MACHINE LEARNING ASSIGNEMENT

Student Alcohol Consumption & Grades in High School

Colin LEVERGER - colinleverger@gmail.com
Anaïs GALISSON - anais.enssat@gmail.com

ENSSAT Informatique, Multimédia et Réseaux
Promotion 2017

Recipient : John D. KELLEHER

May 27, 2016

Contents

1 Dataset Description

1.1 Dataset introduction

The goal of this assessment is to work with machine learning to build some prediction models. Some comparisons will be made between models efficiency and we will be able to choose the best one.

We will study here the alcohol consumption in relation with the grades of students. The goal is to predict the third grade of the year for a student. The predicted grade is supposed to be between 0 and 20. Here, we can notice that we may use two paradigms to treat this problem. We can consider that the grade we will predict is a continuous value; in this case, we can use a linear regression for prediction. Or, we can consider that predicting a grade between 0 and 20 comes down to classification with 21 rankings: 0,1, …,20. In this case, we can also reduce the rankings in order to keep the classification efficient (in the first rank, grades between 0 and 5, etc.)

The dataset provided contains 33 columns and approximately 400 rows. At first glance, we can say that there are not a lot of data, if we consider that some models can be fed with hundred thousand values.

The data consist in a description of each student's personal data, such as age, sex, address… It also includes data about their life environment: family environment (size of the family, parents' job, situation of the parents). Furthermore, there are some details about very personal data: is this student in a relationship, is he romantic, etc. Finally, there are data about the students' habits concerning parties and alcohol consumption.

To predict the grades, we also have in the dataset three grades that the students had during the year: G1, G2 and G3. The prediction we will provide should be on G3.

The dataset we have is very clean: no missing values, and everything is standardized… We won't even need to clean it up. The only thing we have to do on the data is to change categorical features in boolean (because scikit lib only use numbers). We will also work on reducing the rankings for the grades.

1.2 Focus on Features & Quick Observations

'*address*' and '*famsize*' features are encoded with a letter, which is mapped to a value. For example, '*famsize*' can be LE3 or GT3 (less or greater than 3).

We had a concern about four features, which were '*traveltime*', '*studytime*', '*medu*' and '*fedu*' (respectively travel time to school, study time at home, mother education, father education). We have realised that these features were a numeric mapping. For example, for the '*traveltime*' feature, 1 represents '*<15 min.*', 2 represents '*15 to 30 min.*', etc. We know that sometimes, it is not a good thing to have mapping values in features because scikit will interpret these features as ordered. After analysing the features, we saw that finally it was not a problem because all of the features considered are actually ordered.

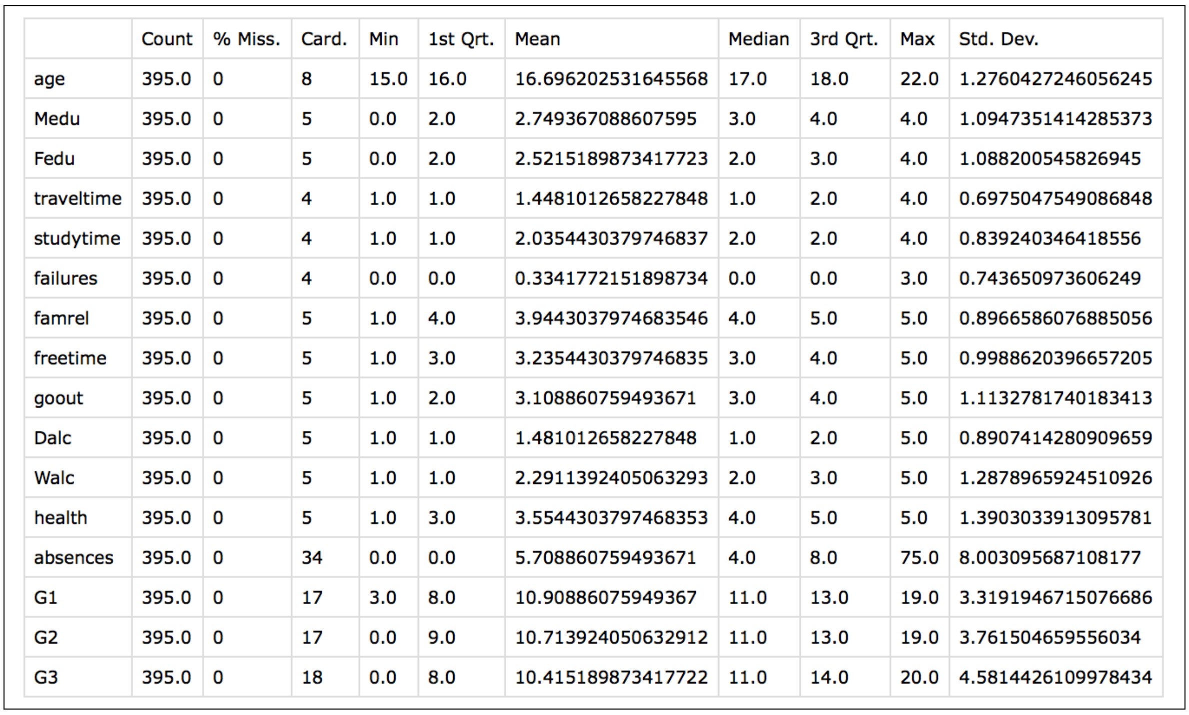
The '*failure*' feature has a high threshold of 4.

