Machine Learning (CSE 343) - Assignment 2: Report

- Arnav Goel (Roll No: 2021519)

Section A:

Q1)

A)

Random forest is an ensemble of decision trees, and the variance of the ensemble is defined as due to all trees being identically distributed:

$$Var = \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

Where ρ is the positive pairwise correlation coefficient between the trees, B is the number of trees and σ^2 is the variance of each tree. We can see that the number of trees and the correlation has a strong relation with the variance of the ensemble. Thus, the interplay of these two variables tweaks the variance and causes the model to overfit or underfit the data. There is an interplay between the correlation and diversity of the ensemble. We need diverse trees to ensure that the weaknesses of each classifier can be compensated for in the ensemble and every possible wrong prediction can be covered.

However, a completely random selection of trees or features can cause each classifier to be extremely weak, which can't be compensated for even with an ensemble. Thus, there needs to exist some correlation between the trees and how they predict their outputs. However, here, a high correlation can overfit the noise in the data. Thus, it is important for correlation to exist till a certain level and maintain some diversity. This is done through bootstrapping samples and randomly selecting feature subsets.

B)

The "curse of dimensionality" refers to the problems which arise when organising, managing and working with high-dimensional data. In the context of a Naive Bayes Classifier, it can lead to the following problems:

i) Sparsity in Data: As dimensionality increases, data can become sparse. This sparsity can give rise to features with zero frequency. It can cause problems with inference using the Naive Bayes Classifier.

ii) Computationally Expensive: Naive Bayes Classifiers are known to be computationally efficient due to a non-iterative training process. However, the presence of high dimensions will cause the process of inference and training to increase substantially due to conditional probabilities being calculated for each feature. This will be especially problematic if the features are continuous and require a probability density function.

iii) Irrelevant Features: Since NB gives equal importance to each feature, it will also take into account features which are irrelevant to the final prediction.

Some techniques which can be used to mitigate this include **Feature Selection** (through various hypothesis testing) and **Feature Extraction** through dimensionality reduction algorithms like PCA, LDA and t-SNE.

C)

The Naive Bayes Classifier makes this naive assumption of conditional independence between the features of the dataset. However, encountering attributes in the testing set which were not present while training the model is a real problem with this classifier. **Yes,** this will affect the inference results, referred to as the **"zero-frequency problem".** The classifier, when met with a new attribute, will assign a probability of **0** to this particular occurrence. Since features are assumed to be independent, the probabilities are multiplied, and this will cause the overall prediction to be zero, which is grossly inaccurate and will lead to an incorrect inference.

An example of this could be a spam classifier wherein we encounter a word not in the training dataset. This assignment of zero will thus skew the probability distribution and worsen our model. Some of the techniques to handle this problem are:

i) **Laplace Smoothing:** Add one to each count of the feature-label combination when calculating the probabilities.

ii) **m-estimate:** A generalisation of laplace smoothing wherein a parameter "m" is estimated and multiplied by "p", which is the prior estimate of the probability of the feature. This is then added instead of one.

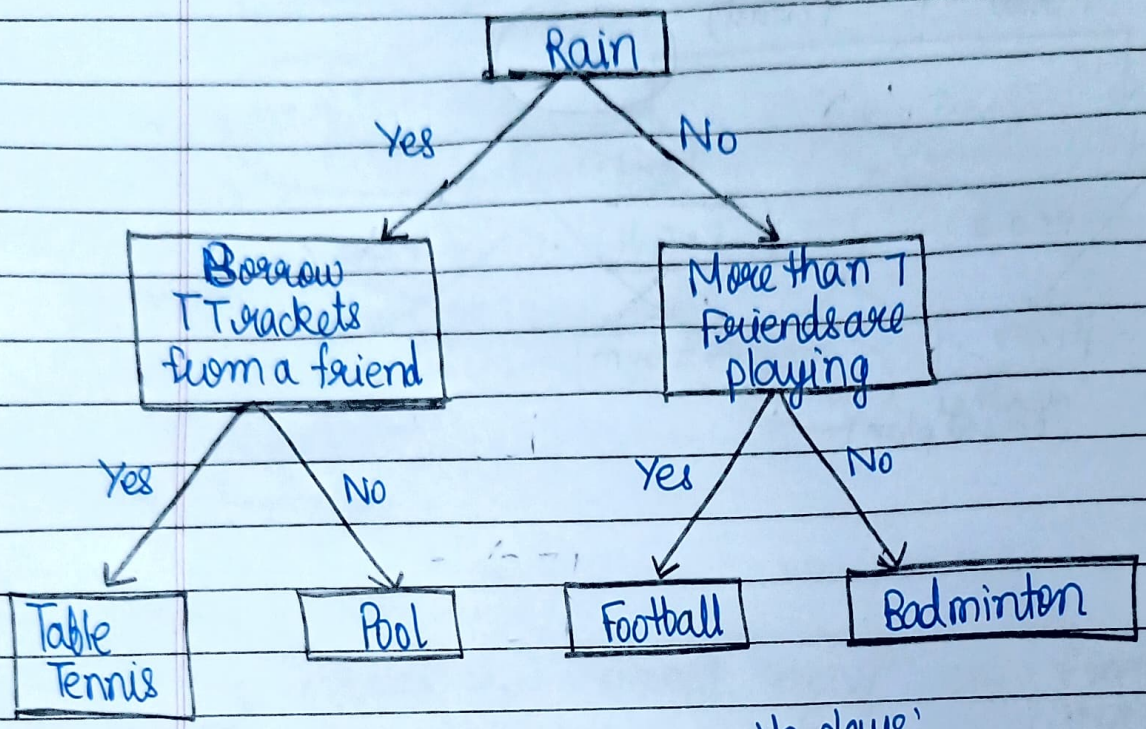
D)

