CSE 343: Machine Learning - Assignment 2

Akshat Gupta

November 5, 2023

Section A

1. (a) With respect to Random Forests:
   - **Correlation -** This refers to how similar or related the individual trees in the ensemble are to each other, i.e. how similar will the predictions be for that set of trees.
   - **Diversity -**This refers to how distinct the individual trees are from each other. This indicates how varied the predictions can be for the set of trees.

   High correlation can lead to over-fitting; this is because highly correlated trees would imply that the ensemble is just learning the one pattern that is depicted by the rote data. If all the trees in the ensemble make the same errors on a certain data point, the result will most likely be incorrect for that data point. Ideally-correlated trees would provide convergent predictions and thus, increase the confidence in the predictions for unseen data. Hence, correlation is desirable to maintain the strength of the ensemble, but only to a certain extent.
   Extremely low correlation would weaken the performance of the ensemble since there will be a very wide spread in the predictions, thus increasing the variance of the ensemble. This can make the ensemble less reliable, especially when making predictions on unseen data. But, a certain level of diversity must also be maintained to capture complementary patterns without the ensemble becoming overly noisy.

   (b) **Curse of Dimensionality -** As the number of dimensions or features increases, the amount of data needed to train the machine learning model accurately increases exponentially. This is because an increase in dimensions would likely make the data sparse, and thus would require a larger number of samples to generalize the data better.
   In Naive Bayes, as the number of dimensions increases, the complexity of the model would also increase, thus making it computationally expensive and increasing the risk of over-fitting. Also, some feature combinations may have few or no samples in the training data, thus not allowing the model to make accurate predictions for such data points.
   To mitigate this problem, the following approaches can be taken:
   - **Feature Selection -** Choose the most relevant features to the problem by incorporating any domain knowledge available, thus reducing the number of dimensions the model needs to operate on and making more focused predictions.
   - **Dimensionality Reduction Techniques -** Employing techniques like PCA or t-SNE can help reduce the dimensionality of the data keeping only the most statistically significant attributes.

   (c) When a Naive Bayes classifier encounters some value of attributes that was not present in the training dataset, the model will most likely not be able to make accurate predictions for the same since it will infer these values as having 0 probability given it has not seen these values of attributes in the training data. This will thus, affect the inference results negatively.
   For example, consider a dataset with a categorical attribute A which has values 0,1, and 2 in the training set. If the test set has a sample point with value 3 for the attribute A, Naive Bayes would give a probability= 0 for the sample with respect to any class. It would thus fail to classify this sample point correctly.
   It is possible to mitigate this issue using:
   - **Laplace Smoothing -** This adds a small count (typically 1) to each attribute-value-class combination, ensuring that for no combination of attribute values, the probability is 0. For c = number of classes, $P(A_i|C) = \frac{N_{ic}+1}{N_c+c}$

- **m-estimate -** Similar to Laplace, this is also a smoothing technique that ensures no combination of values has a 0 probability, but instead of adding a constant value to each combination, it parameterizes the estimation of the probability:
  For c: number of classes, p: prior probability, m: parameter, $P(A_i|C) = \frac{N_{ic}+mp}{N_c+m}$

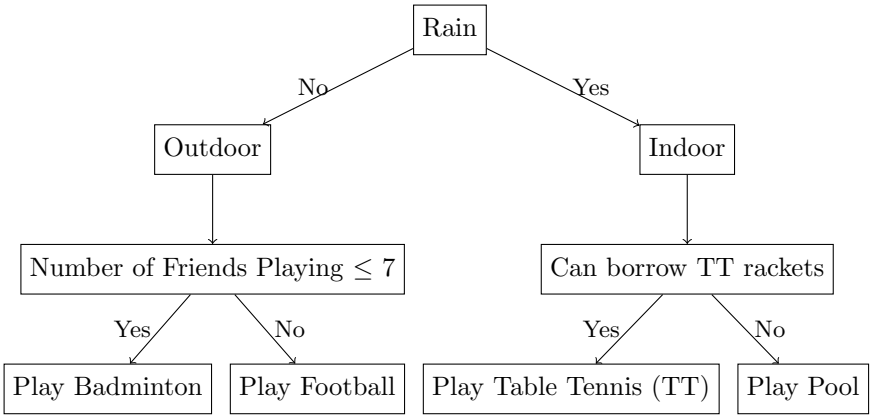
(d) Information Gain can be biased if some attributes have more cardinality than others. This is because Information Gain tends to prefer attributes that have higher cardinality since they result in more distributed splits (more number of subgroups formed), thus reducing the entropy much more than a split with relatively lower number of subgroups. These high cardinality attributes may not be the most significant with respect to the problem, but due to the high cardinality, are used to fit the decision tree. This may also result in a potentially overfit decision tree.
For example, a decision tree being fit to a training set, of say 100 samples, may see a categorical attribute (say A) with 20 different categories as the best attribute for splitting a node since it would result in the most decrease in entropy (as can be seen intuitively). A, however, may not be the most significant attribute with respect to the problem. For instance, the problem may be predicting the weather for a region, and A may be the different cuisine preferences in the region. To mitigate this bias and consider attributes with different cardinalities more fairly, we can use another splitting criterion called **Gain Ratio** (GR. Gain Ratio adjusts Information Gain (IG) by weighing it against **Intrinsic Information** (II):