The '*traveltime*' feature could be put in relation to '*reason*' feature, if the home is close to the school. It could be interesting to see the correlation, and if it is high (>95) merge these two cols.

Final noticeable facts: there are a lot of students in bad health in this university!

2 Data Quality Report and Analysis

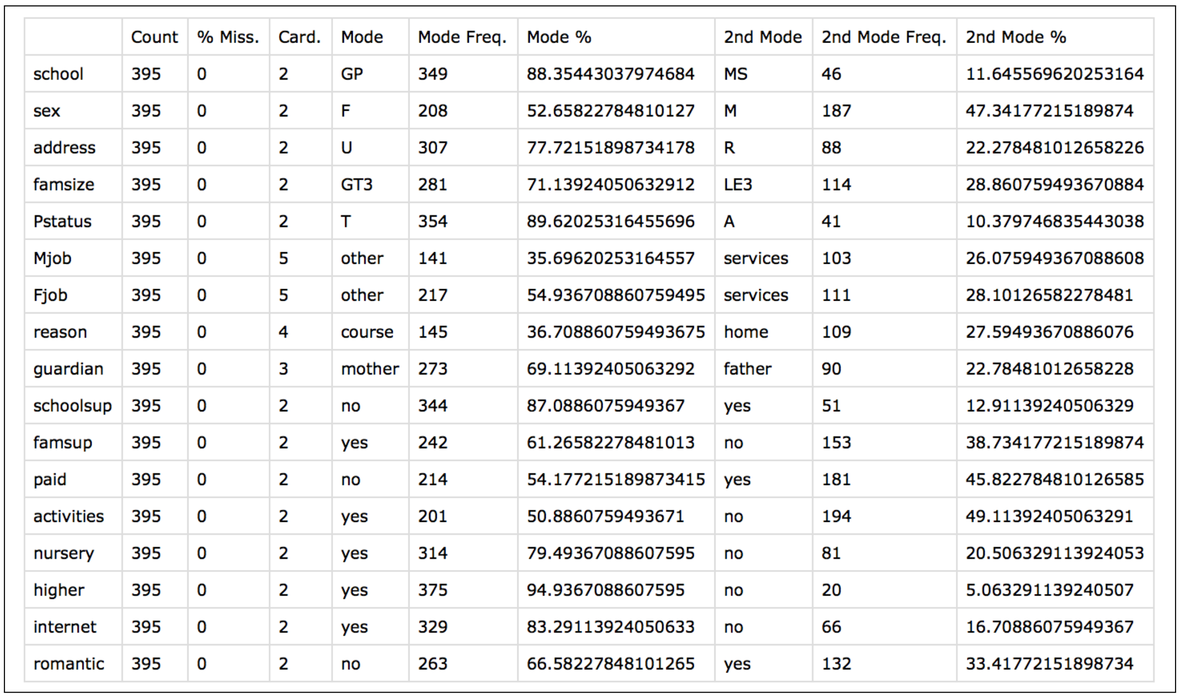
The data quality report we have generated can be found on the figures 1 and 2 respectively on page 2 and 3.



