ITCS453 Data Warehousing and Data Mining (2/2022)

Project 2

1. Project group number**:** 2

2. Member names

- 6388016 Thanawath Huayhongthong
- 6388020 Chalumphu Atjarit
- 6388022 Pattanan Korkiattrakool
- 6388059 Naphat Sookjitsumran
- 6388094 Vipavee Ngamyingsurat

3. Dataset name

- Cure the princess

4. Dataset description and analysis

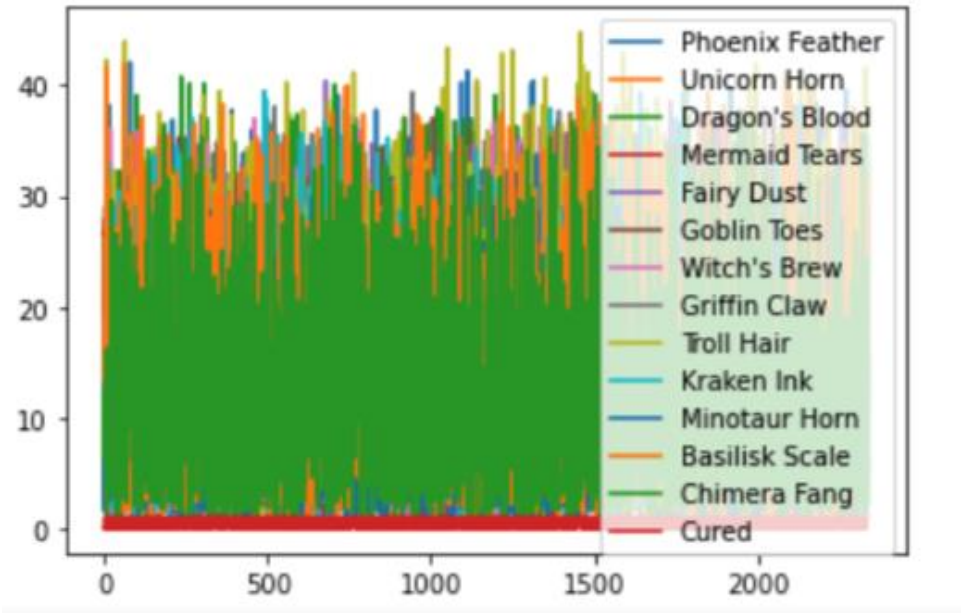
a) Description

Firstly, we know that the dataset consists of information about different combinations of ingredients that have been tried by the villagers to cure various ailments, including the princess. The dataset would likely include the ingredients used in each combination, the quantities used, and the outcome (whether or not the combination was successful in curing the ailment).

Additionally, we know that the ingredients are scarce and volatile, and there may be constraints on the availability of certain ingredients due to droughts. This implies that there may be limitations on the number of combinations that can be tried, and the dataset may be relatively small.

The alchemist would need to use the available data to identify the optimal combination of ingredients that would be most effective in curing the princess. The use of machine learning algorithms would help the alchemist to navigate the complex and potentially nonlinear relationships between the different ingredients and their effects on the ailment.

b) Statistics and graphical analyses



Number of instances: 2338

Attribute: 14 (Feature: 13, Target: 1)

Classes: 2 (0 False, 1 True)

Missing Data: 0

Mean of each attribute:

```
Phoenix Feather    15.365697
Unicorn Horn       10.946749
Dragon's Blood     16.115654
Mermaid Tears      13.627973
Fairy Dust         15.069504
Goblin Toes        14.157271
Witch's Brew       12.328914
Griffin Claw       14.911206
Troll Hair         16.871685
Kraken Ink         14.890590
Minotaur Horn      10.916125
Basilisk Scale     15.371600
Chimera Fang       12.084003
Cured               0.496578
dtype: float64
```

Min of each attribute:

```
Phoenix Feather    1.0
Unicorn Horn       1.0
Dragon's Blood     1.0
Mermaid Tears      1.0
Fairy Dust         1.0
Goblin Toes        1.0
Witch's Brew       1.0
Griffin Claw       1.0
Troll Hair         1.0
Kraken Ink         1.0
Minotaur Horn      1.0
Basilisk Scale     1.0
Chimera Fang       1.0
Cured              0.0
dtype: float64
```