$$GR = \frac{IG}{II}$$

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

Intrinsic Information accounts for the distribution in the attribute locally. This weighs the Information Gain against the attribute cardinality and thus helps in preventing the tree from becoming highly biased towards high-cardinality attributes. Gain Ratio can also be sensitive to attribute cardinality but is better than Information Gain by a large margin.

2. (a) Decision Tree:



Consider:

- B = Play Badminton
- F = Play Football
- TT = Play Table Tennis
- $Pool$ = Play Pool
- R = It Rains
- N_F = Number of Friends Playing
- BR = Can Borrow TT Rackets

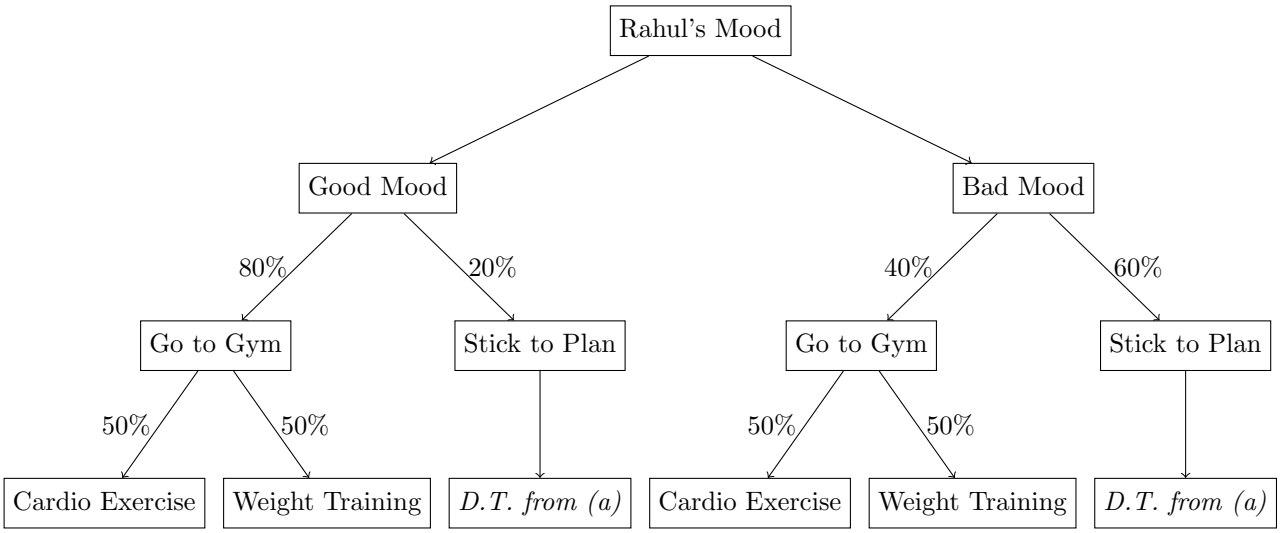
$$P(B) = P(N_F \leq 7|R).P(R)$$
$$P(F) = P(N_F > 7|R).P(R)$$
$$P(TT) = P(BR|(\neg R)).P(\neg R)$$
$$P(Pool) = P((\neg BR)|(\neg R)).P(\neg R)$$