Yes, using Information Gain as a splitting criterion will make splitting more biased towards attributes than higher cardinality (i.e. more number of distinct values). This is mainly because Information Gain depends on entropy reduction, and attributes with larger cardinality will have higher entropies. Attributes with many distinct values can almost perfectly partition the data. In contrast, those with fewer distinct values cannot perform a clean partition and thus would get neglected.

Let's take an example of an attribute like **Customer ID**, where each row has a unique value. Splitting on this will for each unique attribute will create samples with 0 entropy each eventually. However, this is overfitting on the data and is not a desirable outcome. It tells us how Information Gain can mess up due to its dependence on entropy.

An alternative splitting criterion could be the **Gain Ratio,** defined as the ratio of **Information Gain** and **Intrinsic Information (II).** Intrinsic Information captures the intrinsic information of a split (which is high for high-cardinality attributes). This modification of Information Gain thus improves the criterion, reducing its bias to attributes with higher cardinality.

Q2) A2)a) The decision tree for the given scenario:



He plays:

∴ The possible outcomes are: (i) Table Tennis
(ii) Pool
(iii) Football
(iv) Badminton

Let event A → Rain ∴ ~A : No rain

∴ P(A) : Probability of it raining

event B → Borrows TT rackets from a friend

event C → More than 7 friends are playing.

∴ Conditional prob expression for each outcome:

(i) $P(\text{plays TT}) = P(\text{plays TT} \mid B) \cdot P(B \mid A) \cdot P(A)$

(ii) $P(\text{plays pool}) = P(\text{plays pool} \mid \sim B) \cdot P(\sim B \mid A) \cdot P(A)$

(iii) $P(\text{plays football}) = P(\text{plays football} \mid C) \cdot P(C \mid \sim A) \cdot P(\sim A)$

(iv) $P(\text{plays badminton}) = P(\text{plays badminton} \mid \sim C) \cdot P(\sim C \mid \sim A) \cdot P(\sim A)$

(b) i) $R_a \rightarrow$ Event that app pred. "Rainy" iii) $R \rightarrow$ Event that it actually rained

ii) $R_c \rightarrow$ App predicts "clear" $\equiv \sim R_a$

iv) $C \rightarrow$ Event that it actually clear $\boxed{C \equiv \sim R}$

Given: $P(R_a) = 0.3$ — ①

$P(\sim R_a) = P(R_c) = 0.7$ — ②

Accuracy for rainy day:

Given $\rightarrow P(R_a|R) = 0.8$; $P(\sim R_a|R) = 0.2$

for clear:

$P(\sim R_a|C) = 0.9$

$\therefore P(R_a|C) = 0.1$

Q $P(R|R_a) = ?$

For this:

$$P(R|R_a) = \frac{P(R_a|R) \cdot P(R)}{P(R_a)}$$

$$P(R_a) = P(R_a|R) \cdot P(R) + P(R_a|C) \cdot P(C) \quad (\because \sim C \equiv R$$

(Not clear is a rainy day)

