ML
Assignment - 2

Ojasva Singh
2020318

Attempting **Sections A and B**

Section A

1.

a. Random forest is a learning method that combines the predictions of multiple decision trees to improve the accuracy and precision of the model. A trade-off exists between the correlation and diversity in random forests.
There is a need for the trees to be correlated to some extent while maintaining diversity.
In terms of **correlation**, a low correlation between the trees is desirable. If the correlation between the decision trees is high, the decision trees will likely make the same errors and make the exact predictions leading to overfitting. That's why low correlation is preferred so that when we average out the decision trees, we obtain a model that is more accurate and robust.
In terms of **diversity**, high diversity between the trees is desirable. If the trees are diverse, then they can compensate for each other's weaknesses, eventually leading to better generalisation against the noise in the data.
The **trade-off** that exists between correlation and diversity is that if we try to increase the diversity by using different randomisation techniques, then we will also increase the correlation between the trees. Similarly, if we try to lower the correlation by using a larger data size, then we will decrease the diversity among the trees by making them similar.

b. The "curse of dimensionality" refers to the challenges that arise due to the dimensions of the dataset that we are working on. It becomes an issue in the context of Naive Bayes when there are a lot of features in comparison to the size of the dataset.

i. In case the dimensions of the dataset are high in comparison to the dataset, then the model will overfit.

ii. The high number of features will make the data sparse. Therefore, the probability will decrease since the main assumption in Naive Bayes is that the features are independent. This will lead to unreliable probability estimates.

Strategies to mitigate this problem:

iii. Feature Selection: Reduce the features that are irrelevant and are not contributing towards predicting the result.
iv. Regularisation: We can use ridge or lasso techniques to penalise the complex models and prevent them from overfitting the dataset.
v. Different models: Other models can be used on the dataset, considering they are less prone to high-dimensional data.
vi. Laplace Smoothing: To cover the zero probability case due to sparse data, smoothing should be done so that we can avoid the situation and arise at a non-zero probability.

c. In case the Naive Bayes classifier faces some features that are new and were not present in the dataset, then some problems will arise and would affect the inference results.

i. Wrong Predictions: On encountering a new attribute, the model wouldn't know what value to assign to it since it was not present in the dataset, leading to wrong predictions or classifications.
ii. Sparsity: High dimensional data can lead to overfitting the model and less robustness to variations in the testing data.
iii. Zero probability: If a new attribute is encountered, a zero probability would be assigned since Naive Bayes assigns probability based on previous encounters. Therefore, the model will rule out this new value as impossible.

Strategies to mitigate these issues:

i. Laplace Smoothing: This technique helps in assigning a non-zero probable value to new features that are encountered, which helps in including these values in the possible combination of features.
ii. Replace with another value: Replacing the value with one of the most common values can also work in some cases.
iii. Probability estimation: We can use probability density estimation techniques for continuous features to determine the probability for newly encountered values.

d. Yes, splitting a decision tree node using information gain might be biased if some attributes have more cardinality than others.

This bias happens because Information Gain reduces the randomness or confusion in the data, and attributes with more unique values (higher cardinality) can make the data less confused by creating more specific divisions. Consequently, these attributes may seem more valuable,

causing the decision tree algorithm to prefer attributes with higher cardinality, which can result in an unbalanced tree.

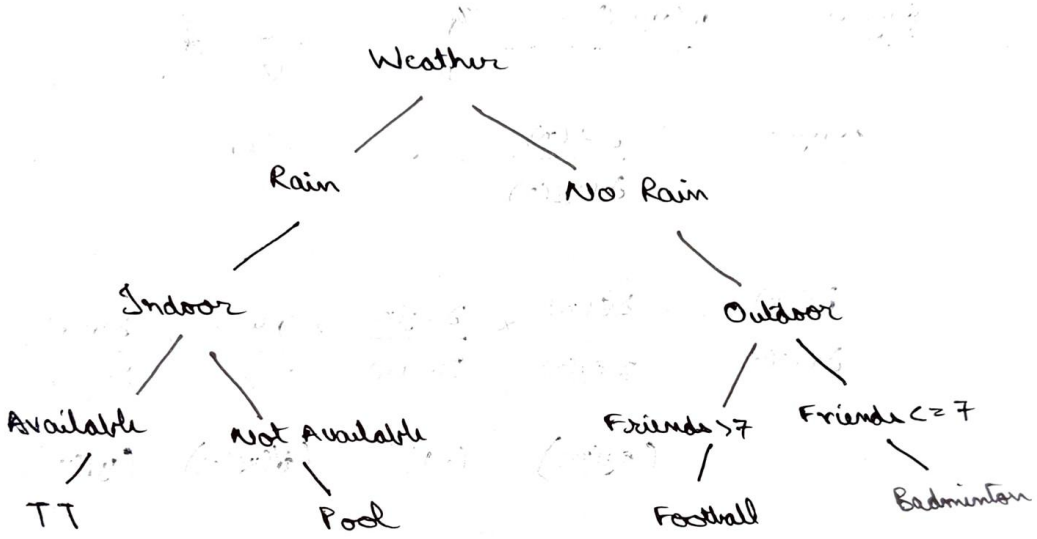
Other criteria for attribute selection:

i. **Gain Ratio**: It is a modification of information gain. It considers the number of branches a split generates and subsequently normalises the information gain. Hence it reduces the bias towards attributes with high cardinatlity.

ii. **Gini Impurity**: It is less sensitive to the cardinality of the attributes as it considers the overall impurity of the dataset and how well an attribute split reduces that impurity.

2.

a. Decision tree for the given scenario:



All possible outcomes and their probabilities:

i. Plays Table Tennis, Indoor, Rain:
P[Rain] * P[Indoor|Rain] * P[Available | Rain and Indoor]

ii. Plays Pool, Indoor, Rain:
P[Rain] * P[Indoor|Rain] * P[Not available | Rain and Indoor]

iii. Plays Football, Outdoor, No Rain:
P[No rain] * P[Outdoor|No rain] * P[Friends>7 | No rain and Outdoor]

iv. Plays Badminton, Outdoor, No Rain:
P[No rain] * P[Outdoor|No rain] * P[Friends<=7 | No rain and Outdoor]

b.

$P(\text{Rainy}) = 0.3$ $\quad P(R'/\text{Rainy}) = 0.8$

$P(\text{Clear}) = 0.7$ $\quad P(C'/\text{Clear}) = 0.9$

$P(R'/\text{Clear}) = 0.1$

$$P(\text{Rainy}/R') = \frac{P(R'/\text{Rainy}) \times P(\text{Rainy})}{P(R')}$$

$$= \frac{P(R'/\text{Rainy}) \times P(\text{Rainy})}{P(R'/\text{Rain}) \times P(\text{Rainy}) + P(R'/\text{Clear}) \times P(\text{Clear})}$$

$$= \frac{0.8 \times 0.3}{0.8 \times 0.3 + 0.1 \times 0.7}$$

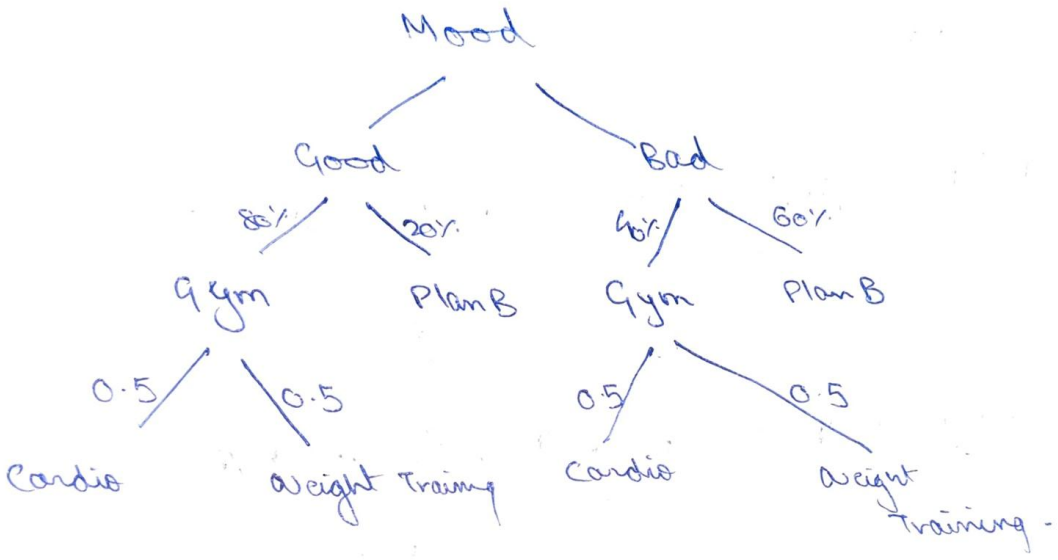
$$= \frac{0.24}{0.24 + 0.07} = \frac{0.24}{0.31}$$

$$= \frac{24}{31}$$

$$= 0.774$$

The probability that it is going to rain on a rainy day(predicted): 0.774

c.