(b) Consider:
R = Rainy,
$C = \neg R$ = Clear,
R_p = App predicts Rainy,
C_p = App predicts Clear

Given that:
$P(R_p|R) = 0.8$
$P(C_p|C) = 0.9$
$\Rightarrow P(R_p|C) = 0.1$
$P(R_p) = 0.3$
$P(C_p) = 0.7$
Since R and C are complementary, $P(C) = 1 - P(R)$

$P(R_p) = P(R_p|R)P(R) + P(R_p|C)P(C)$
$\Rightarrow 0.3 = 0.8 * P(R) + 0.1 * (1 - P(R))$
$\Rightarrow P(R) = \frac{0.2}{0.7} \approx 0.286$
$\Rightarrow P(R|R_p) = P(R_p|R)\frac{P(R)}{P(R_p)}$
$\Rightarrow P(R|R_p) = 0.8 * \frac{0.286}{0.3}$
$\Rightarrow P(R|R_p) \approx 0.761$

(c) Decision Tree:



Consider:

- M_G = Good Mood
- M_B = Bad Mood
- G_C = Cardiological Exercise at Gym
- G_W = Weight Training at Gym
- S = Stick to Plan

$$P(G_C) = 0.5 * 0.8 * P(M_G) + 0.5 * 0.4 * P(M_B)$$
$$P(G_W) = 0.5 * 0.8 * P(M_G) + 0.5 * 0.4 * P(M_B)$$
$$P(S) = 0.2 * P(M_G) + 0.6 * P(M_B)$$

(d) Given $P(F = 7|M_G) = 0.7$, $P(F = 7|M_B) = 0.45$, $P(M_G) = 0.6$, and $P(M_B) = 0.4$:

$\Rightarrow P(F = 7) = P(F = 7|M_G) * P(M_G) + P(F = 7|M_B) * P(M_B)$
$\Rightarrow P(F = 7) = 0.7 * 0.6 + 0.45 * 0.4 = 0.6$
$\Rightarrow P(M_G|F = 7) = P(F = 7|M_G)\frac{P(M_G)}{P(F=7)}$ and $P(M_B|F = 7) = P(F = 7|M_B)\frac{P(M_B)}{P(F=7)}$

$\Rightarrow P(M_G|F=7) = 0.7 * \frac{0.6}{0.6} = 0.7$ and $P(M_B|F=7) = 0.45 * \frac{0.4}{0.6} = 0.3$

$\Rightarrow P(G_C) = 0.5 * 0.8 * P(M_G|F=7) + 0.5 * 0.4 * P(M_B|F=7)$
$\Rightarrow P(G_W) = 0.5 * 0.8 * P(M_G|F=7) + 0.5 * 0.4 * P(M_B|F=7)$
$\Rightarrow P(S) = 0.2 * P(M_G|F=7) + 0.6 * P(M_B|F=7)$

$\Rightarrow P(G_C) = 0.5 * 0.8 * 0.7 + 0.5 * 0.4 * 0.3 = 0.34$
$\Rightarrow P(G_W) = 0.5 * 0.8 * 0.7 + 0.5 * 0.4 * 0.3 = 0.34$
$\Rightarrow P(S) = 0.2 * 0.7 + 0.6 * 0.3 = 0.32$

$\Rightarrow$ Since the probability at the *Stick to Plan* node is 0.32, the probability of all the outcomes possible by splitting that node must be ≤ 0.32 (since 0.32 will be multiplied by some numbers between 0 and 1). Therefore The most likely outcomes are G_C and G_W.

Section B

Data Pre-processing

- The features *ca* and *thal* had some rows with entries as *?*. These were replaced by the mean for the *ca* feature (since it is a regressive feature) and the mode for the *thal* feature (since it is a categorical feature).
- The data in the target column i.e. *label* was converted to a binary-class system where all labels > 0 were considered as class 1 and label 0 was considered as class 0.
- All regressive variables were standardized.
- All categorical variables were one-hot encoded for better performance.

Exploratory Data Analysis

- The respective figures (refer figures 1-5) describe the distribution of the regressive variables in the processed dataset.
- There are no missing values in any of the attributes in the processed dataset.
- The correlation heatmap (refer figure 6) clearly shows that there is not significant correlation between any of the attributes in the processed dataset.

