Politecnico di Torino

PREDICTING STUDENTS' PERFORMANCE

Using Supervised Machine Learning Models

Abstract

A 5-level classification to predict final year results of students, based on information about lifestyle and midterm results.

COMO CLAUDIO

S278006

Contents

1 Introduction

In this project a dataset of students from a Portugal high school it is used to predict their studying performance based on secondary elements as habits, family background, etc…

To classify the data a 5 Level classification has been used based on the Erasmus grade convention system and several machine learning models to find out the best one to fit the problem.

Education it's a key problem in Portugal as it is in Italy. Find a way to understand what influence the students would be a good way to prevents students' failure or to reduce the dropout rate.

Country	I (excellent/very good)	II (good)	III (satisfactory)	IV (sufficient)	V (fail)
Portugal	16-20	14-15	12-13	10-11	0-9

Five Level Classification system

2 Students Data

2.1 Overview

Several countries as the Portugal adopt a 20-point grading scale evaluation, where 0 is the lowest grade and 20 is the perfect score. During the school year, students are evaluated with 3 tests, the final one corresponds to the final grade (G3 attribute). These data have been collected during the 2005-2006 school year from two public schools, from the Alentjo region of Portugal. The database come from 2 source, one, based on information provided by the school (like G3, G2, G1, or number school absence), the other one is based on questionaries for students. The dataset has 33 attributes and 649 instances. The original dataset includes 2 subsets: Portuguese (649 instances) and Math (395 instances) but 382 instances belong to both subsets. For this reason, just the Portuguese subset has been considered, the one with greater number of instances.

2.2 Attributes

Let's inspect each attribute:

Attribute	Description
School	student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
Sex	student's sex (binary: "F" - female or "M" - male)
Age	student's age (numeric: from 15 to 22)
Address	student's home address type (binary: "U" - urban or "R" - rural)
Famsize	family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
Pstatus	parent's cohabitation status (binary: "T" - living together or "A" - apart)
Medu	mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
Fedu	father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
Mjob	mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
Fjob	father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
Reason	reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
Guardian	guardian - student's guardian (nominal: "mother", "father" or "other")
Traveltime	home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
Studytime	studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

Failures	failures - number of past class failures (numeric: n if 1<=n<3, else 4)
Schoolsup	extra educational support (binary: yes or no)
Famsup	family educational support (binary: yes or no)
Paid	extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
Activities	extra-curricular activities (binary: yes or no)
Nursery	attended nursery school (binary: yes or no)
Higher	wants to take higher education (binary: yes or no)
Internet	Internet access at home (binary: yes or no)
Romantic	with a romantic relationship (binary: yes or no)
Famrel	quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
Freetime	free time after school (numeric: from 1 - very low to 5 - very high)
Goout	going out with friends (numeric: from 1 - very low to 5 - very high)
Dalc	workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
Walc	weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
Health	current health status (numeric: from 1 - very bad to 5 - very good)
Abscences	number of school absences (numeric: from 0 to 93)
G1	first period grade (numeric: from 0 to 20)
G2	second period grade (numeric: from 0 to 20)

G3	final grade (numeric: from 0 to 20, output target)

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	13	13

5 rows × 33 columns

FIGURE 2.1: DATASET DESCRIBE

3 Data Visualization and Exploration

This part concerns the visualization of the data. Important steps are look for null values, correlation matrix for attributes, presence of outliers and data distribution in bins made up by the 5 levels classification.

3.1 Check info and null value

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64
 7   Fedu        649 non-null    int64
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64
 13  studytime   649 non-null    int64
 14  failures    649 non-null    int64
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
```

FIGURE 3.1: DATASET INFO

```
df.isnull().sum()

school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

FIGURE 3.2: NULL VALUES CHECK

This dataset hasn't null values as it is possible to see in Fig. 3.1. With Fig. 3.2 it is possible to have a better idea of instances and attribute type.

3.2 Correlation Matrix

Using correlation matrix is a good way to understand how every attribute is correlated to another one. An attribute can be positively correlated or negatively correlated to another one. In a range from -1 to 1, an attribute is highly (positively or negatively) correlated if the value generated with the other attribute is closer to 1 or -1. To generate the correlation matrix, it is needed to convert every categorical attribute to numbers, this can be achieved using the **OrdinalEncoder()** form **scikit-learn library**, this encoder give an incremental number to every unique value of an attribute. In this way even binary value are converted correctly to 0,1. A copy of the dataset is created to pursuit these pre-processing operations.