Max of each attribute:

```
Phoenix Feather    42.1
Unicorn Horn       34.1
Dragon's Blood     40.8
Mermaid Tears      35.8
Fairy Dust         40.4
Goblin Toes        37.8
Witch's Brew       37.3
Griffin Claw       39.4
Troll Hair         44.8
Kraken Ink         39.5
Minotaur Horn      33.7
Basilisk Scale     42.0
Chimera Fang       37.8
Cured               1.0
dtype: float64
```

5. Experiment

a) Chosen algorithms

kNN Algorithm

- k- Nearest

  The k-Nearest Neighbors (kNN) algorithm is a prominent non-parametric supervised machine learning classifier technique that can be used for classification and regression problems. The algorithm works by finding the k closest training instances (neighbors) in the feature space to a given test sample and predicting based on their labels or values. There are two main advantages to the kNN algorithm: first, its simplicity and ease of implementation. The kNN algorithm does not make any assumptions about the underlying distribution of the data, which makes it suitable for datasets with complex and nonlinear relationships between variables. Additionally, the kNN can handle both classification and regression tasks and can be used with a variety of distance metrics to measure the similarity between data points. Another advantage of kNN is that it is a lazy learning algorithm, meaning that it does not require a training phase and can quickly adapt to new data.

- **Why kNN is the chosen algorithm**

  Our group has chosen the kNN algorithm to process the ‘cure the princess’ dataset because the kNN algorithm is capable of handling multi-class classification problems. The output of kNN is probabilistic, which enables us to make predictions with high accuracy. Furthermore, the kNN algorithm is a non-parametric model, meaning it can handle complex decision boundaries, making kNN more flexible to use with the princess dataset because of this dataset has a lot of attribute which are 14 thing to create a medicine to cure the princess and this algorithm can handle the complex of the information.

Decision Tree Algorithm

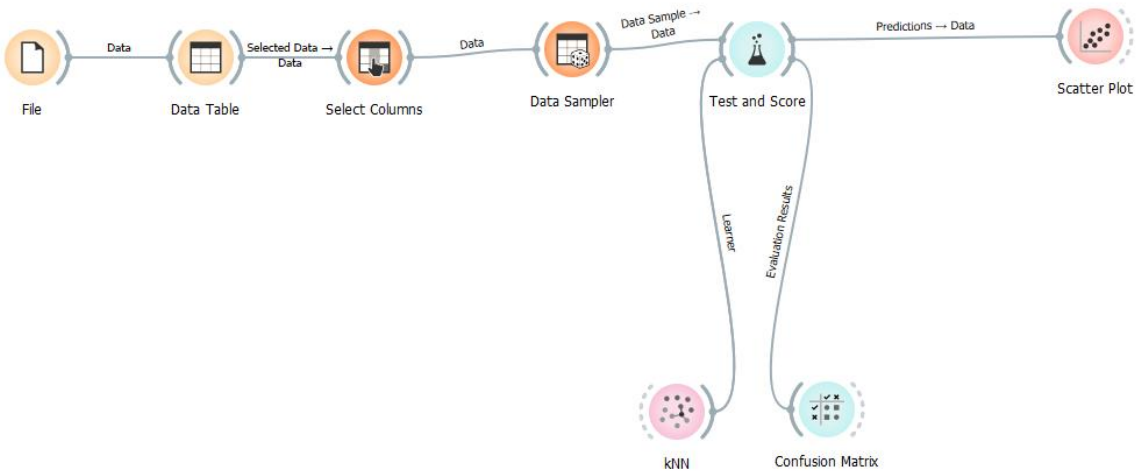
- A decision tree is a machine learning algorithm commonly used for classification and regression tasks. Its tree-like structure consists of nodes that have indicated features, attributes, and branches that represent possible outcomes. The algorithm is processed by recursively splitting the data based on the most significant features at each node until a stopping criterion is reached. The results in a tree can be used to make predictions by traversing from the root to the leaf node that corresponds to the input data point. One major advantage of decision trees is their interpretability and visualization, which can provide insights into how the algorithm makes decisions. Decision trees can handle both categorical and numerical data, as well as missing values with no requiring imputation. Additionally, decision trees can capture interactions between variables, making them useful for complex datasets. Another benefit is that decision trees are relatively fast and can handle large datasets. Therefore, decision trees can be a good choice for the "cure the princess" dataset, as they can efficiently identify important features and help to understand the algorithm's decision-making process.

