**Project 1.1**

# Computing - Tabulating - Recording and a bit of (Machine) Learning

**Pietro Bonizzi, Martijn Bouss´e, Philippe Dreesen, Kurt Driessens, Evgueni Smirnov**

**Student team:**

**Beatrice Boccolari**
**Alejandro Guijarro**
**Zena Saez Portillo**
**Dmitrii Sakharov**
**Mihai Solescu**

**Period 1.1, 1.2, and 1.3**
**Academic year 2023-2024**
**Data Science and Artificial Intelligence**
**Faculty of Science and Engineering**
**Maastricht University**

## Abstract

This study focuses on the comprehensive analysis and prediction of academic grades, after taking into consideration records of students' university grades and some additional properties of these students. The objective was to thoroughly describe our data and make predictions for students' missing grades using various data analysis methods.

We extracted valuable insights, including the identification of the temporal structure of the program, and student passing percentages. Our predictive models give an estimation of future student grades, enhancing our understanding of individual student performance. One of the conclusions of our study is that taking the weighted average of the best decision stumps by variance reduction is flawed, giving a worse overall prediction than taking the individual best decision stump

## Introduction

The goal of this study is to provide the students involved in the project with a high level of understanding of data representation, analysis and interpretation and the opportunity to make use of and apply the knowledge gained from the lectures on given sets of data with various properties.

We used a fictitious dataset of 30 features representing course final grades (FGs) of 18321 graduates and 1128 current students of a university program. For the current students, we have used 4 additional features, representing some attributes of students. Moreover, the current students are missing values for the courses they haven't completed.

The main objective of the project is to figure out the extent to which predictions of the student's accomplishments can be made based on data analysis techniques and the data at hand, to be able to answer questions such as:

- What is the structure of the academic program of the students in the dataset, i.e. how many years of study are there and to what year do courses belong?
- Which are the most challenging courses in the dataset?
- Which of the current students in the dataset will graduate cum-laude?
- What are the course passing rates for the current students in the dataset?
- To what accuracy can we predict students' missing grades?

The algorithms we have considered for predicting the missing values are Decision Stump (DS), an ad-hoc ensemble of Decision Stumps, Decision Tree(DT), AdaBoost (AB) and Random Forest (RF).

Our process for answering these questions was as follows:

- Sort the courses by the number of missing values to find out their order and group them into years.

- Compute additional features, such as student GPAs, median FGs and the year the student is in.
- Implement the above ML techniques to make predictions.
- Implement a cross-validation method for computing model accuracies.
- Use hyperparameter tuning to improve the prediction accuracies of our models;
- Evaluate the method using prediction accuracy, mean absolute error (MAE) and mean squared error (MSE).
- Create visualizations to express our results.

## Literature Review

A study published in IEEE Access dealing with a dataset consisting of 12 academic features (STEM course scores) of 4266 students from a 4-year university program, taken across 14 years, found that in predicting the score for a 13th STEM course an RF algorithm had an accuracy of 91% using various resampling techniques (Aya Nabil et al., 2021).

## Methods

We adopted a variance reduction method to find the best property for splitting data, and our ensemble prediction incorporated weighted means based on total variance. Additionally, we also used the AdaBoost method to predict more accurately the missing grades.

**1st and 2nd phase**

In the first step, we evaluated the courses' difficulties based on their average grade using Graduate Grades. We have also found cum-laude  graduate students using Graduate Grades. We considered cum-laude students as students that have GPAs for all the given courses more or equal to 8. In order to compare courses between themselves we used 3 different metrics: Pearson correlation, Cosine similarity, and Euclidean Distance.

The formula for evaluating the Pearson correlation coefficient (Clarke, G. M., & Cooke, D., 1998), where x are the grades for the first course in the pair and y are the grades for the second course, with \bar x and \bar y the means of the courses:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

We calculated this coefficient for each pair of courses.

The formula for cosine similarity (Jeffrey, A., 1995), where A is a vector of grades for the first course in the pair and B is a vector for the second course. There is a dot product of these vectors in the numerator and the product of the length (euclidean L2-norm) of these vectors in the denominator.

$$\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

We calculated cosine similarity for each pair of courses.

The formula for Euclidean distance between courses, where A is a vector of grades for the first course in the pair and B is a vector for the second course. Ai and Bi are the components of these vectors (grades).

$$\text{Euclidean\_distance}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

We calculated the Euclidean distance between each pair of courses.

In the second step we had the main goal of predicting how many students would graduate in the current year. To achieve that we first assumed that since there are 30 exams in total, each year would be composed of 10 exams.

We began by identifying potential graduation candidates among students who needed to complete fewer than 10 exams. Subsequently, we determined the passing percentage for each course in the third year, analyzing every exam. Following this, we computed the average grade and the average number of exams remaining for each student. By calculating the average probability of passing for each individual, we then integrated this percentage with the count of eligible students, providing an estimation of the overall graduation rate.
Another of our goals was to arrange the exams based on how many NG values each exam had in order to be able to extrapolate some insights for the creation of the forest. To do so we took advantage of the bubble sort algorithm(Kunjwal, P. , 2015).

One of the main goals of our project is to predict missing grades of Current Students with the information about other students' grades and additional student information dataset for the particular missing grades of some students for some courses.
Our method of prediction is called "Decision Stump", which will be advanced to "Decision Trees Forest" in Phase 3. "Decision Stump Forest" is a "forest" of one-level decision trees, so decision trees with one splitting feature (*Decision Stump*, 2023). Forest means that it is an ensemble (*Veronica*, 2020) of different models, in our case it is the ensemble of one-level trees.
First, there are some specifics in our implementation of decision stumps. When we are dealing with continuous features as predicates, we make splits by all the possible values of this feature and then we pick the best split. Our method of picking the best split will be discussed in the next paragraph. While dealing with categorical features, we are considering the n amount of branches, and therefore n amount of children coming from the root, where n is the number of categories in this property.

In order to find the best property along which to split the data we make use of a method called *variance reduction* (Ganapathi, 2021).

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n},$$

where $\sigma^2$ is the variance and $\bar{x}$ is the mean value.

Essentially, for each of the subsets into which the students are split by property, we want to minimize the average distance from the mean of the grade given the expected property a student might have (in order to incorporate the probability of a student having a specific value for that property we weight the variance of the subgroup of students having that value against the subgroup's count).
The variance reduction per split is the difference between the grade variance of the student total and the weighted variance of the subgroups where for each $i$ representing a subset in the given subset collection $n_i$ is the number of students in the subset, and

$$Variance\ reduction = \sum_{i \in I} \frac{\sigma^2}{n_i},$$

$$Reduced\ variance = Original\ variance - Variance\ Reduction$$

This method works best for slow, continuous changes in the data (Note that even though the student grades are discrete, we are treating them as hidden continuous variables that get rounded to the nearest integer). As such, the real-life case of a student who suddenly gets a lower grade than normal would never be predicted, as it was not part of the initial dataset. Moreover, since the data set does not contain grades below 6, then such a grade would never be predicted.

Now, the method of uniting predictions together using different decision stumps will be explained. There are different methods in ensembles that can deduce the final prediction. The most used are voting for classification and averaging for regression. In our case, we are using averaging because of the data type of our target feature, but we have decided to enhance the basic averaging a bit and we have the following formulas for the final prediction:

$$\text{Mean}_{\text{ weighted}} = \sum_i \text{mean}_i \times \left( \frac{1}{\text{totalVariance}_i} \right)$$

$$\text{Prediction} = \frac{\text{Mean}_{\text{ weighted}}}{\sum_i \frac{1}{\text{totalVariance}_i}}$$

In this formula, $\text{mean}_i$ is the mean of the target feature in an $i^{th}$ decision stump, according to a student's property for the feature by which the decision stump itself is split by, and $\text{totalVariance}_i$ is the total variance of the split in the $i^{th}$ split. So the idea is to weight our means of each decision stump on the total variance of each to make decision stumps with less total variance to play a greater role in the prediction because these decision stumps split the data better.

**3rd Phase. Predictions with Random Forest.**

In the last phase of the project we predicted the missing grades not just with decision stumps but with the forest of decision trees. Let us start with the concept of the random forest.

**Random Forest**

Random forest is an ensemble of Decision Trees which takes the predictions from many different trees that have been trained on different datasets which have been randomly taken from our whole training dataset. *(Russell, S. J., & Norvig, P., 2016)* As long as the Decision Tree is the key element in our algorithm, we should first discuss the essence of a decision tree.

So, Decision Tree is an algorithm for regression or classification problems, which consists in applying decision stumps sequentially and therefore getting the dataset splitted into many "leafs". So the training of a decision tree implies finding the best splits (earlier in this paper we discussed how these best splits can be found), that splits the whole dataset into small homogenous samples. The prediction of a decision tree is basically passing a new object, for which we need to predict the target feature through the algorithm and checking in what "leaf" it fell. Then, if it is the classification problem we are just taking the most popular target class in the "leaf" as a prediction for our object and if it is the regression problem, we are taking some statistical value (for example, the mean or the median). *(Russell, S. J., & Norvig, P., 2016)*.