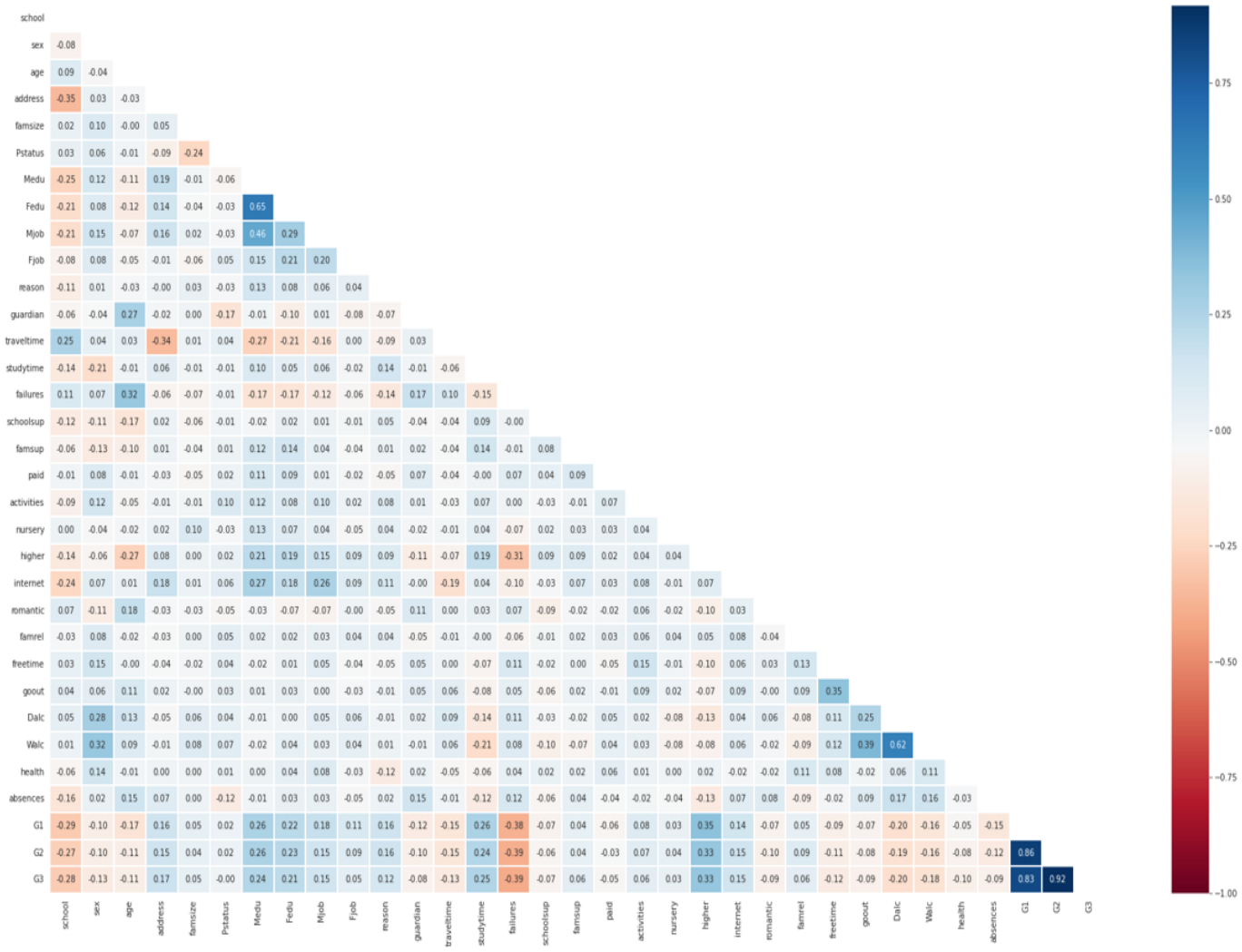

FIGURE 3.3: CORRELATION MATRIX

In Fig. 3.3 it is possible to note that “G1” and “G2” with “G3” attribute are highly positively correlated, this mean that they will influence in a meaningful way the result on classification model, other interesting attributes are “Medu”, “Fedu”, “Studytime” and “Higher”, these attributes are positively correlated with “G3” so they can impact in a good way during the classification, those attributes are positively correlated even with “G1” and “G2”, so it seems they have an impact during all the scholastic year. “Dalc”, “Walc” and “Traveltime” are negatively correlated with “G3”. The assumption of alcohol can be a serious problem and can influence the final grade result. As the travel time form school to home. When it’s higher can makes difficult manage the time in a right way to do homework and other stuff.

Lastly, we can see that “School” and “Failures” attribute are negatively correlated with “G3”, for the failures it makes sense, for school it can be maybe related to the quality of the school?

Some other information that we can take from the correlation matrix are the slightly high correlation between “Fedu” and “Medu”, not so important for our study but curious. And the negative correlation between “Higher” and “Failures”, even this obvious but interesting.

3.3 Data distributions with bins

To visualize our data distribution we need, first, to add a new column to our dataset (“final_grade_evaluation”) that convert G3 range votes in 5 bins, one per classification level. After that we can plot the G3 distribution and the 5 levels distribution.

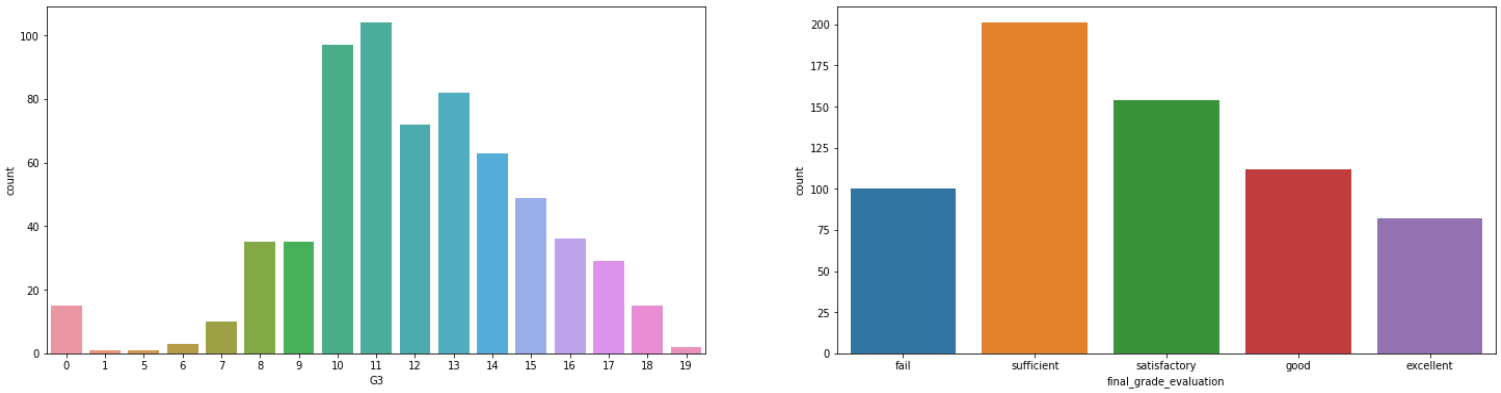

FIGURE 3. 4: DISTRIBUTION BY G3 ON LEFT AND BY FINAL GRADE ON RIGHT

As it is possible to see in Fig. 3.4 the distribution of our data in G3 and final_grade_evaluation is gaussian and data are slightly unbalanced as it is possible to observe in Fig. 5. For these reasons is not needed to oversample the dataset.

sufficient	30.970724
satisfactory	23.728814
good	17.257319
fail	15.408320
excellent	12.634823

FIGURE 6

3.4 Outliers detection

Dataset can have some data points that differs significantly from other observations, these are known as outliers. These data points can cause serious problems to our predictive models but is even important to understand if a data point occurred for a measurement error, so it can be discarded, or is due to a heavy-tiled distribution so it is needed to be cautious on choose the rights tool to process our data.

To detect outliers on our dataset non-categorial attributes have been taken in consideration using a violin plot. These attributes are “Age”, “Absences” and “G3”.

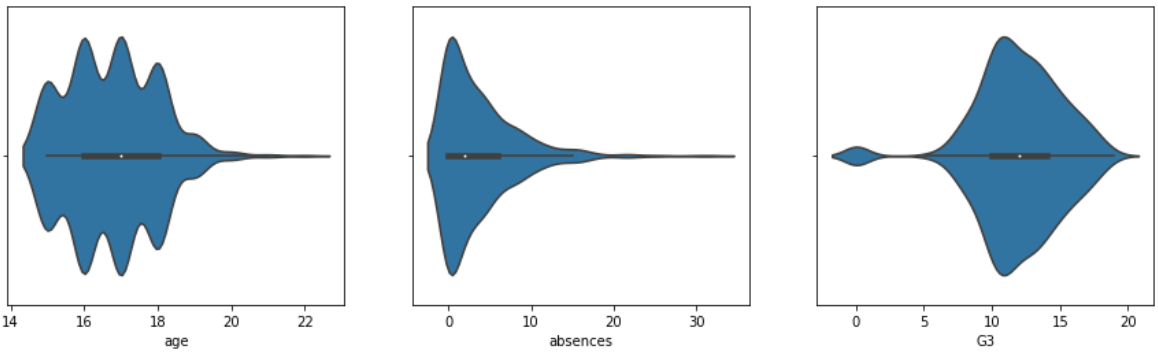

FIGURE 3.5

In Fig.3.5 it is possible to observe that large part of the data are distributed between 0 to 15 for absences, 15 to 20 for age and 5 to 20 for G3. But some other data points are still present and far form the range of distribution. These datapoints can be considered outliers.