Basic Decision Tree

- A decision tree was trained for the splitting criterion of `gini` and `entropy` each.
- The performance for `entropy` was better than the `gini` criterion:
  Using `gini` as the splitting criterion:
  Accuracy: **0.6721311475409836**
  Sensitivity: 0.5
  Specificity: 0.8620689655172413

  Using `entropy` as the splitting criterion:
  Accuracy: **0.6885245901639344**
  Sensitivity: 0.5625
  Specificity: 0.8275862068965517

Grid Search

- A grid search for the hyperparameters `max_depth` and `min_samples_split` yielded the results:
  Best values for hyperparameters obtained from `GridSearch`:
  `max_features`: **sqrt**
  `min_samples_split`: **22**
- Using the best hyperparameters obtained from the `GridSearch` for Decision Tree:
  Accuracy: **0.8032786885245902**
  Sensitivity: 0.71875
  Specificity: 0.896551724137931

Random Forest

- A `RandomForestClassifier` was trained for the given dataset.
- The hyperparameters for the classifier were chosen by a grid search.
- The results from the grid search are as follows:
  Best values for hyperparameters obtained from Grid Search:
  `n_estimators`: **30**
  `max_depth`: **None**
  `min_samples_split`: **10**

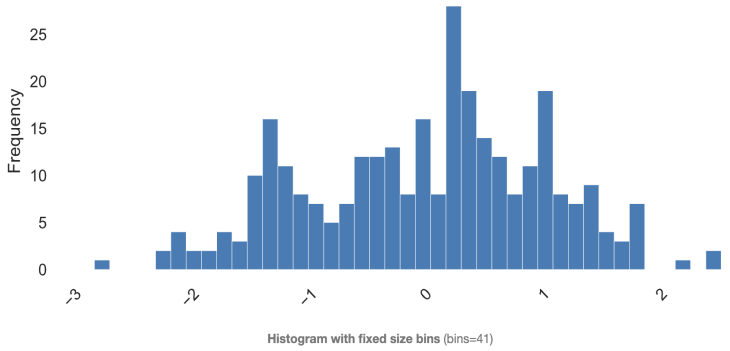


Figure 1: **age**

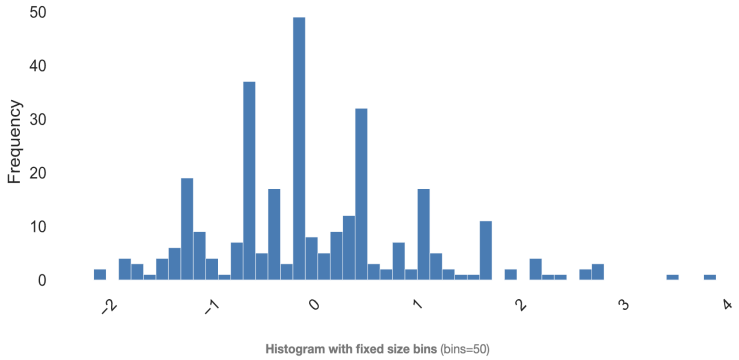


Figure 2: **trestbps**

- The results from the Random Forest are:
  Using the best hyperparameters obtained from the `GridSearch` for Random Forest:
  Accuracy: **0.7704918032786885**
  Sensitivity: 0.6875
  Specificity: 0.8620689655172413

