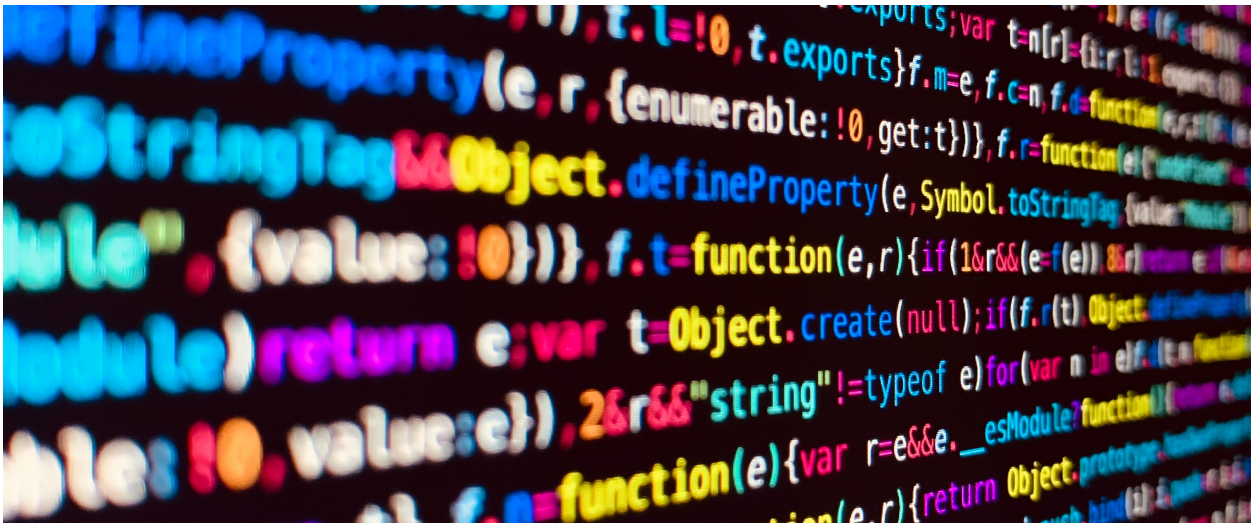


Machine Learning Assignment-2

SECTION A

Q1.

(a) **Correlation** in random forests refers to the extent to which individual trees in the ensemble are similar in their predictions.

When trees in the ensemble are highly correlated to each other then they tend to make similar predictions, errors, and biases, For example, If Tree 1 makes some prediction for some input then Trees 2 and 3 will also tend to make the same prediction on the same input and produce same errors also, so higher correlation doesn't provide strong predictive power and hence high correlation can also lead to overfitting, where the model becomes too specialized to the training data and does not generalize well to new, unseen data.

Diversity in random forests refers to the variation/difference in the individual tree in an ensemble which means each tree will have different predictive power, different bias, and will make different errors, so combining predictions from diverse models can lead to a more robust and accurate overall prediction.

So, we can conclude that as the correlation between individual trees increases diversity among them decreases so it is very obvious that there should be a trade-off between

both terms we neither want complete independence between individual trees nor complete dependence.

Why is it important for the trees to be correlated up to a certain extent while maintaining diversity?

Correlation among the individual trees also affects the bias-variance trade-off, Complete independence among the trees can lead to high variance which causes the model to overfit on training data. (**High diversity →Large variance**)

If the correlation among the trees is very high, it means they tend to make the same errors and have similar biases so the model will be too biased towards the training data only and will not generalize to new, unseen data. (**High Correlation →Large bias**)

Hence the trees need to be correlated to a certain extent while maintaining the diversity to avoid the above described issues.

We use techniques like bootstrapping (sampling with replacement) and random feature selection to encourage diversity among the individual decision trees while maintaining some level of correlation.

(b) The term "**Curse of Dimensionality**" is used to address the issues that arise when the number of features in our dataset is even larger than the number of data points itself, Generally Model's performance increases with the increasing number of features (when the model was underfitting) up to a certain extent, after this stage increasing the number of features leads to overfitting, increased model complexity, and poor model performance.

Curse of dimensionality can become an issue in Naive Bayes in below described conditions:

1. Sparse data: In high dimensional spaces datasets contain fewer data points than the number of features and it becomes unlikely for any data point to have non-zero values for all features which causes the Naive Bayes to make predictions accurately due to lack of sufficient data points.
2. Naive Assumptions: As Naive Bayes assumes that all the features are independent of each other and contribute equally to the prediction, as the number of features increases It will be more unlikely that they are still independent and the distribution

of data will be more spread out and their distribution may not remain uniform, and hence these naive assumptions may not remain true in high dimensional spaces.

To mitigate the curse of dimensionality we can following strategies:

1. Dimensionality Reduction: We can use techniques like PCA (Principal component analysis) and LDA (Linear Discriminant Analysis) to reduce features to avoid above described issues.
2. Regularization: We can use regularization techniques, such as Laplace smoothing (add-one smoothing), to prevent zero probabilities, which can be problematic with sparse data. It helps to smooth the probability estimates.
3. Feature Selection: Selecting a subset of features from all available features that are most relevant solves the problem of having the features with zero or empty values. This reduces dimensionality and can lead to better model performance.