Considering that this dataset has not many data and outliers for age and absences seems not so influence, it well be deleted just data points with G3=0 because it means that students maybe didn't understand the subject at all.

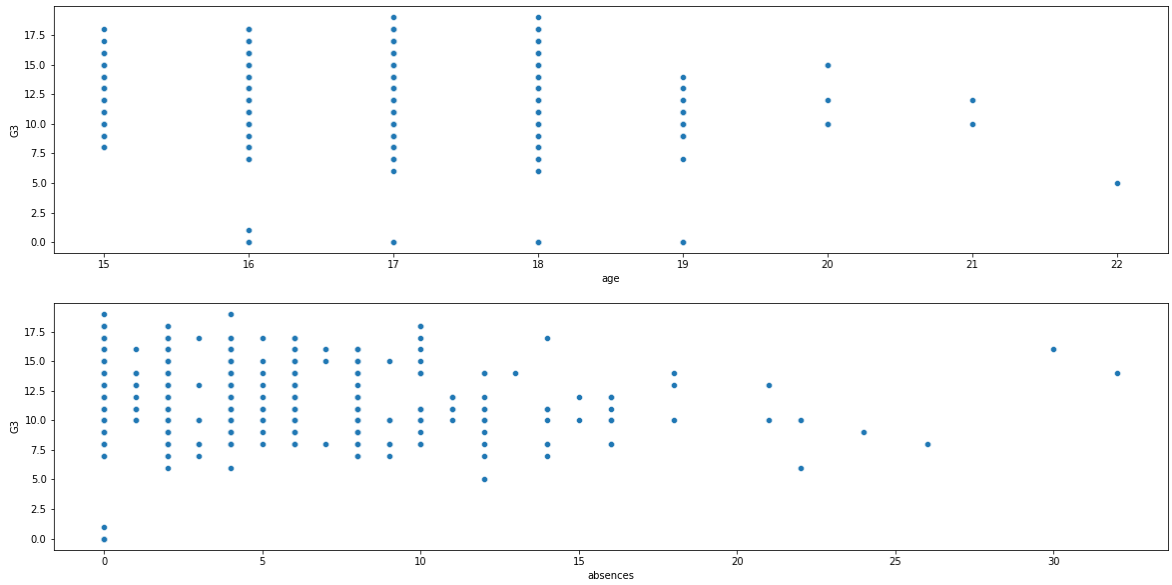

FIGURE 3.6

4 Feature Extraction

4.1 Splitting and scaling

To perform the following operation the dataset it is splitted in **train_set** and **test_set** with a proportion of 80/20. Then, categorical attributes are converted in numeric ones. There are two types of categorical number: binary and nominal. Former are converted with **OrdinalEncoder()**, to have them in a binary numerical form, latter are converted with a function of **pandas** called **get_dummies()** which convert every unique value of the attribute in a binary form. Before doing these operations the 'final_grade_evaluation' column and 'G3' are dropped from the dataset, the former is saved as label, the latter is dropped because the values for the label are generated from this attribute.

Now, all the data are in a numeric form, looking to these numbers it can be possible to notice that they are not in the same range, for this reason they must be scaled. There are more type of scalers, like **MinMaxScaler()** or **StandardScaler()**. The latter one is used for these data. This method standardizes in a way that the transformed feature has mean 0 and standard deviation of 1. The standard score for the feature is calculated as:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Where '**x**' is the feature's value, $\bar{\mathbf{x}}$ the mean and **σ** the standard deviation.

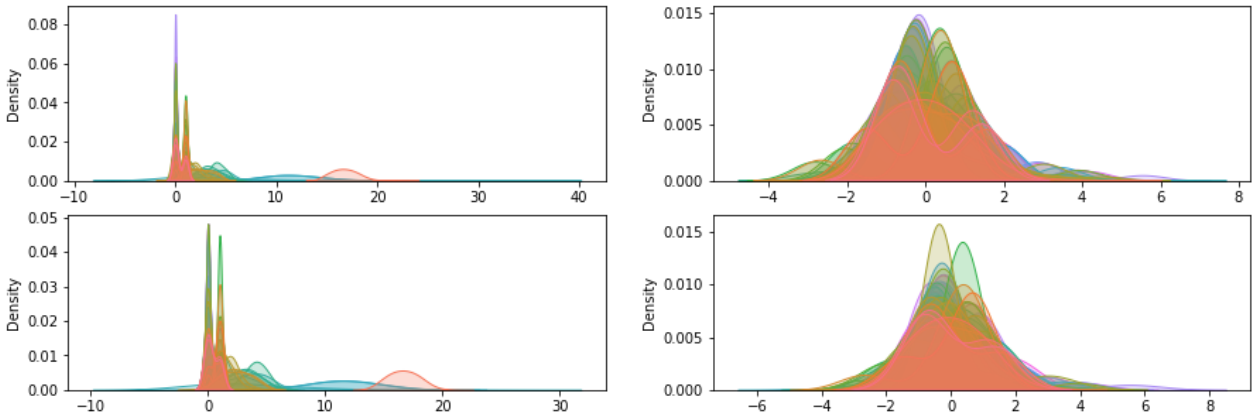

FIGURE 4.1

In Fig.4.1 the first raw represent the density distribution of train set before and after the scaling, the second raw is the test set one.

4.2 PCA vs LDA

One of the challenging problems of machine learning is dealing with large amounts of sample or features. Some variables can be redundant, correlated or not relevant at all. Here it comes to help 2 dimensionality reduction algorithms, **Principal Component Analysis** (**PCA**) and **Linear Discriminant Analysis** (**LDA**). In this project both algorithms have been used and compared to find the best one, even if this dataset is not so large.

First, we should talk about what is PCA and how it works. It's an unsupervised reduction method, it reduces the features into a smaller subset of orthogonal variables, called principal component, linear combinations of the original variables. The main idea of PCA is to maximize the data's variability while reducing the dataset's dimensionality.

Instead, LDA is a supervised machine learning and linear algebra approach for dimensionality reduction. It is used for classification tasks since the class label is known. LDA finds the linear discriminant to maximize the variance between the different categories while minimizing the variance within the class.