All possible outcomes:

i. Does Cardio, Goes to Gym, Good Mood:
P[Good] * P[Gym|Good] * P[Cardio | Gym and Good]

ii. Does Weight Training, Goes to Gym, Good Mood:
P[Good] * P[Gym|Good] * P[Weight| Gym and Good]

iii. Plan B, Good Mood:
P[Good] * P[PlanB|Good]

iv. Does Cardio, Goes to Gym, Bad Mood:
P[Bad] * P[Gym|Bad] * P[Cardio | Gym and Bad]

v. Does Weight Training, Goes to Gym, Bad Mood:
P[Bad] * P[Gym|Bad] * P[Weight| Gym and Bad]

vi. Plan B, Bad Mood:
P[Bad] * P[PlanB|Bad]

d.

d) $P(RG) = 0.6$

$P(RB) = 0.4$

$P(F=7 / RG) = 0.7$ $\quad P(F=7 / RB) = 0.45$

$$P(\text{Good Mood} / F=7) = \frac{P(F/RG) \times P(RG)}{P(F=7)}$$

$$= \frac{0.6 \times 0.7}{0.6 \times 0.7 + 0.45 \times 0.4}$$

$$= \frac{0.42}{0.42 + 0.18}$$

$$= \frac{0.42}{0.60}$$

$$= 0.7$$

$$P(\text{Bad Mood} / F=7) = \frac{P(F/RB) \times P(RB)}{P(F=7)}$$

$$= \frac{0.45 \times 0.4}{0.60}$$

$$= \frac{0.18}{0.60}$$

$$= 0.3$$

Now we have the probability that Rahul will have a good mood or a bad mood depending on 7 hours of sleep.

i. Does Cardio, Goes to Gym, Good Mood:
P[Good] * P[Gym|Good] * P[Cardio | Gym and Good]
0.7*0.8*0.5 = 0.28

ii. Does Weight Training, Goes to Gym, Good Mood:
P[Good] * P[Gym|Good] * P[Weight| Gym and Good]
0.7*0.8*0.5 = 0.28

iii. Plan B, Good Mood:
P[Good] * P[PlanB|Good]
0.7*0.2 = 0.14

iv. Does Cardio, Goes to Gym, Bad Mood:
P[Bad] * P[Gym|Bad] * P[Cardio | Gym and Bad]
0.3*0.4*0.5 = 0.06

v. Does Weight Training, Goes to Gym, Bad Mood:
P[Bad] * P[Gym|Bad] * P[Weight| Gym and Bad]
0.3*0.4*0.5 = 0.06

vi. Plan B, Bad Mood:
P[Bad] * P[PlanB|Bad]
0.3*0.6 = 0.18

The most likely outcome is that Rahul will go to the gym and will either do cardio or weight training since both of them have the same probability, 0.28.