But there is a problem with a simple decision tree, it can easily be overfitted, it is really sensitive on training dataset (as it can't extrapolate) and splits that are made at the top of the tree affects the splits in the bottom really much. *(Sinitsin P., 2021).* But also, there is an advantage of a decision tree, if it is deep enough it greatly diminishes bias (because if we want to we can make a leaf for every object in our training dataset, so the prediction accuracy will be 100 percent), so the problem that should be solved somehow is high variance. That's where the idea of bootstrap aggregation and ensembles come. Bootstrap is the process of sub-samples generation by means of sampling with return. Basically, with bootstrap we just create a new sample which has approximately the same distribution as our previous sample but still is different. *(Sarkar, D., & Natarajan, V., 2019).* So, in order to lower the variance of the decision tree algorithm, a forest of trees can be made, each tree in which will be trained on a different data sub-sample. The reason for variance to decrease will be proved below. A proof is taken from the source 10 *(Elistratova, E., Gubko, P., 2021).*

Let's assume that the size of the training dataset is n. Let's take k bootstrapped samples from it. We will call the i-th sample $X^i$. We will call the i-th model which predicts a target feature for object x $b_i(x) = b(x, X^i)$, where the value of the function on the left side of equality is our prediction for that object. Prediction of forest for object x is $a(x) = \frac{1}{k}(b_1(x) + \cdots + b_k(x))$. Let's find out what will be the variance of our ensemble of models.

$$\mathbb{V}_X[a(x, X)] = \mathbb{E}_X[a(x, X) - \mathbb{E}_X[a(x, X)]]^2 =$$

As our prediction consists of predictions of k models, let's substitute it with the prediction of those models.

$$= \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^{k} b(x, X^i) - \mathbb{E}_X \left[ \frac{1}{k} \sum_{i=1}^{k} b(x, X^i) \right] \right]^2 =$$

The mathematical expectation of the averaged sum of our models is just a math expectation of our model, so let's converge it to this view and bracket the multiplier 1/k.

$$= \frac{1}{k^2} \mathbb{E}_X \left[ \sum_{i=1}^{k} \left( b(x, X^i) - \mathbb{E}_X b(x, X^i) \right) \right]^2 =$$

Proceeding to the next step involves disclosing the square of the sum. This means that each term of the sum is multiplied by itself and by every other term of the sum, resulting in two separate sums.

$$= \frac{1}{k^2} \sum_{i=1}^{k} \mathbb{E}_X \left( b(x, X^i) - \mathbb{E}_X b(x, X^i) \right)^2 +$$

$$+ \frac{1}{k^2} \sum_{k_1 \neq k_2} \mathbb{E}_X \left[ \left( b(x, X^{k_1}) - \mathbb{E}_X b(x, X^{k_1}) \right) \left( b(x, X^{k_2}) - \mathbb{E}_X b(x, X^{k_2}) \right) \right] =$$

We see that the first sum is just the variance of the i-th model and the second one really looks like the covariance of the k1 and k2 model.

$$= \frac{1}{k^2} \sum_{i=1}^{k} \mathbb{V}_X b(x, X^i) + \frac{1}{k^2} \sum_{k_1 \neq k_2} \operatorname{cov} \left( b(x, X^{k_1}), b(x, X^{k_2}) \right)$$

Here we are starting to understand why we took different samples to train our models, we want the covariance between models to be as low as we can get (if it is 0 then we decreased the variance of the ensemble by k times because the variance of the ensemble is basically what we get from the first summand: $\frac{1}{k} \mathbb{V}_X b(x, X)$ and the variance of each algorithm is $\mathbb{V}_X b(x, X)$. So now our goal is to make our models as low correlated as possible, that is why we want to take random features for each decision stump in our tree, so our models will be different. The idea of restricting the features that can be taken into consideration while finding the best split in a decision stump on each iteration of a decision tree is one of the main ideas of the random forest.

**Branching factor and prediction function**

In the first phase of the project, we described that we used variance reduction as a branching factor (a function that is used to find the best split in the decision stump) and we used mean in a leaf to predict a grade. For the sake of getting higher accuracy, we have implemented additional branching and prediction functions. While using variance reduction as a way to detect the best split, the MSE is calculated, we have always implemented a branching

function based on MAE. To minimize MAE of the prediction: $L(y_i, c) = |y_i - c|$,
where L is loss function and y_i is a real value of the object and c is our constant prediction, the median of the targets of training objects should be predicted, because exactly in that case MAE is minimized *(Sinitsin P., 2021)*. Therefore, the information metric function will look like this: $H(X_m) = \sum_{(x_i,y_i)\in X_m} \frac{|y_i - MEDIAN(Y)|}{|X_m|}$
where Xm is our dataset of objects in the node, y_i is correct grades for that training set. So, the information gain will have the following formula:

$$H(X_m) - \frac{|X_l|}{|X_m|}H(X_l) - \frac{|X_r|}{|X_m|}H(X_r)$$

If we allow algorithms to use this function (the accuracy of this approach will be checked during tuning the model), we will also create a new prediction function that will predict not the mean of the leaf but the median.

**AdaBoost algorithm**

AdaBoost is a boosting algorithm that creates a sequence of weak learners, where the training data of each learner is influenced by the importance and misclassifications of previous learners *(Freund, Y., & Schapire, 1996)*.

The algorithm begins by assigning equal sample weights to all records in the dataset. The first base learner is then formed by creating stumps for each feature, and the learner with the least Entropy value is selected.

Following the assessment of the learner's total error (the sum of all the errors in the classified record for sample weights) and performance, important adjustments are made to the sample weights.

The importance, or “amount to say” is calculated with the formula:

$$\frac{1}{2}\log_e\left(\frac{1 - Total\ Error}{Total\ Error}\right)$$

The algorithm gives more weight to records that the learner misclassified, highlighting the need to correct these mistakes in the next rounds. On the other side, records that were correctly classified get lighter weights. To maintain fairness and balance, the algorithm then normalizes these weights. This normalization ensures that when you add up all the weights, it equals 1.

This balancing act prevents any single record from having too much influence on the overall decision of the group of learners, making sure each record plays a fair role.
Following the weight update, a new dataset is created based on these weights, giving precedence to misclassified records. This process is iteratively repeated, creating a sequence of learners with updated weights, until minimal error is achieved compared to the initial normalized weights.

**Metric**

In order to find the best models for our target feature we should decide on a metric, which we will try to maximize. As we have been clarified that despite the discrete nature of grades we still should solve a regression problem, we decided to select a regression metric and not classification. Due to the fact that grades are discrete, we chose final metric showed below:

$$\text{Accuracy } = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\text{round}(\hat{y}_i) = y_i)$$

**Models' hyperparameters and selected methods**

In this section hyperparameters available for basic models and specific methods and decisions we have made within our basic models will be discussed.

1. **Decision Stump**

Within just a basic model of a Decision Stump we have only one hyperparameter which is *"branching function"* which was discussed above. Also, while making a Random Forest a Decision Stump will have restricted courses (which are restricted to make a forest random), but it will be discussed in the Random Forest part.
Certain solutions we have made in a model of Decision Stump are having only binary splits both for continuous and categorical variables. This decision was made for continuous variables due to the enhanced prediction performance in such splits because each binary split is focused on separating the data based on a single, most significant attribute at each step. Also it eases implementation, saves us from having leaves with 0 objects and enhances computational efficiency.
As for the categorical variables, we found an ordinal pattern in our categorical features, so the ordinal encoding and binary splitting made sense in our case and boosted computational effectiveness (because splitting categorical variable at each possible combinations take a lot of time and splitting variables into multiple branches might lead to the problems with prediction accuracy). How the features were encoded will be discussed in the following sections of the report.

2. **Decision Tree**

Within a basic Decision Tree model we have a hyperparameter: *"prediction function*"which was discussed above in the report. For the stopping criteria (the criteria which stops the branching and makes node a leaf when a certain condition is satisfied) we had: *Max Tree Depth* (maximum amount of sequent splits) , *Min Leaf Size* (minimal amount of objects in a leaf)*, Min Variance* (to state it correctly, minimal loss function value at a node, MSE or

MAE) and *Min Information Gain* (when the information gain for the next split is too low, we just make node a leaf). These criteria help us tune a tree, make it overfitted or under-fitted, therefore that lets us find a perfect combination for predicting on the data, algorithm hasn't "seen". *(Sinitsin P., 2021)*

3. **Random Forest**

The Random Forest itself has a hyperparameter *Tree Number* which indicates the number of trees in the forest.
As for the certain solutions in the Random Forest Model, we have decided to take max(n/3, 2) properties to choose from at each node, where n is the amount of available features for the whole Decision Tree (available features will be discussed later), which are taken randomly. *(Elistratova, E., Gubko, P., 2021).* Moreover, we are bootstrapping samples the same size as the whole training dataset for each Decision Tree.

4. **AdaBoost**

The one hyperparameter that an AdaBoosted ensemble has is its depth, or the number of iterations to run the algorithm, adding this many weak learners to the ensemble.

## Implementation

As we have a complex problem of predicting different target features for objects with different sets of available predictors, we should develop a pipeline to solve this problem for any objects in our dataset and for objects beyond our initial dataset which might have a different set of available predictors.

**Dealing with NGs in the data**

The first issue we solved is a lot of NGs in our data that are missing for a reason. In order to do that, we have detected which courses are 1st, 2nd, 3rd year and which courses haven't been taken at all. We distributed courses in these groups based on the amount of NGs for each course. You can see the distribution on chart 12.

The notation "n-th year course" is used here just to distribute courses into different groups based on NGs values, because we will take this into consideration while creating a model robust to missing values, so the actual meaning of these groups might be different.
The distribution we got:
1st year: 'ATE-003', 'ATE-214', 'DSE-003', 'MTE-004', 'LOE-103', 'BKO-800', 'DSE-007'
2nd year: 'LPG-307', 'PPL-239', 'FEA-907', 'TGL-013', 'SLE-332', 'WOT-104', 'MON-014', 'JTW-004', 'ATE-008', 'LUU-003'
3rd year: DSE-005', 'JJP-001', 'JHF-101', 'GHL-823', 'LDE-009', 'KMO-007', 'JTE-234', 'ATE-014', 'WDM-974', 'HLU-200'
3rd year fully missing: 'BKO-801', 'TSO-010', 'PLO-132'.

