Project Description

Data Structures (CS-2001)

Project instructions

1. **Project is allowed with group of two individuals. The project should be distributed as every member of the group must present his/her contribution.**
2. **Plagiarism is strictly prohibited.**
3. **Due date: 22 December 2021**

Description

The ability to predict the performance tendency of students is very important to improve their skills. It has become valuable knowledge that can be used for different purposes; for example, a strategic plan can be applied for the development of quality or keeping the track of their performance. This Project proposes the application of data mining techniques to predict the grades of students, based on their historical data. In this project, you are going to use two well-known data mining techniques (decision tree and random forest) on an educational dataset related to your grades given with this project.

Rules to keep in mind to solve this problem

1. Preprocessing of the dataset
2. Feature selection
3. Training Model (Generate the forest of trees)
4. Testing (Prediction Phase)
5. Traversals to display the trees
6. Display the predictions of trees with mode

1. Preprocessing of the dataset

In the given dataset with the name of "Student's Dataset.xlsx", the records of 1780 undergraduate students in BS's program were collected from the FAST school computing NUCES. These records include Sr. No and 2 semester's courses with data structures taken during their bachelors' study with their grades. Table 1 shows the list of the main used attributes for one student.

Sr. No	Semester	Course Co	Course Tit	Credit Hou	Grade	Grade Poi	SGPA	CGPA	Warning
1	Fall 2016	MT104	Linear Alg	3	B	3	3.27	3.27	0
1	Fall 2016	MT101	Calculus -	3	B+	3.33	3.27	3.27	0
1	Fall 2016	CS101	Introducti	3	A	4	3.27	3.27	0
1	Fall 2016	CL101	Introducti	1	A+	4	3.27	3.27	0
1	Fall 2016	EE182	Basic Elect	3	C-	1.67	3.27	3.27	0
1	Fall 2016	SL101	English La	1	A-	3.67	3.27	3.27	0
1	Fall 2016	SS101	English La	3	A+	4	3.27	3.27	0
1	Fall 2017	CS201	Data Struc	3	A	4	3.75	3.57	0
1	Spring 2017	EE227	Digital Log	3	A+	4	3.71	3.49	0
1	Spring 2017	SS122	English Co	3	A	4	3.71	3.49	0
1	Spring 2017	MT115	Calculus -	3	A-	3.67	3.71	3.49	0
1	Spring 2017	SS111	Islamic an	3	B	3	3.71	3.49	0
1	Spring 2017	CS103	Computer	3	A	4	3.71	3.49	0
1	Spring 2017	EL227	Digital Log	1	B	3	3.71	3.49	0
1	Spring 2017	CL103	Computer	1	A+	4	3.71	3.49	0

1.1. Instructions for preprocessing

You are required to read the "Student's Dataset.xlsx" file using Java and restructure this dataset by arranging each student's record in one tuple, sample is given below.

Sr. No	MT104	MT119	CS118	CL118	EE182	SL101	SS101	EE227	SS122	MT224	SS111	CS217	EL227	CL217	CGPA	Warning	CS201(Data Stratures)
1	3	3.33	4	4	1.67	3.67	4	4	4	3.67	3	4	3	4	3.49	0	4
2	4	4	3.67	4	3.33	3	4	4	4	4	3.33	4	2.33	4	3.66	0	4
3	3.33	2.67	2.67	3.33	2.33	3.33	3	3	4	3	3.67	3.67	3	4	3.21	0	3.33

Each Tuple (Feature vector) must contain all subjects of the first two semesters, CGPA , Warning and pick the data structure as a label or target class. Do not consider or select those students who did not have studied the data structures. There are around 312 students available in this dataset who have studied the data structures with their 1st and 2nd semesters record. So after prepossessing you will get a dataset with around 312 instances. The data shown in this sample might be changed from data given to you, however, you got the idea of how to prepare the training examples for data training. Yes, each tuple/ feature vector will be used as a training example in the training of the trees and you are required to create around 312 tuples (Training examples/instances).

Now split the dataset examples into two with name *training_dataset* and *testing_dataset*. Assign the first 250 examples as *training_dataset* and 62 examples as *testing_dataset*.