(c) As soon as the Naive Bayes classifier encounters the attributes that were not present in the training dataset it will assign zero probability to them, In a Naive Bayes classifier, when we multiply probabilities together and one of them is zero, the entire posterior probability becomes zero. This can lead to incorrect predictions.

Also, Naive Bayes assumes independence between attributes, which might not hold if unseen attribute values introduce dependencies. This can lead to a loss of important information and potentially result in misclassifications.

Approaches to mitigate the problems:-

1. Laplace smoothing (additive smoothing) is a common technique to address the zero probability issue. Instead of assigning zero probability to unseen values, a small pseudocount is added to the count of each attribute value for each class. This ensures that no probability becomes zero, and the classifier becomes more robust.
2. Use more complex models like decision trees, random forests, or deep learning models when independence assumptions are too restrictive for the data. These models can capture interactions between attributes more effectively.

Example:-

Suppose we have a problem of spam email classification problem task and in the training dataset we observe that the word “discount” occurs multiple times in spam

emails and the word “Lottery” occurs multiple times in non-spam emails.

Suppose the probability of spam is defined as:

P(discount|spam) = (Number of times "discount" appears in spam emails) / (Total number of words in spam emails)

Now suppose we have received a new mail and encountered the new word “Marketing” multiple This word was not present in the training dataset, so the classifier does not have any information about its association with "spam" or "non-spam" emails. so the naive Bayes classifier will assign this new feature a zero probability.

With a probability of zero, the Naive Bayes classifier would be strongly inclined to classify this email as non-spam, which might not be accurate.

To address this issue, we can apply Laplace (Add-One) Smoothing. With Laplace smoothing, we add a small constant (usually 1) to the count of each word in both spam and non-spam classes. Now, for the new email with "supermarket," the probability becomes:

P(supermarket|spam) = (0 + 1) / (12 + Vocabulary Size)

Here, the count of "supermarket" in spam emails is adjusted by adding 1, and the denominator is adjusted by adding the vocabulary size. This avoids the zero probability issue and allows the classifier to make a more informed decision. Depending on the vocabulary size and the degree of smoothing, P(supermarket|spam) may be a small positive value.

By using Laplace smoothing, the Naive Bayes classifier becomes more robust to unseen words and is less likely to make overly confident predictions based on the absence of training data for specific attribute values like "supermarket."

(d) Yes, splitting a decision tree node using Information gain can be biased when attributes have different cardinalities. This bias occurs because Information Gain is sensitive to the number of categories or values an attribute can take, favoring attributes with a higher cardinality. Attributes with more categories are more likely to result in a higher Information Gain.

On the other hand, The gain Ratio takes into account the number of categories an attribute has and normalizes the Information gained by the intrinsic information of the attribute as described in the below formulas.

The formula of Intrinsic Information is:

$$II = -(\sum \frac{|D_j|}{|D|} * \log_2 \frac{|D_j|}{|D|})$$

|Dj|/|D| = number of samples in j-th subset / number of total samples available

The Gain Ratio is:

$$GainRatio = \frac{\text{Information Gain}}{\text{Intrinsic Information}}$$

One more important criterion for attribute selection with proper is the Gini Impurity.

Gini Impurity (GI) measures the probability of misclassifying a randomly chosen element if it is randomly classified according to the class distribution in a dataset.

The formula to calculate Gini Impurity (GI) for a dataset S is:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

where pi is the proportion of examples in class ‘i’ within the dataset.

To select an attribute, the algorithm calculates the weighted average of Gini Impurity for each subset created by splitting on that attribute. The attribute with the lowest Gini Impurity after the split is chosen.

Example

Suppose we have a dataset of students and their grades, and we want to build a decision tree to predict whether a student will pass or fail a class. We have two attributes: "GPA" with high cardinality (e.g., 100 different GPAs) and "Attendance" with low cardinality (e.g., 3 categories: high, medium, low).

If we use Information gain alone to predict whether a student would pass or fail, the system would be biased towards attribute GPA. This is because there are many different possible GPAs, so the system could create more rules for predicting whether a student would pass or fail based on their GPA.

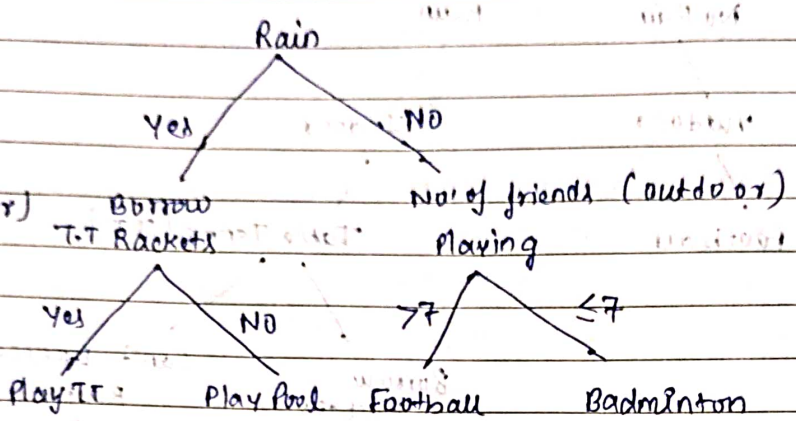
However, "Attendance" may still be an essential attribute in predicting whether a student will pass or fail.