Section B

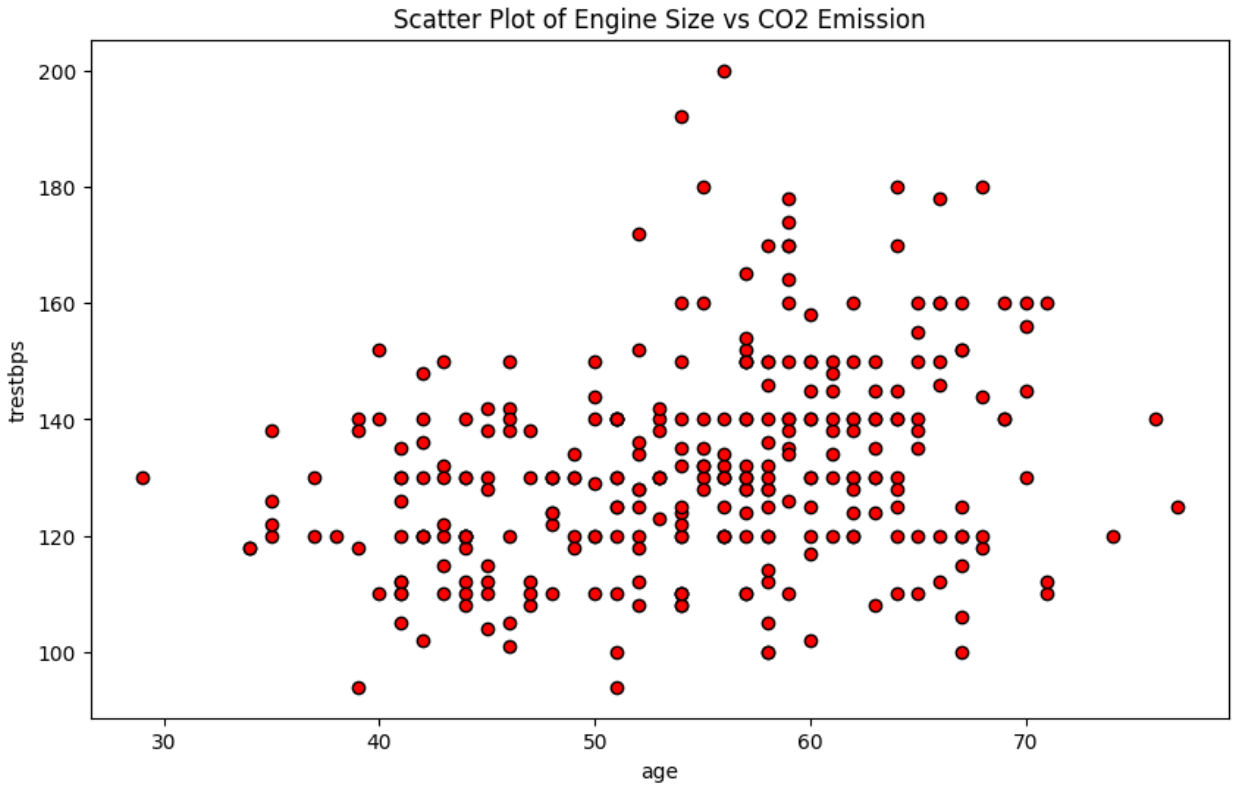
1.

Preprocess the dataset if required and perform Exploratory Data Analysis.

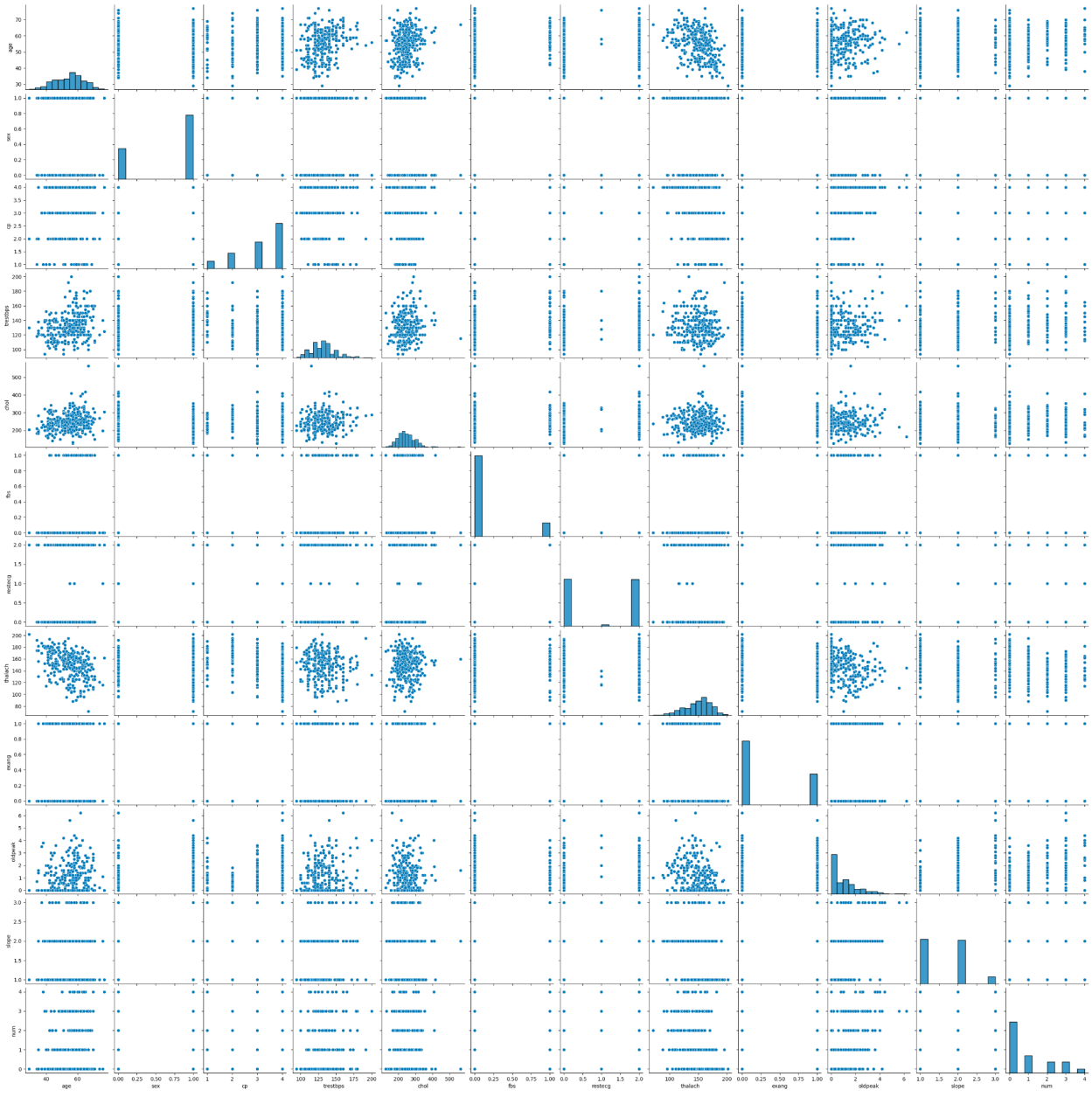
-> For **preprocessing**, replaced the "?' values with NA and then dropped all the rows that contained NA and got the final data frame with a total of 297 rows. Also did multi-mapping of the data and mapped the values greater than 0 to 1.