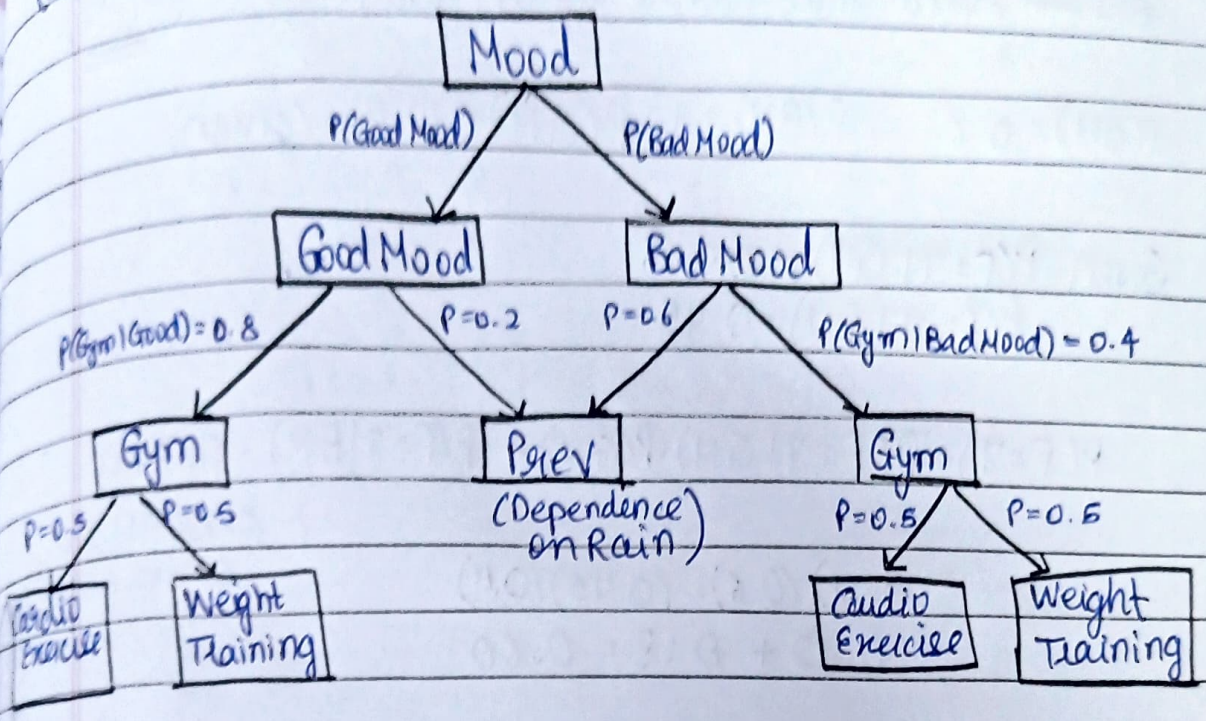
$$0.3 = 0.8P + 0.1(1-P)$$

$$\Rightarrow 0.2 = 0.7P \Rightarrow \boxed{P(R) = 2/7} \text{ — ①}$$

$$\therefore P(R|R_a) = (0.8)\left(\frac{2}{7}\right) \times \frac{1}{0.3} = \frac{16}{21} \sim 0.762$$

Ans. The probability $P(R|R_a)$ is thus 0.762.

(c) The decision tree for this situation will be:



The possible outcomes are: i) Cardio exercise ii) Weight training iii) Previous (↓ from part A)

Let event G → Good Mood ; ~G → Bad Mood

M → Goes to Gym

C → Cardio Exercise W → Does weight training

P → Previous does as in part (A)

(i) $P(\text{cardio exercise}) = P(C|M) \cdot P(M|G) \cdot P(G) + P(C|M) \cdot P(M|\sim G) \cdot P(\sim G)$

(ii) $P(\text{weight training}) = P(W|M) \cdot P(M|G) \cdot P(G) + P(W|M) \cdot P(M|\sim G) \cdot P(\sim G)$

(iii) $P(\text{does as previously}) = P(P|G) \cdot P(G) + P(P|\sim G) \cdot P(\sim G)$

Above are the conditional expressions, for each possible outcome.

(d) GM → Event that Rahul is in good mood
BM → Event that Rahul is in bad mood ≡ (~GM)

P(GM) = 0.6 ; P(BM) = 1 − P(GM) = 0.4 (given)

Given P(F=7 | GM) = 0.7
P(F=7 | BM) = 0.45

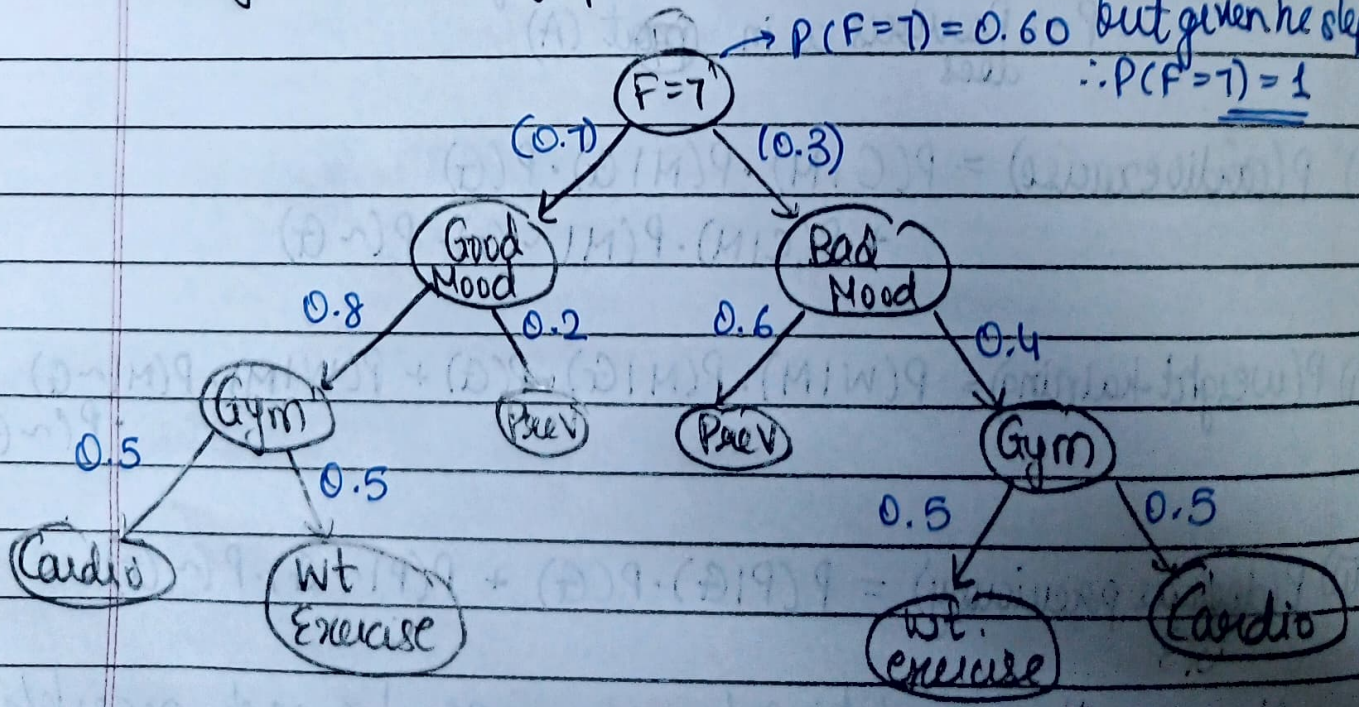
∴ P(F=7) = P(F=7 | GM) · P(GM) + P(F=7 | BM) · P(BM)
∵ GM & BM are comp. events