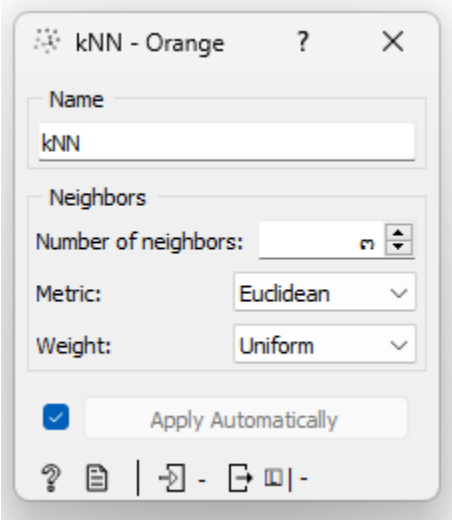
- **Why Decision Tree is the chosen algorithm**

  Since our dataset is large and has diverse attributes, it may take a long time to clean the dataset to assure that it is usable. With a lot of data, there is a chance of missing values and outliers. Therefore, choosing the decision tree model is considered appropriate for our dataset because the decision tree model is less affected by the presence of missing values and outliers. Another reason for choosing the decision tree model is that it can mix different types of data, which can predict the value of each data, and also that the model can reduce the time to clean data.

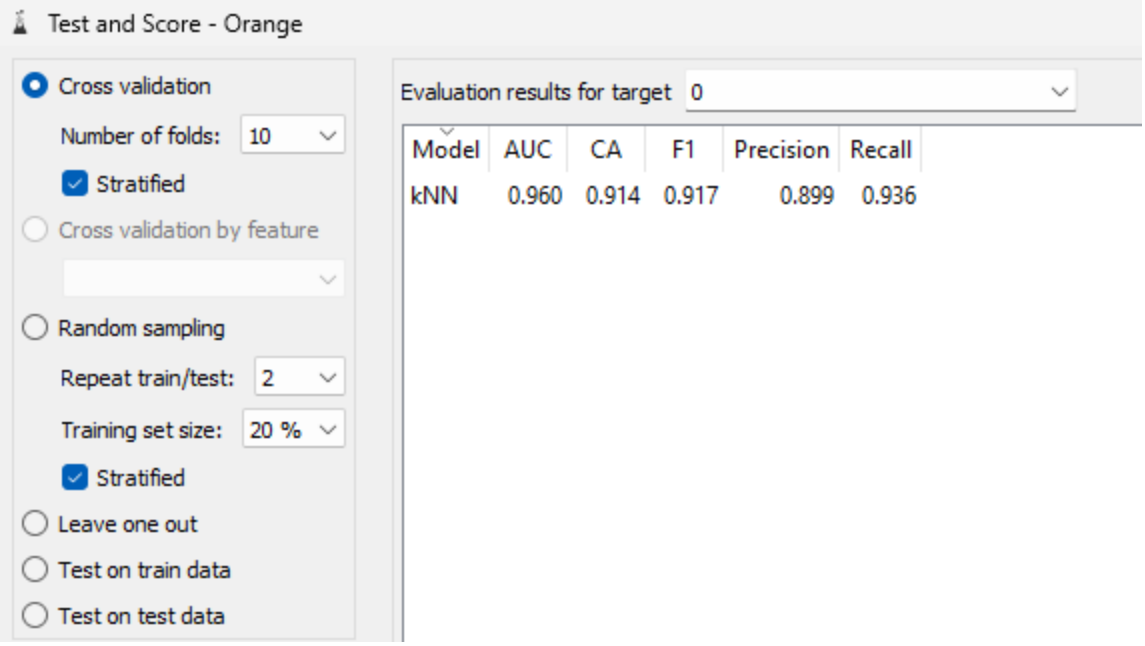
b) Experiment description (including screenshot(s) of Orange pipeline)

- **kNN**





kNN: for the kNN model we choose the number of neighbor equal to 3 to set the kNN model to look at the three closest neighbors to a given data point and use their class label for classification or target value for regression to make a prediction