-> **EDA**

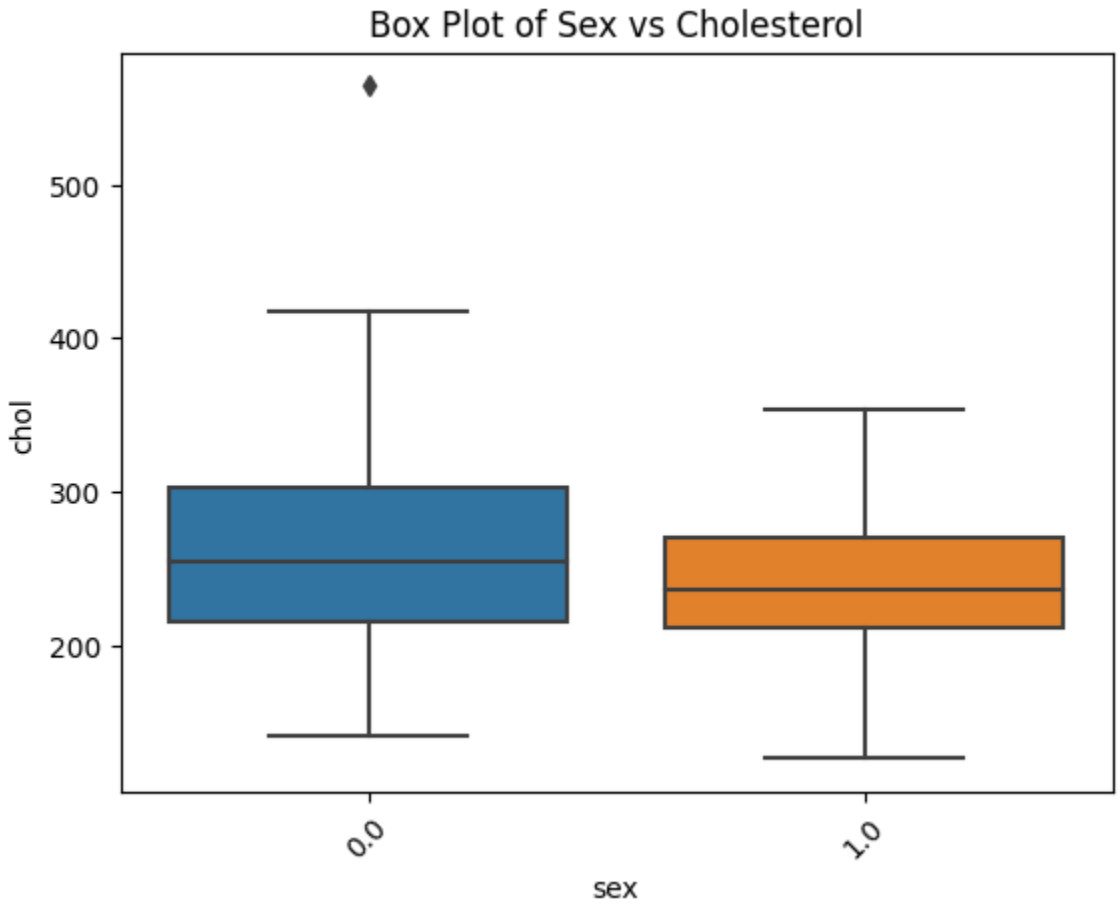
`Scatter plot`



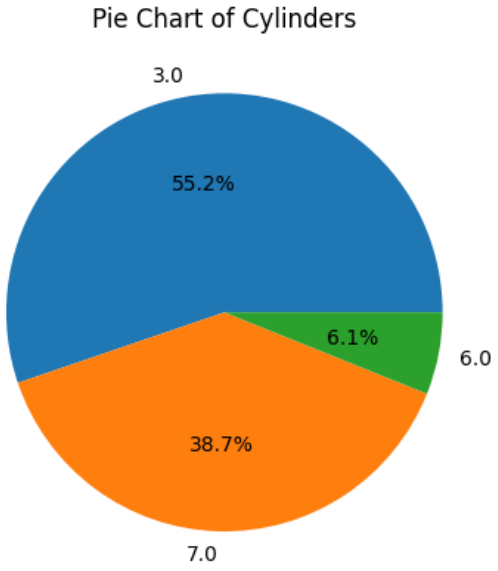
Pair plot


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
num

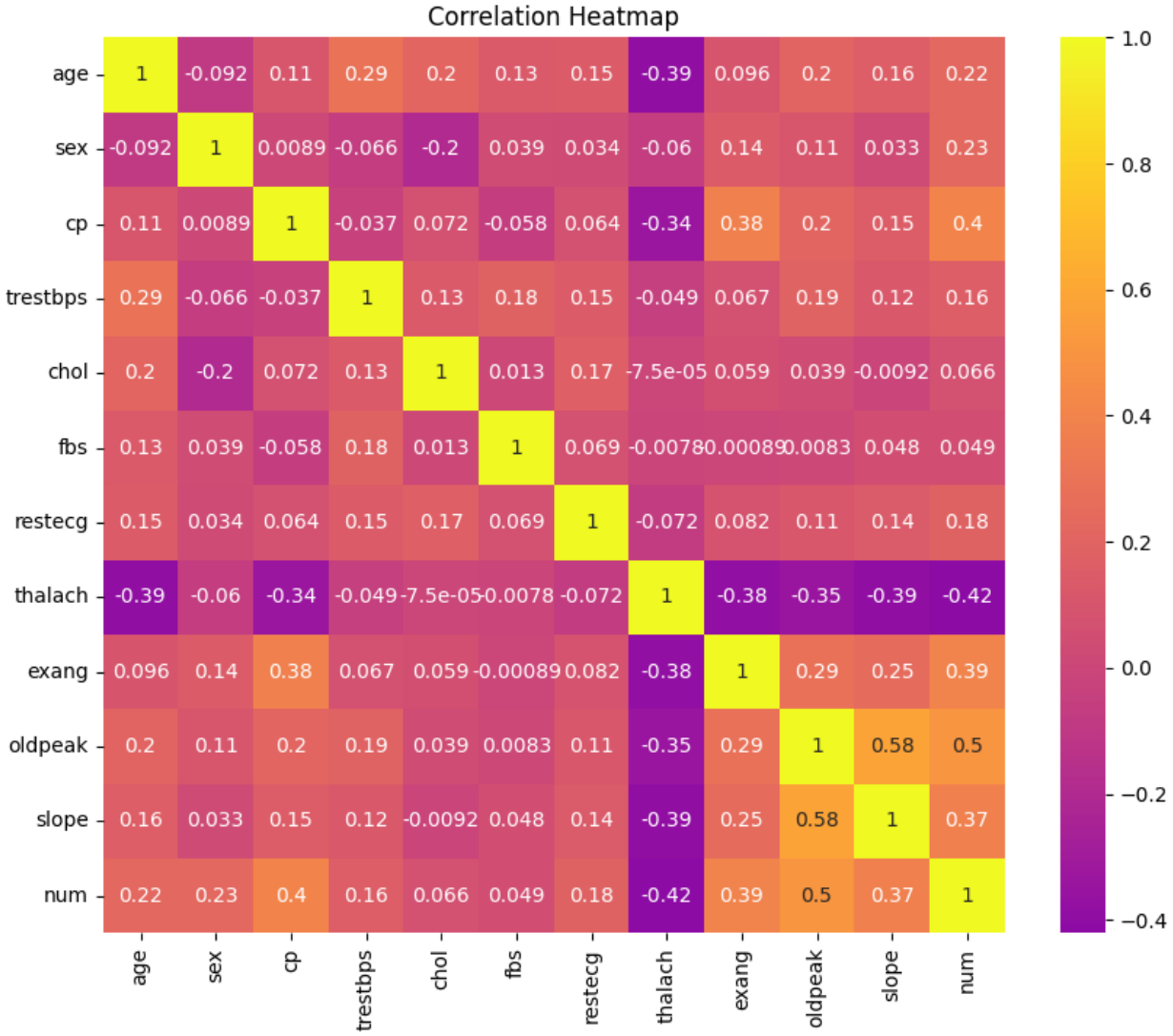
Box plot



Pie Chart



```
Correlation Heatmap
```



2.

Split the data into train: test, 80:20

```
[27]
x = reduced.drop(columns=["num"])
y = reduced["num"]

multi_mapping = {0: 0, 1: 1, 2: 1, 3: 1, 4: 1}
y = y.map(multi_mapping)
#print(y.value_counts())
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2,random_state=42)
✓ 0.0s
```

3.

Entropy and Gini Impurity

c) Entropy and Gini Impurity

```
#Entropy criterion
entropy_tree = DecisionTreeClassifier(criterion='entropy', random_state=42)
entropy_tree.fit(xtrain, ytrain)
ypred_entropy = entropy_tree.predict(xtest)
accuracy_entropy = accuracy_score(ytest, ypred_entropy)

#Gini criterion
gini_tree = DecisionTreeClassifier(criterion='gini', random_state=42)
gini_tree.fit(xtrain, ytrain)
ypred_gini = gini_tree.predict(xtest)
accuracy_gini = accuracy_score(ytest, ypred_gini)
```

[34] ✓ 0.0s

```
print("Accuracy of Decision Tree with Entropy Criterion: ", accuracy_entropy)
print()
print("Accuracy of Decision Tree with Gini Criterion: ", accuracy_gini)
```

[35] ✓ 0.0s

```
Accuracy of Decision Tree with Entropy Criterion:  0.7833333333333333

Accuracy of Decision Tree with Gini Criterion:  0.7833333333333333
```

We can see from the outputs that the accuracy of both entropy and gini criteria are the same.
