MTL782: Data Mining Assignment 1

Archisman Biswas	Atul	Harshit Goyal
2021MT10254	2021MT10623	2021MT10143

April 20, 2024

1 Data

We are given the evaluation data from a previous offering of this course. Our first step was to explore the data, for instance :

- What type of attributes are present?
- How are the attributes distributed?
- How does attendance affect the grade?

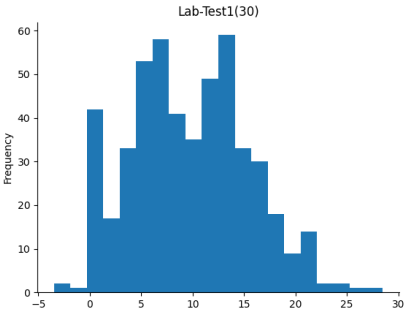
We observe that in total, there are **500** examples in the dataset. This is a **classification** task, where we predict the student's **grade**, which takes 8 values, in {A , A- , B , B- , C , C- , D , E}.

Attribute Description		
Attribute	Type	Classes discrete Range (continuous)
Lab-Test1	Continuous	0 - 30
Lab-Test2	Continuous	0 - 24
Midsem Test	Continuous	0 - 90
Gender	Discrete	{Male , Female}
Attendance	Ordinal	{Low , Moderate , High}

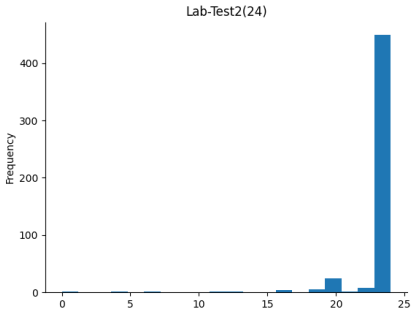
When we split the dataset using the **70-30 rule** into **training set** and **test set**, we get the following partition of data.

Data split	
Set	Number of samples
Training	350
Test	150

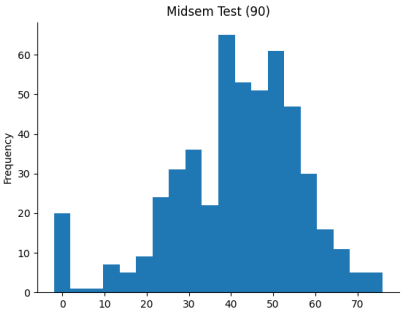
Here are some exploratory visualisations for the given data:



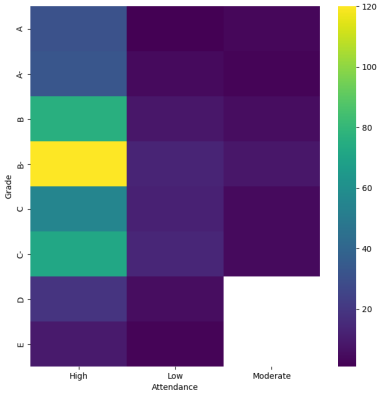
Histogram: Lab Test 1 scores



Histogram: Lab Test 2 scores



Histogram: Mid-sem scores



Heatmap: Attendance vs Grade

2 Results

2.1 Random Forest

Our first task is to train Random Forest models. We use the default CART random forests from Tensorflow. In the first part, we compare two versions of the random forests:

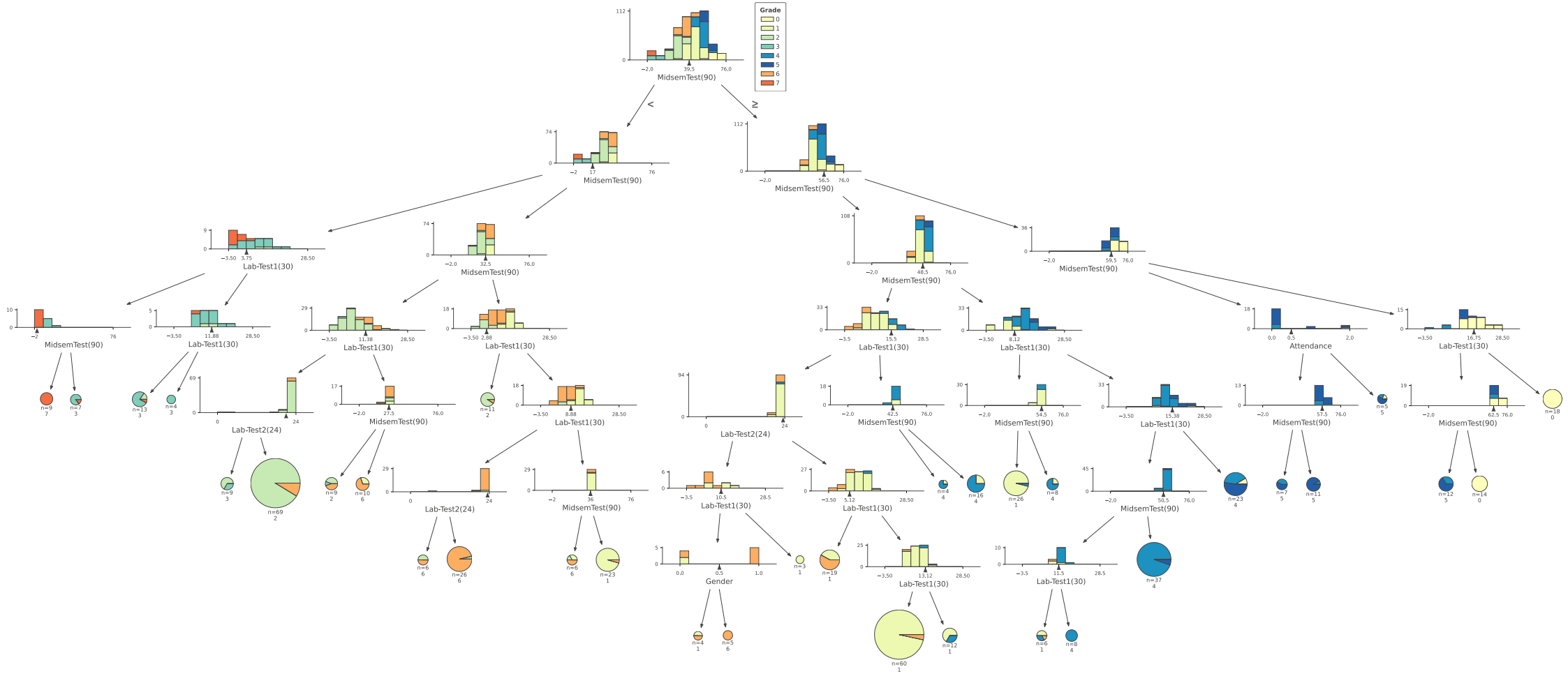
- Passing raw data to the model
- Preprocessing the data by encoding the Gender, Attendance and Grade columns into numbers. For example, Grade A goes to 0, A- goes to 1 and so on.

We train Random Forests with the default hyperparameter values for `num_trees` and `max_depth`.

`num_trees`=300, `max_depth`=16			
Pre-processing	**Train-Test split** (%)	**Training OOB Accuracy (%)**	**Testing OOB Accuracy (%)**
Raw	100 : 0	96.39	-
Gender, Attendance, Grade encoded	100 : 0	97.00	-
Raw	70 : 30	94.74	86.33
Gender, Attendance, Grade encoded	70 : 30	94.69	83.85

2.2 Tree Visualization

This is the first tree in the Random Forest constructed on data after the **70-30 split** and encoding of categorical attributes. It is built using the `dtreeviz` module.


Grade
0
1
2
3
4
5
6
7
MidsemTest(90)
<
≥
Lab-Test1(30)
Lab-Test2(24)
Attendance
Gender

2.3 Gradient Boosted Trees

Next up, we train Gradient Boosted trees, and observe their performance with default hyperparameter values. Here are the results:

`num_trees`=300, `max_depth`=16			
Pre-processing	**Train-Test split** (%)	**Training OOB Accuracy (%)**	**Testing OOB Accuracy (%)**
Raw	100 : 0	99.80	-
Gender, Attendance, Grade encoded	100 : 0	99.40	-
Raw	70 : 30	97.51	90.65
Gender, Attendance, Grade encoded	70 : 30	97.93	86.33

In comparison to random forests for the same parameters, we observe that gradient boosted trees performed better. This is consistent with the plots presented during lecture.

3 Comparison

We train a **Random Forest** and **Gradient Boosted Trees** on the same split, with **30** decision trees in each and the default max-depth, i.e. 16.

`num_trees`=30, `max_depth`=16		
Model	**Random Forest OOB Accuracy (%)**	**Gradient Boosted Trees OOB Accuracy(%)**
Training	93.35	97.50
Test	85.61	90.64

We observe that GBDT clearly outperforms RF on train and test dataset. In fact, both of them cross the required threshold of 85%.