= (0.7)(0.6) + (0.45)(0.4)
= 0.42 + 0.18 = 0.60

∴ P(GM | P=7) = $\frac{(0.7)(0.6)}{0.6}$ = 0.7 — ①

P(~GM | P=7) = $\frac{(0.45)(0.4)}{0.6}$ = 0.3 — ②

Now
using the following prob. tree:



(i) P(cardio) = P(cardio|gym) . P(gym|good Mood) . P(GM|F) . P(F)
+ P(cardio|gym) . P(gym|BM) . P(BM|F) . P(F)

= (0.5)(0.8)(0.7)(1) + (0.6)(0.4)(0.3)(1)
= 0.28 + 0.06 = 0.34

(ii) P(wt. training) = P(wt|Gym) . P(Gym|GM) . P(GM|F) . P(F)
+ P(wt|Gym) . P(Gym|BM) . P(BM|F) . P(F)
= (0.5)(0.8)(0.7)(1) + (0.5)(0.4)(0.3)(1)
= 0.34

(iii) P(does as prev as in Pauta) = P(P|GM) . P(GM|F) . P(F) + P(P|~GM) . P(~GM|F) . P(F)

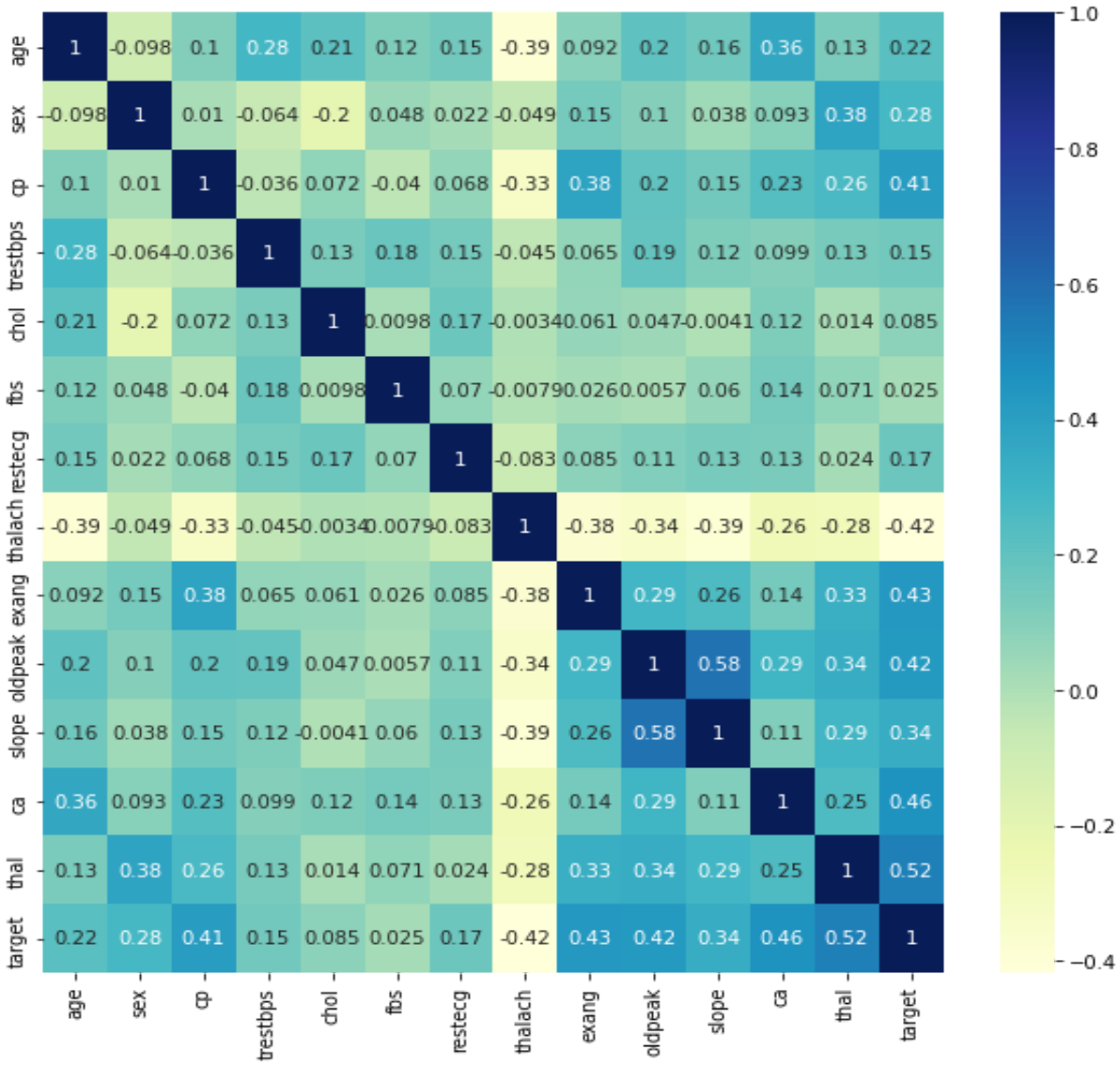
= (0.2)(0.7)(1) + (0.6)(0.3)(1)
= 0.14 + 0.18 = 0.32