Since Gini results in simpler and more understandable trees, and it is computationally less intensive, I used Gini criteria and considered it a better choice in this case.

4.

Hyperparameter search using GridSearch

```
    print("Best Criteria: ",best_criterion)
    print("Best Parameters: ",best_params)
    print("Best Score: ",best_score)
    print("Test Accuracy of the best Decision Tree Classifier: ", test_accuracy)
```

```
1]  ✓  0.0s
```

```
Best Criteria:  gini
Best Parameters:  {'max_features': None, 'min_samples_split': 20}
Best Score:  0.7383865248226951
Test Accuracy of the best Decision Tree Classifier:  0.7666666666666667
```

The output tells us that the best score, 0.738.. was produced when the hyperparameters were set to None, 20.
The accuracy of the decision tree when applied to the test data was 0.766....7.

5.

Random Forest Classifier

```
Best Hyperparameters: {'max_depth': None, 'min_samples_split': 10, 'n_estimators': 150}
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.86      0.89        36
           1       0.81      0.88      0.84        24

    accuracy                           0.87        60
   macro avg       0.86      0.87      0.86        60
weighted avg       0.87      0.87      0.87        60
```

The output tells us that the best score was produced when the hyperparameters were set to Max_depth = None, Min_samples_split = 10 and n_estimators = 150. This could've varied depending on the parameters that were set.

The f1 score of the random forest classifier obtained is 0.87.
The report also shows the classification of 0 and 1 classes with precision, recall, f1 score and support.