The main difference between these two approaches is that PCA is an unsupervised method, so he has no idea about class labels, instead LDA take the labels into account, and it assumes that data corresponding to a class follows a Gaussian distribution with a common variance and different mean. For this reason, it's sensitive to outliers.

To compare these 2 methods first we need to find the number of principal components to select. To do so we fix a threshold of explainable variance at 80%. Then, we create a dataframe where the cumulative explained variance corresponds to a certain quantity, after that we create a filter with our threshold, and we apply it to the dataframe taking the first row that is equal or greater than 80%.

In this project we defined 2 datasets for classification that will be explained later, one is called A, the other is called B.

On PCA we had these results:

A)

```
25 Components capture 81.64% of variability of the data
```

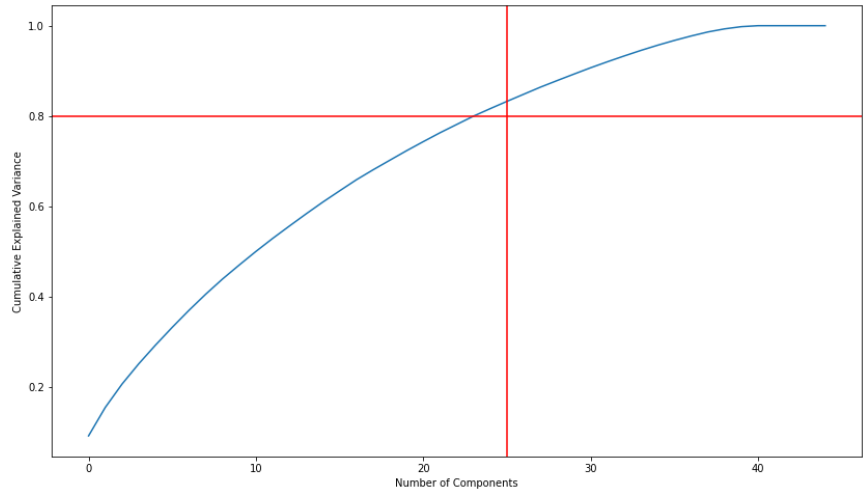

FIGURE 4.2

B)

```
24 Components capture 80.77% of variability of the data
```

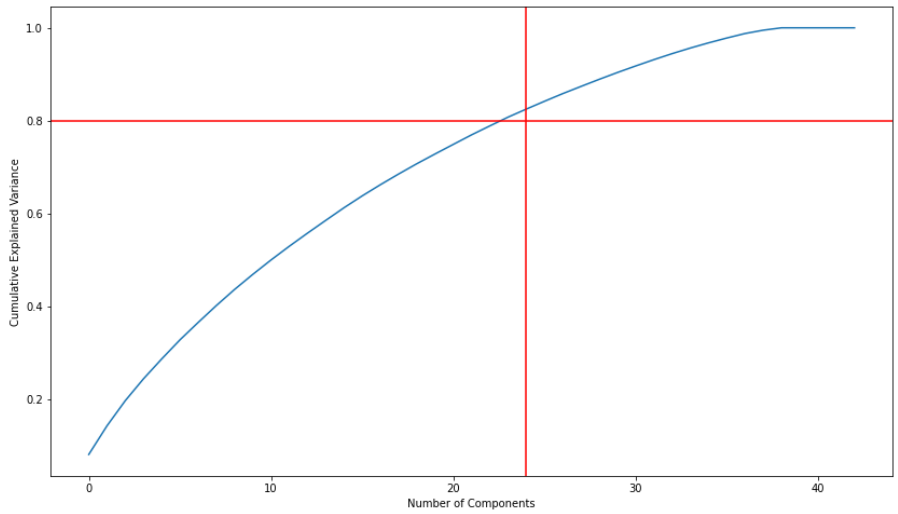

FIGURE 4.3

Then we reduce the train data set using those components and we check how much data variance each principal component explains, as we can see in Fig. 4.4 and 4.5 the first 2 components explain 8%, or more in the case of train A, of the total variability.

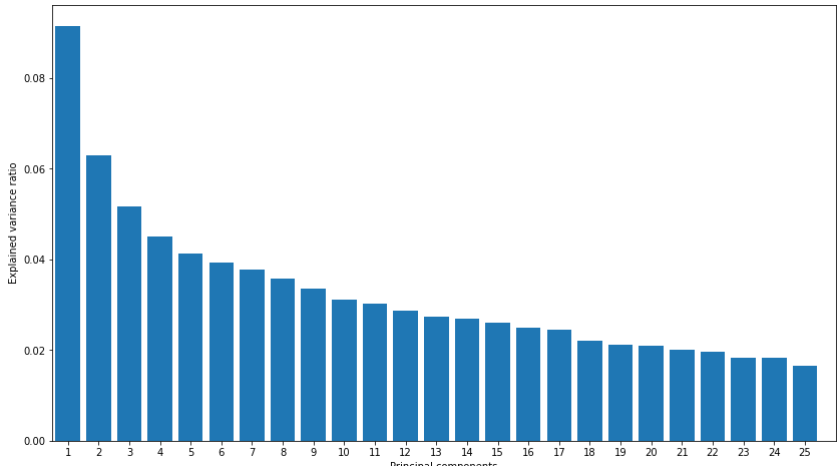

FIGURE 4.4: TRAIN A

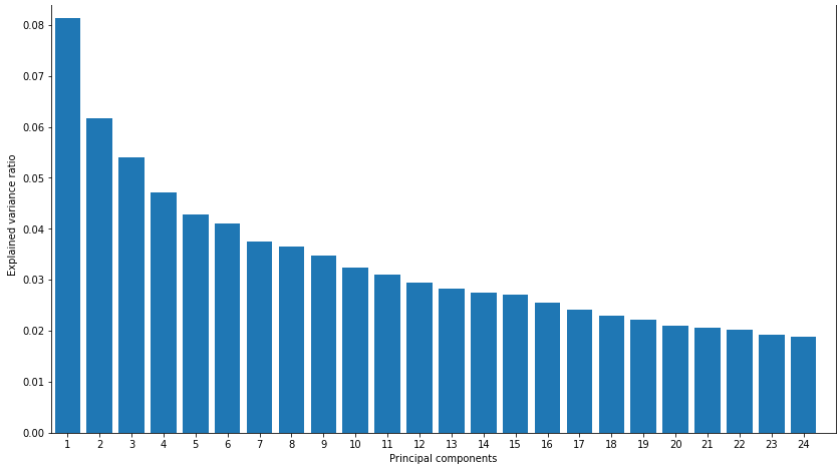

FIGURE 4.5: TRAIN B

In Fig. 4.4 and Fig. 4.5 we can see that large part of the data variance is explained in the first principal component, less in the second one and so on until the last one.