The gain Ratio solves this problem by penalizing features with a high cardinality (GPA). It does this by dividing the Information Gain of a feature by the entropy of the feature itself. This penalizes features with a high cardinality, as they will have a higher entropy.

As a result, the Gain Ratio is a more robust measure of how informative a feature is, regardless of its cardinality. This makes it a good choice for selecting attributes in decision trees when there is a significant difference in the cardinality of different attributes.

Q2.

Q2 (a) Decision Tree



Conditional probability Expression for all possible outcomes:-

1. If there is rain and he can borrow T-T rackets then he will play TT

   P(play TT | Rain ∧ Borrow TT rackets)

2. If there is ~~no~~ rain and he can't borrow T-T rackets then he will play Pool

   P(play Pool | Rain ∧ can't borrow TT Rackets)

3. If there is no rain and more than 7 of his friends are playing then he will play football.

   P(playfootball | ¬Rain ∧ No. of friends playing >7)

4. If there is no rain and no more than 7 of his friends are playing then he will play badminton

   P(play badminton | ¬Rain ∧ No. of friends playing ≤7)

Q2 (b)

PR → App predicts rainy
PC → App predicts clear
R → Actually, rainy
C → Actually clear.

$P(PR) = 0.3$
$P(PR|R) = 0.8$
$P(PC|R) = 0.2$
$P(PC) = 0.7$
$P(PC|\neg C) = 0.9$
$P(PR|C) = 0.1$

We need to calculate $P(R|PR)$

So, According to Bayes' theorem

$$P(R|PR) = \frac{P(PR|R) \times P(R)}{P(PR)}$$

Now event PR can be written as,

$PR = (PR \wedge C) + (PR \wedge R)$ —(I)

↳ predicts Rain and Actually Rain

↳ predicts Rain and Actually clear

We have $P(PR|R) = 0.8 = \frac{P(PR \wedge R)}{P(R)}$

$\Rightarrow P(PR \wedge R) = 0.8 \times P(R)$ —(II)

also $P(PR|C) = 0.1$

$\Rightarrow \frac{P(PR \wedge C)}{P(C)} = 0.1$

$\Rightarrow P(PR \wedge C) = 0.1\, P(C) = 0.1(1 - P(R))$ —(III)