2.Feature selection

You are required to build different trees, each of them built over a specific extraction of the observations from the dataset and a specific extraction of the features. Not every tree sees all the features or all the observations, and this guarantees that the trees are de-correlated and therefore less prone to over-fitting. Each tree is also a sequence of grade predictions based on a single or combination of features. At each node you are required to choose the feature that best split the dataset, the split divides the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. Therefore, the importance of each feature is derived from how "pure" each of the buckets is, Node purity is a measure of how homogeneous a node is. An example of node purity is information entropy.

Beginning with the question Which attribute should be tested at the root of the tree? Each instance attribute is evaluated using a statistical test (Information Gain and entropy) to determine how well it alone classifies the training examples. The "best attribute" is selected and used as the root node of the tree by calculating the information gain and entropy.

How to calculate information gain and entropy?

2.1.Instructions for Feature selection

Every training example contains around 16 features (subject grades, waring and cgpa) with 1 Label class (Data structure).

Step 1: Get the Windows Size in Int type variable

Step 2: Select the first 5 features out of 16 features, assume that the value of Windows_Size is 5, if the Windows_Size is 3 then you are required to select the first 3 features.

Step 3: Calculate the entropy and Gain of each feature subset which is selected in the step 2.

Step 4: Choose attribute with the largest information gain as the root node, divide the dataset to its branches by grades value (each node must have the 16 Childs (A+, A, A-, B+, B, B-, C+, C, C- , D+ , D, F)).

Step 5: Grow the tree and repeat the same process of feature selection on every node but on each node shift the window by round robin fashion (a fixed time slot is assigned in a cyclic way) on features by 2 and do not select the already selected feature.

Example: Features (s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, cgpa, warning) If the Windows_Size =5 then subset= {s1,s2,s3,s4,s5} and root node must be from these 5 features according to the value of the Gain.

For the next node, shift the windows by 2, then subset= {s3,s4,s5,s6,s7,s8} and the next node must be from these 5 features according to the value of the Gain. Remember previously selected features cannot be placed on the same branch of the tree.

Step 6a: A branch with the entropy of 0 is a leaf node.

Step 6b: A branch with entropy more than 0 needs further splitting.

Step 7: This function runs recursively on the non-leaf branches until all data is classified.

3. Training Model (Generate the forest of trees)

Here, you are going to generate multiple decision trees, every tree must predict the grade of data structures, collect the predictions from every decision tree and return mode (majority voting) of the predictions that will be your final grade of data structure, if the two or more classes have the same mode, then all the classes with same mode will be your possible grades. Prior to the tree generation, you have must have the following parameters Windows_Size and No_of_trees in order to generate the trees.

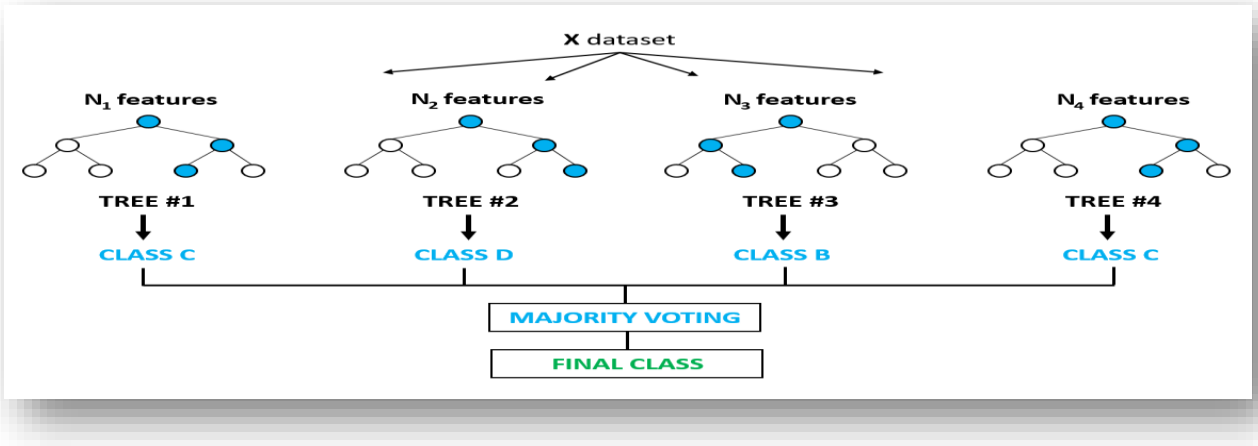
3.1.Instructions to generate the trees

Repeat the same process of feature selection on every Tree but each tree must be distinct from the previously generated tree, each time after generating 1 tree shift the Window by 2 in round robin fashion and don't select the root node that is already selected.