At this point it can be useful plot the data to visualize their distribution

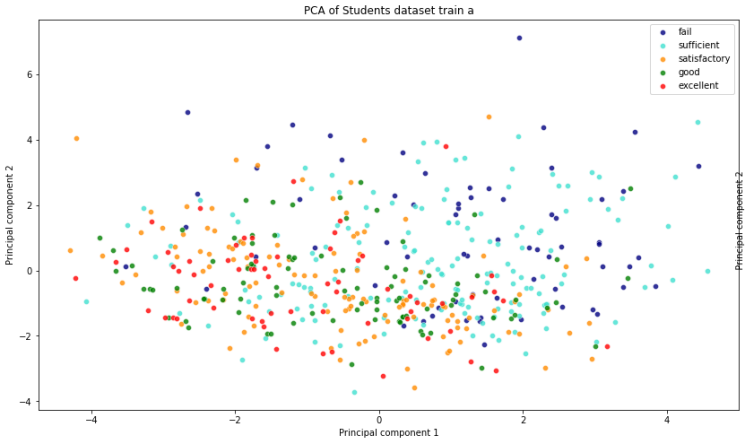

FIGURE 4.6: TRAIN A, 2 PC

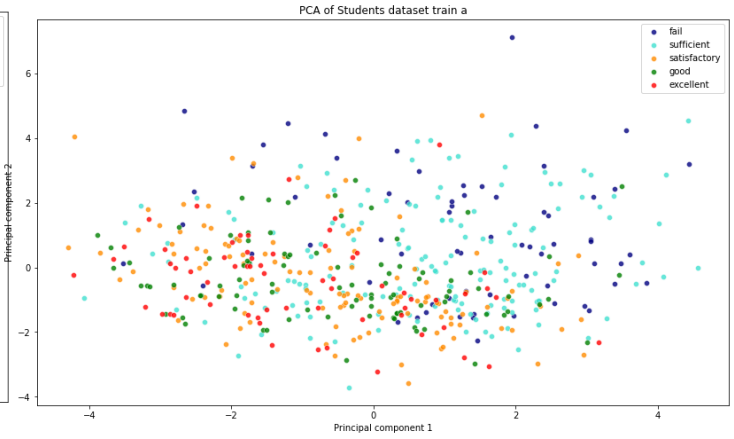

FIGURE 4.7: TRAIN B, 2 PC

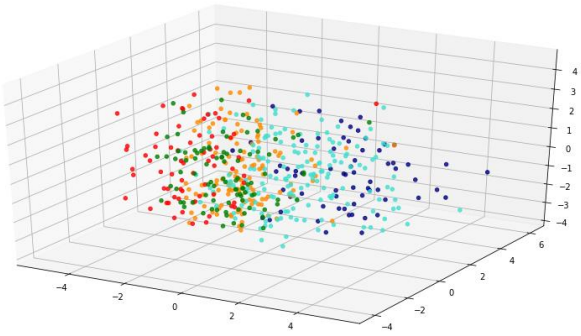

FIGURE 4.8: TRAIN A, 3 PC

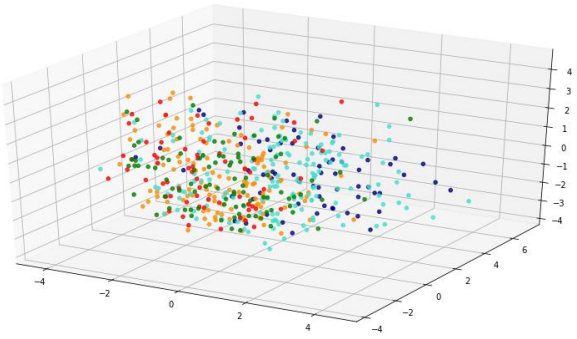

FIGURE 4.9: TRAIN B, 3 PC

Unfortunately, with PCA is not easier to distinguish every class, let's try with LDA.

The procedure is the same as PCA one, the difference here is that number of components can't be more than n.classes – 1, so in our case the max number of components is 4 :

A)

```
1 Discriminant Components explain 94.51% of variability between
CLASSES
```

B)

```
2 Discriminant Components explain 87.13% of variability between
CLASSES
```

We apply LDA with the number of components that we found, and we plot the distribution of the data:

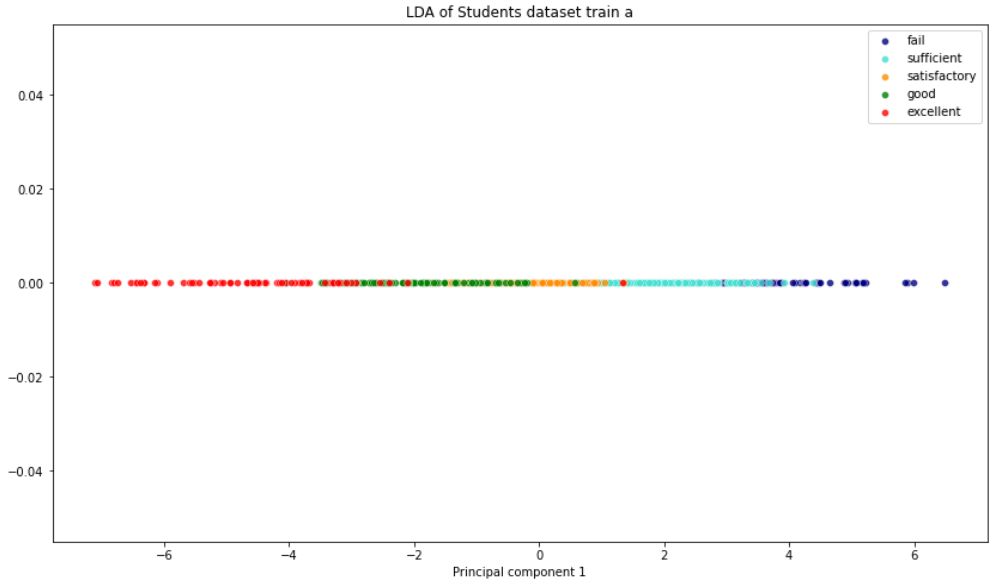

FIGURE 4.10: TRAIN A, LDA 1 PC

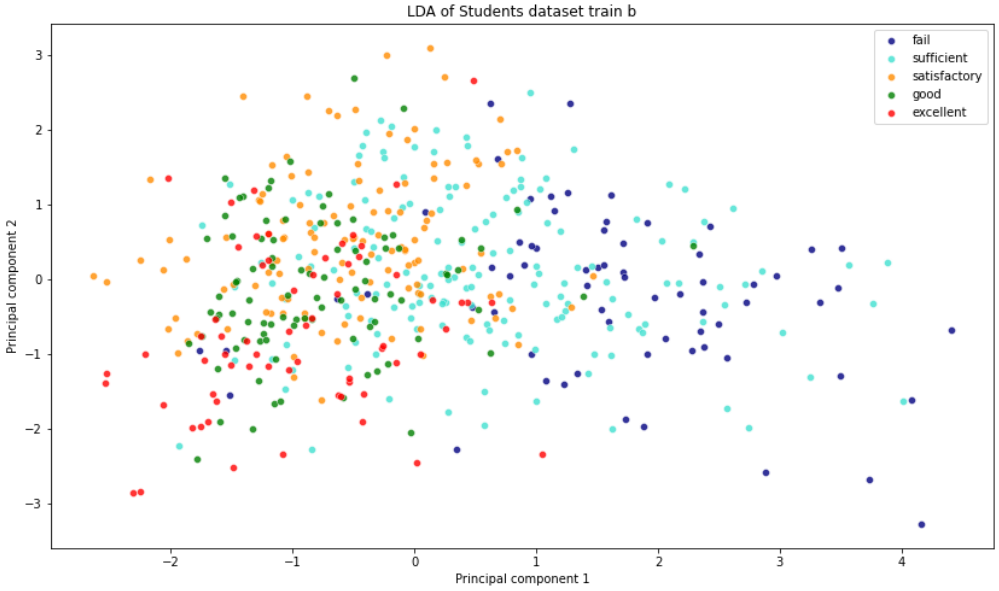

FIGURE 4.11: TRAIN B, LDA 2 PC

In Fig. 4.10 we see how data seems better classified with some overlapped points for training set A, in Fig. 4.11 the distribution of the data points is still not clear and pronounced.

Considering the 2 algorithms we decided to apply LDA on both training set because it seems perform really well respect to PCA and speed up the models that we will use in the project.

During the project we will compare the model's result with and without using LDA.

5 Classification

As mentioned before in this project 2 configuration will be used to classify our data:

1. Configuration A: with all attributes except G3
2. Configuration B: same as A without G1 and G2

The idea is to understand if even without information about school results it can be possible to predict which result the student will achieve at the end of the year.

The dataset has been split before with a random state, also known as seed, of 170. This number is needed to randomly shuffle the data and have a similar distribution between training and testing set.

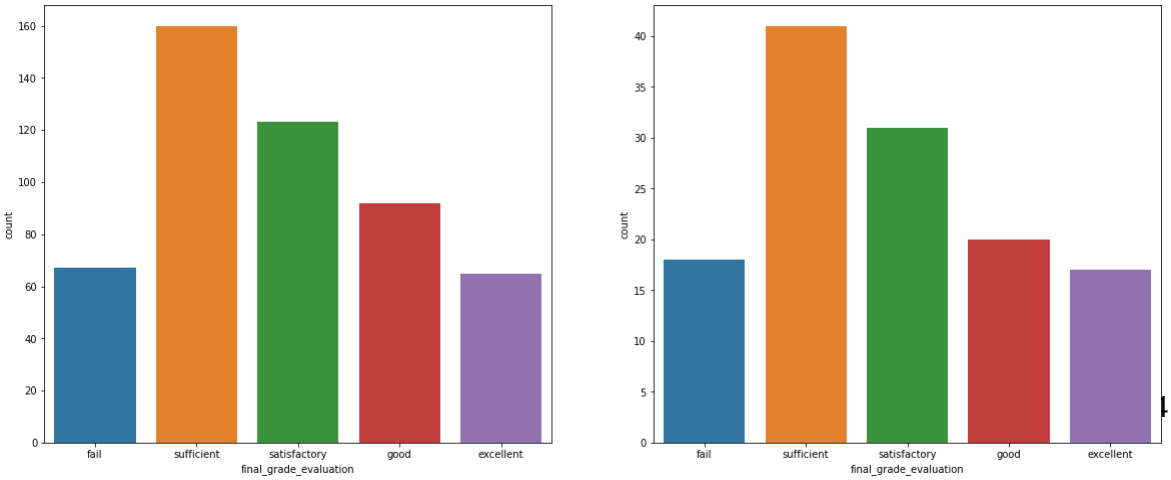

FIGURE 5.1: TRAINING SET ON LEFT AND TESTING SET ON RIGHT

Except of ‘good evaluation’ data points the distribution seems quite similar.

This section will show how the machine learning models are used to solve the classification task and we will compare in the next section their result using the most common evaluation metrics. But first, we need to explain the parameters used and their meaning:

- **True Positive (TP):** number of times the model predicts true, and the target value is true
- **True Negative (TN):** number of times the model predicts false, and the target value is false
- **False Positive (FP):** number of times the model predicts true, and the target value is false
- **False Negative (FN):** number of times the model predicts false, and the target value is true

- **Accuracy:** it defines how accurate is our model; it is measured as: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** it defines how many times our model correctly predicted a positive value over the total of positive predictions; it is measured as: $\frac{TP}{TP+FP}$
- **Recall:** it defines how many of our actual positive case we were able to predict; it is measured as: $\frac{TP}{TP+FN}$
- **F1 score:** combine Precision and Recall and it is at maximum when Precision is equal to Recall; it is measured as: $2 * \frac{Precision*Recall}{Precision+Recall}$

These equations are for binary classification, but they can be adapted for multiclassification as in our case adding the value for each class.

5.1 Multinomial Logistic Regression

Logistic Regression is a technique used when the dependent variable is categorical (or nominal). For Binary Logistic Regression the number of dependent variables is two, for multinomial logistic regression is more than two.

In Logistic Regression, a logistic function is used to modelling the probability that data belong to a specific class:

$$p(X) = \frac{e^{(a+b1X1+b2X2+b3X3+\cdots)}}{1 + e^{(a+b1X1+b2X2+b3X3+\cdots)}}$$

Where p(X) is the probability that a case is in a particular category, ‘a’ the constant of the equation and ‘b’ the coefficient of the predictor or independent variable.

In our case we use the Multinomial Logistic Regression to predict membership of 5 classes. It works basically in the same way as binary logistic regression but in this case the analysis breaks the outcome variable down into a series of comparison between two categories.

To apply the Multinomial Logistic Regression it has been used the **LogisticRegression()** class offered by **Scikit-learn** with setting 'multi_class = multinomial' and 'solver=lbfgs' as parameters. In combination with **Reapeted-StratifiedKfold()** class to perform a k-fold cross validation. Splitting the train set in 5 subset and performing validating 3 times.