Test and Score

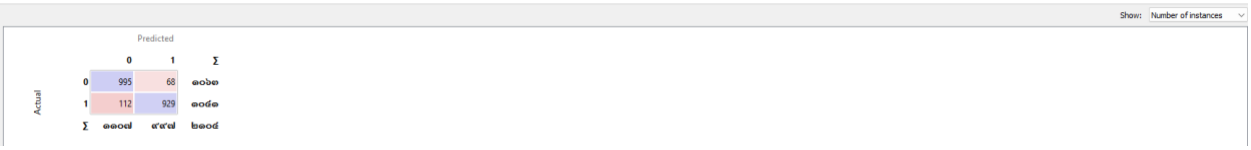
AUC: This stands for Area Under the Curve and is a common metric used to evaluate the performance of a binary classification model for the AUC value is equal to the 1 which is 0.96 can indicate that the model has excellent discriminatory power and this is able to correctly distinguish between positive and negative cases with a high degree of accuracy.

CA: This stands for Classification Accuracy and is another common metric used to evaluate the performance of a classification model it represents the true positive and true negative out of all instance in the dataset, In our results a CA score is 0.914 can indicate that model has a high overall accuracy in classifying the data.

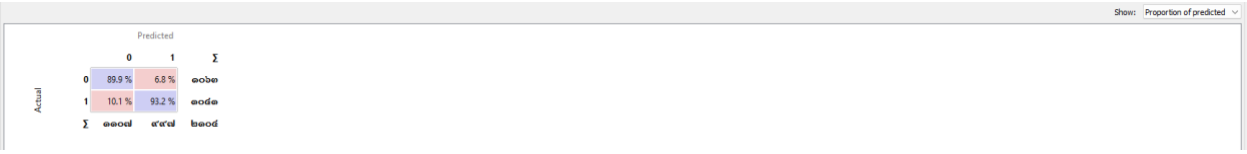
F1: This is a measure of the balance between precision and recall in a binary classification model. It ranges from 0 to 1, with a score of 1 indicating perfect balance between precision and recall. A score of 0.917 indicates that the model achieves a good balance between precision and recall.

Precision: This is the proportion of true positive predictions (correctly predicted positive cases) out of all positive predictions made by the model. In this case, a precision score of 0.899 indicates that the model is precise in predicting positive cases.

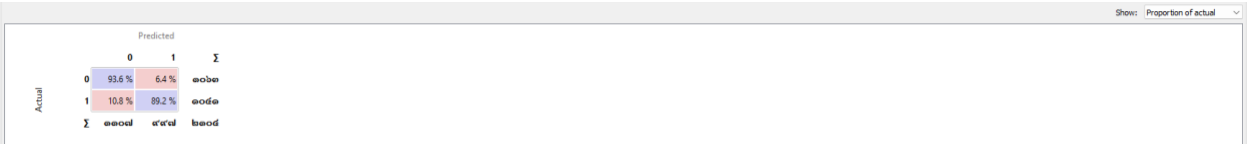
Recall: This is the proportion of true positive predictions out of all actual positive cases in the dataset. In this case, a recall score of 0.936 indicates that the model is able to capture a high proportion of positive cases in the dataset.



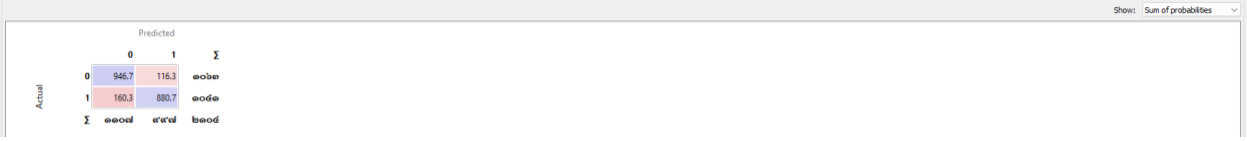
Number of instances



Proportion of predicted



Proportion of actual



Sum of probabilities

A confusion matrix provides a summary of the actual and predicted class labels for a set of instances, breaking down the predictions into four categories: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These values can be used to calculate various performance metrics, such as accuracy, precision, recall, F1 score, and others, that can help us evaluate the model's performance and identify areas where it can be improved.

