Decision Tree Report

1) Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute and each leaf nodes corresponds to a class label.

The decision of making strategic splits heavily affects a tree's accuracy. The algorithm selection is also based on type of target variables. The four most commonly used algorithms in decision tree are:

Gini Index: Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.It works with categorical target variable "Yes" or "No". It performs only Binary splits. Higher the value of Gini higher the homogeneity. CART (Classification and Regression Tree) uses Gini method to create binary splits.

Gini Calculation: Calculate Gini for sub-nodes, using formula sum of square of probability for yes and no (p^2+q^2) then calculate Gini for split using weighted Gini score of each node of that split.

Chi-Square: It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it by sum of squares of standardized differences between observed and expected frequencies of target variable.It works with categorical target variable "Yes" or "No". It can perform two or more splits. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node. Chi-Square of each node is calculated using formula,

Chi-square = $((\text{Actual} — \text{Expected})^2 / \text{Expected})^{1}/2$

Chi-square Calculation: Calculate Chi-square for individual node by calculating the deviation for Yes and No both then calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

Information Gain: Less impure node requires less information to describe it. And, more impure node requires more information. Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% — 50%), it has entropy of one.

Entropy can be calculated using formula:- Entropy = -p log2 p — q log2q

Here p and q is probability of success and failure respectively in that node. Entropy is also used with categorical target variable. It chooses the split which has lowest entropy compared to parent node and other splits. The lesser the entropy, the better it is.

Entropy Calculation: Calculate entropy of parent node, then calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split. We can derive information gain from entropy as **1- Entropy.**

Reduction in Variance

Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Above X-bar is mean of the values, X is actual and n is number of values.

Variance Calculation: Calculate variance for each node then calculate variance for each split as weighted average of each node variance.

Naive Bayes classifier

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the posterior probability of *class* (A, *target*) given *predictor* (B, *attributes*).
- $P(A)$ is the prior probability of *class*.
- $P(B|A)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(B)$ is the prior probability of *predictor*.

There are three types of Naive Bayes model under the scikit-learn library:

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

3 Applications of Naive Bayes Algorithms

- Real time Prediction
- Multi class Prediction
- Text classification/ Spam Filtering/ Sentiment Analysis

2)Dataset description:

The dataset used in the assignment is cardio vascular disease dataset from Kaggle. This has 11 features and a target. The features include

1. Age | int (days)
2. Height | int (cm)
3. Weight | float (kg)|
4. Gender
5. Systolic blood pressure
6. Diastolic blood pressure
7. Cholesterol | 1: normal, 2: above normal, 3: well above normal
8. Glucose | 1: normal, 2: above normal, 3: well above normal
9. Smoking
10. Alcohol intake
11. Physical activity

Target:

12. Presence or absence of cardiovascular disease | Target Variable | binary(0,1)

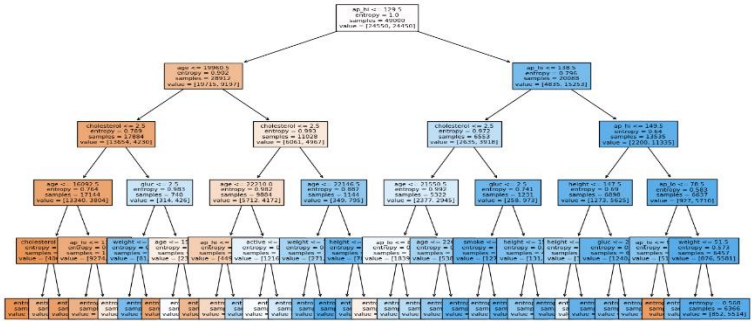
This data has some interesting values for height and weight. For example, there are weights as low as 10kg and a height upto 250cm. I feel that data is not legitimate and some randomly generated values.

3) Preprocessing:

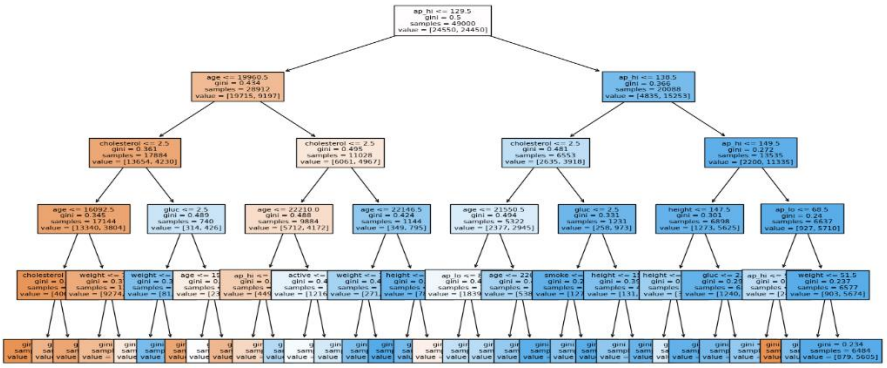
1) Data is loaded as a csv file into the pandas dataframe using ";" as a separator.
2) There are no null values in the dataset , so no rows are omitted or interpolated to get the values.
3) All the features are stored into a feature variable and target is stored into a target variable.
4) Id is dropped from the features as it is not a significant feature for the classification.
5) The data is split into train_data and test_data(70%:30%) using train_test_split method from sklearn package.

4) Visualizing the decesion tree for gini and entropy

Gini: Here is the decision tree generated based on Gini index classification. This decision is obtained by calculating Gini index and information gain for each of the attributes in the dataset.



Entropy: Here is the decision tree generated based on Entropy classification. This Decision Tree is constructed by using the values of entropy and Miscalculation Error for the attribute.



5) Gini vs Entropy

Visualisation of decision tree using gini and entropy are attached as PNG files.

Gini and entropy are used to select the node for splitting on classification/regression trees. They are just different methods to calculate Information Gain based on which the nodes are split. Gini and entropy are calculated as below:

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

Results for Gini and entropy for the heart dataset can be analysed below:

	Entropy	gini
Confusion matrix	[[8119 2352] [3305 7224]]	[[8119 2352] [3305 7224]]
Accuracy	73.06	73.06

As you can see both the methods performed similar. The accuracy is same using the both methods. The confusion matrix also looks very similar. Hence any of the methods gives us similar results.

Naïve Bayes:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is the posterior probability of *class* (A, *target*) given *predictor* (B, *attributes*).
- $P(A)$ is the prior probability of *class*.
- $P(B|A)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(B)$ is the prior probability of *predictor*.

The following table show the results of accuracy and confusion matrix for given dataset by using naïve Bayes

Accuracy	0.5944
Confusion Matrix	[[9228 1243] [7274 3255]]

Accuracy for Naïve Bayes is way less than decision tree . Hence we can conclude that decision tree performed far better than Naïve Bayes.

References (Sources)

1. https://www.datacamp.com/community/tutorials/decision-tree-classification-python
2. https://stackabuse.com/decision-trees-in-python-with-scikit-learn/