The came out for both configurations are:

Test metrics Configuration A with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.61	1	0.75	-
Sufficient	0.93	0.76	0.84	-
Good	0.52	0.64	0.57	-
Satisfactory	0.65	0.48	0.55	-
Excellent	0.76	0.93	0.84	-
Tot.	0.69	0.76	0.71	0.71

Test metrics Configuration B with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.11	0.4	0.17	-
Sufficient	0.49	0.38	0.43	-
Good	0.48	0.3	0.37	-
Satisfactory	0.05	0.08	0.06	-
Excellent	0.06	0.14	0.08	-
Tot.	0.23	0.26	0.22	0.31

Test metrics Configuration A without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.61	0.85	0.71	-
Sufficient	0.78	0.74	0.76	-
Good	0.61	0.53	0.57	-
Satisfactory	0.55	0.55	0.55	-
Excellent	0.70	0.8	0.75	-
Tot.	0.65	0.69	0.67	0.67

Test metrics Configuration B without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.28	0.56	0.37	-
Sufficient	0.46	0.48	0.47	-
Good	0.52	0.33	0.41	-
Satisfactory	0.25	0.23	0.24	-
Excellent	0.11	0.25	0.16	-
Tot.	0.32	0.37	0.33	0.37

The mean train accuracy for configuration A with LDA is 76.7% and without 62.5%, for configuration B with LDA 44.8 % and without LDA 33.3%.

Configuration A as expected perform almost twice better than configuration B.

5.2 Support Vector Machine (SVM)

Support Vector Machine is a Supervised Machine Learning Algorithm used for classification and/or regression. Basically, SVM finds a hyperplane that create a boundary between the types of data. In 2-dimensional space, this hyperplane is a line. SVM plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next find the optimal hyperplane to separate the data. SVM should be more indicated for a binary classification, but it can be used even for Multi-Class classification, in this case it will be create a binary classifier for each class of data. The result for each classifier will be 'datapoint belong or NOT to the class'.

Data not always are linearly separable, but Kernalized SVM work well with this kind of data. In fact, SVM can map each data point in a higher dimension point until they can be linearly separable. To understand how similar are datapoints in the new space, kernel

functions are used. The interesting thing is that SVM doesn't actually need to transform each point to compute this calculation, it does the calculation by a similarity function that is actually the kernel. Important parameters for SVM are:

- The Kernel type: it is selected based on the data and the type of Transformation. By default, use Radial Basis Function Kernel
- Gamma: this parameter decides how far the influence of a single training example reaches during training, which in turn affects how tightly the decision boundaries end up surrounding points in the input space.
- The 'C' parameter: this parameter controls the amount of regularization applied to the data. Large values mean low regularization, less tolerant of errors. Lower values mean high regularization and more toleration of errors. Used to generate the Soft Margin SVM.

The kernel type, C and gamma are all hyperparameters. It means they can be changed to find the best model to fit the data.

To perform this algorithm it has been used **SVC() class** and hyperparameters were tuned with **GridSearchCV()** class which by default use a cross validation with 5 folders.

Value used to tune the hyperparameters:

- Kernel: linear, rbf;
- C: 0.1, 1, 10, 100, 1000
- γ: 1, 0.1, 0.01, 0.001, 0.0001

Here the results:

Test metrics Configuration A with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.61	1	0.76	-
Sufficient	0.93	0.76	0.84	-
Good	0.67	0.65	0.67	-
Satisfactory	0.6	0.6	0.6	-
Excellent	0.76	0.93	0.84	-
Tot.	0.71	0.79	0.74	0.75

Test metrics Configuration B with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.22	0.5	0.31	-
Sufficient	0.51	0.41	0.46	-
Good	0.42	0.31	0.36	-
Satisfactory	0.15	0.15	0.15	-
Excellent	0.06	0.17	0.09	-
Tot.	0.27	0.31	0.27	0.33

Test metrics Configuration A without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.61	0.79	0.69	-
Sufficient	0.73	0.71	0.72	-
Good	0.51	0.55	0.53	-
Satisfactory	0.65	0.52	0.58	-
Excellent	0.71	0.71	0.71	-
Tot.	0.64	0.66	0.65	0.65

Test metrics Configuration B without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0	0	0	-
Sufficient	0.51	0.40	0.45	-
Good	0.52	0.31	0.39	-
Satisfactory	0.2	0.24	0.22	-
Excellent	0.06	0.2	0.09	-
Tot.	0.26	0.23	0.23	0.33

Hyperparameters and mean train accuracy with LDA:

A) Kernel= 'rbf', C=10, γ=0.1, mean train accuracy: 78.2%
B) Kernel = 'rbf', C=1000, γ=0.1, mean train accuracy: 33.1%

Hyperparameters and mean train accuracy without LDA:

A) Kernel= 'linear', C=10, γ=0.1, mean train accuracy: 65.6%
B) Kernel = 'rbf', C=10, γ=0.001, mean train accuracy: 36.6%

Overall SVM perform better with LDA especially for configuration A. On configuration B perform as without LDA in terms of accuracy but has some slight improvement on Recall, Precision and F1 score.

5.3 K-Nearest Neighbours (KNN)

K-nearest neighbours is a type of supervised machine learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then selects the K number points which is closet to the test data and the most common class among them it's assigned to the test-data. Usually, to calculate the distance it is used the Euclidian Distance (minkowski distance = 2). The hardest part when we use this algorithm is to choose a right value for K, because a smaller value of K can bring our model to be more sensitive to noise, it can reach a good accuracy on training set but a poor one on the test set. If K is higher the model can be too much generalized and prone to overfitting. To search a good value for K in the project it has been used **GridSearchCV()** as in SVM model, in this case try to choose a good value in a range from 1 to 101.

Test metrics Configuration A with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.61	1	0.76	-
Sufficient	0.93	0.76	0.83	-
Good	0.68	0.66	0.67	-
Satisfactory	0.6	0.6	0.6	-
Excellent	0.76	0.93	0.84	-
Tot.	0.72	0.79	0.74	0.75

Test metrics Configuration B with LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.22	0.36	0.28	-
Sufficient	0.36	0.39	0.38	-
Good	0.35	0.28	0.31	-
Satisfactory	0.25	0.19	0.21	-
Excellent	0.06	0.08	0.07	-
Tot.	0.25	0.26	0.25	0.28

Test metrics Configuration A without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0.06	1	0.11	-
Sufficient	0.57	0.58	0.57	-
Good	0.61	0.34	0.44	-
Satisfactory	0.25	0.22	0.23	-
Excellent	0.29	0.71	0.42	-
Tot.	0.35	0.57	0.35	0.42

Test metrics Configuration B without LDA

Class	Recall	Precision	F1_score	Accuracy
Fail	0	0	0	-
Sufficient	0.44	0.38	0.41	-
Good	0.61	0.28	0.38	-
Satisfactory	0.2	0.36	0.26	-
Excellent	0	0	0	-
Tot.	0.25	0.20	0.21	0.32

Result with LDA:

A) K = 45, mean train accuracy: 78.7%
B) K = 27, mean train accuracy: 47.5%

Result without LDA:

A) K=23, mean train accuracy: 44.8%
B) K=100, mean train accuracy: 36.3 %

Even in this case both configurations perform better with LDA on almost all the metrics, except for Accuracy in configuration B.

