
302    False  False  False  False   False

[303 rows x 14 columns]
```

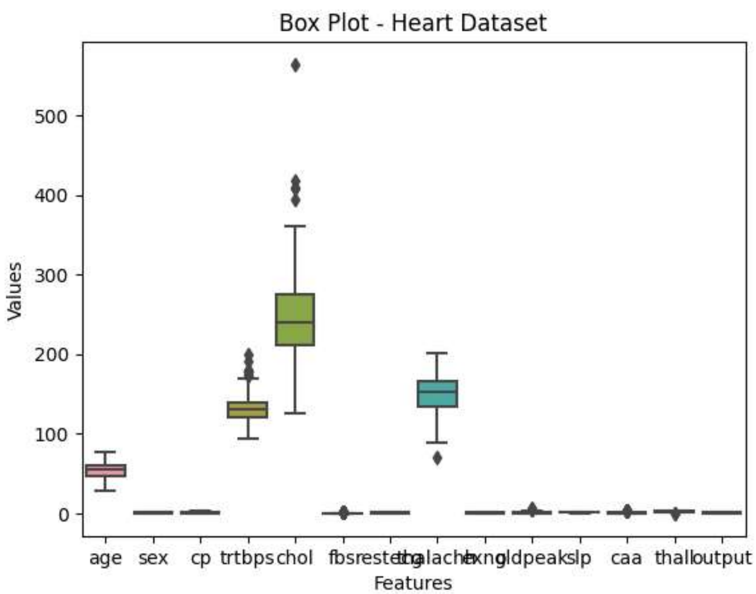



```
z_scores = (heart - heart.mean()) / heart.std()

outliers = (z_scores > 3) | (z_scores < -3)

heart_no_outliers = heart[~outliers.any(axis=1)]

heart_no_outliers.reset_index(drop=True, inplace=True)

print("Heart Dataset without Outliers:")
print(heart_no_outliers)
```

```
Heart Dataset without Outliers:
     age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  \
0     63    1   3     145   233    1        0       150     0      2.3    0
1     37    1   2     130   250    0        1       187     0      3.5    0
2     41    0   1     130   204    0        0       172     0      1.4    2
3     56    1   1     120   236    0        1       178     0      0.8    2
4     57    0   0     120   354    0        1       163     1      0.6    2
..   ...  ...  ..     ...   ...  ...      ...       ...   ...      ...  ...
282   57    0   0     140   241    0        1       123     1      0.2    1
283   45    1   3     110   264    0        1       132     0      1.2    1
284   68    1   0     144   193    1        1       141     0      3.4    1
285   57    1   0     130   131    0        1       115     1      1.2    1
```

```
286   57    0   1     130   236    0        0       174     0      0.0    1

       caa  thall  output
0        0      1       1
1        0      2       1
2        0      2       1
3        0      2       1
4        0      2       1
..     ...    ...     ...
282      0      3       0
283      0      3       0
284      2      3       0
285      1      3       0
286      1      2       0

[287 rows x 14 columns]
```

```
sns.boxplot(data=heart)

plt.title("Box Plot - Heart Dataset")
plt.xlabel("Features")
plt.ylabel("Values")
plt.show()
```



a)Perform logistic regression, check if the assumptions are valid and evaluate the performance of the model using confusion matrix and draw ROC curve.Interpret the results and the efficacy of the model in prediction of the event under study

```
X = heart.drop('output', axis=1)
y = heart ['output']
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

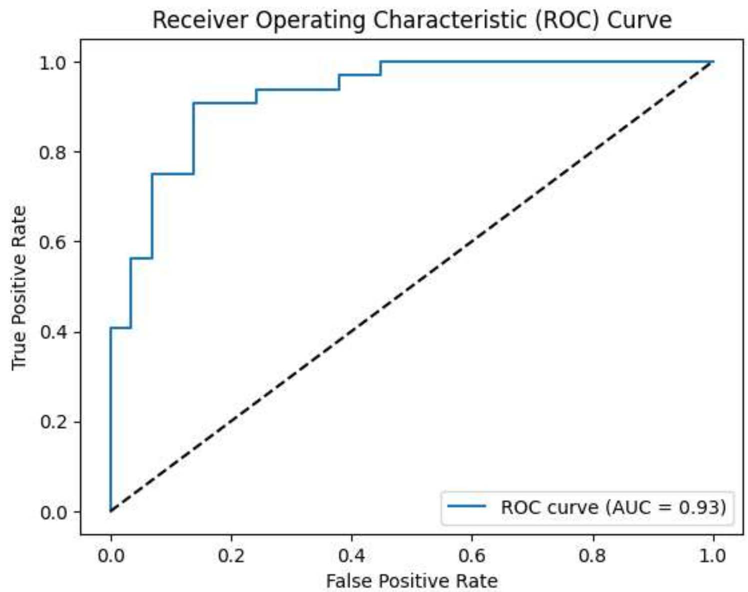
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = LogisticRegression(max_iter=1000, solver='liblinear')
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, solver='liblinear')
```

```
y_pred = model.predict(X_test)
```

```
y_pred = model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

y_pred_prob = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
auc = roc_auc_score(y_test, y_pred_prob)

plt.plot(fpr, tpr, label='ROC curve (AUC = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

```
Confusion Matrix:
[[25  4]
 [ 5 27]]
```