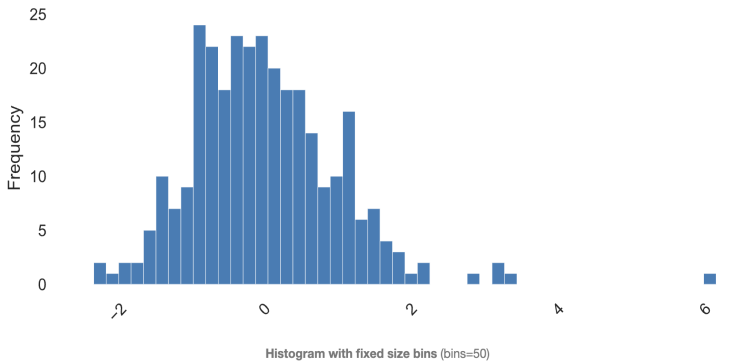


Figure 3: **chol**

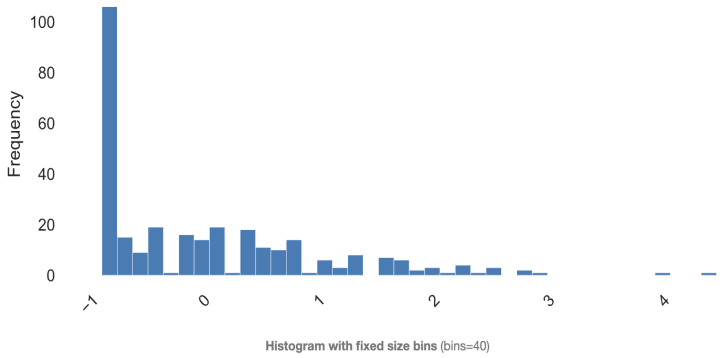


Figure 4: **oldpeak**

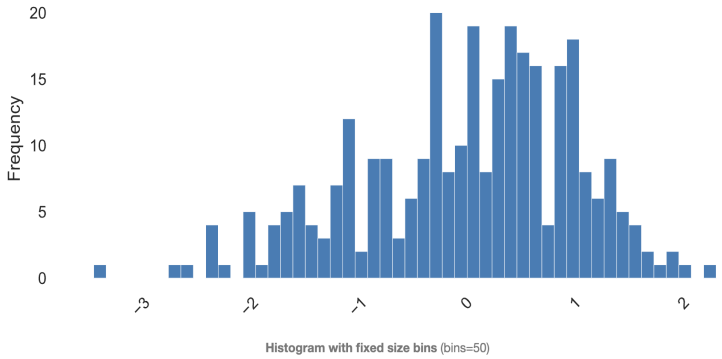


Figure 5: **thalach**

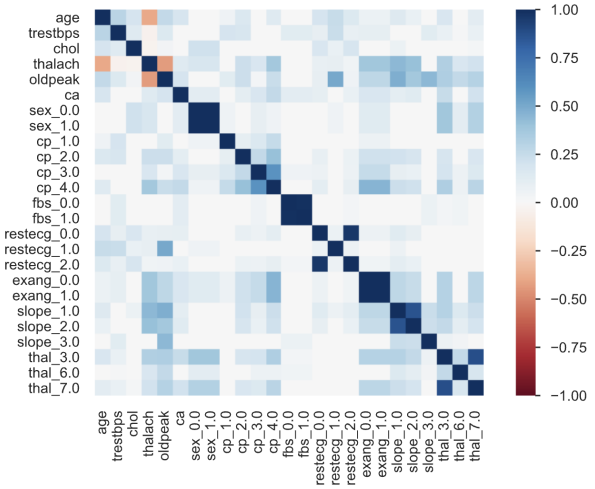


Figure 6: **Correlaton Heatmap**

Section C

Data Pre-processing

- The rows with the value of the attribute $sex = 0$ were dropped since they cannot be determined by mode. The column *TBG* was dropped since it only has one value throughout the dataset.
- The columns that indicate whether certain regressive variables were measured were dropped and the values for the data points that did not measure the respective regressive variable were set to 0.
- All regressive variables were standardized.
- All categorical variables were one-hot encoded for better performance.
- The columns that have boolean variables were changed with f values changed to 0, and t values changed to 1.

Class `MyDecisionTree`

- A class `MyDecisionTree` was designed that implements a decision tree with the following functions:
  - `__init__()`: Initialise the decision tree and set the attribute `max_depth` to the value passed to this instance.
  - `fit(X,y)`: Trains the decision tree to the dataset `X` with labels `y` using a recursive approach with the internal function `_build_tree`
  - `make_split(X,y)`: Make the best split for the dataset `X` and labels `y` using the internal function `_split`
  - `cost_function(left,right)`: Evaluates the impurity of a node for the given split (`left`,`right`) using `gini` as the splitting criterion
  - `predict(X)`: Predicts the output of the decision tree for the given set of test samples `X`
  - `score(X,y)`: Evaluates the performance of the model on the test set of samples `X` with labels `y`; computes the *Accuracy*, *Sensitivity* and *Specificity*
- The results for an instance of the `MyDecisionTree`, trained on the given dataset for an 80 : 20 split, are as follows:
  Accuracy: **0.9907063197026022**
  Sensitivity: **0.7857142857142857**
  Specificity: **0.9961832061068702**