5.4 Decision Tree

Decision tree is one of the predictive models used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about item's target value. Decision tree model can be used for regression in case we have continuous value or for classification in presence of discrete values. In our case we will use a classification tree.

In a classification tree we predict that each observation belongs to the most commonly occurring class of training observation in the region to which belongs.

To do so, Decision Tree employs a divide and conqueror strategy by conducting a greedy search to identify the optimal split point within a tree. This process of binary splitting is then repeated in a top-down manner until all, or most records have been classified under a specific class label.

To perform the binary splitting two criterion can be used: 'Gini-index' and 'cross-entropy'.

The Gini Index is defined by:

$$G = \sum_{k=1}^{K} pmk(1 - pmk)$$

A measure of total variance across the K classes. Where *pmk* represent the portion of training observations in the mth region that are from the *kth* class. The Gini index takes on a small value if all the *pmk*'s are close to zero or one. For this reason, is referred as a measure of node-purity, a small value indicates that node contains predominantly observations from a single class.

The cross entropy is measured as:

$$D = -\sum_{k=1}^{K} pmk \log(pmk)$$

Like in the Gini index, smaller is the value purer is *mth* node.

The high risk of using this algorithm is bound to the overfitting of the model, because this classifier can adapt too much to the test set and have a bad accuracy on the test set, for this reason some prune methods exist.

To apply this algorithm it has been used the **DecisionTreeClassifier()** class and the **GridSearchCV()** to find the best hyperparameters.

For the tree methods it will be used the pure train and test set, without scaling and LDA.

Hyperparameters:

- Criterion: gini, entropy;
- Max_depth: all values in the range [1,30];
- Min_samples_split: all values in the range [2,30]

Test metrics Configuration A

Class	Recall	Precision	F1_score	Accuracy
Fail	0.5	1	0.67	-
Sufficient	0.98	0.74	0.84	-
Good	0.77	0.72	0.75	-
Satisfactory	0.65	0.72	0.68	-
Excellent	0.76	1	0.87	-
Tot.	0.73	0.84	0.76	0.78

Test metrics Configuration B

Class	Recall	Precision	F1_score	Accuracy
Fail	0.39	0.43	0.41	
Sufficient	0.34	0.48	0.4	
Good	0.42	0.38	0.4	
Satisfactory	0.35	0.18	0.24	
Excellent	0.12	0.2	0.15	
Tot.	0.32	0.34	0.32	0.34

Hyperparameters chosen and mean train accuracy:

A) Criterion: entropy, max_depth=3, min_sample_split=2, mean train accuracy 75,7%
B) Criterion: entropy, max_depth=12, min_sample_split=12, mean train accuracy 35,7%

Great result for configuration A and B and contrary to predictions seems the model is not overfitted.

5.5 Random Forest

Random forest is a supervised learning algorithm that grows and combines multiple decision tree to create a "forest". It can be used for both regression and classification.

The logic behind the Random Forest is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone. When using Random Forest each tree gives a classification, the forest chooses the most given classification. Usually, this algorithm performs better than decision tree because every tree in the forest is uncorrelated, so the forest is less prone to overfitting. Moreover, every tree is built with a random subset of features. This gives more possibility even to that features that have less impact in a classic decision tree. This method is called "bagging method" and it is a type of ensemble machine learning algorithm called Bootstrap Aggregation.

To apply this model we used the **RandomForestClassifer()** class in combination with **GridSearchCV()** to tune the hyperparameters

Hyperparameters:

- Criterion: 'gini','entropy';
- Max_depth: in a range of [1,30];
- Min_samples_split: in a range of [2,30];
- N_estimators: range(10, 101, 10);
- Max_features: sqrt, log2;

Test metrics Configuration A

Class	Recall	Precision	F1_score	Accuracy
Fail	0.39	0.88	0.54	-
Sufficient	0.95	0.68	0.80	-
Good	0.61	0.68	0.64	-
Satisfactory	0.65	0.62	0.63	-
Excellent	0.76	1	0.87	-
Tot.	0.67	0.77	0.70	0.72

Test metrics Configuration B

Class	Recall	Precision	F1_score	Accuracy
Fail	0.11	0.67	0.19	-
Sufficient	0.63	0.43	0.51	-
Good	0.48	0.33	0.39	-
Satisfactory	0.1	0.15	0.12	-
Excellent	0.06	0.17	0.09	-
Tot.	0.28	0.35	0.26	0.36

Hyperparameters and mean train accuracy:

A) Criterion: gini, max_depth = 11, max_features = log2, min_samples_split = 9, n_estimators= 30, mean train accuracy 73.8%
B) Criterion: entropy, max_depth=16, max_features=sqrt, min_samples_split=20, n_estimators=80, mean train accuracy 43%

6 Conclusion

Configuration A test and train results compare

Model	Recall	Precision	F1_score	Accuracy Test	Accuracy Train
LogisticReg	0.69	0.76	0.71	0.71	76.7%
	0.65	0.69	0.67	0.67	62.5%
SVM	0.71	0.79	0.74	0.75	78.2%
	0.64	0.66	0.65	0.65	65.6%
K-NN	0.72	0.79	0.74	0.75	**78.7%**
	0.35	0.57	0.35	0.42	44.8%
DecisionTree	**0.73**	**0.84**	**0.76**	**0.78**	75.7%
RandomForest	0.67	0.77	0.70	0.72	73.8%

Configuration B test and train results compare

Model	Recall	Precision	F1_score	Accuracy Test	Accuracy Train
LogisticReg	0.23	0.26	0.22	0.31	44.8%
	0.32	**0.37**	**0.33**	**0.37**	33.3%
SVM	0.27	0.31	0.27	0.33	33.1%
	0.26	0.23	0.23	0.33	36.6%
K-NN	0.25	0.26	0.25	0.28	**47.5%**
	0.25	0.20	0.21	0.32	36.3%
DecisionTree	**0.32**	0.34	0.32	0.34	35.7%
RandmForest	0.28	0.35	0.26	0.36	43%

As expected, the configuration A performs better than Configuration B, thanks to the information given by attributes ‘G1’ and ‘G2’. The application of LDA gave a sensible boost on **Configuration A** results but at least the **Decision Tree** (trained with the original dataset) had the best result on almost every metrics. Even if we can see how SVM and KNN with LDA are really close to Decision Tree results. For configuration B all the results are not so good, seems that having no information about results during the school year is not enough to predict well which results will achieve the student at the end of the year. The best result for **Configuration B** is obtained by **Logistic Regression** without LDA in terms of test. Instead, about the training set, KNN with LDA had the best train accuracy but, considering the results on test set probably the model overfitted a bit.

7 References:

P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.