Based on our distribution we developed a pipeline of predicting courses iteratively, so in that case we don't have to deal with missing values inside our algorithms. You can see the idea of

the pipeline on Figure 1, where StudentInfo column - all Student Info Properties, 1st year - all 1st year courses, 2nd - all 2nd year courses, 3rd - all 3rd year courses, Missing - all fully missing courses. Orange columns - columns that are used as predictors for each iteration, green cells - predicted grades.
So, at first we predict and insert all missing 1st year courses with StudentInfo properties, then we predict and insert all 2nd year courses with StudentInfo and 1st year courses, then we predict and insert all 3rd year courses with StudentInfo, 1st and 2nd year courses and then we predict missing courses with all years courses but not StudentInfo.

**Predicting missing courses**

In the pipeline, described above we haven't mentioned how we are predicting missing courses if we don't have training data for them in Current Grades. The solution we came up with is to train our model on all other courses in Graduate Grades to predict these 3 missing courses. Though the distributions of the same courses in Graduate Grades and Current Grades are slightly different (check charts 5 and 6 for the comparison of the same course distribution between Graduate and Current) and there are no grades lower than 6 in Graduate Grades, that is the only possible solution. As for advantages for using Graduate Grades we can highlight that we have a huge training dataset with no missing values.

**Feature engineering**

In this part of the project we transformed existing features and added new ones in order to help algorithms get a hold of all the patterns in data.
In the previous parts of the report it has been stated that our categorical features actually have ordinal patterns. So, before encoding them into numbers we checked the average GPA for all the courses for people with certain classes of these categorical features. You can see on the charts 7,8,9. While Hurni Level and Suruna Value have an obvious order, which we will take into consideration when encoding, Volta doesn't have it, but still, there is a difference between average GPA for all courses between classes of Volta: 7.153182625829043 for "1 star" class, 7.3145549861418075 for "2 stars" class, 7.2453608409362324 for "3 stars"class, 7.259209424216109 for "4 stars" class and 7.2827803457422124 for "5 stars" class, so we will encode according to that. Final encoding:
"Suruna Value": lobi - 0, nulp - 1, doot - 2;
"Hurni Level": nothing - 0, low - 1, medium - 2, high - 3, full - 4;
"Volta": 1 star - 0, 2 stars - 4, 3 stars - 3, 4 stars - 1, 5 stars - 2;

As for the new features, we added *GPA and median grade* columns for each student for the 1st year courses group, 2nd year courses group and 3rd year courses group. So, GPA 1st year column is basically a sum of all the first year courses grades divided by the amount of first year courses. These features will help the algorithm easily detect cum-laude students. The reason for having 6 added columns instead of just 2 (GPA overall and median grade overall) lies within the issue of missing values and the issue of different course difficulties. So, the problem is that we can't take all the courses into calculation of average GPA, because

students don't have all the courses, and we can't take just the average GPA of the courses which a student has, because courses difficulties differ and average GPA of a student with a certain set of courses available will be higher than the other's student average GPA, just because courses the latter did were harder.
That's why we are adding GPA and medians for a group of courses. We add these columns according to our prediction pipeline (predicting first year courses for everybody - > adding columns for first year - > predicting second year courses etc.).
Moreover, we have added a Student *Year* column which indicates in which year the student is currently (so, if a student has most of the 3rd year courses, he is in the 3rd year, if a student has 2nd year courses but doesn't have 3rd year courses, he is in the 2nd year, etc.). This feature is really important for prediction accuracy because a certain course now and the same course 2 years ago might have different difficulties and different grade distribution (examples of distributions of the same course for 2nd year students and 3rd year students on charts 10, 11).

**AdaBoost**
Our implementation of the AdaBoost algorithm uses decision trees instead of stumps, as in our tests we have obtained higher prediction accuracies this way rather than with stumps.

**Regression Line**
We add a regression line to a scatter plot in our application using the formula from *(Russell, S. J., & Norvig, P., 19.6.1 Univariate linear regression 2016).*

## Experiments

**Accumulated error**
Due to the fact that we predict courses iteratively an obvious problem might occur. When we are predicting a second year course, only approximately 70 percent of the predictions are correct, nevertheless we use them to predict 3rd year courses. The solution to this problem might be taking only StudentInfo properties and 1st year courses for predicting 3rd year courses who also miss second year grades. But, we can't be sure which approach is better, we basically don't know if our predictions are good enough to make new predictions based on them. Therefore, we prepared correct samples to test this exact issue and conducted a series of experiments, in which we tested which of the approaches is better. At first we trained 2 models, one of which was trained on StudentInfo, 1st year and 2nd year courses and other have been trained only on StudentInfo and 1st years courses. Then, we took a sample of students that had both 2nd and 3rd year courses. We predicted 2nd year grades with a different training set and substituted the actual grades with the predicted ones. That was a testing dataset for both models. The results of these series of experiments showed that the accuracy of the model that also took our predicted 2nd year courses was by average 1.441 percent better, which is not a significant difference, but it made it clear that our second year predictions were good enough to use them as predictors.

## Results

The programme is split into three years, having 10 courses each, and the current grades are taken from before the last three courses of the year. (See *Chart 5* in the appendix)

17.9% of students are eligible for graduation within the next year, while only 17% are estimated to graduate.

Out of all the student attributes, Hurni level is the best overall predictor for student performance, followed by Suruna count (See *Chart 1* and *Chart 2* in the appendix), while Volta and Lal count are poor overall predictors. (See *Chart 3* and *Chart 4* in the appendix)

The algorithm using multiple prediction stumps has a lower overall accuracy than that using a single prediction stump. (See *Table 1* in the appendix)

With RF and by hypertuning, we get accuracies for target courses (excl. the last three in the third year) in the range of 60.69% - 80.19%, the highest from our choice of algorithms. For DT, we get ~50.31%. For AB, we get ~49.68%. See Table 3 for exact results of RF.

## Discussion

Our model's ability to predict missing grades showcases its effectiveness in handling complex academic data. However, it's important to recognize certain limitations in this approach, such as the assumption that the grades will remain uniform without sudden drops in the performance, or the absence of grades below 3 in the dataset. Additionally, predictions using the stump forest method are not as accurate as those made by a single decision stump, and they take longer to run. Looking forward we plan to try a different approach: use a random forest with individual predictions to improve accuracy.

The predictions using our stump ensemble method fell behind those made by using a single decision stump, in addition to the program taking considerably more time to run.

## Conclusion

Our investigation explored academic grades using data analysis, revealing insights into course difficulties, identified cum laude graduates, estimated graduation rates, and developed predictive models for missing grades. To reiterate, we discovered the structure of the programme, the order in which student attributes by overall prediction accuracy, the passing percentages for the current year, and that the prediction method based on multiple decision stumps is worse than the one based on a single decision stump.. We also predicted that 83%

of eligible students will graduate. In the future, we aim to make our predictions more accurate by using more precise and complex modeling techniques.

