**Section A:**

1. Decision trees for classification

2. Decision trees for classification are also simply know as classification trees. At a very general level, the classification tree algorithm seeks to split data sets into homogenous groups, ultimately identifying and creating groups that have similar variables to one another. It does so by analyzing the data set and using the analysis to create a set of rules/ questions, breaking down the dataset into smaller and smaller subsets. The creation of these rules is not governed by a human being and are determined by the algorithm itself. The variables which go into the classification can be numerical or categorical, although the output of the classification tree will always be a discrete category (class).

The decision tree can be thought of as a tree structure which has been flipped upside down, with the root being the starting point of the algorithm. At each node, the algorithm breaks the data set into smaller subsets which produce the most information about a class. It is important to note that each node is usually split into only 2 additional nodes, with each resulting node continuing to split until it reaches it’s given stopping criterion. The leaf node is the very last node at the end of a branch, which does not contain any nodes. A branch node is a node which “branches out” into additional nodes, which can either be more branch or leaf nodes. Most classification trees would often be overfitted to the dataset it was trained on, rendering it less effective in classifying other datasets accurately. This is where the tree is pruned, a process which involves turning branch nodes into leaf nodes or removing leaf nodes from the original branch. Pruning continues until the smallest tree that fits the data is obtained.

Given that the decision tree for classification produces a discrete category, it can be used to help interpret data. Use cases range from predicting the candidate someone might vote for in a presidential election, working as a decision framework when screening job candidates, or understanding consumer behavior on websites. A decision tree classifier is a two-steps process: Learning step which consist of training the model based on training data set, and prediction step which will consist of testing the model with test data. A decision tree can handle high dimensional data with good accuracy. Decision tree algorithm divides the tree in recursive way. (Datacamp.com)

To perform any decision tree algorithm, we must follow three steps (Datacamp.com):

* Select the best attribute using Attribute Selection Measures (ASM)
* Take that attribute as a decision tree node and divides rest of the dataset into smaller subsets.
* Starts building the tree by repeating recursively this process until there are no more attribute, or all the tuples belong to the same attribute value.

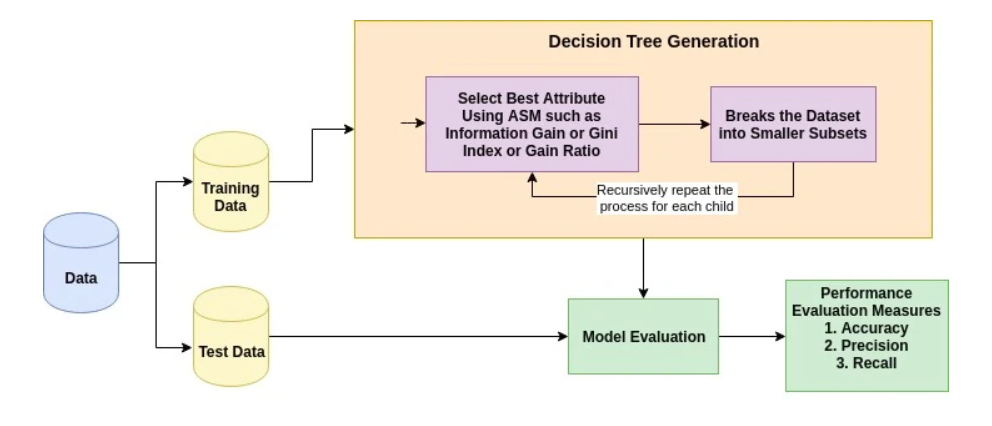


Figure 1: Decision Tree Generation

3. The following formulae help create the decision tree

This is the equation to calculate entropy, the degree of randomness of elements, which can be interpreted as a measure of impurity. It is used to determine the homogeneity of a sample, with E = 0 meaning that the sample is completely homogenous, while E = 1 means that the sample is equally divided. With the given number of attributes/variables in the data set, the data set is initially split on each attribute, with the entropy for each branch being calculated. The entropy for each branch is proportionally added to get the total entropy for the split. The entropy is subtracted from the entropy before the split, resulting in decreased entropy.

The formula above is known as the Gini Impurity formula. It is a measure of impurity at the node level of the decision tree. It measures impurity by calculating the probability that two different categories will be created after a node is split. The creation of two identical categories indicates the complete purity/ homogeneity of the category, provided by .The measure of the probability of measuring Gini impurity is thus 1 minus the sum of creating identical categorical nodes, giving us the probability of obtaining two different outputs, for every possible category from j to k.