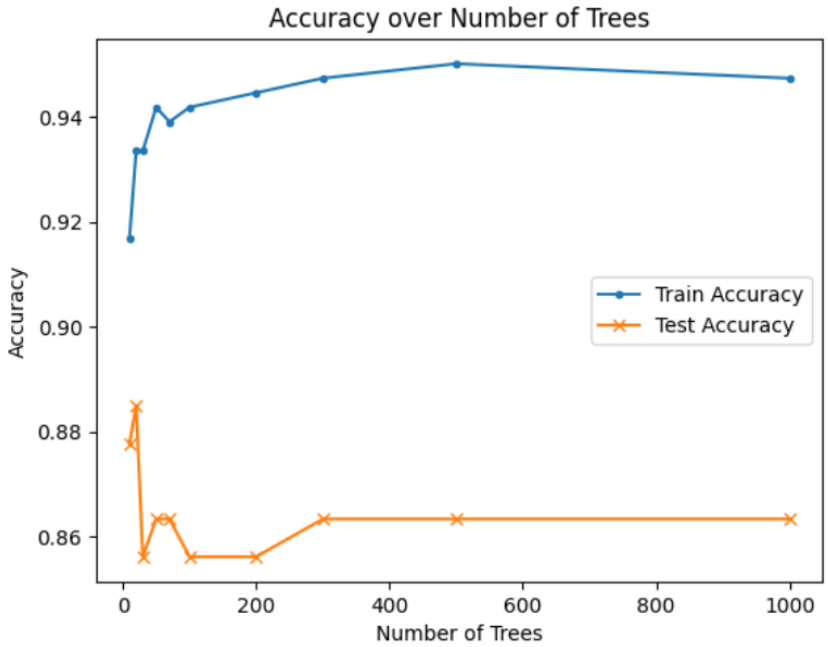
4 Tuning the hyperparameters

Note that the default value of *num_ trees* is 300, and *max_ depth* is 16.

4.1 Random Forests Tuning

4.1.1 Number of Trees

We experimented with the number of trees in the Random Forest, keeping max-depth as the default value, i.e. 16

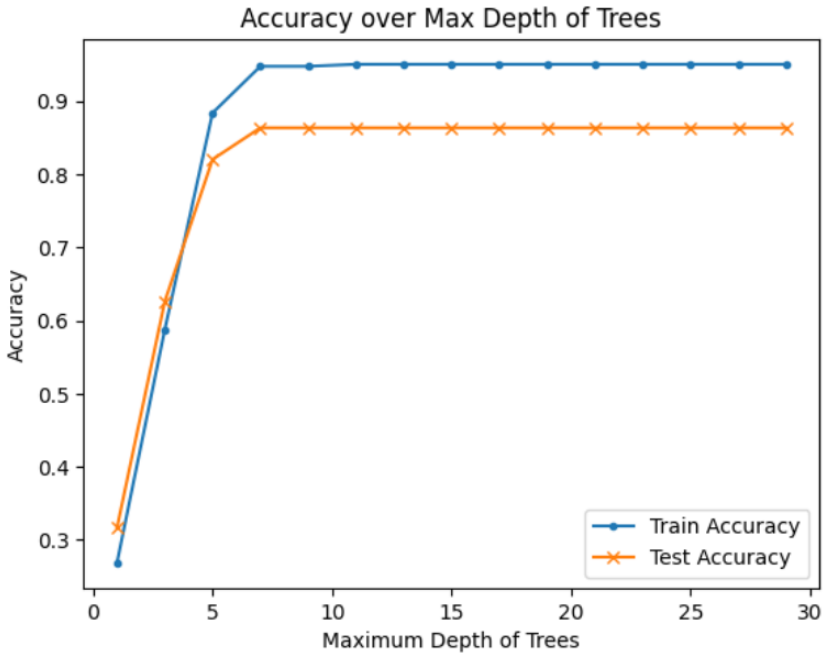


Train and test accuracy against varying number of trees

Here, we observe that after a point, both training and testing accuracy stop increasing.

4.1.2 Max-depth

We experimented with the max-depth in the Random Forest, keeping nunber of trees as 500, fixed. This is obtained by observing the previous graph. Here are the results of the experiment.



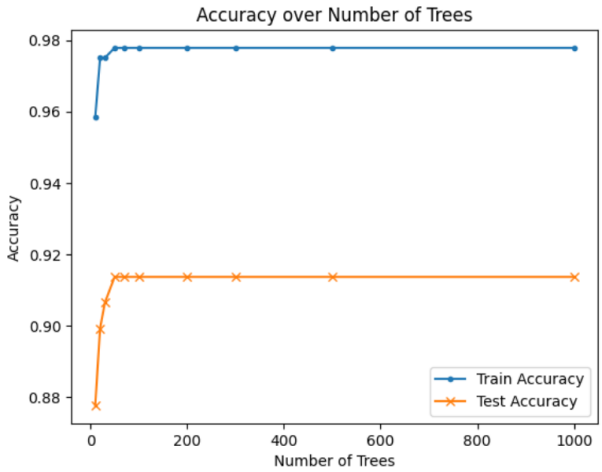
Train and test accuracy against varying max-depth

We can see that when the number of trees reach 7, the accuracy values settle. Keeping a smaller number also helps in avoiding overfitting. Also, initially due to extremely small depth, the model is underfitted and the train accuracy is even lower than the test accuracy.

