**Project 3: Classification Algorithms**

**CSE 601: Data Mining and Bioinformatics**

**Team members:**

Arvind Thirumurugan athirumu 50289656

Saketh Varma Pericherla sakethva 50288206

Vijay Jagannathan vijayjag 50290947

**Part 1: Implementing Classification Algorithms:**

**1.0 Results:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dataset** | **Measures** | **Algorithms** | | | |
| **Accuracy** | **Precision** | **Recall** | **F1** |
| **1** | **K-nn**  **(K = 5)** | 0.914285 | 0.834282 | 0.976476 | 0.893758 |
| **Naïve Bayes** |  |  |  |  |
| **Decision Tree** | 0.917418 | 0.884163 | 0.901467 | 0.889868 |
| **Random Forest** |  |  |  |  |
| **2** | **K-nn** | 0.678260 | 0.533307 | 0.493544 | 0.503983 |
| **Naïve Bayes** |  |  |  |  |
| **Decision Tree** | 0.599814 | 0.424779 | 0.467053 | 0.441297 |
| **Random Forest** |  |  |  |  |
| **3\_train** | **K-nn** |  |  |  |  |
| **Naïve Bayes** |  |  |  |  |
| **Decision Tree** | 0.8875 | 0.845714 | 0.787380 | 0.810158 |
| **Random Forest** | 1.0 | 0.7 | 0.7 | 0.7 |
| **3\_train** | **K-nn** |  |  |  |  |
| **Naïve Bayes** |  |  |  |  |
| **Decision Tree** | 1.0 | 1.0 | 1.0 | 1.0 |
| **Random Forest** | 1.0 | 1.0 | 1.0 | 1.0 |
| **4** | **K-nn** |  |  |  |  |
| **Naïve Bayes** |  |  |  |  |
| **Decision Tree** | 1.0 | 1.0 | 1.0 | 1.0 |
| **Random Forest** | 1.0 | 1.0 | 1.0 | 1.0 |

The above table shows the results obtained by our implementation of K-nearest neighbours, Decision Tree, Naïve Bayes and Random forest algorithms for the four datasets provided.

* Datasets 1 and 2 are trained using 10-fold cross validation while dataset 3 is trained with a separate training set and tested on a separate testing set provided. Dataset 4 is trained and tested on the same dataset due to the lack of sufficient data and a separate testing set.
* The various hyper-parameters chosen for these different algorithms to get these final results and further analysis on the results are discussed in the respective sections below.
  1. **K-Nearest Neighbors:**
  2. **Naïve Bayes:**
  3. **Decision Tree:**

A decision tree makes sequential, hierarchical decision about the outcomes variable based on the predictor data. Also known as CART (Classification and Regression Tree), a decision tree model is represented as a binary tree.

**1.3.0 Algorithm Description:**

**Building a tree:**

* Before we build a tree we need to know how to create a node and make a binary split.
* There are many possible trees that can be derived for a given dataset. To choose a tree that provides the best split we follow the greedy strategy i.e. Split the records based on an attribute test that optimizes certain criterion.
* To implement the greedy strategy, we will calculate GINI index at each node which indicates how pure the nodes are, where node purity refers to how mixed the training data assigned to each node is. The GINI index is calculated as:



* Further, Information GAIN measures reduction in impurity achieved because of the split. This measure helps us choose the split that achieves most reduction
* If GAIN is zero, make the node a leaf node – this node contains an output variable which is used to make a final prediction.
* If the GAIN is non-zero, then make two branches left and right that form the decision nodes – these nodes can further be split into left and right nodes based on their respective optimal GAIN calculations.



**Making a prediction:**

* To make a prediction with the built decision tree we navigate the tree with the given row of data.
* We implement a recursive function, where the same prediction function is called again with left or right child nodes, depending on the condition at which the tree splits the data.
* If the type of node is “decision node” we continue stepping through the tree based on the satisfied condition until we reach a leaf node.
* If the type of node is “leaf node” we return the prediction provided by the leaf node. This is the terminating condition of the prediction.



* + 1. **Parameters:**

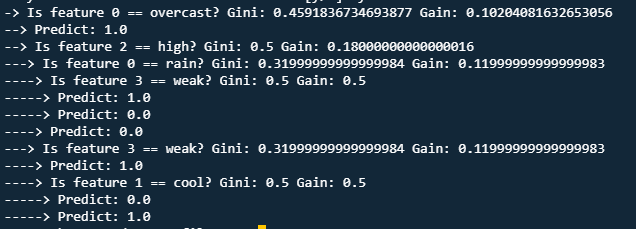
**Continuous & Categorical features:**

****

As shown in the above snippet we detect if the feature is a continuous variable or a categorical variable using python 3’s “**isinstance(value, datatype)”** function. If the column has continuous data, our condition to split the tree is “>=” whereas if the column has categorical data our condition to split the tree is “==”

**Best features:**

We use GINI and Gain calculations to find out the best split at a given node. The below tree shows the tree that we obtained for dataset 4:



**Stop criteria:**

There are two scenarios in which the algorithm stops building the tree. They are:

- If the information gain at a node is found to be zero, we cannot split it further, so we make it a leaf node and once there are no decision nodes to be filled the tree is said to be completely built.

- We provide maximum depth as an optional parameter. If specified, once the tree reaches the maximum depth specified we terminate the process of building the tree. We also provide an optional parameter called “n\_features” which restricts the tree to utilize only “n” number of features specified in order to build the tree. These parameters are used judiciously to avoid overfitting the data.

**1.3.2 Analysis:**

Accuracy analysis here

Cross validation here

**Pros:**

* Decision tree takes a non-parametric approach to do classification i.e. it doesn’t need to know the probability distributions of various classes and other properties of the dataset.
* They are extremely fast at doing predictions at O(logn) time complexity where n is the maximum depth of the tree.
* Even though the idea of decision trees is trivial, they provide great accuracy that can be comparable with any modern algorithm.
* Decision trees are intuitive and easy to interpret and which makes it relatively easier to analyse and optimize for the given dataset.

**Cons:**

* Overfitting is a huge concern for decision trees since a decision tree is a rule based classifier. If parameters like maximum depth, number of features etc. are not provided, the tree tries to fit in all the data perfectly and ends up performing poorly on test dataset.
* Decision trees draw rectangular decision boundaries which is not optimal for datasets having more complex decision boundaries.
* Without pruning, decision tree can generate leaf nodes that do not have sufficient number of records to make the right prediction with confidence.
* Decision trees can produce trees that can be difficult to interpret unless we provide parameters like maximum depth, reduced number of features, minimum count at leaf node etc.
  1. **Random Forest:**

Random forest is an ensemble learning algorithm that constructs a multitude of decision trees at training time and outputs the mode of the classes predicted by these trees for classification.

**1.4.0 Algorithm Description:**

**Training:**

* We choose “T” number of trees to build.
* For each tree “t” in the forest:
  + We generate “b” number of bags of data by choosing “s” number of data points from the training set with replacement.
  + We choose “m” number of features to select from randomly. Generally, m < M (where M is total number of features) to avoid complexity and overfitting the model.

**Prediction:**

We run predictions on all the above trees and the class which has more votes across the trees is made the final prediction. We essentially calculate the mode of all the classes predicted across the trees for each data point to make the final prediction.

**1.4.1 Parameters:**

RF: #trees, #features, #bootstrap

**1.4.2 Analysis:**

**Pros:**

**Cons:**