AUC: Calculate by plotting the true positive rate

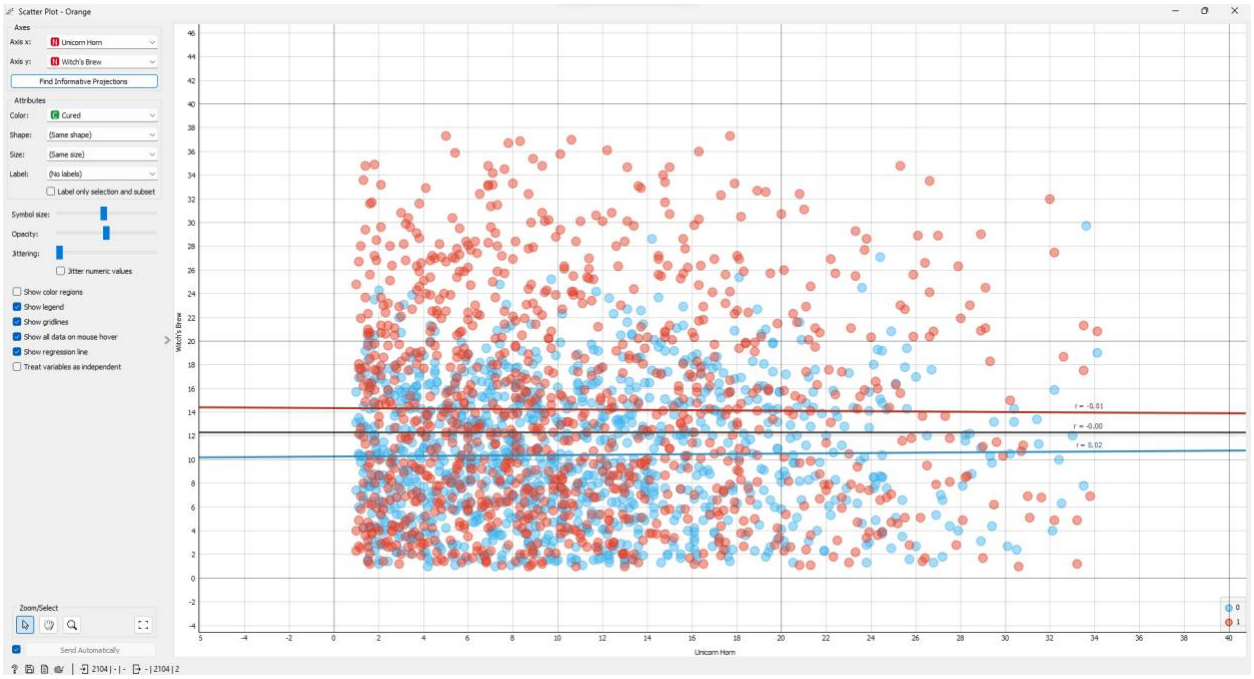
CA: It calculated the number of instance by (TP+TN)/(TP+TN+FP+FN)

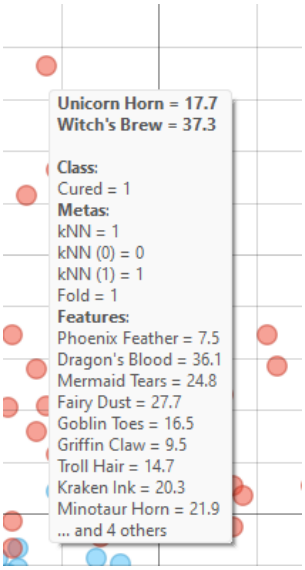
F1: It calculated by 2* ((precision * recall) / (precision+recall))

Precision: This measure by the proportion of instance that were predicted as positive that actually belong to the positive class by TP/(TP+FP)

