Assessment_Course_1

October 8, 2020

1 Student Performance Dataset

This is a notebook for the Final Course Project with the topic **Exploratory Data Analysis for Machine Learning**.

In this notebook the open dataset *Student Performance Data Set* from the *UCI Machine Learning Repository* was used: (https://archive.ics.uci.edu/ml/datasets/student%2Bperformance)

Reference: 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.

Following sections will be answered with code examples: 1. Brief description of the data set and a summary of its attributes 2. Initial plan for data exploration 3. Actions taken for data cleaning and feature engineering 4. Key Findings and Insights, which synthesizes the results of Exploratory Data Analysis insightfully and actionable 5. Formulating at least 3 hypothesis about this data 6. Conducting a formal significance test for one of the hypotheses and discuss the results 7. Suggestions for next steps in analyzing this data 8. A paragraph that summarizes the quality of this data set and a request for additional data if needed

1.1 1. Brief desription of the data set and a summary of its attributes

The *Student Performance Dataset* includes data from achievements of secondary education students participating in courses of two different schools in Portugal. There are data from math courses and from Portuguese courses. For this assigment only the data for the math courses will be used.

The structure of the data, which were provided as .csv from the *UCI Machine Learning Repository* are as follows:

In the documentation of the dataset the dataset is described as follows:

1.1.1 Attributes for both student-mat.csv (Math course) and student-por.csv (Portuguese language course) datasets:

1. **school** - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)
2. **sex** - student's sex (binary: "F" - female or "M" - male)
3. **age** - student's age (numeric: from 15 to 22)
4. **address** - student's home address type (binary: "U" - urban or "R" - rural)
5. **famsize** - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)
6. **Pstatus** - parent's cohabitation status (binary: "T" - living together or "A" - apart)
7. **Medu** - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)

8. **Fedu** - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 – 5th to 9th grade, 3 – secondary education or 4 – higher education)
9. **Mjob** - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
10. **Fjob** - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at_home" or "other")
11. **reason** - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")
12. **guardian** - student's guardian (nominal: "mother", "father" or "other")
13. **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)
14. **studytime** - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15. **failures** - number of past class failures (numeric: n if 1<=n<3, else 4)
16. **schoolsup** - extra educational support (binary: yes or no)
17. **famsup** - family educational support (binary: yes or no)
18. **paid** - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19. **activities** - extra-curricular activities (binary: yes or no)
20. **nursery** - attended nursery school (binary: yes or no)
21. **higher** - wants to take higher education (binary: yes or no)
22. **internet** - Internet access at home (binary: yes or no)
23. **romantic** - with a romantic relationship (binary: yes or no)
24. **famrel** - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25. **freetime** - free time after school (numeric: from 1 - very low to 5 - very high)
26. **goout** - going out with friends (numeric: from 1 - very low to 5 - very high)
27. **Dalc** workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28. **Walc** - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29. **health** - current health status (numeric: from 1 - very bad to 5 - very good)
30. **absences** - number of school absences (numeric: from 0 to 93)

1.1.2 these grades are related with the course subject Math:

31. **G1** - first period grade (numeric: from 0 to 20)
32. **G2** - second period grade (numeric: from 0 to 20)
33. **G3** final grade (numeric: from 0 to 20, output target)

1.2 2. Initial Plan for data exploration

My steps for data exploration include: 0. keeping place for all packages needed in the notebook 1. read in the data and print the first 5 rows 2. find out how many rows and columns are in the dataset 3. find out which columns have numeric variables and which not 4. extract the columns with numeric values and do some maths 5. group the data by some attributes and do some vizualisations

1.2.1 0. keeping place for all packages needed in the notebook

```
In [44]: # importing relevant packages for the notebook
         import pandas as pd
```

```
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt

import seaborn as sns
sns.set()

from scipy.stats import binom
```

1.2.2 1. read in the data and print the first 5 rows

In [26]:
```
# read the data in, do a copy for later and display the first 5 rows - transposed dis
# can be seen. The engine='python' command was recommended by jupyter-python3 for par
# "\;" must be written as RegEx statement due to version changes

path = "student-mat.csv"

df = pd.read_csv(path, sep="\;", engine='python')
data = df.copy()

df.head().T
```

Out[26]:

	0	1	2	3	4
school	"GP"	"GP"	"GP"	"GP"	"GP"
sex	"F"	"F"	"F"	"F"	"F"
age	18	17	15	15	16
address	"U"	"U"	"U"	"U"	"U"
famsize	"GT3"	"GT3"	"LE3"	"GT3"	"GT3"
Pstatus	"A"	"T"	"T"	"T"	"T"
Medu	4	1	1	4	3
Fedu	4	1	1	2	3
Mjob	"at_home"	"at_home"	"at_home"	"health"	"other"
Fjob	"teacher"	"other"	"other"	"services"	"other"
reason	"course"	"course"	"other"	"home"	"home"
guardian	"mother"	"father"	"mother"	"mother"	"father"
traveltime	2	1	1	1	1
studytime	2	2	2	3	2
failures	0	0	3	0	0
schoolsup	"yes"	"no"	"yes"	"no"	"no"
famsup	"no"	"yes"	"no"	"yes"	"yes"
paid	"no"	"no"	"yes"	"yes"	"yes"
activities	"no"	"no"	"no"	"yes"	"no"
nursery	"yes"	"no"	"yes"	"yes"	"yes"
higher	"yes"	"yes"	"yes"	"yes"	"yes"
internet	"no"	"yes"	"yes"	"yes"	"no"
romantic	"no"	"no"	"no"	"yes"	"no"
famrel	4	5	4	3	4

freetime	3	3	3	2	3
goout	4	3	2	2	2
Dalc	1	1	2	1	1
Walc	1	1	3	1	2
health	3	3	3	5	5
absences	6	4	10	2	4
G1	"5"	"5"	"7"	"15"	"6"
G2	"6"	"5"	"8"	"14"	"10"
G3	6	6	10	15	10

1.2.3 2. find out how many rows and columns are in the dataset

```
In [27]: # printing with the shape-attribute the number of rows and columns

         df.shape
```

```
Out[27]: (395, 33)
```

There are 395 *observations* (rows) and 33 *attributes* (columns). The 33. column (G3, final grade) was declared as *target*, so there are for now 32 x 395 (12 640) different *feature* values (sex, age, mother's and father's education,...), which can be used to train and validate a model to predict a special grade (*target*) in student courses.

1.2.4 3. find out which columns have numeric variables and which not

```
In [28]: # use the info-function to display all columns with datatype

         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
school        395 non-null object
sex           395 non-null object
age           395 non-null int64
address       395 non-null object
famsize       395 non-null object
Pstatus       395 non-null object
Medu          395 non-null int64
Fedu          395 non-null int64
Mjob          395 non-null object
Fjob          395 non-null object
reason        395 non-null object
guardian      395 non-null object
traveltime    395 non-null int64
studytime     395 non-null int64
failures      395 non-null int64
schoolsup     395 non-null object
famsup        395 non-null object
```

```
paid          395 non-null object
activities    395 non-null object
nursery       395 non-null object
higher        395 non-null object
internet      395 non-null object
romantic      395 non-null object
famrel        395 non-null int64
freetime      395 non-null int64
goout         395 non-null int64
Dalc          395 non-null int64
Walc          395 non-null int64
health        395 non-null int64
absences      395 non-null int64
G1            395 non-null object
G2            395 non-null object
G3            395 non-null int64
dtypes: int64(14), object(19)
memory usage: 101.9+ KB
```

1.2.5 4. extract the columns with numeric values and do some maths

```
In [29]: # save all numeric values columns into a list

         numeric_columns = df.dtypes[df.dtypes != np.object].index.to_list()

         numeric_columns

Out[29]: ['age',
          'Medu',
          'Fedu',
          'traveltime',
          'studytime',
          'failures',
          'famrel',
          'freetime',
          'goout',
          'Dalc',
          'Walc',
          'health',
          'absences',
          'G3']

In [30]: # build a smaller df (df_num) with all columns with numeric values

         df_num = df[numeric_columns]

         df_num.columns
```

```
Out[30]: Index(['age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'famrel',
                'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G3'],
               dtype='object')
```

```
In [31]: # do some maths with these numeric values

         # print some stats with the describe-funtion for age, studytyime, and G3

         df_num[['age', 'studytime', 'G3']].round().describe().T
```

Out[31]:

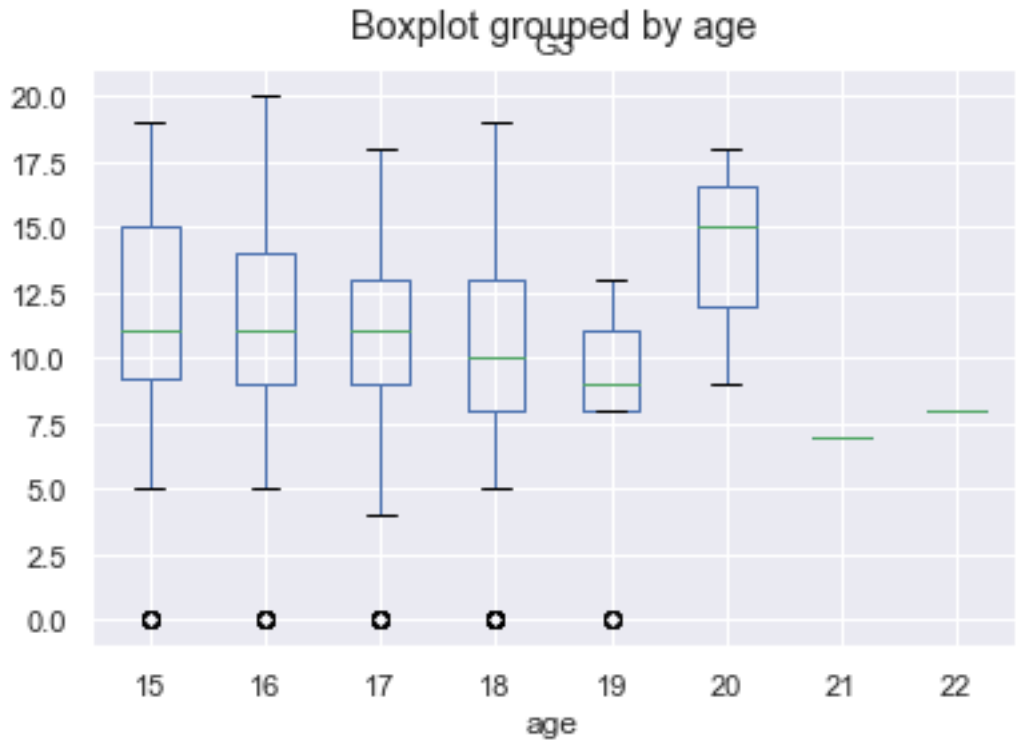
	count	mean	std	min	25%	50%	75%	max
age	395.0	16.696203	1.276043	15.0	16.0	17.0	18.0	22.0
studytime	395.0	2.035443	0.839240	1.0	1.0	2.0	2.0	4.0
G3	395.0	10.415190	4.581443	0.0	8.0	11.0	14.0	20.0

In the mean the students are about 16 - 17 years old, the youngest is 15 years old and the oldest 22 years. Furher in the mean they learn 2 - 5 hours (see data description for the meaning of "2" for studytime), and have something above 10 from 20 points in the final grade. There is also at least one student who has 0 points in the final grade. So quite average students... :-)

1.2.6 5. group the data by some attributes and/or do some vizualisations

```
In [32]: # create a barplot grouped by age and the grades (G)

         df.boxplot(column='G3', by='age');
```