Example: Feature (s1,s2,s3,s4,s5,s6,s7,s8,s9,s10) If the Windows_Size =5 and No_of_trees=5, then

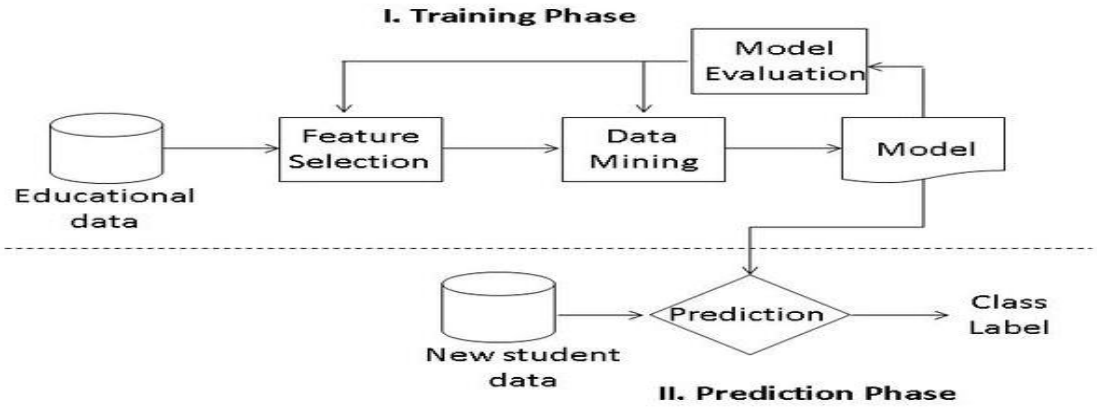
- subset1 for root-selection of tree1 = {s1,s2,s3,s4,s5} and root node must be from these 5 features according to the value of the Gain.
- Subset2 for root-selection of tree2= {s3,s4,s5,s6,s7} and root node must be from these 5 features according to the value of the Gain.
- Subset3 for root-selection of tree3= {s5,s6,s7,s8,s9} and root node must be from these 5 features according to the value of the Gain.
- Subset4 for root-selection of tree4= {s2,s3,s4,s5,s6} and root node must be from these 5 features according to the value of the Gain.
- Subset5 for root-selection of tree5= {s8,s9,s10,s1,s2} and root node must be from these 5 features according to the value of the Gain.

Note: collect the predictions from every decision tree and the mode (majority voting) of the predictions will be your final grade of data structure. After completing these steps your training model is ready for testing.



4. Testing (Prediction Phase)

Never evaluate performance on training data. The conclusion would be optimistically biased. therefore, used the testing_dataset on testing instances. Write a function that expect the test instance and return the predicted class label after parsing it from all the tress. After parsing all the 62 testing instances from all the tress, now you have 62 predicted results, compare those results with real values available in the testing examples with the name of CS201(Data structures) and calculate the prediction accuracy, Formula=link. Here is the is entire model for your better understanding.



5. Traversals to display the trees

At this point you have successfully trained and test the model. Now all you have to do is push a test instance in your trained model which creates the trees depending upon the parameters (Windows_Size and No_of_trees) and display the all trees for single instance using three different travels(Inorder , Preorder, Postorder).

5.1.Inorder traversal

- First, visit all the nodes in the left subtree

- Then the root node
- Visit all the nodes in the right subtree

5.2.Preorder traversal

- Visit root node
- Visit all the nodes in the left subtree
- Visit all the nodes in the right subtree

5.3.Postorder traversal

- Visit all the nodes in the left subtree
- Visit all the nodes in the right subtree
- Visit the root node

6. Display the predictions of trees with mode

Push a test instance in your trained model which creates the trees depending upon the parameters (Windows_Size and No_of_trees) and display prediction of each tree with mode (Majority voting) if the two or more classes have the same mode, then all the classes with same mode will be your possible grades.

Best of Luck ☺