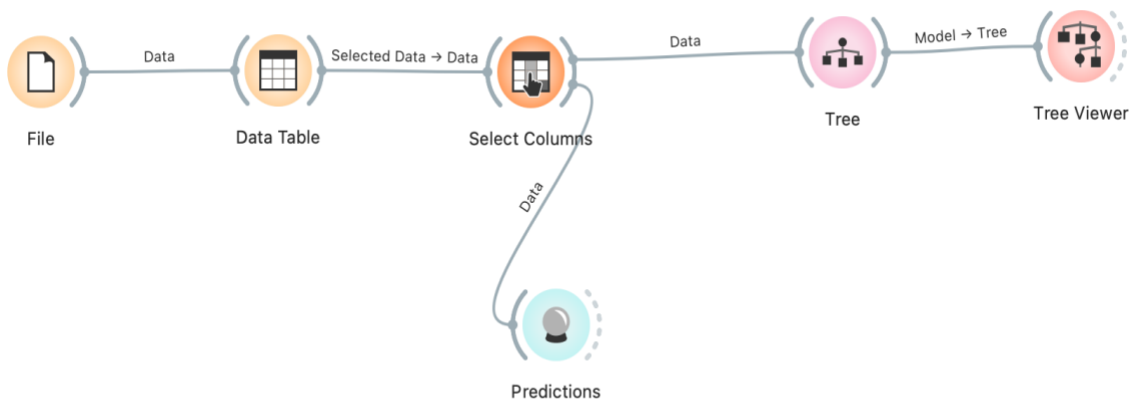
Recall: This measures the proportion of positive instances that were correctly classified as positive. It is calculated as TP / (TP + FN)

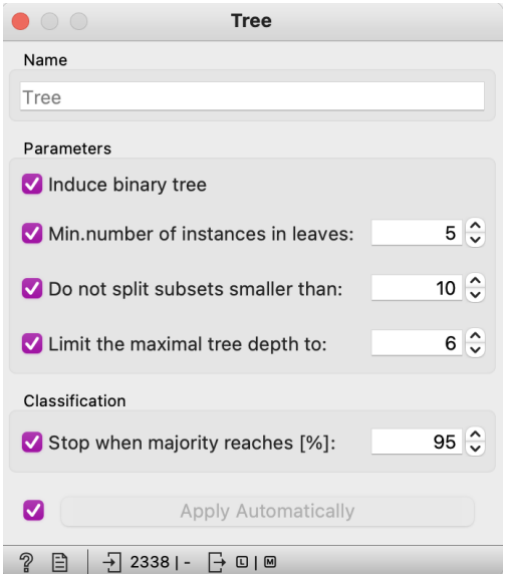
In the end of the result if you see the results in scatter plot that can prediction about the attribute that using to create the medicine to princess, for example if we set the Axis x is Unicorn Horn and Axis Y to be the Witch’s brew the red dot refer to this attribute can cured the princess in the 17.7 unicorn horn ratio and 37.3 witch's brew ratio.





- **Decision Tree**





Decision Tree:

Induce binary tree: build a binary tree (split into two child nodes)

Min. number of instances in leaves: if checked, the algorithm will never construct a split which would put less than the specified number of training examples into any of the branches.

Do not split subsets smaller than: forbids the algorithm to split the nodes with less than the given number of instances.

Limit the maximal tree depth: limits the depth of the classification tree to the specified number of node levels.

Stop when majority reaches [%]: stop splitting the nodes after a specified majority threshold is reached.

For the decision tree model, we have set values of parameters in tree model:

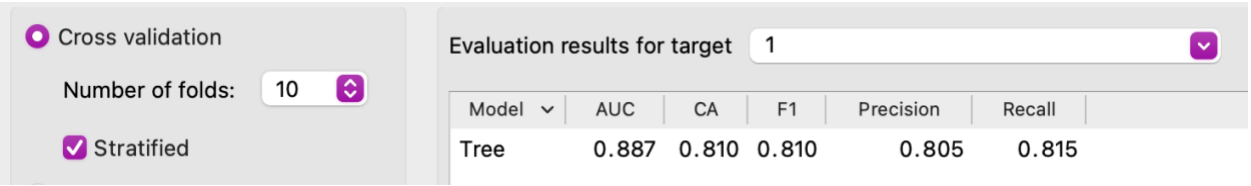
- Induce binary tree: Yes
- Min. number of instances in leaves: 5
- Do not split subsets smaller than: 10
- Limit the maximal tree depth: 6
- Stop when majority reaches [%]: 95

This model can be summarized as having at least 5 instances in the leaf, at least 10 instances in the internal nodes. and at depths up to 6, the tree stops splitting when the majority of nodes reaches 95% in classification.