	Count	% Miss.	Card.	Min	1st Qrt.	Mean	Median	3rd Qrt.	Max	Std. Dev.
age	395.0	0	8	15.0	16.0	16.696202531645568	17.0	18.0	22.0	1.2760427246056245
Medu	395.0	0	5	0.0	2.0	2.749367088607595	3.0	4.0	4.0	1.0947351414285373
Fedu	395.0	0	5	0.0	2.0	2.5215189873417723	2.0	3.0	4.0	1.088200545826945
traveltime	395.0	0	4	1.0	1.0	1.4481012658227848	1.0	2.0	4.0	0.6975047549086848
studytime	395.0	0	4	1.0	1.0	2.0354430379746837	2.0	2.0	4.0	0.839240346418556
failures	395.0	0	4	0.0	0.0	0.3341772151898734	0.0	0.0	3.0	0.743650973606249
famrel	395.0	0	5	1.0	4.0	3.9443037974683546	4.0	5.0	5.0	0.8966586076885056
freetime	395.0	0	5	1.0	3.0	3.2354430379746835	3.0	4.0	5.0	0.9988620396657205
goout	395.0	0	5	1.0	2.0	3.108860759493671	3.0	4.0	5.0	1.1132781740183413
Dalc	395.0	0	5	1.0	1.0	1.481012658227848	1.0	2.0	5.0	0.8907414280909659
Walc	395.0	0	5	1.0	1.0	2.2911392405063293	2.0	3.0	5.0	1.2878965924510926
health	395.0	0	5	1.0	3.0	3.5544303797468353	4.0	5.0	5.0	1.3903033913095781
absences	395.0	0	34	0.0	0.0	5.708860759493671	4.0	8.0	75.0	8.003095687108177
G1	395.0	0	17	3.0	8.0	10.90886075949367	11.0	13.0	19.0	3.3191946715076686
G2	395.0	0	17	0.0	9.0	10.713924050632912	11.0	13.0	19.0	3.761504659556034
G3	395.0	0	18	0.0	8.0	10.415189873417722	11.0	14.0	20.0	4.5814426109978434

Figure 1: DQR: Continuous Features

In this report, we can first notice that everything seems to be clear and clean for the continuous data. Nothing is missing and there are apparently no strange data: as said before, the set provided here is nice and clean.



	Count	% Miss.	Card.	Mode	Mode Freq.	Mode %	2nd Mode	2nd Mode Freq.	2nd Mode %
school	395	0	2	GP	349	88.35443037974684	MS	46	11.645569620253164
sex	395	0	2	F	208	52.65822784810127	M	187	47.34177215189874
address	395	0	2	U	307	77.72151898734178	R	88	22.278481012658226
famsize	395	0	2	GT3	281	71.13924050632912	LE3	114	28.860759493670884
Pstatus	395	0	2	T	354	89.62025316455696	A	41	10.379746835443038
Mjob	395	0	5	other	141	35.69620253164557	services	103	26.075949367088608
Fjob	395	0	5	other	217	54.936708860759495	services	111	28.10126582278481
reason	395	0	4	course	145	36.708860759493675	home	109	27.59493670886076
guardian	395	0	3	mother	273	69.11392405063292	father	90	22.78481012658228
schoolsup	395	0	2	no	344	87.0886075949367	yes	51	12.91139240506329
famsup	395	0	2	yes	242	61.26582278481013	no	153	38.734177215189874
paid	395	0	2	no	214	54.177215189873415	yes	181	45.822784810126585
activities	395	0	2	yes	201	50.8860759493671	no	194	49.11392405063291
nursery	395	0	2	yes	314	79.49367088607595	no	81	20.506329113924053
higher	395	0	2	yes	375	94.9367088607595	no	20	5.063291139240507
internet	395	0	2	yes	329	83.29113924050633	no	66	16.70886075949367
romantic	395	0	2	no	263	66.58227848101265	yes	132	33.41772151898734

Figure 2: DQR: Categorical Features

Same interpretation can be made for the categorical features, nothing special to declare.