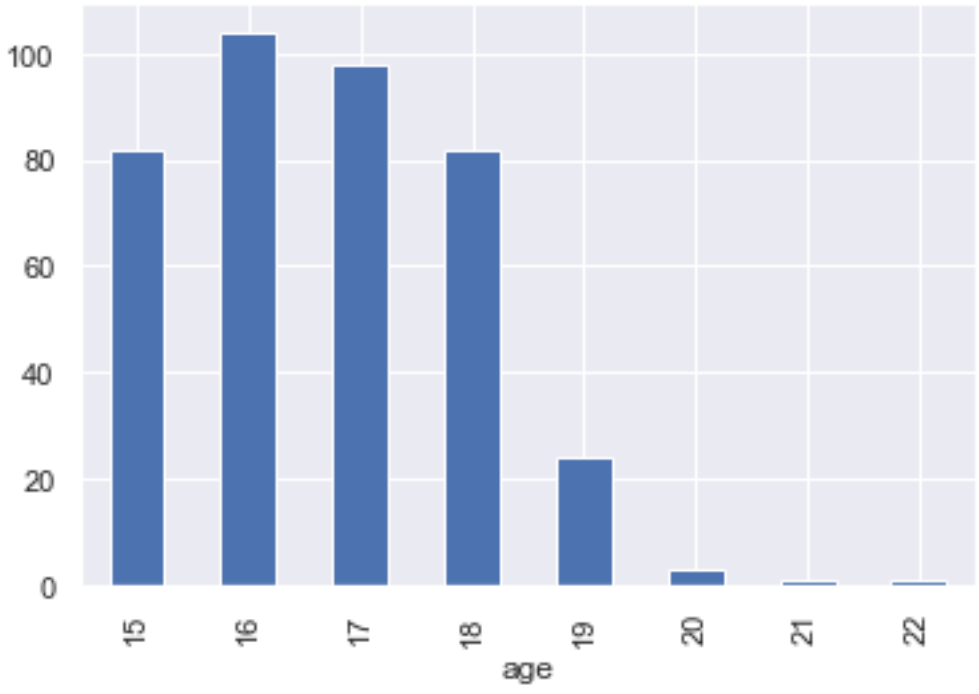


So it seams that students with the age 20 has the most points for final grade. For students with age 21 and age 22, there seems to very few data in the dataset. Further there seem to be outliers concerning the students aged 15 until 19 years. Now, let's investigate the number of students in each age class.

```
In [33]: x= df_num.groupby(['age'])['age'].count()
         x

Out[33]: age
         15     82
         16    104
         17     98
         18     82
         19     24
         20      3
         21      1
         22      1
         Name: age, dtype: int64

In [34]: x.plot(kind='bar');
```



And how many male and female students are there?

```
In [35]: # counting male and female students
         df.groupby('sex')['sex'].count()
```

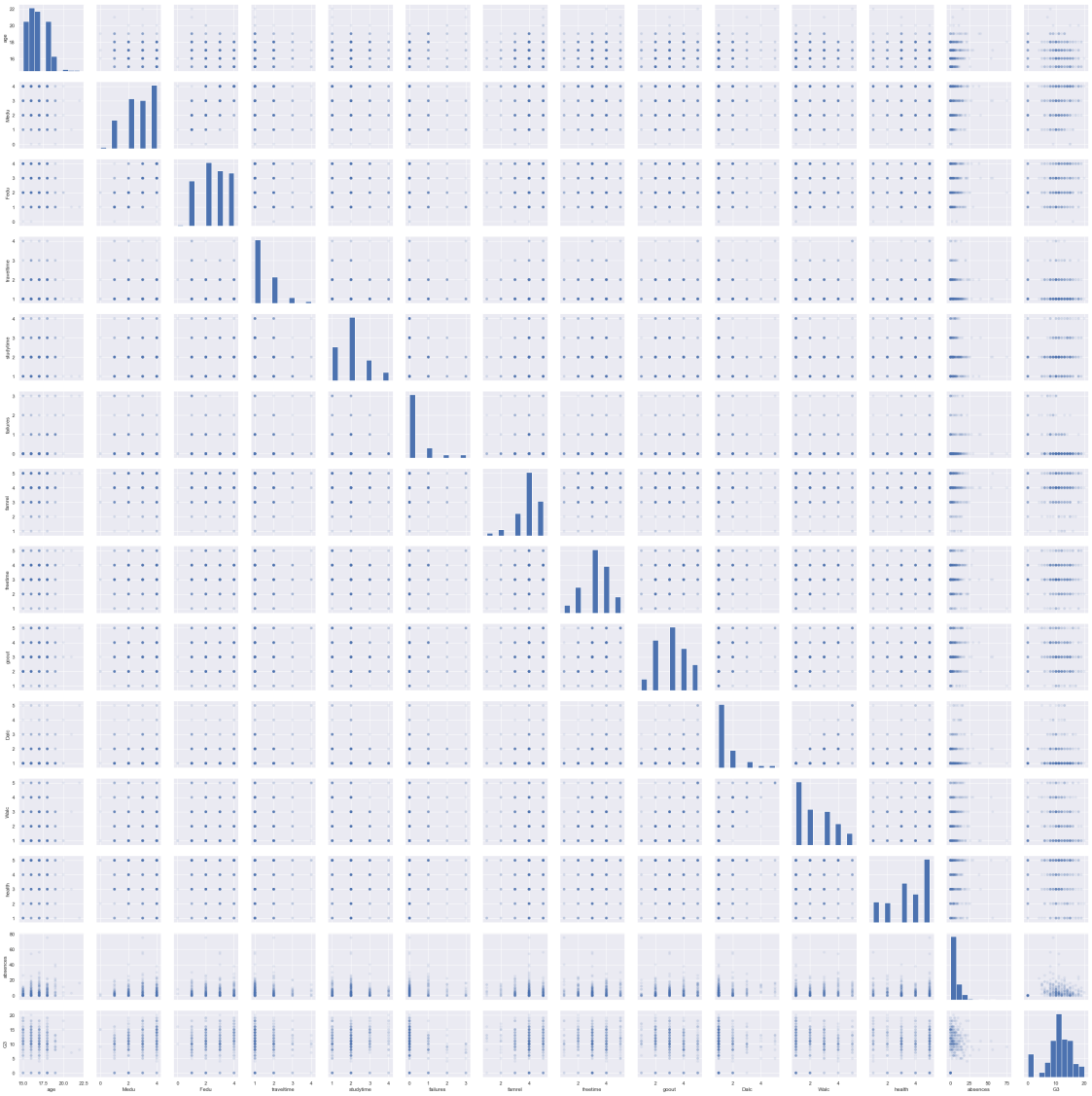
```
Out[35]: sex
         "F"    208
         "M"    187
         Name: sex, dtype: int64
```

```
In [36]: # calculating the percentage of female students
         fem_perc = round((208*100)/395,1)
         print('{}%'.format(fem_perc))
```

```
52.7%
```