putting (II) and (III) in (I) we get

$P(PR) = 0.8 \times P(R) + 0.1(1 - P(R))$

we have $P(PR) = 0.3$

so we get

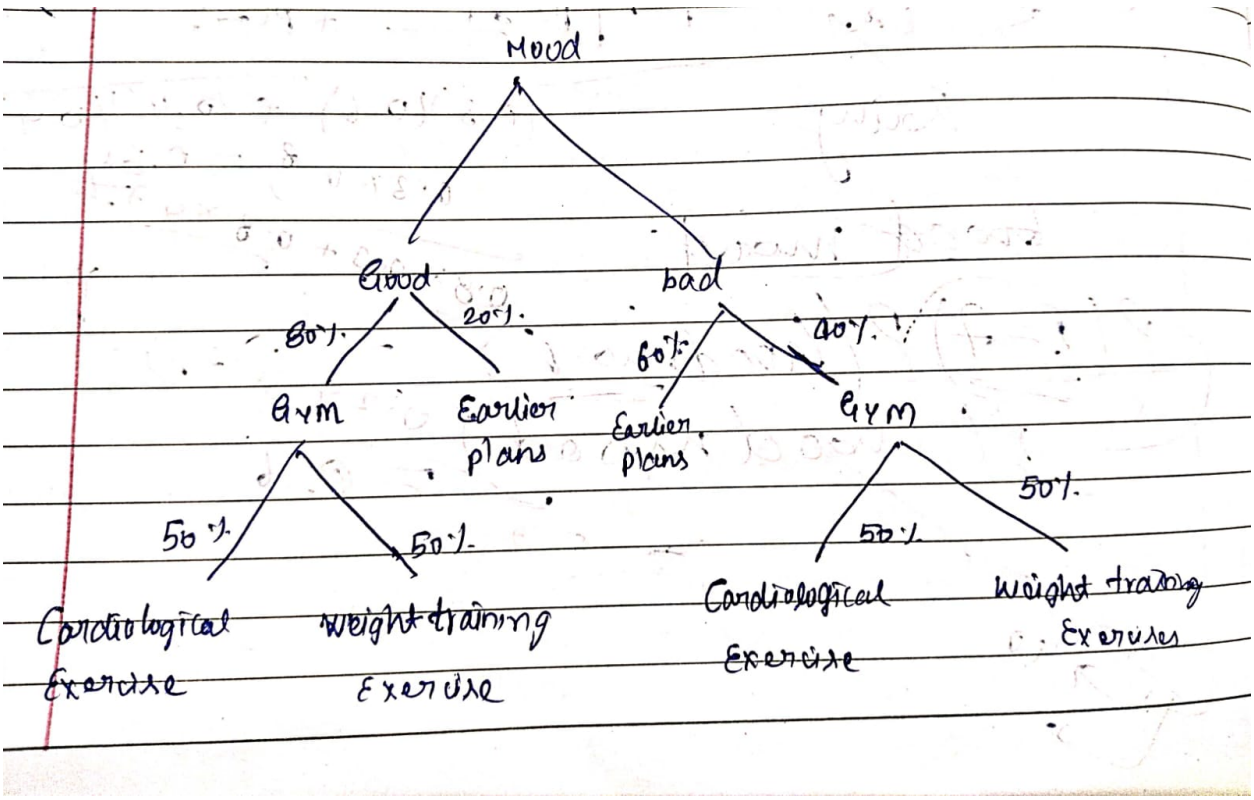
$0.3 = 0.7\,P(R) + 0.1$

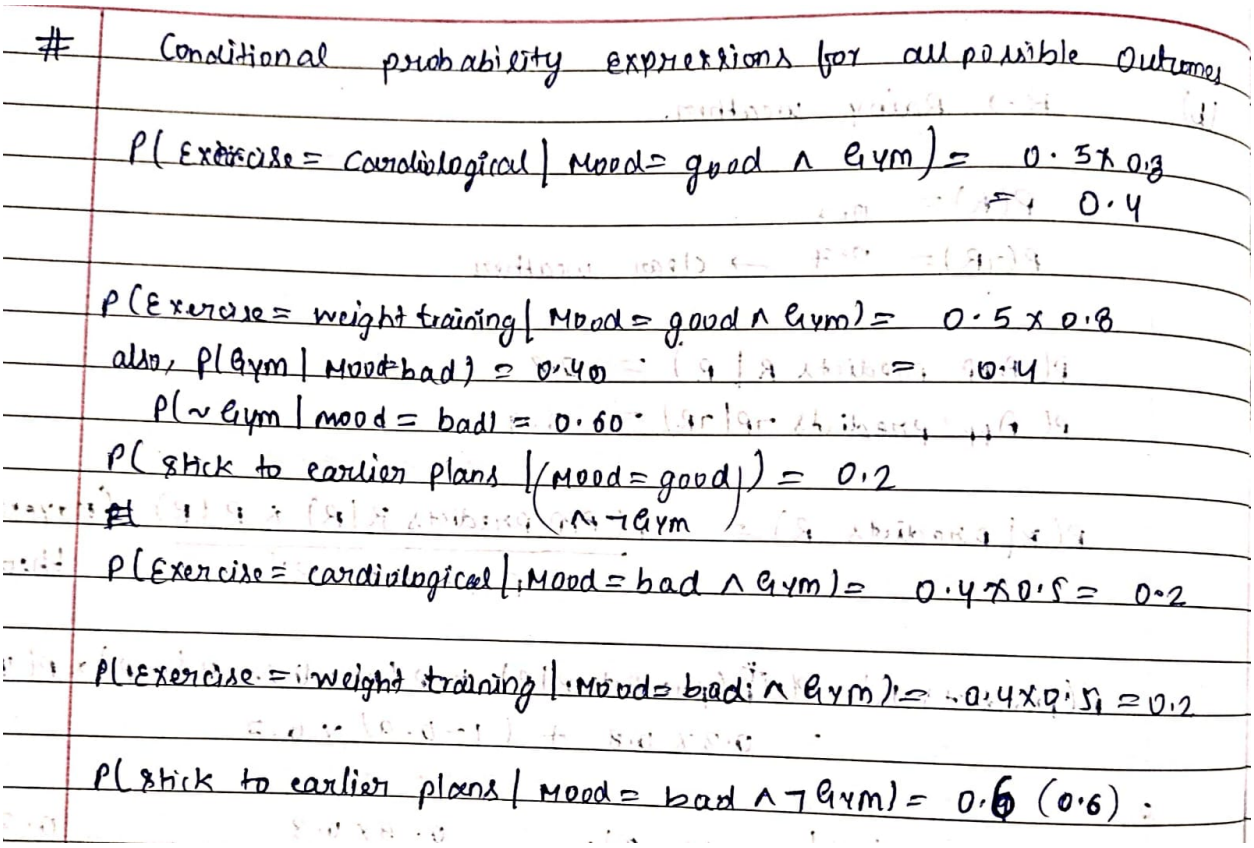
$\Rightarrow P(R) = \frac{2}{7}$

Hence, $P(R|PR) = \frac{P(PR|R) \times P(R)}{P(PR)} = \frac{0.8 \times \frac{2}{7}}{0.3}$

$= \frac{16}{21} \approx 0.7619$

(c)




Conditional probability expressions for all possible Outcomes
P(Exercise = Cardiological | Mood = good ∧ Gym) = 0.5 × 0.8 = 0.4
P(Exercise = weight training | Mood = good ∧ Gym) = 0.5 × 0.8 = 0.4
also, P(Gym | Mood = bad) = 0.40
P(~Gym | mood = bad) = 0.60
P(stick to earlier plans | (Mood = good) ∧ ¬Gym) = 0.2
P(Exercise = cardiological | Mood = bad ∧ Gym) = 0.4 × 0.5 = 0.2
P(Exercise = weight training | Mood = bad ∧ Gym) = 0.4 × 0.5 = 0.2
P(stick to earlier plans | Mood = bad ∧ ¬Gym) = 0.6 (0.6)

(d)