We have thought that it could be interesting to take a closer look at the relation between the fields '*reason*' and '*travel*' because they may be related: a student who lives 10 meters away from the school will probably choose this one rather than the one which is far away...

In conclusion to this DQR, we have a high-quality dataset and we did not identify noticeable problems.

3 Data Handling

First of all, in the dataset we received, we identified some of the continuous features that may be categorical. Indeed, the '*traveltime*', '*studytime*', '*Medu*', '*Fedu*' may be considered as categorical features because they are mapped values: 1 represents '*<15 minutes*' for '*traveltime*' field, for example.

Regarding the target which is the column '*G3*': It can have 20 different values so there are a lot of values. At first glance, we did not know if we could consider G3 as a categorical or continuous target because there are too many values to be considered as categorical and not enough to be considered as continuous.

So, we have decided to realise the prediction in two different ways:

- Classification for data with discrete labels
- Regression for data with continuous labels

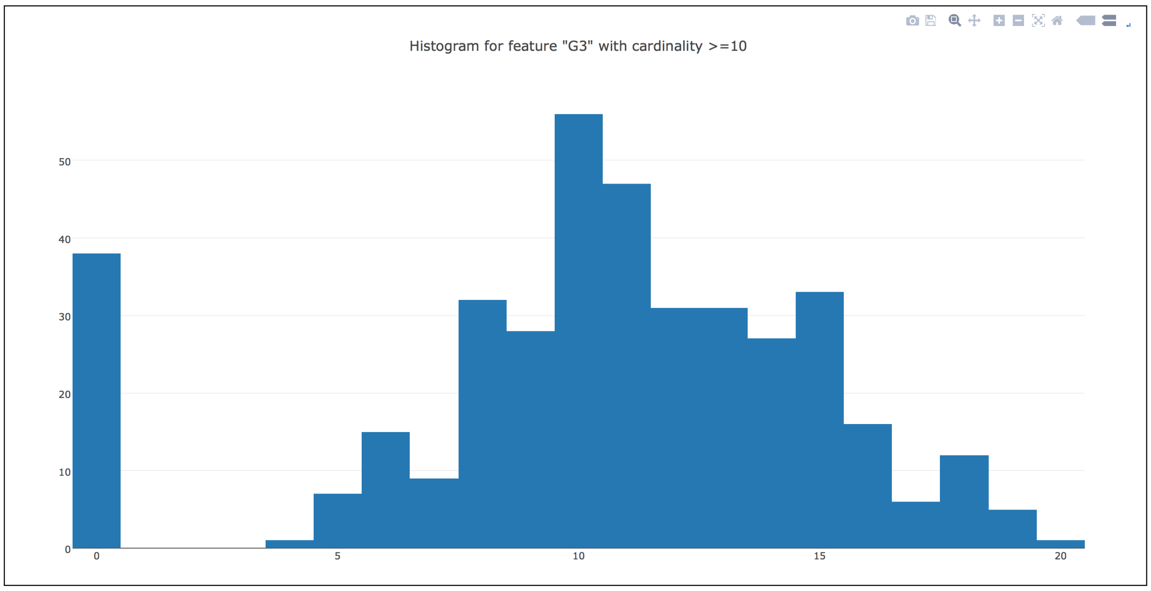


Figure 3: Histogram for G3 feature (normally distributed)

For the classification, we have decided to reduce to 5 categories instead of 21. It will highly reduce complexity, and the accuracy of the algorithm will be much better. For example, with 21 categories, a first naive algorithm gave us an accuracy of > 40%, and with 5 categories an accuracy of > 65%! So, we have used a 'Frequency Distribution' to map the grades. In the mapping, we have called the first category '1', the second '2', etc.