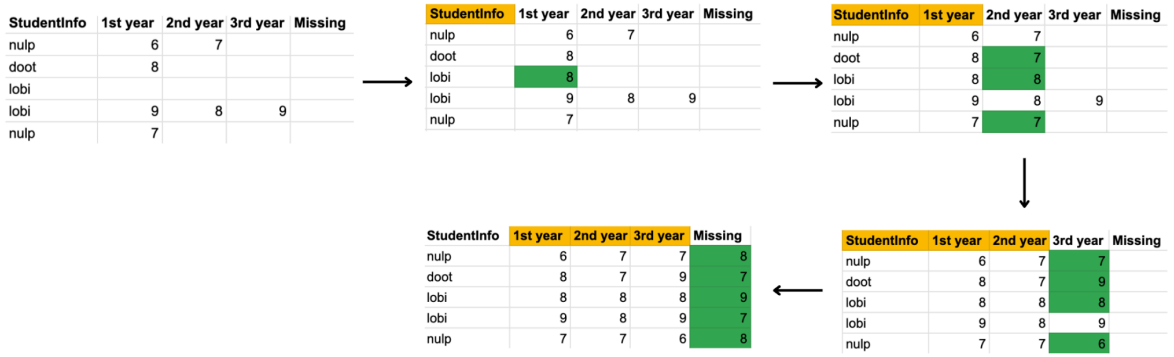
## References


1) Rice, J. A. (2007). *Mathematical statistics and data analysis* (Third edition, International). Brooks/Cole Cengage Learning.
2) International Conference on Similarity Search and Applications (2023). *Similarity search and applications*: 16th international conference, SISAP 2023, A Coruña, Spain, October 9-11, 2023, proceedings. (O. Pedreira & V. Estivill-Castro, Eds.) (Ser. Lecture notes in computer science, 14289). Springer. https://doi.org/10.1007/978-3-031-46994-7
3) *Decision Stump*. The Hitchhiker's Guide to Machine Learning Algorithms. (2023, October 22). November 22, 2023. https://serpdotai.gitbook.io/the-hitchhikers-guide-to-machine-learning-algorithms/chapters/decision-stump
4) Veronica, A. (2020, May 13). *Ensemble Learning Methods in Machine Learning*. Medium. November 22, 2023. https://medium.com/analytics-vidhya/ensemble-learning-methods-in-machine-learning-5d2f849192f8
5) Ganapathi, S. (2021, March 17). Decision Trees and Splitting Functions *(Gini, Information Gain and Variance Reduction)*. Medium. November 22, 2023. https://medium.com/nerd-for-tech/decision-trees-and-splitting-functions-gini-information-gain-and-variance-reduction-23192f639048
6) Freund, Y., & Schapire, R. E. (1996.). Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*.
7) Kunjwal, P. (2015). BUBBLE SORT. *International Journal of Innovative Research in Technology*, *Volume 2 Issue 6*. https://ijirt.org/master/publishedpaper/IJIRT142753_PAPER.pdf
8) Sarkar, D., & Natarajan, V. (2019). Ensemble machine learning cookbook : over 35 practical recipes to explore ensemble machine learning techniques using python. Packt Publishing.
9) Russell, S. J., & Norvig, P. (2016). Artificial intelligence : a modern approach (Third, Ser. Prentice hall series in artificial intelligence). Pearson Education.
10) Elistratova, E., Gubko, P. (2021). Ensembles in machine learning. Yandex Machine learning Textbook. Retrieved January 22, 2024, from https://education.yandex.ru/handbook/ml/article/ansambli-v-mashinnom-obuchenii
11) Sinitsin P. (2021). Decision Trees. Yandex Machine learning Textbook. Retrieved January 22, 2024, from https://education.yandex.ru/handbook/ml/article/reshayushchiye-derevya
12) Nabil, A., Seyam, M., & AbouElfetouh, A. (2021). Prediction of students' academic performance based on courses' grades using deep neural networks. IEEE Access, 9, 140731–140746. https://doi.org/10.1109/access.2021.3119596

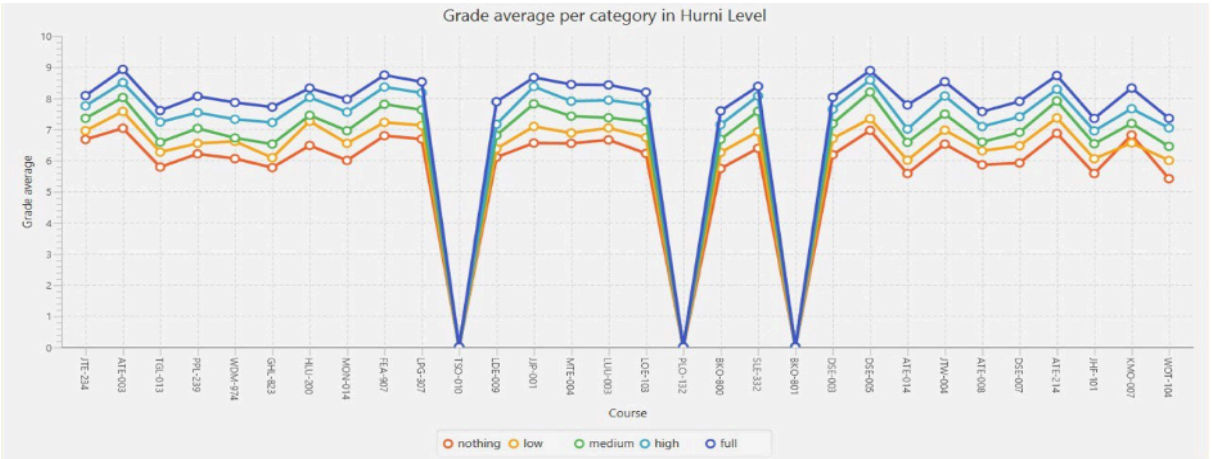
13) Jeffrey, A. (1995). Handbook of mathematical formulas and integrals. Academic Press.
14) Clarke, G. M., & Cooke, D. (1998). A basic course in statistics (4th ed.). Arnold.

## Appendix

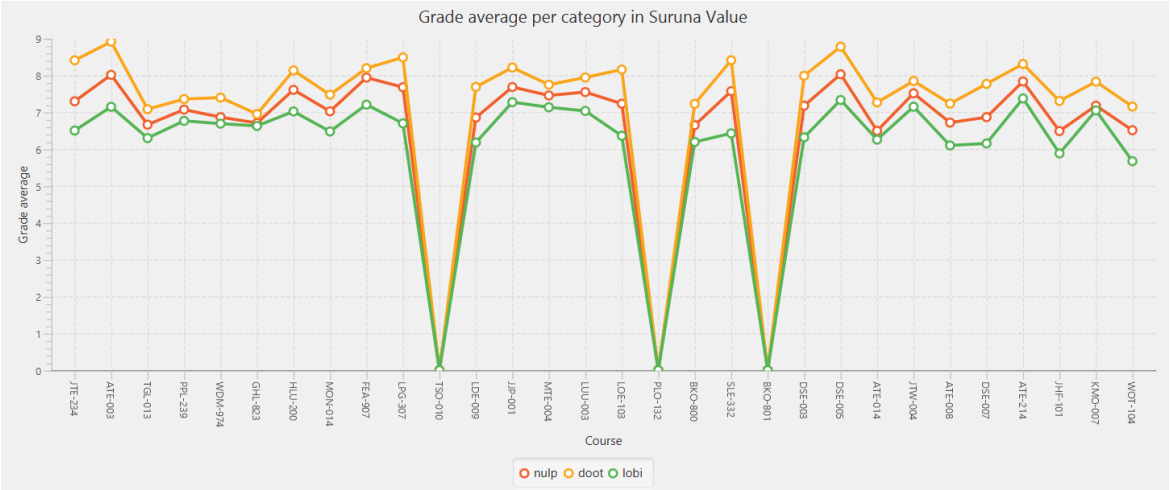
The basic tools we used to perform all our analyses and methods are the Java Development Kit (JDK 21) and the JavaFX library (JavaFX 21) for building an application that visualizes our findings. In the following paragraphs, specific methods used for each of the steps from the 1st phase of the project will be discussed.



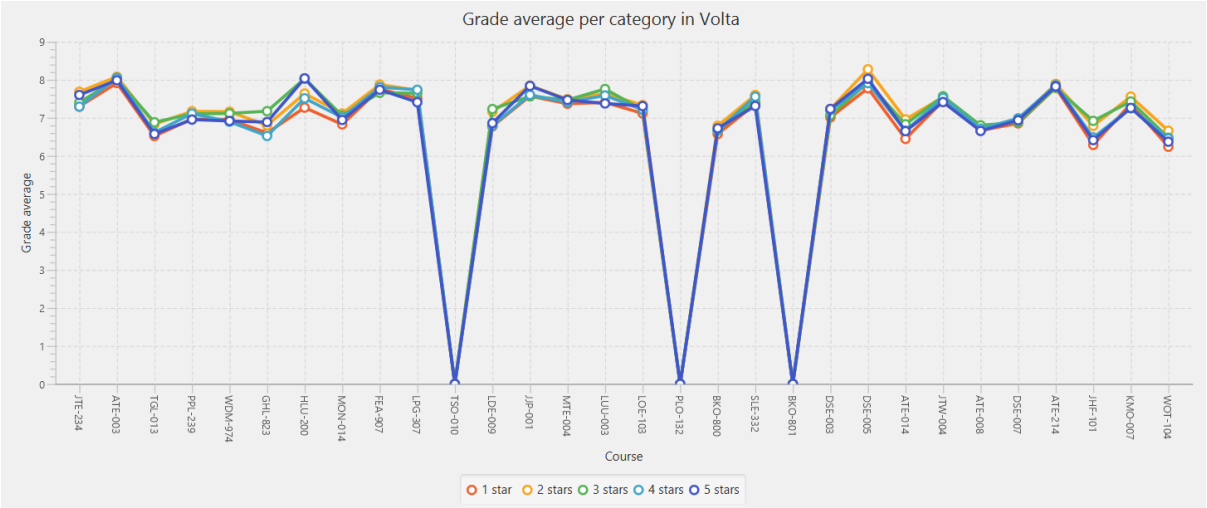

[Figure 1]



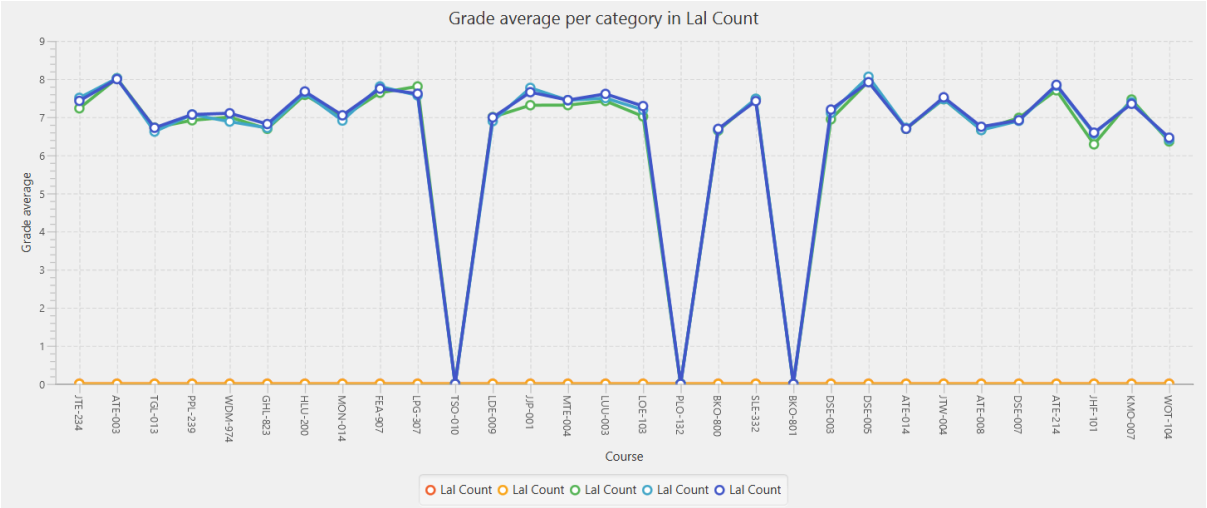

[Chart 1]