(d) $P(F=7 \mid \text{Good mood}) = 0.7$

$P(F=7 \mid \text{Bad mood}) = 0.45$

$P(\text{Good mood}) = 0.6$

$P(\text{Bad mood}) = 0.4$

Now,

$P(F=7) = P(F=7 \wedge \text{Good mood}) + P(F=7 \wedge \text{Bad mood})$

$= P(F=7 \mid \text{Good mood}) \times P(\text{Good mood}) + P(F=7 \mid \text{Bad mood}) \times P(\text{Bad mood})$

$= 0.7 \times 0.6 + 0.45 \times 0.4$

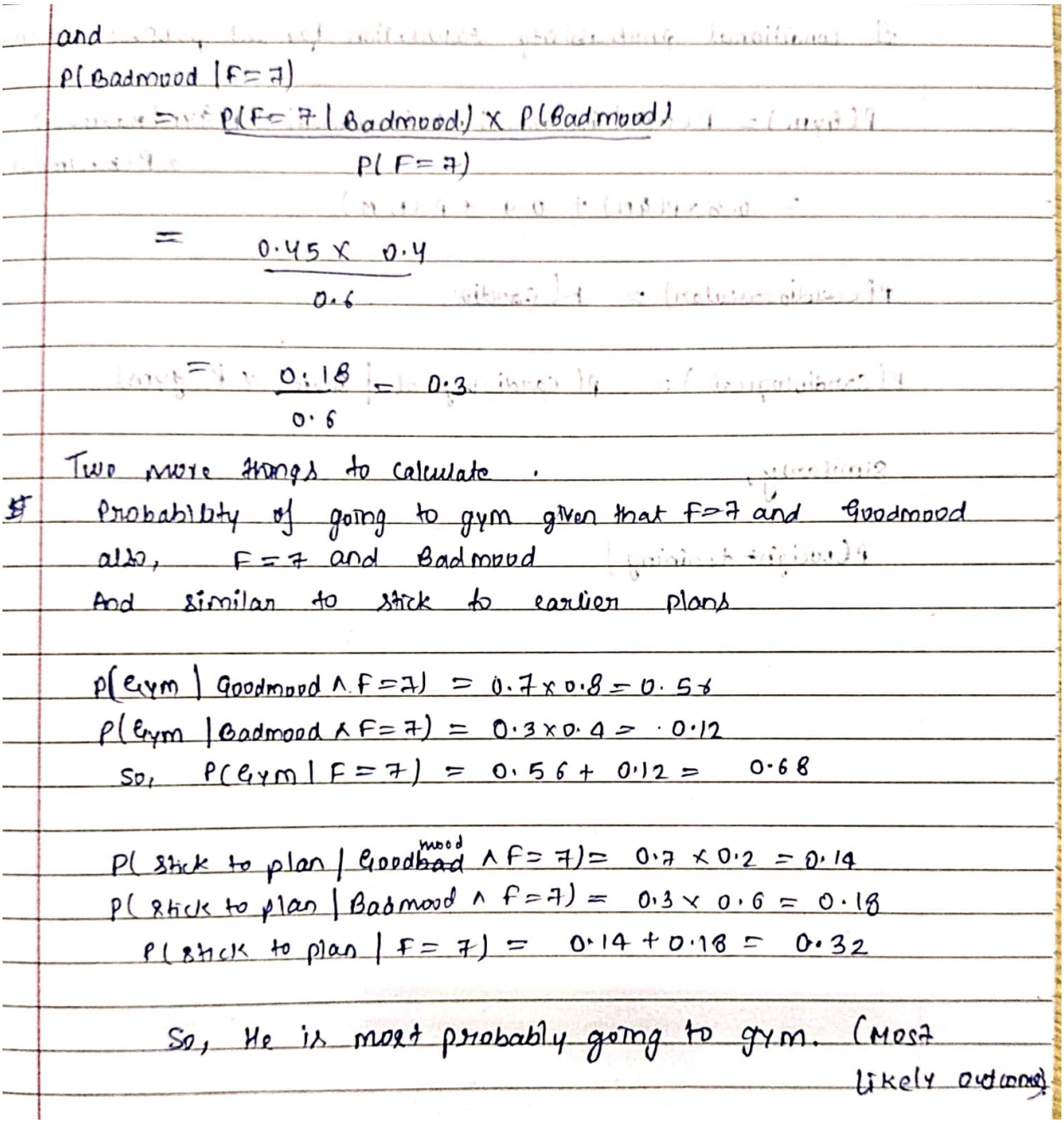
$= 0.42 + 0.18$

$= 0.6$

Hence, $P(F=7) = 0.6$

Also, $P(\text{Good mood} \mid F=7) = \dfrac{P(F=7 \mid \text{Good mood}) \times P(\text{Good mood})}{P(F=7)}$