Thanks to the histogram above (see 3 on page 4), we have noticed that the distribution of the grades looks like the normal distribution. In fact, there are more people who have a mark between 8 and 12 than 15–20 or 1–5. The only problem is the 0 because there are nearly 40 persons who have this mark. About this problem, we have decided to suppose that it was totally normally distributed.

As the distribution is not uniform, we can't just divide the target values in 5 equal parts like this:

$1 : [0, 1, 2, 3]$
$2 : [4, 5, 6, 7]$
$3 : [8, 9, 10, 11]$
$4 : [12, 13, 14, 15]$
$5 : [16, 17, 18, 19, 20]$

There is 395 rows of values in the dataset, and we want 395/5 = 79 values in each part. The goal is really to divide our dataset in fair parts.

The new distribution obtained is:

$1 : [0, 4, 5, 6, 7]$
$2 : [8, 9]$
$3 : [10, 11]$
$4 : [12, 13]$
$5 : [14, 15, 16, 17, 18, 19, 20]$

4 Modelling

For the modelisation, we always followed the same procedure. The goal is first to run the cross validation 5 times on a model with the training data. This will give us 5 results, 5 different scores that we will use to compare our models. For each model, it is also useful to do tests by tuning some hyperparameters, and to decide which one is the best. So, for each model we ran the cross validation with each of the hyperparameters we wanted to change. After that, by applying the '*mean*' function to the array given back by the cross validation, we were able to choose the best hyperparameters, and to compare models at the end.

In the scope of this project, we have tried 3 different classification models and 2 linear models, and we have finally compared all of these.

4.1 Classification

4.1.1 Model Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

To know which criterion is the best, we have calculated the scores of two different ones, which were: 'gini' and 'entropy'. The output of the program can be the following:

```
For criterion entropy the scores are [0.59090909 0.55384615 0.53125 0.72131148 0.71666667]
For criterion gini the scores are [0.62121212 0.64615385 0.640625 0.68852459 0.66666667]
The best hyper parameter in this case for the Model Decision Tree was gini
```

After the choice of the hyper parameter, it was time to benchmark the model by computing the accuracy. The accuracy could be something like *0.607594936709*

We also obtained the following confusion matrix (see 4 on page 6):

We can see in this matrix that the algorithm predicts that a lot of people will have grades in the 4th ranking, which represents the grades 12–13. Globally, there is not a lot of miss that are too far from the diagonal.

4.1.2 Random Forest

When you use a random forest, you are basically running several decision trees; this process can be incremental, and the algorithm will make the last state of the data automatically tune its behaviour.

We have followed the exact same method as the model decision tree. One output was:

```
For criterion entropy the scores are [0.68181818 0.66153846 0.609375 0.6557377 0.7 ]
For criterion gini the scores are [0.75757576 0.67692308 0.6875 0.70491803 0.63333333]
The best hyper parameter in this case for the Model Random Forest was gini
```

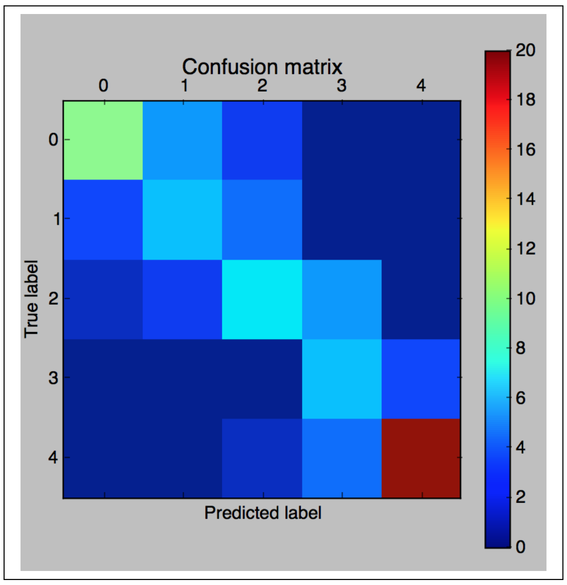


Figure 4: Confusion matrix for Decision Tree

We can notice that between the random forest and the decision tree, gini was always the best criterion. So, by using this criterion, we had an accuracy of *0.645569620253* after a run.