Section B:
Library Implementation of a Decision Tree and Random Forest Classifier:

Part A:
We take the data from the UCI ML repository and perform the following standard pre-processing on the data:
i) We check for missing values. We found some, so we replaced them with the mean of the column.
ii) We normalised the data using **Z-score normalisation,** i.e. subtracting the mean of the column and dividing each value by the standard deviation.

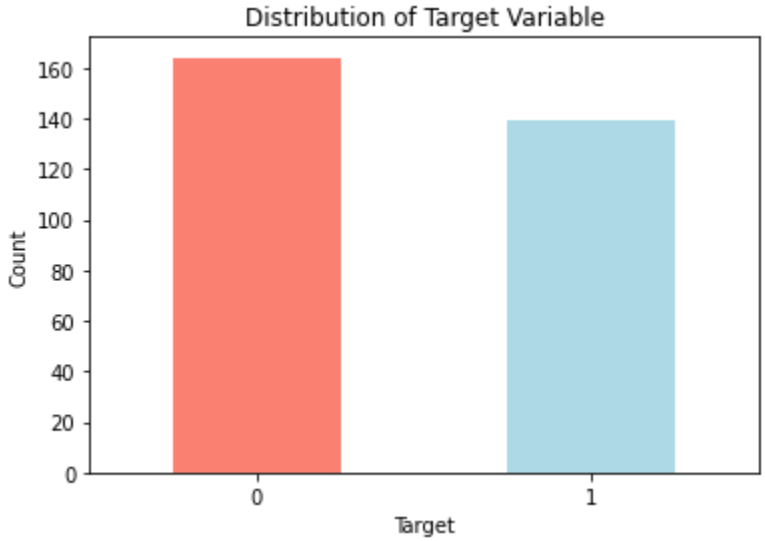
We performed some Exploratory Data Analysis before the normalisation of data, and it is presented below:

i) Correlation Heatmap:

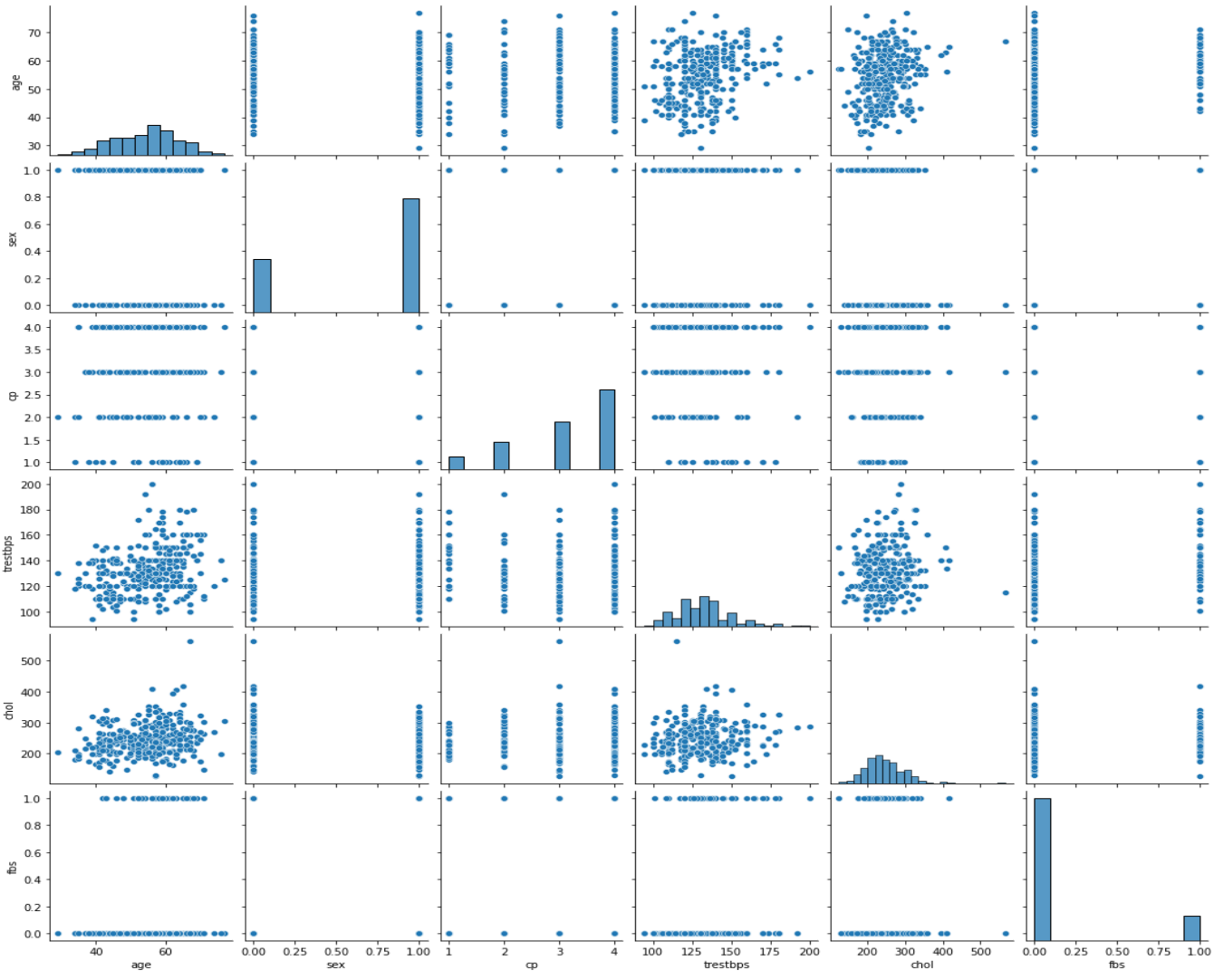


ii) Visualisation of Target Variable:

We merged classes 2, 3 and 4 into class 1 as asked by the question to help perform binary classification. The graph is shown below:



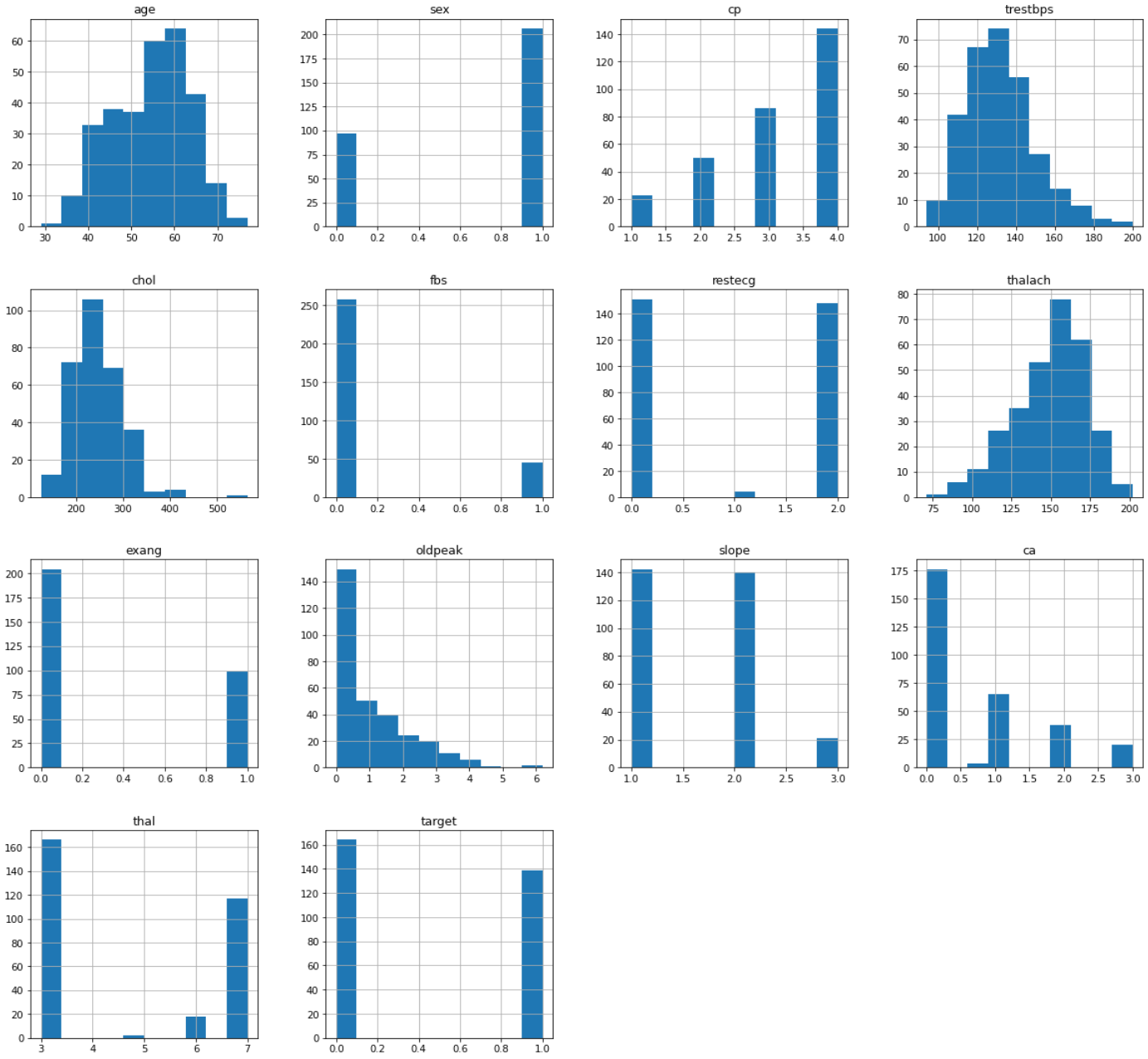
iii) Pairplots:



This shows the pair plots between the first six features of the dataset.