$= \dfrac{0.7 \times 0.6}{0.6} = 0.7$



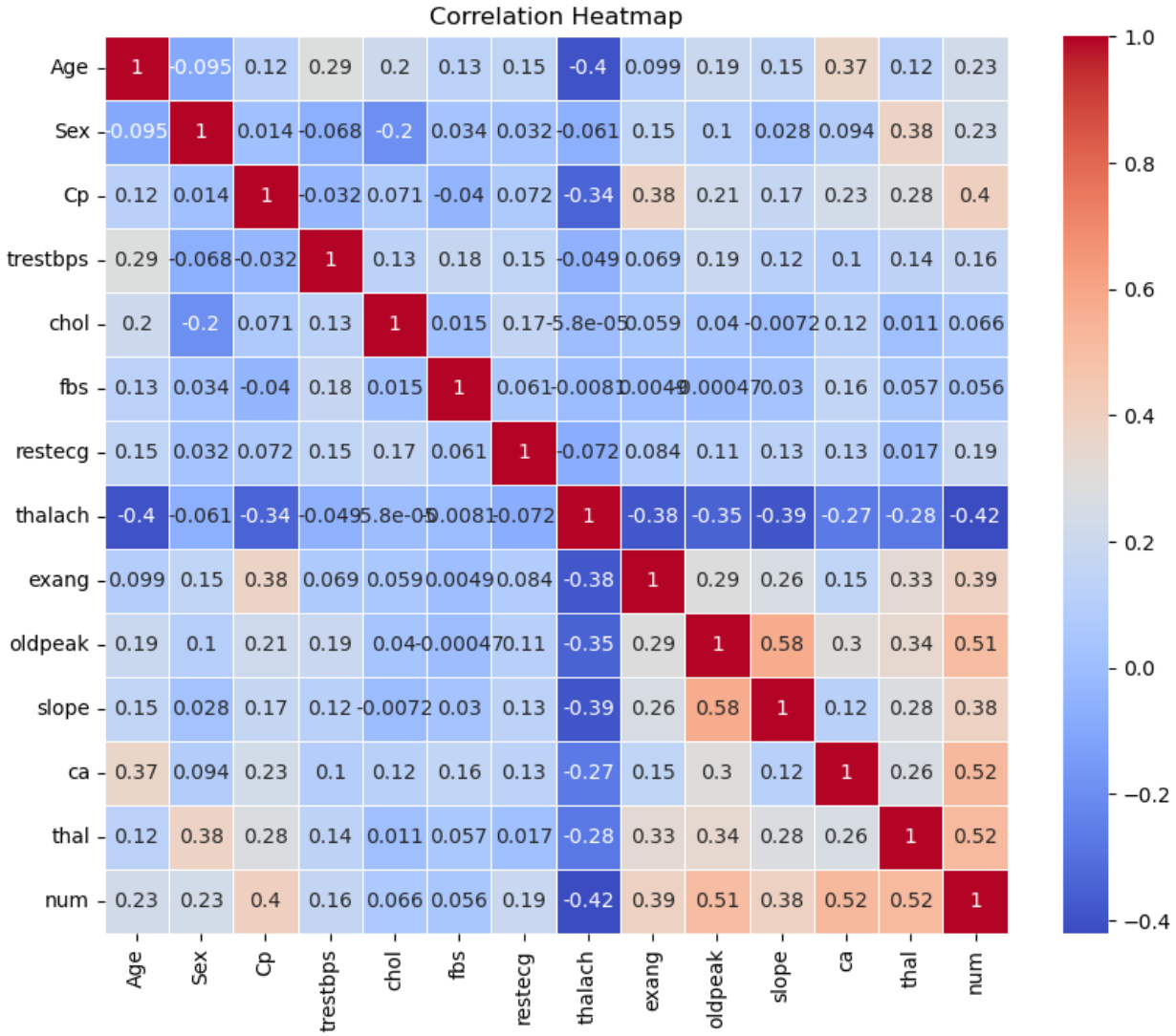
SECTION B:

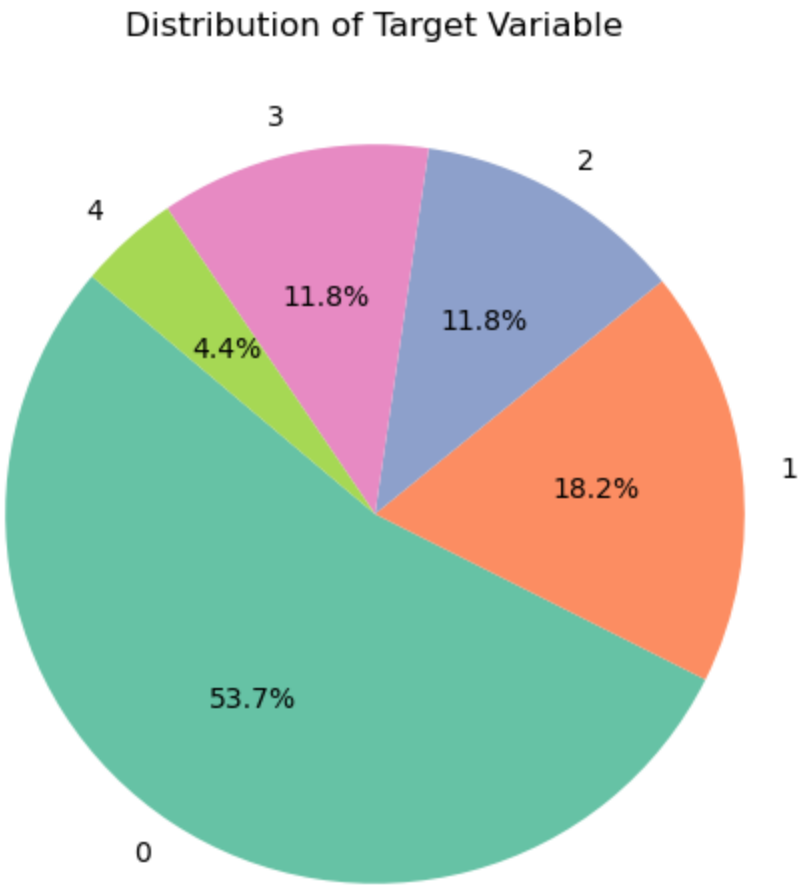
Preprocessing the dataset involves checking the null values and printing basic information and summary statistics of the dataset:

```
RangeIndex: 296 entries, 0 to 295
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Age       296 non-null    float64
 1   Sex       296 non-null    float64
 2   Cp        296 non-null    float64
 3   trestbps  296 non-null    float64
 4   chol      296 non-null    float64
 5   fbs       296 non-null    float64
 6   restecg   296 non-null    float64
 7   thalach   296 non-null    float64
 8   exang     296 non-null    float64
 9   oldpeak   296 non-null    float64
 10  slope     296 non-null    float64
 11  ca        296 non-null    float64
 12  thal      296 non-null    float64
 13  num       296 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 32.5 KB
None
```