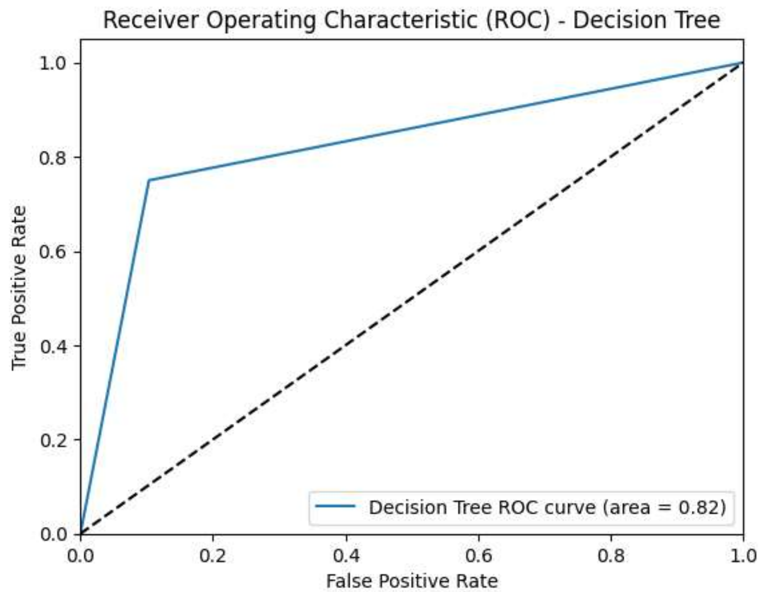


c) Employ decision tree analysis for the data in part a) of this assignment and compare the results of logistic regression and decision tree.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve
import matplotlib.pyplot as plt

dt_model = DecisionTreeClassifier()

dt_model.fit(X_train, y_train)

dt_y_pred_prob = dt_model.predict_proba(X_test)[:, 1]

dt_fpr, dt_tpr, dt_thresholds = roc_curve(y_test, dt_y_pred_prob)

dt_auc_roc = roc_auc_score(y_test, dt_y_pred_prob)

dt_y_pred = dt_model.predict(X_test)

dt_accuracy = accuracy_score(y_test, dt_y_pred)
dt_confusion = confusion_matrix(y_test, dt_y_pred)

print("Decision Tree Accuracy:", dt_accuracy)
print("Confusion Matrix (Decision Tree):")
```

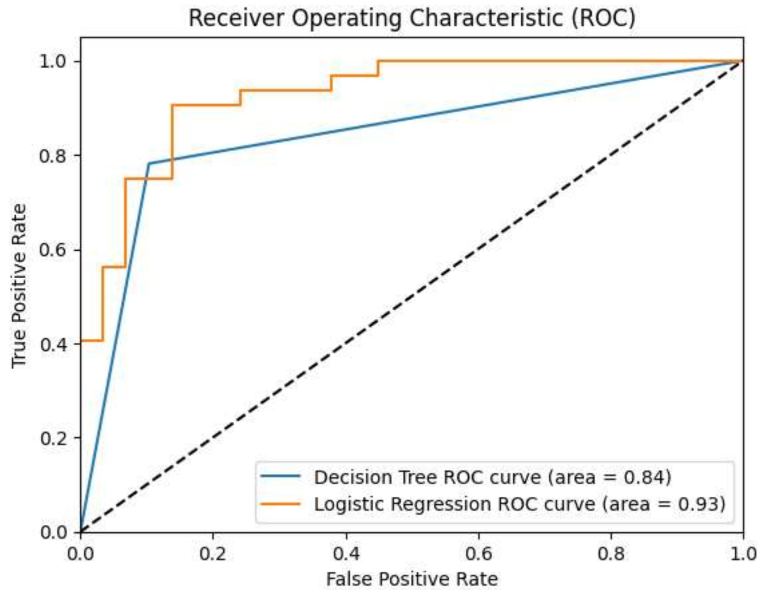
```
print(dt_confusion)
print("AUC-ROC Score (Decision Tree):", dt_auc_roc)

plt.plot(dt_fpr, dt_tpr, label='Decision Tree ROC curve (area = %0.2f)' % dt_auc_roc)
plt.plot([0, 1], [0, 1], 'k--')  # Random classifier line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) - Decision Tree')
plt.legend(loc="lower right")
plt.show()
```

```
Decision Tree Accuracy: 0.819672131147541
Confusion Matrix (Decision Tree):
[[26  3]
 [ 8 24]]
AUC-ROC Score (Decision Tree): 0.8232758620689656
```




```
from sklearn.tree import DecisionTreeClassifier

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

dt_y_pred_prob = dt_model.predict_proba(X_test)[:, 1]

dt_fpr, dt_tpr, dt_thresholds = roc_curve(y_test, dt_y_pred_prob)

dt_auc_roc = roc_auc_score(y_test, dt_y_pred_prob)

print("AUC-ROC Score (Decision Tree):", dt_auc_roc)

plt.plot(dt_fpr, dt_tpr, label='Decision Tree ROC curve (area = %0.2f)' % dt_auc_roc)

plt.plot(fpr, tpr, label='Logistic Regression ROC curve (area = %0.2f)' % auc)

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

```
AUC-ROC Score (Decision Tree): 0.8389008620689655
```



✓ 0s completed at 9:50 AM