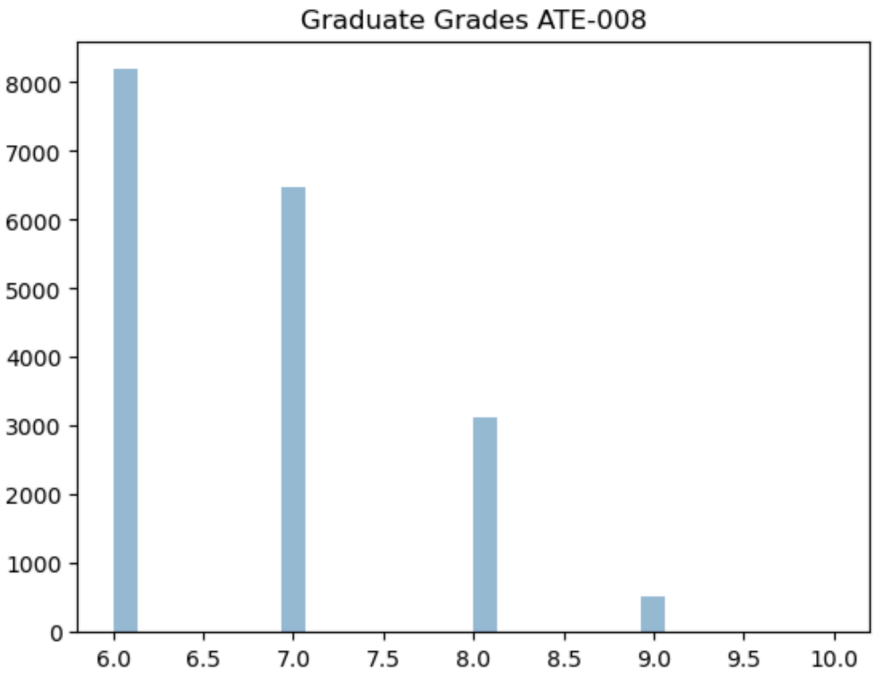

[Chart 2]



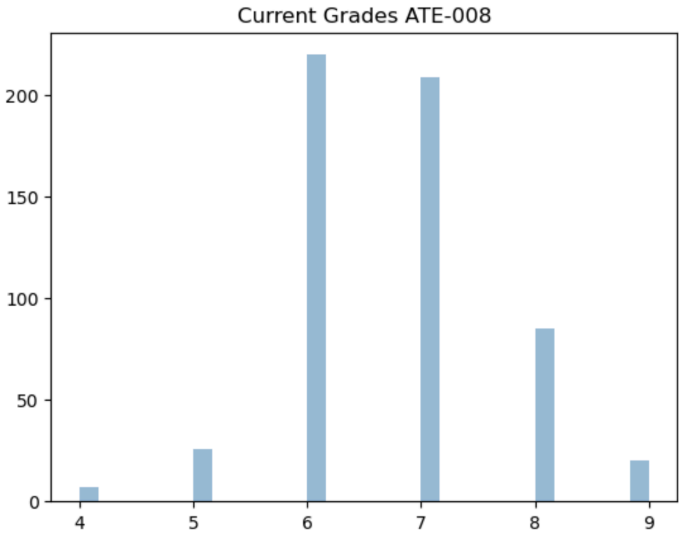

[Chart 3]



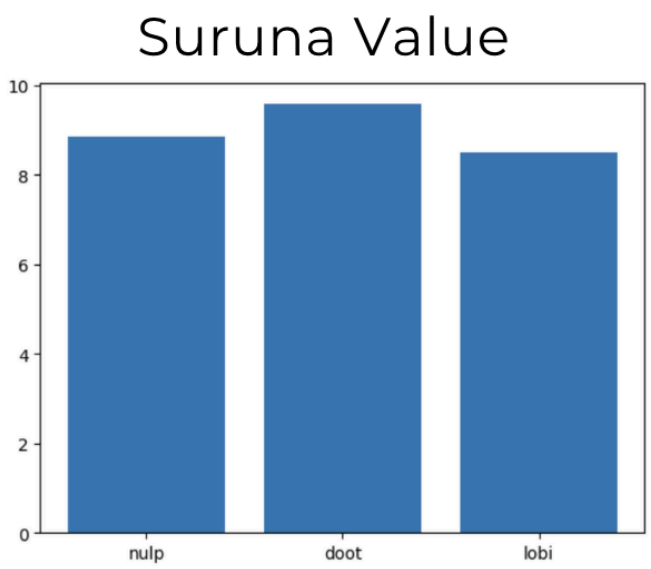

[Chart 4]



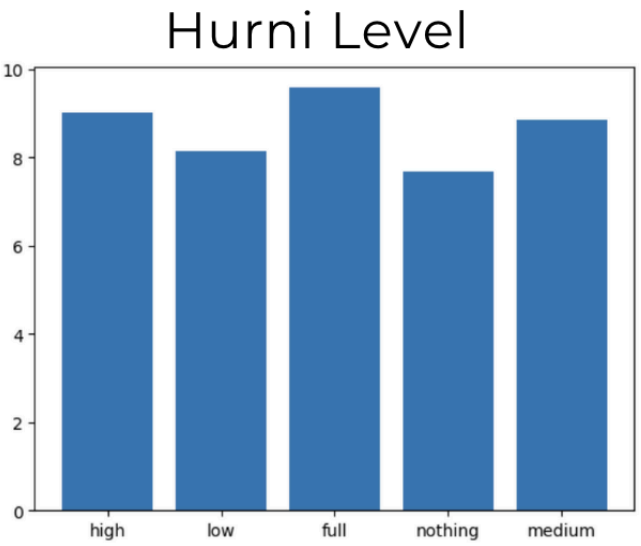

[Chart 5]



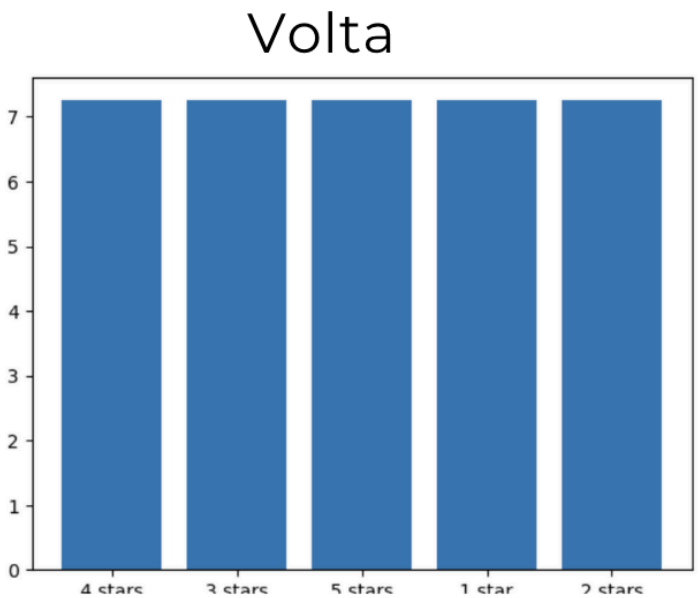

[Chart 6]



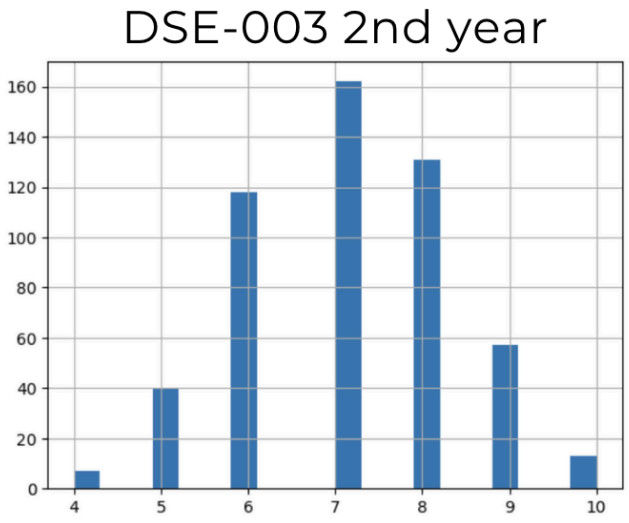

[Chart 7]



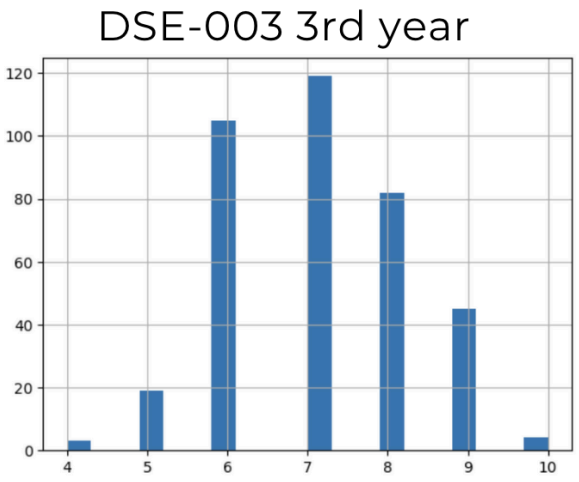

[Chart 8]