4.2 Gradient Boosted Decision Tree Tuning

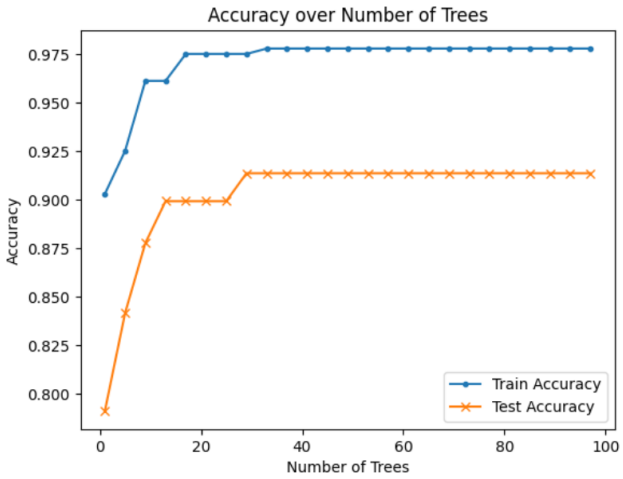
4.2.1 Number of Trees

We experimented with the number of trees, keeping max-depth as 15



Train and test accuracy against varying number of trees

We observe that in case of GBDT, the accuracies plateau faster as compared to RF. So, we test again, but on more num_trees values between 0 and 100. Here are the results:

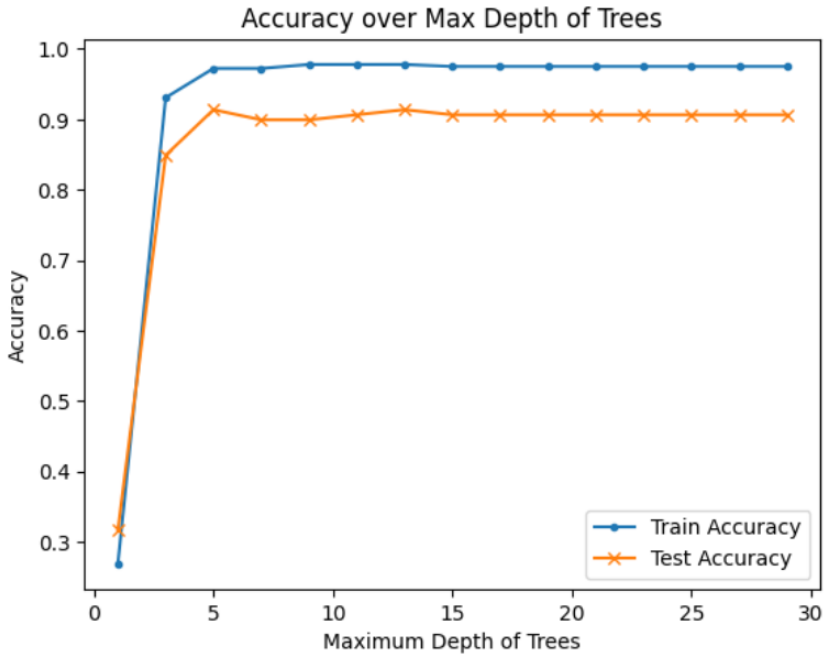


Train and test accuracy against varying number of trees

Note that for `num_trees` till 30, the accuracy increases, and thereafter, it becomes mostly constant

4.2.2 Max-depth

We experimented with the max-depth in the Random Forest, keeping number of trees as 30, fixed. This is obtained from previous graph.



Train and test accuracy against varying max-depth

The accuracy increases with increasing max-depth till 5, then a slight stagnation. It becomes maximum at around 14, and then decreases due to overfitting. Here too, initially the model is underfitted due to extremely small depth.

4.3 Final Models with Reasonable Accuracy

This section presents the final hyperparameters which achieve a reasonable accuracy over the given dataset.

4.3.1 Random Forest (RF) model

Hyperparameters:

- `num_trees`: 500
- `max_depth`: 9

RF Accuracy:

- Train = 94.73%,
- Test = 86.33%

4.3.2 Gradient Boosted Decision Tree (GBDT) model

Hyperparameters:

- `num_trees`: 30
- `max_depth`: 5

GBDT Accuracy:

- Train = 97.72%,
- Test = 91.36%

Hence, it can be seen that Gradient Boosted Random Trees achieve a better accuracy than Random Forests at much smaller costs, i.e. it requires much less number of trees and less maximum-depth to achieve it.