The confusion matrix for the random forest can be seen below (see 5 on page 6).

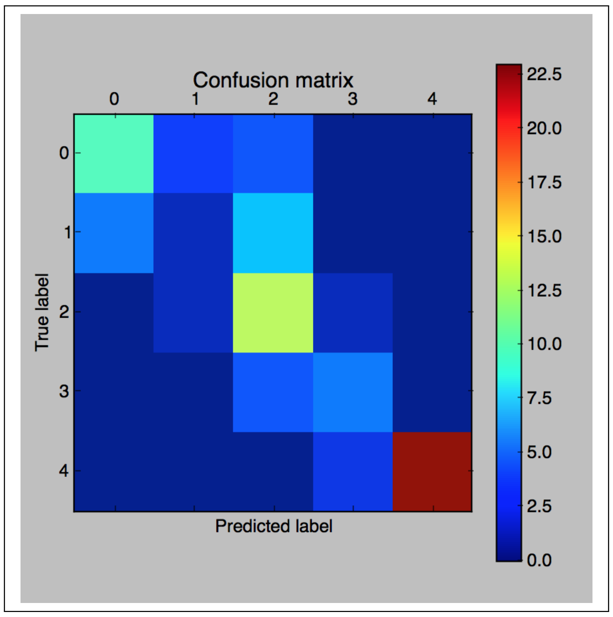


Figure 5: Confusion matrix for Random Tree

4.1.3 Nearest Neighbors

The principle behind nearest neighbor methods is to find a pre-defined amount of training samples closest in distance to the new point, and predict the label from these. The criterions that we will tune here are the number of neighbors. We have decided to range neighbors from 1 to 49 to test purposes; here, the best number of neighbors for this dataset is 15.

For n 1 the scores are [0.57575758 0.56923077 0.5625 0.57377049 0.53333333]
For n 2 the scores are [0.48484848 0.53846154 0.5625 0.59016393 0.63333333]

[...]

For n 48 the scores are [0.57575758 0.47692308 0.578125 0.59016393 0.66666667]
For n 49 the scores are [0.57575758 0.47692308 0.578125 0.59016393 0.66666667]

Here, we have computed an Accuracy of *0.632911392405*.

The confusion matrix for the Nearest Neighbors can be seen below (see 6 on page 7).

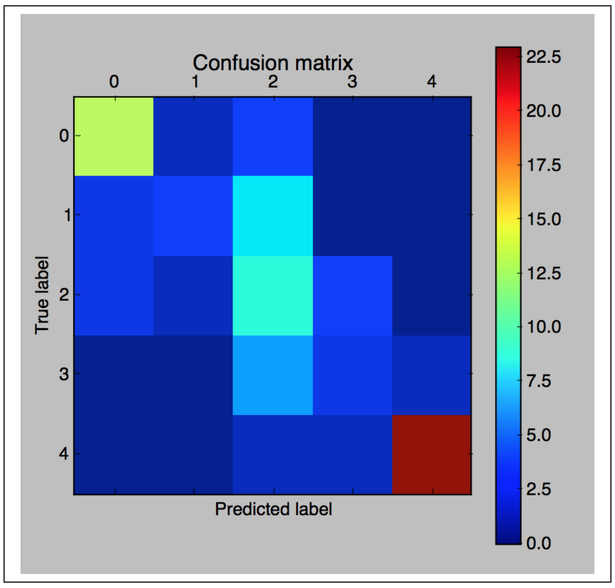


Figure 6: Confusion matrix for Nearest Neighbors

4.2 Linear Regression

For Linear Regression, the method was a bit different. In this particular case, we did not tune any of the hyperparameters, but we just used the functions as they were. Here, we have decided to use lasso regression and simple regression, and to print the scores for each of those.

Using a linear regression, the best way to compute the accuracy is to compute R2, and to compare it with the variance score.