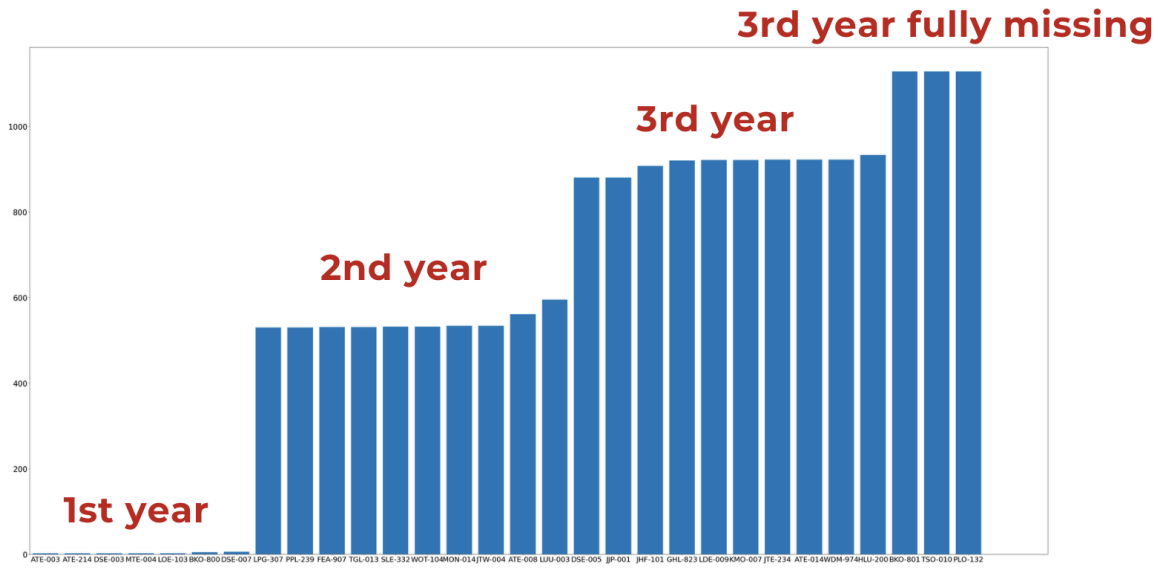

[Chart 9]



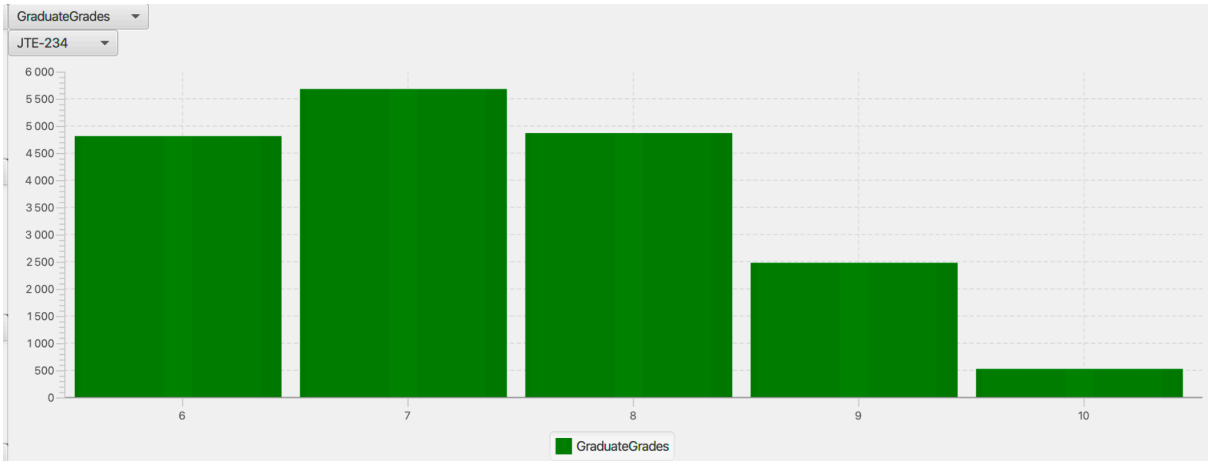

[Chart 10]



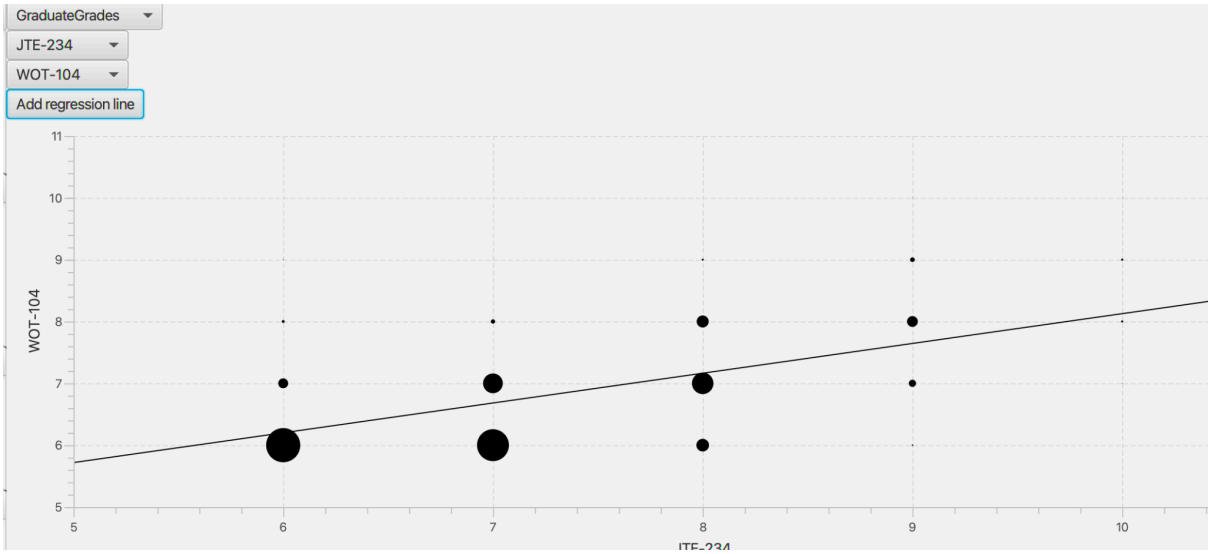

[Chart 11]



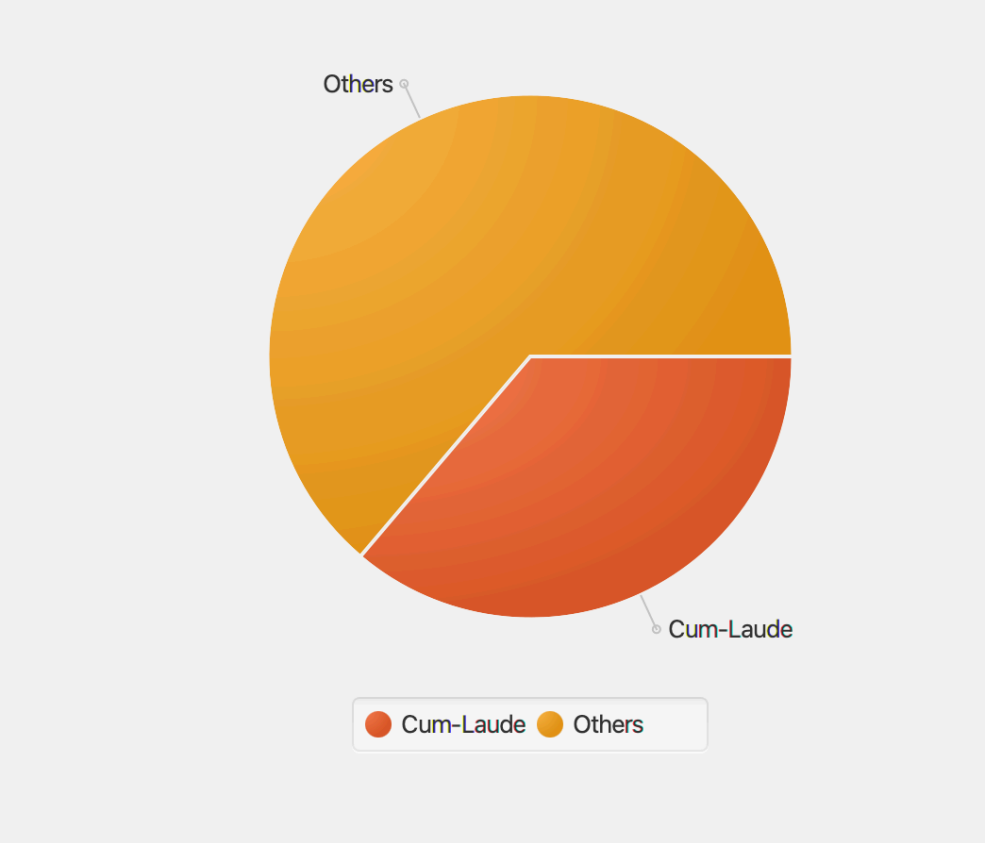

[Chart 12]