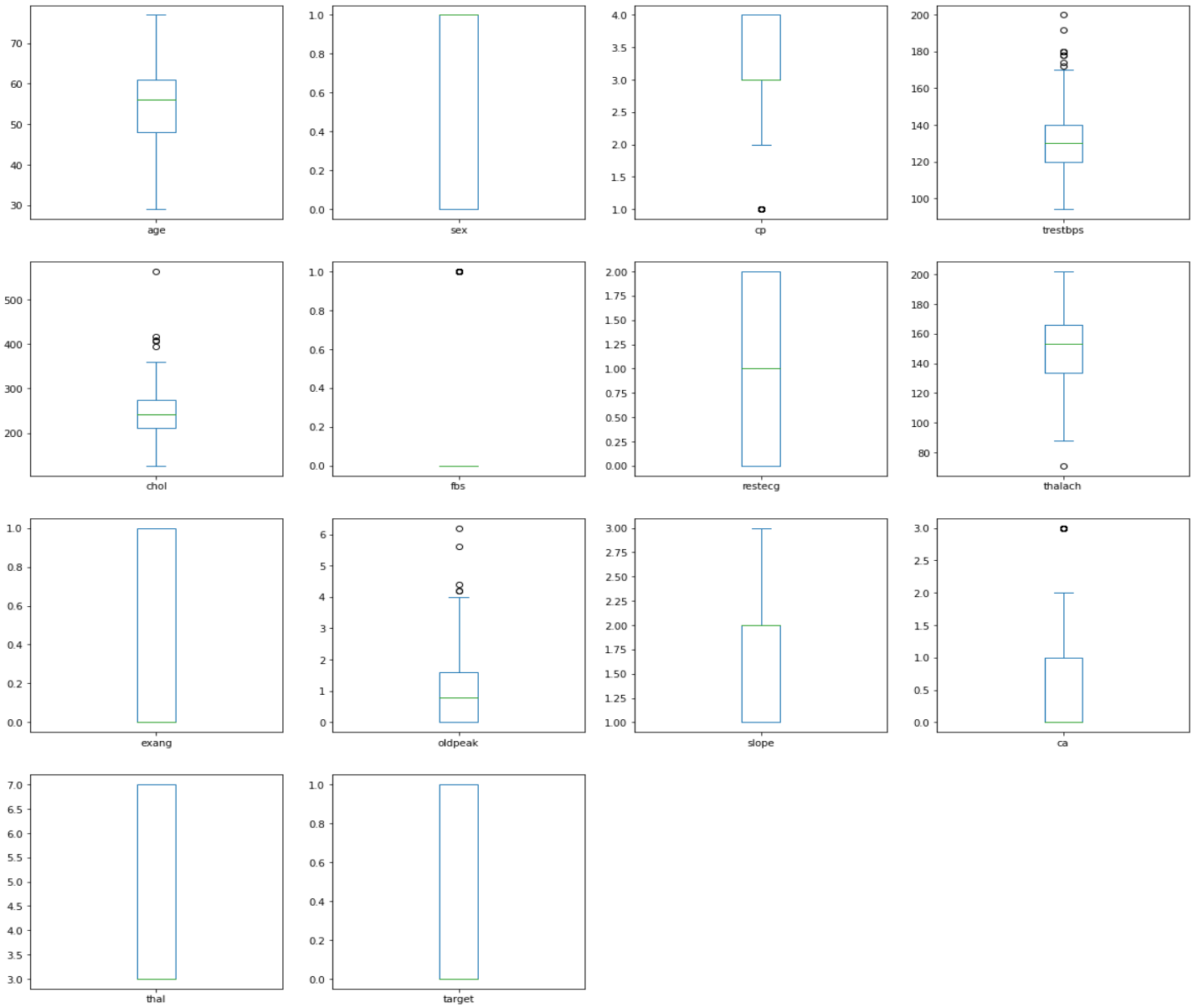
iv) Histograms:

The diagram below shows the histograms for all features in the dataset:



v) Boxplots:

The plot shows the boxplots between all features in the dataset:



Part B:

Part B:

- Split the dataset into train and tests in the ratio 80:20

+ Code + Markdown

```
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 0.0s Python

All computations are done ahead at **random state: 42**

Part C:

We trained two decision trees with “entropy” and “gini” criteria, and the following results were obtained:

```
Accuracy of Decision Tree Classifier with Entropy Criterion:  0.8032786885245902
Accuracy of Decision Tree Classifier with Gini Criterion:  0.7540983606557377


Decision Tree Classifier with Entropy Criterion is better.
```

- We pick Entropy Criterion

Part D:

This is the range of hyperparameters we performed grid-search on using 10-fold cross-validation:

Part D

+ Code + Markdown

```
# Grid Search to find the best parameters for Decision Tree Classifier
dtc = DecisionTreeClassifier(criterion=criteria_chosen, random_state=42)

# Defining the parameters
parameters = {'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10],
              'max_features': [None, 'sqrt', 'log2', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}

# Using GridSearchCV to find the best parameters
grid_search = GridSearchCV(estimator=dtc, param_grid=parameters, scoring='accuracy', cv=10, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)
```

[34] ✓ 1.1s Pytho

```
Fitting 10 folds for each of 117 candidates, totalling 1170 fits
```

This is the best hyperparameters we get and the best score:

```
Best Score:  0.8019999999999999
Best Params:  {'max_features': 9, 'min_samples_split': 5}
```

Part E:

The figure below shows the range of hyperparameters we tested for the random forest classifier and the best hyperparameters we got from this range. Note that we took the splitting criterion as “entropy” only as asked in the assignment.

Part E

```
# Grid Search to find the best parameters for Random Forest Classifier
rfc = RandomForestClassifier(random_state=0, criterion=criteria_chosen)

# Defining the parameters
parameters = {'n_estimators': [5, 10, 20, 30, 40, 50, 60], 'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10],
              'max_depth': [None, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}

# Using GridSearchCV to find the best parameters using 10-fold cross validation
grid_search = GridSearchCV(estimator=rfc, param_grid=parameters, scoring='accuracy', cv=10, n_jobs=-1, verbose=1)

# Fitting the model
grid_search.fit(X_train, y_train)

# Displaying the best parameters
print("Best Hyper Parameters:\n", grid_search.best_params_)
```

[20] ✓ 1m 0.5s Pyth

```
Fitting 10 folds for each of 693 candidates, totalling 6930 fits
Best Hyper Parameters:
 {'max_depth': 2, 'min_samples_split': 2, 'n_estimators': 60}
```

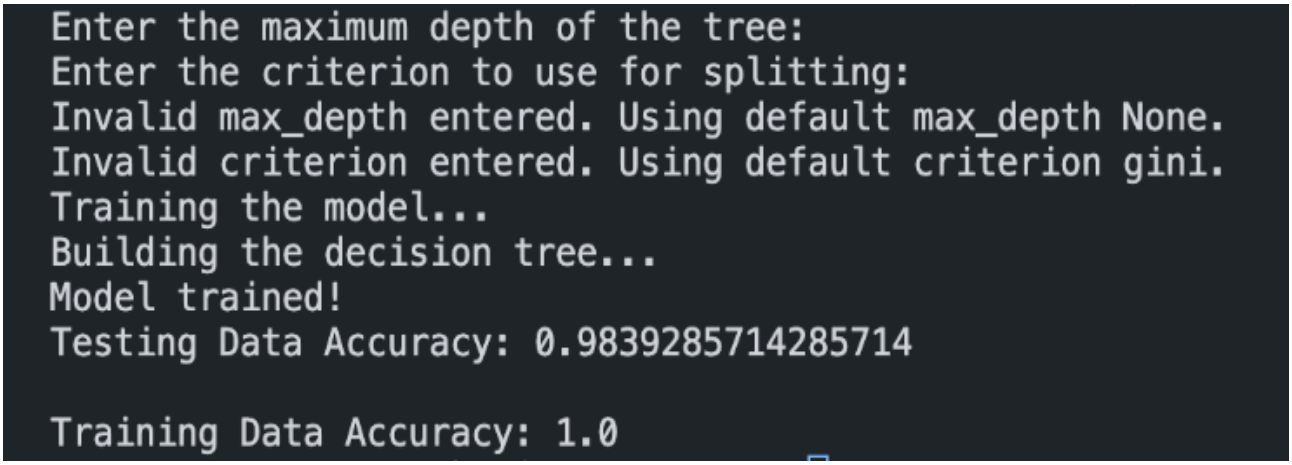
Now, we prepared a random forest classifier with the best hyperparameters and got the following accuracy on the testing data and the corresponding classification report:

```
Accuracy of Random Forest Classifier with Best Parameters:  0.9180327868852459
              precision    recall  f1-score   support

           0       0.88      0.97      0.92        29
           1       0.97      0.88      0.92        32

    accuracy                           0.92        61
   macro avg       0.92      0.92      0.92        61
weighted avg       0.92      0.92      0.92        61
```

Section C:

- I implemented a Decision Tree from scratch, which used both Information Gain and Gini Index for splitting the nodes. The user can input which criterion they want to use, and accordingly, it will be used for splitting the nodes. The default criterion is set as **"gini".**
- The Max_Depth is also input by the user, and the default is set as **None.**
- **A well commented and documented code-file is available for submissions and can be checked.**

The following results were obtained by my Decision Tree on the given dataset:
For max_depth = None and criterion = gini



For max_depth = 5 and criterion = entropy