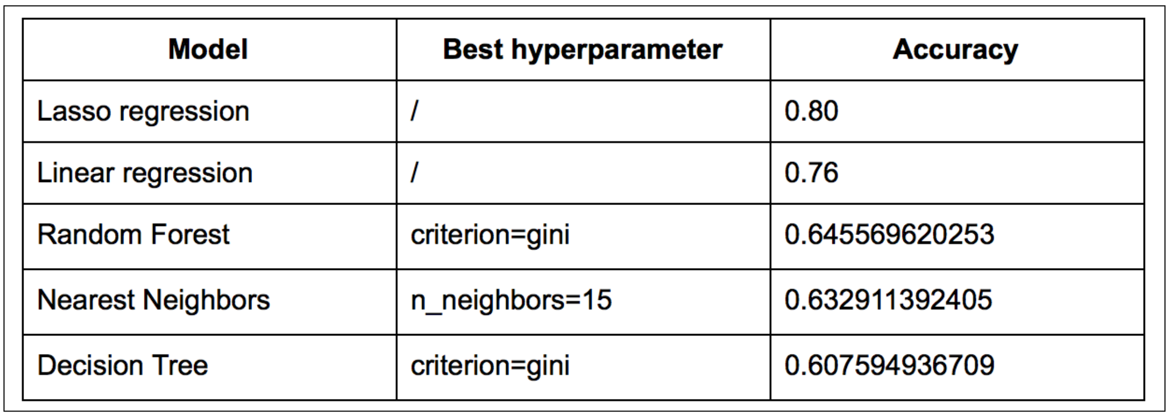
As said before, we have tried a Lasso model, but we think that it is not relevant. Indeed, lasso could be useful if we had a very high number of variables, which is definitely not the case. But we wanted to see the accuracy anyway.

With this part, we had the following results:

Variance score for LinearRegression: 0.71
R2 score for LinearRegression: 0.76
Variance score for Lasso: 0.64
R2 score for Lasso: 0.80

5 Comparison between algorithms

Here is a comparison between algorithms, ordered from the best to the worst (see 7 on page 8):



Model	Best hyperparameter	Accuracy
Lasso regression	/	0.80
Linear regression	/	0.76
Random Forest	criterion=gini	0.645569620253
Nearest Neighbors	n_neighbors=15	0.632911392405
Decision Tree	criterion=gini	0.607594936709

Figure 7: Comparison between the models' accuracy

We can see that in this case, the Lasso regression is the best model and the decision tree the worst. We had a concern about the comparison between algorithms: is it possible to compare linear regression and classification? This question can be relevant because with classification, we are only saying that 'well, this hit is a miss', whatever the miss is. In linear regression, we are more talking about the distance between the real value and the guess value. But, because we have computed R2 for every linear regression model, we are then able to compare them to the classification models because we are working with the same units.

Note that these results (fig 7) are only sampled from the execution n°i of the scripts. It may be possible that the execution n° i+1 will provide different results... It could have been possible to make a loop to test the models a hundred times (and *mean* the results) to improve the accuracy of this scoreboard. Unfortunately we did not have enough time to do it.

6 Jazz

We have explained before all our jazz, but if you need to remember what we did:

- Linear regression: we wanted to try this because we thought that predicting a note between 0 and 20 could be seen as the prediction of a continuous value.
- Frequency Distribution: we have used a Frequency Distribution to reduce the number of categories for the classification. The accuracy of the algorithm was much better after this.
- Changed some cols from continuous to categorical because it seems to be a better choice.

7 Conclusion

This project was very interesting and it gave us the opportunity to work as Data Analyst during a few days. Thinking about how to handle and manage data to fit the business needs is a pleasant aspect of the work, and we tried to find the best problematic to solve the given problem.

With the code provided, we had a lot of examples and we were able to mix them up to fit our needs. We did understand the goal of the *cross_validation* relatively late, as we were computing accuracy and confusion matrix on every iteration. At this time, we were not choosing any parameters and we were doing the wrong process. But we had enough time to recode this part and to provide expected results.

It appears that there were several sets of data provided, but we did not have enough time to merge properly, nor to split the data in data for training/data for learning. Though it could have been interesting to do it. Python is a really good scripting language. We appreciated it much and we hope that we will use it more in other projects.

Being in Dublin during one week and having an introduction to machine learning was a really good experience and it will probably lead us to new horizons.

List of Figures