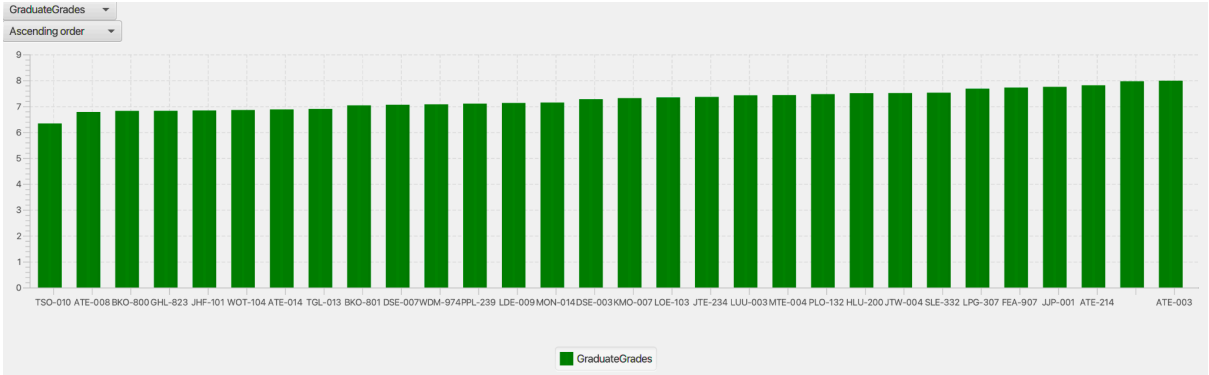

[Chart 13]




[Chart 14]




[Chart 15]

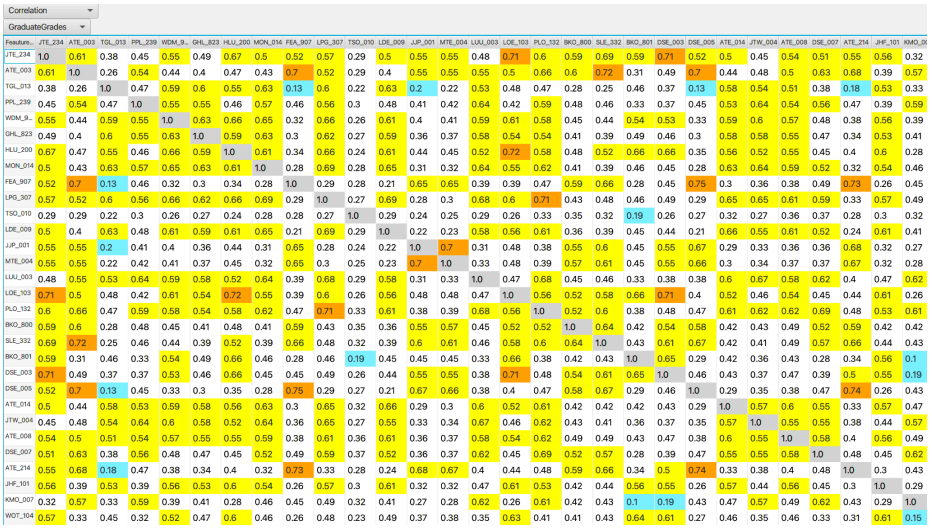



[Chart 16]

| Weighted Euclidean distances | ATE-003 | MON-014 | BKO-800 |
|---|---|---|---|
| Decision stump forests using 10 best splits | 0.03997641421222504 | 0.046447326811630926 | 0.030777198835001105 |
| Decision stump forests using 3 best splits | 0.04074617845248129 | 0.047967048292976704 | 0.0309949320338817 |
| Single-decision stumps | 0.028359498229991415 | 0.03191855453985822 | 0.02320836897422511 |

| | | | |
|---|---|---|---|
| using the best split | | | |

[Table 1]