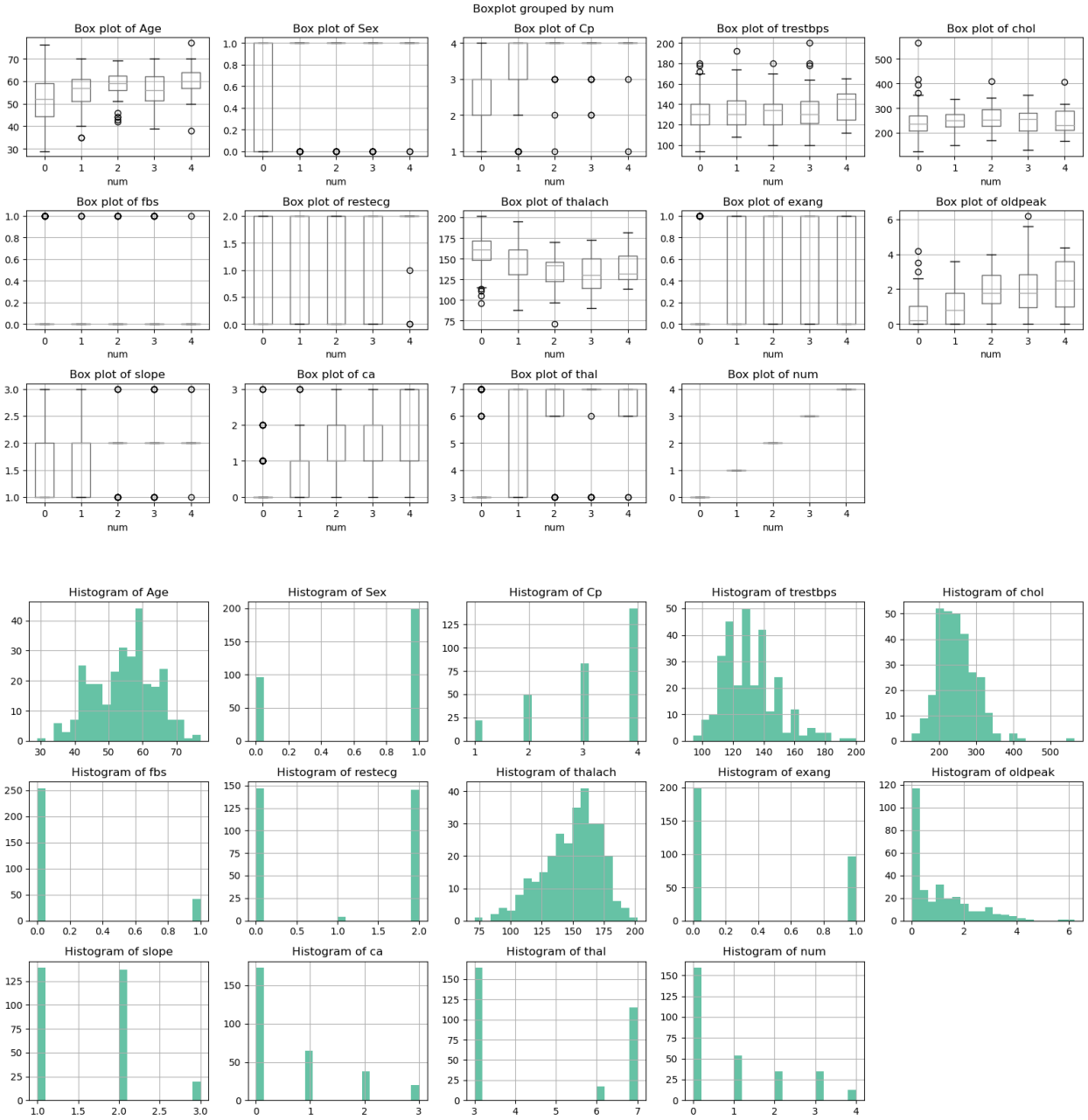
Exploratory data analysis includes a bar plot, pair plot, histograms, heatmap, piechart, boxplots, and Histograms as shown below:


Distribution of Target Variable
Count
160
140
120
100
80
60
40
20
0
0
1
2
3
4
Target Class


Correlation Heatmap
Age 1 -0.095 0.12 0.29 0.2 0.13 0.15 -0.4 0.099 0.19 0.15 0.37 0.12 0.23
Sex -0.095 1 0.014 -0.068 -0.2 0.034 0.032 -0.061 0.15 0.1 0.028 0.094 0.38 0.23
Cp 0.12 0.014 1 -0.032 0.071 -0.04 0.072 -0.34 0.38 0.21 0.17 0.23 0.28 0.4
trestbps 0.29 -0.068 -0.032 1 0.13 0.18 0.15 -0.049 0.069 0.19 0.12 0.1 0.14 0.16
chol 0.2 -0.2 0.071 0.13 1 0.015 0.17 -5.8e-05 0.059 0.04 -0.0072 0.12 0.011 0.066
fbs 0.13 0.034 -0.04 0.18 0.015 1 0.061 -0.0081 0.0049 0.00047 0.03 0.16 0.057 0.056
restecg 0.15 0.032 0.072 0.15 0.17 0.061 1 -0.072 0.084 0.11 0.13 0.13 0.017 0.19
thalach -0.4 -0.061 -0.34 -0.049 5.8e-05 0.0081 -0.072 1 -0.38 -0.35 -0.39 -0.27 -0.28 -0.42
exang 0.099 0.15 0.38 0.069 0.059 0.0049 0.084 -0.38 1 0.29 0.26 0.15 0.33 0.39
oldpeak 0.19 0.1 0.21 0.19 0.04 -0.00047 0.11 -0.35 0.29 1 0.58 0.3 0.34 0.51
slope 0.15 0.028 0.17 0.12 -0.0072 0.03 0.13 -0.39 0.26 0.58 1 0.12 0.28 0.38
ca 0.37 0.094 0.23 0.1 0.12 0.16 0.13 -0.27 0.15 0.3 0.12 1 0.26 0.52
thal 0.12 0.38 0.28 0.14 0.011 0.057 0.017 -0.28 0.33 0.34 0.28 0.26 1 0.52
num 0.23 0.23 0.4 0.16 0.066 0.056 0.19 -0.42 0.39 0.51 0.38 0.52 0.52 1
Age Sex Cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


Distribution of Target Variable
3
2
4
11.8%
11.8%
4.4%
1
18.2%
53.7%
0


Age
Sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
num



The target variable was classified into 5 classes 0 to 4 so for binary classification i took 0 as one negative class and 1-4 as another positive class.

After training decision trees using ‘entropy’ and ‘gini impurity’ as the splitting criterion training and testing dataset (80:20) I got:

Accuracy scores based on criteria:
entropy: 0.7416
gini: 0.7164
The best criterion for attribute selection is: entropy
Test accuracy with the best criterion: 0.7667

Then performed a grid search on variables “min_samples_split” and “max_features” got their best values as follows:

```
Best hyperparameters found by Grid Search:
{'max_features': None, 'min_samples_split': 20}
Test accuracy with the best hyperparameters: 0.7667
```

Finally trained a random forest classifier on the same dataset and performed a Grid Search for the parameters n_estimators, max_depth and min_samples_split.

I got results as follows:

```
Best hyperparameters found by Grid Search:
{'max_depth': None, 'min_samples_split': 10, 'n_estimators': 100}
Classification Report on Test Data:
              precision    recall  f1-score   support

           0       0.87      0.92      0.89        36
           1       0.86      0.79      0.83        24

    accuracy                           0.87        60
   macro avg       0.87      0.85      0.86        60
weighted avg       0.87      0.87      0.87        60
```

SECTION C:

1. _`build_tree(self, X, y, depth)`

- Recursively constructs the decision tree nodes.
- Takes the input features `X`, target values `y`, and the current depth of the tree.
- Stops the recursion if the maximum depth is reached or if all target values in a node are the same.
- Finds the best split for the current node using `_find_best_split`.
- Recursively constructs left and right subtrees.

2. `_find_best_split(self, X, y)`
    - Finds the best split for a node by iterating through features and thresholds.
    - Calculate the Gini impurity for each split and select the one with the lowest impurity.
    - Returns the feature, threshold, and indices for the left and right child nodes.
3. `_compute_gini_impurity(self, y, left_indices, right_indices)`
    - Computes the Gini impurity for a split.
    - Takes the target values `y`, left, and right indices.
    - Calculates the Gini impurity for both left and right child nodes and combines them.
4. `_gini_index(self, y)`
    - Computes the Gini index for a set of target values.
    - Takes the target values `y`.
    - Calculates the Gini index, a measure of impurity for the given node.

I made a split of 80: 20 from the given dataset as training and testing, The target variable was classified into 4 categories: “negative”, “hyperthyroid”, “T3 toxic”, and “goitre” so for binary classification I took negative as 0 (negative class) and rest as 1 (positive class) and encoded the dataset accordingly.

After training and evaluating the above implementation on the given dataset, I observed that accuracy increased as I increased the max_depth of the tree and finally, it gave 100% accuracy on the training dataset and 98.0357%.