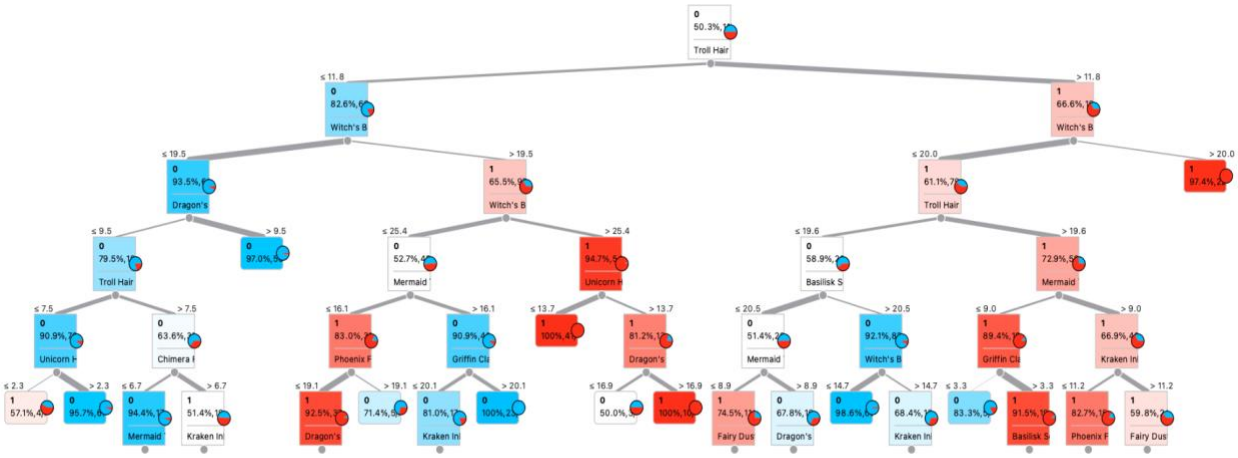
Data in Tree:

- Data instances: 2338
- Features: Phoenix Feather, Unicorn Horn, Dragon's Blood, Mermaid Tears, Fairy Dust, Goblin Toes, Witch's Brew, Griffin Claw, Troll Hair, Kraken Ink, Minotaur Horn, Basilisk Scale, Chimera Fang (total: 13 features) [Cured: 0(not cured), 1(cured)]
- Target: Cured

Test and Score:



Tree viewer:



According to the Tree result, it has shown that the parent node which is ‘Troll hair’ has not much difference in positive and negative outcome values. In the ‘Troll hair’ node, It turned out to be negative at 50.3% (1177 instances) while positive at 49.7% (1161 instances) with the distribution of ‘cured’. The parent node will continue to be split into 'witch's brew' node which can be seen that if ‘Troll hair’ and ‘Witch’s brew’ are used in the amount of ‘Troll hair’ with a dose less than or equal to 11.8, the result is 0, that is, it is incurable, yet if ‘Troll hair’ is used in doses greater than 11.8, the result will be 1, meaning that the treatment is successful (cured). Thus, from the tree model it can be concluded that the deeper the layers of the tree model, the more detailed the data.

c) Discussion of performance of learned classifiers

After we use the kNN model to classify the dataset the kNN algorithm can compute the most accurate model with high accuracy predictions which attribute can create the medicine and can cure the princess like the example that we showed above. On the other hand, Tree decision model's algorithm shows that the results are highly accurate as well, but the results of the data are not as detailed as kNN because the tree model is broken down into layers where the detail and completeness of the data is dependent on the greater number of tree layers to be displayed greater detail of the data. In conclusion, for the 'cure the princess' dataset, kNN may be a more suitable algorithmic model since kNN is more readable and it shows the ratio of data more clearly than Decision tree algorithm.

d) Comparison of performance using t-test

Paired *t* test results

P value and statistical significance:

The two-tailed P value equals 0.0002

By conventional criteria, this difference is considered to be extremely statistically significant.

Confidence interval:

The mean of kNN minus decisionTree equals 0.09980

95% confidence interval of this difference: From 0.07767 to 0.12193

Intermediate values used in calculations:

t = 12.5201

df = 4

standard error of difference = 0.008

Review your data:

Group	kNN	decisionTree
Mean	0.92520	0.82540
SD	0.02349	0.03462
SEM	0.01050	0.01548
N	5	5

After, comparing the t-test score is showing the kNN algorithm has the mean score more than decision tree model with a 0.92520 means the kNN algorithm can work well to compute the ratio to creating medicine to cure the princess.