| Featurn... | JTE_234 | ATE_003 | TGL_013 | PPL_239 | WDM_9... | GHL_823 | HLU_200 | MON_014 | FEA_907 | LPG_307 | TSO_010 | LDE_009 | JJP_001 | MTE_004 | LUU_003 | LOE_103 | PLO_332 | BKO_800 | SLE_332 | BKO_801 | DSE_003 | DSE_005 | ATE_014 | JTW_004 | ATE_008 | DSE_007 | ATE_214 | JHF_101 | KMO_0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JTE_234 | 1.0 | 0.61 | 0.38 | 0.45 | 0.55 | 0.49 | 0.67 | 0.5 | 0.52 | 0.57 | 0.29 | 0.5 | 0.55 | 0.55 | 0.48 | 0.71 | 0.6 | 0.59 | 0.69 | 0.59 | 0.71 | 0.52 | 0.5 | 0.45 | 0.54 | 0.51 | 0.55 | 0.56 | 0.32 |
| ATE_003 | 0.61 | 1.0 | 0.26 | 0.54 | 0.44 | 0.4 | 0.47 | 0.43 | 0.7 | 0.52 | 0.29 | 0.4 | 0.55 | 0.55 | 0.55 | 0.5 | 0.66 | 0.6 | 0.72 | 0.31 | 0.49 | 0.7 | 0.44 | 0.48 | 0.5 | 0.63 | 0.68 | 0.39 | 0.57 |
| TGL_013 | 0.38 | 0.26 | 1.0 | 0.47 | 0.59 | 0.6 | 0.55 | 0.63 | 0.13 | 0.6 | 0.22 | 0.63 | 0.2 | 0.22 | 0.53 | 0.48 | 0.47 | 0.28 | 0.25 | 0.46 | 0.37 | 0.13 | 0.58 | 0.54 | 0.51 | 0.38 | 0.18 | 0.53 | 0.33 |
| PPL_239 | 0.45 | 0.54 | 0.47 | 1.0 | 0.55 | 0.55 | 0.46 | 0.57 | 0.46 | 0.56 | 0.3 | 0.48 | 0.41 | 0.42 | 0.64 | 0.42 | 0.59 | 0.48 | 0.46 | 0.33 | 0.37 | 0.45 | 0.53 | 0.64 | 0.54 | 0.56 | 0.47 | 0.39 | 0.59 |
| WDM_9... | 0.55 | 0.44 | 0.59 | 0.55 | 1.0 | 0.63 | 0.66 | 0.65 | 0.32 | 0.66 | 0.26 | 0.61 | 0.4 | 0.41 | 0.59 | 0.61 | 0.58 | 0.45 | 0.44 | 0.54 | 0.53 | 0.33 | 0.59 | 0.6 | 0.57 | 0.48 | 0.38 | 0.56 | 0.39 |
| GHL_823 | 0.49 | 0.4 | 0.6 | 0.55 | 0.63 | 1.0 | 0.59 | 0.63 | 0.3 | 0.62 | 0.27 | 0.59 | 0.36 | 0.37 | 0.58 | 0.54 | 0.54 | 0.41 | 0.39 | 0.49 | 0.46 | 0.3 | 0.58 | 0.58 | 0.55 | 0.47 | 0.34 | 0.53 | 0.41 |
| HLU_200 | 0.67 | 0.47 | 0.55 | 0.46 | 0.66 | 0.59 | 1.0 | 0.61 | 0.34 | 0.66 | 0.24 | 0.61 | 0.44 | 0.45 | 0.52 | 0.72 | 0.58 | 0.48 | 0.52 | 0.66 | 0.66 | 0.35 | 0.56 | 0.52 | 0.55 | 0.45 | 0.4 | 0.6 | 0.28 |
| MON_014 | 0.5 | 0.43 | 0.63 | 0.57 | 0.65 | 0.63 | 0.61 | 1.0 | 0.28 | 0.69 | 0.28 | 0.65 | 0.31 | 0.32 | 0.64 | 0.55 | 0.62 | 0.41 | 0.39 | 0.46 | 0.45 | 0.28 | 0.63 | 0.64 | 0.59 | 0.52 | 0.32 | 0.54 | 0.46 |
| FEA_907 | 0.52 | 0.7 | 0.13 | 0.46 | 0.32 | 0.3 | 0.34 | 0.28 | 1.0 | 0.29 | 0.28 | 0.21 | 0.65 | 0.65 | 0.39 | 0.39 | 0.47 | 0.59 | 0.66 | 0.28 | 0.45 | 0.75 | 0.3 | 0.36 | 0.38 | 0.49 | 0.73 | 0.26 | 0.45 |
| LPG_307 | 0.57 | 0.52 | 0.6 | 0.56 | 0.66 | 0.62 | 0.66 | 0.69 | 0.29 | 1.0 | 0.27 | 0.69 | 0.28 | 0.3 | 0.68 | 0.6 | 0.71 | 0.43 | 0.48 | 0.46 | 0.49 | 0.29 | 0.65 | 0.65 | 0.61 | 0.59 | 0.33 | 0.57 | 0.49 |
| TSO_010 | 0.29 | 0.29 | 0.22 | 0.3 | 0.26 | 0.27 | 0.24 | 0.28 | 0.28 | 0.27 | 1.0 | 0.29 | 0.24 | 0.25 | 0.29 | 0.26 | 0.33 | 0.35 | 0.32 | 0.19 | 0.26 | 0.27 | 0.32 | 0.27 | 0.36 | 0.37 | 0.28 | 0.3 | 0.32 |
| LDE_009 | 0.5 | 0.4 | 0.63 | 0.48 | 0.61 | 0.59 | 0.61 | 0.65 | 0.21 | 0.69 | 0.29 | 1.0 | 0.22 | 0.23 | 0.58 | 0.56 | 0.61 | 0.36 | 0.39 | 0.45 | 0.44 | 0.21 | 0.66 | 0.55 | 0.61 | 0.52 | 0.24 | 0.61 | 0.41 |
| JJP_001 | 0.55 | 0.55 | 0.2 | 0.41 | 0.4 | 0.36 | 0.44 | 0.31 | 0.65 | 0.28 | 0.24 | 0.22 | 1.0 | 0.7 | 0.31 | 0.48 | 0.38 | 0.55 | 0.6 | 0.45 | 0.55 | 0.67 | 0.29 | 0.33 | 0.36 | 0.36 | 0.68 | 0.32 | 0.27 |
| MTE_004 | 0.55 | 0.55 | 0.22 | 0.42 | 0.41 | 0.37 | 0.45 | 0.32 | 0.65 | 0.3 | 0.25 | 0.23 | 0.7 | 1.0 | 0.33 | 0.48 | 0.39 | 0.57 | 0.61 | 0.45 | 0.55 | 0.66 | 0.3 | 0.34 | 0.37 | 0.37 | 0.67 | 0.32 | 0.28 |
| LUU_003 | 0.48 | 0.55 | 0.53 | 0.64 | 0.59 | 0.58 | 0.52 | 0.64 | 0.39 | 0.68 | 0.29 | 0.58 | 0.31 | 0.33 | 1.0 | 0.47 | 0.68 | 0.45 | 0.46 | 0.33 | 0.38 | 0.38 | 0.6 | 0.67 | 0.58 | 0.62 | 0.4 | 0.47 | 0.62 |
| LOE_103 | 0.71 | 0.5 | 0.48 | 0.42 | 0.61 | 0.54 | 0.72 | 0.55 | 0.39 | 0.6 | 0.26 | 0.56 | 0.48 | 0.48 | 0.47 | 1.0 | 0.56 | 0.52 | 0.58 | 0.66 | 0.71 | 0.4 | 0.52 | 0.46 | 0.54 | 0.45 | 0.44 | 0.61 | 0.26 |
| PLO_332 | 0.6 | 0.66 | 0.47 | 0.59 | 0.58 | 0.54 | 0.58 | 0.62 | 0.47 | 0.71 | 0.33 | 0.61 | 0.38 | 0.39 | 0.68 | 0.56 | 1.0 | 0.52 | 0.6 | 0.38 | 0.48 | 0.47 | 0.61 | 0.62 | 0.62 | 0.69 | 0.48 | 0.53 | 0.61 |
| BKO_800 | 0.59 | 0.6 | 0.28 | 0.48 | 0.45 | 0.41 | 0.48 | 0.41 | 0.59 | 0.43 | 0.35 | 0.36 | 0.55 | 0.57 | 0.45 | 0.52 | 0.52 | 1.0 | 0.64 | 0.42 | 0.54 | 0.58 | 0.42 | 0.43 | 0.49 | 0.52 | 0.59 | 0.42 | 0.42 |
| SLE_332 | 0.69 | 0.72 | 0.25 | 0.46 | 0.44 | 0.39 | 0.52 | 0.39 | 0.66 | 0.48 | 0.32 | 0.39 | 0.6 | 0.61 | 0.46 | 0.58 | 0.6 | 0.64 | 1.0 | 0.43 | 0.61 | 0.67 | 0.42 | 0.41 | 0.49 | 0.57 | 0.66 | 0.44 | 0.43 |
| BKO_801 | 0.59 | 0.31 | 0.46 | 0.33 | 0.54 | 0.49 | 0.66 | 0.46 | 0.28 | 0.46 | 0.19 | 0.45 | 0.45 | 0.45 | 0.33 | 0.66 | 0.38 | 0.42 | 0.43 | 1.0 | 0.65 | 0.29 | 0.42 | 0.36 | 0.43 | 0.28 | 0.34 | 0.56 | 0.1 |
| DSE_003 | 0.71 | 0.49 | 0.37 | 0.37 | 0.53 | 0.46 | 0.66 | 0.45 | 0.45 | 0.49 | 0.26 | 0.44 | 0.55 | 0.55 | 0.38 | 0.71 | 0.48 | 0.54 | 0.61 | 0.65 | 1.0 | 0.46 | 0.43 | 0.37 | 0.47 | 0.39 | 0.5 | 0.55 | 0.19 |
| DSE_005 | 0.52 | 0.7 | 0.13 | 0.45 | 0.33 | 0.3 | 0.35 | 0.28 | 0.75 | 0.29 | 0.27 | 0.21 | 0.67 | 0.66 | 0.38 | 0.4 | 0.47 | 0.58 | 0.67 | 0.29 | 0.46 | 1.0 | 0.29 | 0.35 | 0.38 | 0.47 | 0.74 | 0.26 | 0.43 |
| ATE_014 | 0.5 | 0.44 | 0.58 | 0.53 | 0.59 | 0.58 | 0.56 | 0.63 | 0.3 | 0.65 | 0.32 | 0.66 | 0.29 | 0.3 | 0.6 | 0.52 | 0.61 | 0.42 | 0.42 | 0.42 | 0.43 | 0.29 | 1.0 | 0.57 | 0.6 | 0.55 | 0.33 | 0.57 | 0.47 |
| JTW_004 | 0.45 | 0.48 | 0.54 | 0.64 | 0.6 | 0.58 | 0.52 | 0.64 | 0.36 | 0.65 | 0.27 | 0.55 | 0.33 | 0.34 | 0.67 | 0.46 | 0.62 | 0.43 | 0.41 | 0.36 | 0.37 | 0.35 | 0.57 | 1.0 | 0.55 | 0.55 | 0.38 | 0.44 | 0.57 |
| ATE_008 | 0.54 | 0.5 | 0.51 | 0.54 | 0.57 | 0.55 | 0.55 | 0.59 | 0.38 | 0.61 | 0.36 | 0.61 | 0.36 | 0.37 | 0.58 | 0.54 | 0.62 | 0.49 | 0.49 | 0.43 | 0.47 | 0.38 | 0.6 | 0.55 | 1.0 | 0.58 | 0.4 | 0.56 | 0.49 |
| DSE_007 | 0.51 | 0.63 | 0.38 | 0.56 | 0.48 | 0.47 | 0.45 | 0.52 | 0.49 | 0.59 | 0.37 | 0.52 | 0.36 | 0.37 | 0.62 | 0.45 | 0.69 | 0.52 | 0.57 | 0.28 | 0.39 | 0.47 | 0.55 | 0.55 | 0.58 | 1.0 | 0.48 | 0.45 | 0.62 |
| ATE_214 | 0.55 | 0.68 | 0.18 | 0.47 | 0.38 | 0.34 | 0.4 | 0.32 | 0.73 | 0.33 | 0.28 | 0.24 | 0.68 | 0.67 | 0.4 | 0.44 | 0.48 | 0.59 | 0.66 | 0.34 | 0.5 | 0.74 | 0.33 | 0.38 | 0.4 | 0.48 | 1.0 | 0.3 | 0.43 |
| JHF_101 | 0.56 | 0.39 | 0.53 | 0.39 | 0.56 | 0.53 | 0.6 | 0.54 | 0.26 | 0.57 | 0.3 | 0.61 | 0.32 | 0.32 | 0.47 | 0.61 | 0.53 | 0.42 | 0.44 | 0.56 | 0.55 | 0.26 | 0.57 | 0.44 | 0.56 | 0.45 | 0.3 | 1.0 | 0.29 |
| KMO_007 | 0.32 | 0.57 | 0.33 | 0.59 | 0.39 | 0.41 | 0.28 | 0.46 | 0.45 | 0.49 | 0.32 | 0.41 | 0.27 | 0.28 | 0.62 | 0.26 | 0.61 | 0.42 | 0.43 | 0.1 | 0.19 | 0.43 | 0.47 | 0.57 | 0.49 | 0.62 | 0.43 | 0.29 | 1.0 |
| WOT_104 | 0.57 | 0.33 | 0.45 | 0.32 | 0.52 | 0.47 | 0.6 | 0.46 | 0.26 | 0.48 | 0.23 | 0.49 | 0.37 | 0.38 | 0.35 | 0.63 | 0.41 | 0.41 | 0.43 | 0.64 | 0.61 | 0.27 | 0.46 | 0.35 | 0.46 | 0.33 | 0.31 | 0.61 | 0.15 |

[Table 2]

| Year | Target | Best Accuracy |
|---|---|---|
| 1 | 1 | 0.636514 |
| 1 | 13 | 0.744387 |
| 1 | 15 | 0.718088 |
| 1 | 17 | 0.730785 |
| 1 | 20 | 0.736409 |
| 1 | 25 | 0.701093 |
| 1 | 26 | 0.631681 |
| 2 | 2 | 0.626694 |
| 2 | 3 | 0.741318 |
| 2 | 7 | 0.662418 |
| 2 | 8 | 0.654961 |
| 2 | 9 | 0.661307 |
| 2 | 14 | 0.651546 |
| 2 | 18 | 0.690532 |
| 2 | 23 | 0.656024 |
| 2 | 24 | 0.761829 |
| 2 | 29 | 0.653449 |
| 3 | 0 | 0.801961 |
| 3 | 4 | 0.696339 |
| 3 | 5 | 0.705903 |
| 3 | 6 | 0.648398 |
| 3 | 11 | 0.683546 |
| 3 | 12 | 0.681392 |
| 3 | 21 | 0.733067 |
| 3 | 22 | 0.734419 |
| 3 | 27 | 0.755438 |
| 3 | 28 | 0.606965 |
| Missing | 10 | 0.811790 |
| Missing | 16 | 0.650655 |
| Missing | 19 | 0.657860 |

[Table 3]