4. (a) The algorithm can be applied to categorical data and numerical data with less than 100 categories

(b) As decision trees are non-linear algorithms, they require a lot more data. Given that decision trees can grow significantly by depth and size, it is recommended that the amount of data be in the thousands. The decision tree classifier does not well suit for small data set as It will lead to overfitting and high variance between the actual and predicted results.

(c) Overfitting problem. Not easy to determine how deep to grow the tree.  
  
5. (a) If the data has not been cleaned well, the decision tree classification algorithm becomes very sensitive to noisy data, resulting in the creation of new branch or leaf nodes. These nodes will obviously be based off erroneous data, creating an entire tree of false/meaningless data. It may also lead to boundaries of classes/categories overlapping, both lead to inaccurate classifications of data. Decision trees algorithms also suffer when not provided enough data points. Decision trees formed on small sets of data will suffer from high variance in the results (target function) when changing training set data. This is because the target function, a function which knows and maps full relationships between the input variables/features to the output variable/response, is not trained to accommodate data sets with far more datapoints and input variables.

(b) One potential limitation, which is quite common, is the likelihood of overfitting the algorithm to the training data. This occurs when the algorithm develops hypotheses to reduce training set error which often leads to an increase in test set error. This is because the algorithm captures the noise and outliers in the data, tailoring the algorithm to only work well with the provided data set and not others. Decision trees are also vulnerable to bias in classes which make up a substantial portion of the dataset. These classes tend to be selected as nodes within the decision tree, resulting in a tree that classifies perfectly for data with these variables but poorly with less common variables. Training the algorithm is also computationally extremely intensive, requiring a large amount of time and processing power to run these algorithms. This is because experts must determine the different possible branches to determine the best split of a nod and, select the optimal weights to prune algorithms.

6. To run a decision tree classifier, we must import the libraries below.  
import pandas as pd  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
from sklearn.tree import export\_graphviz  
from sklearn.externals.six import StringIO  
from IPython.display import Image   
import pydotplus  
We also need to split the data set into two sets: A training set, and test set, And then create a decision tree classifier object decisionTree =DecisionTreeClassifier(). We train the model by calling the method fit on the object by given the appropriates arguments. We predict the response for the test data by calling the method predict. We can also measure the accuracy of the decision tree by calling the methods accuracy score by comparing the predicted values and the tested ones.

7. To determine how well a test condition performs, we need to compare the degree of impurity of the parent before splitting with degree of the impurity of the child nodes after splitting. The larger their difference, the better the test condition. To evaluate the performance of model, we can use the function accuracy\_score(), which will return the accuracy of the model as a percentage. It is the ratio of the correctly predicted datapoints to all the predicted datapoints.

8. The data was retrieved contains information about the passengers of the Titanic, containing variables that could potentially help us predict what types of passengers were most likely to survive the tragic accident. It contains a mix of 10 categorical and numerical variables, all of which were obtained from the website: https://www.kaggle.com/c/titanic

**Section B:**

|  |  |  |
| --- | --- | --- |
| 1. | https://towardsdatascience.com/a-beginners-guide-to-decision-tree-classification-6d3209353ea | Information on classification tree algorithm |
| 2. | <https://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf> | More information about decision trees for classification |
| 3. | https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248 | Information of how the decision tree splits data |
| 4. | https://machinelearningmastery.com/much-training-data-required-machine-learning/ | How much data is needed |
| 5. | https://sci2s.ugr.es/noisydata#Introduction%20to%20Noise%20in%20Data%20Mining | Issues with algorithm |
| 6. | https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956 | Pros and cons of decision trees |
| 7. | https://www.saedsayad.com/decision\_tree\_overfitting.htm | Effects of overfitting |
| 8. | https://www.saedsayad.com/decision\_tree\_super.htm | Dangers of biased classes |
| 9. | https://www.brighthubpm.com/project-planning/106005-disadvantages-to-using-decision-trees/ | Disadvantages of decision trees |
| 10. | https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/ | Information about bias and variance |
| 11. | https://www.datacamp.com/community/tutorials/decision-tree-classification-python | Coding decision tree in python and info on formulas |
| 12. | https://pixorblog.wordpress.com/2017/11/06/interpretation-of-the-gini-impurity/ | Information on Gini Impurity |
| 13. | http://dataaspirant.com/2017/02/01/decision-tree-algorithm-python-with-scikit-learn/ | How to create a decision tree in python |