There are a few more female students as male students, exactly 52.7% of all students.

```
In [37]: # doing a pairplot with the numeric values
         sns.pairplot(df_num, plot_kws=dict(alpha=.1, edgecolor='none'));
```



As can be seen in the vizualisation there are very few students aged older than 19. So let's do some feature engineering by summarizing groups and finally get all non-numric values into numeric ones, so that we can fit a model.

1.3 3. actions taken for data cleaning and feature engineering

Since some age groups have very few students we will summarize some of them. Further since there are many non-numeric values in the data, we will transform them by one-hot-encoding.

```
In [38]: # summarize some age-groups - Part 1
         age_groups = df.age.value_counts()
         age_groups
```

```
Out[38]: 16    104
         17     98
         18     82
         15     82
         19     24
         20      3
         22      1
         21      1
         Name: age, dtype: int64
```

```
In [39]: # summarize some age-groups - Part 2
         old = list(age_groups[age_groups <= 24].index)
         old
```

```
Out[39]: [19, 20, 22, 21]
```

```
In [40]: # summarize some age_groups - Part 3
         df['age'] = df['age'].replace(old, '19+')
         df['age'].value_counts()
```

```
Out[40]: 16     104
         17      98
         18      82
         15      82
         19+     29
         Name: age, dtype: int64
```

Now there are from 8 age-groups 5 left. The new age-groups has the name '19+' and summarizes all students between 19 - 22 years old.

```
In [41]: ### transform non-numeric data in numeric data by one-code-encoding
         one_hot_encode_cols = df.dtypes[df.dtypes == np.object]  # filtering by string catego
         one_hot_encode_cols = one_hot_encode_cols.index.tolist()  # list of categorical field

         df[one_hot_encode_cols].head().T
```

Out[41]:

	0	1	2	3	4
school	"GP"	"GP"	"GP"	"GP"	"GP"
sex	"F"	"F"	"F"	"F"	"F"
age	18	17	15	15	16
address	"U"	"U"	"U"	"U"	"U"
famsize	"GT3"	"GT3"	"LE3"	"GT3"	"GT3"
Pstatus	"A"	"T"	"T"	"T"	"T"
Mjob	"at_home"	"at_home"	"at_home"	"health"	"other"
Fjob	"teacher"	"other"	"other"	"services"	"other"
reason	"course"	"course"	"other"	"home"	"home"
guardian	"mother"	"father"	"mother"	"mother"	"father"
schoolsup	"yes"	"no"	"yes"	"no"	"no"
famsup	"no"	"yes"	"no"	"yes"	"yes"
paid	"no"	"no"	"yes"	"yes"	"yes"
activities	"no"	"no"	"no"	"yes"	"no"
nursery	"yes"	"no"	"yes"	"yes"	"yes"
higher	"yes"	"yes"	"yes"	"yes"	"yes"
internet	"no"	"yes"	"yes"	"yes"	"no"
romantic	"no"	"no"	"no"	"yes"	"no"
G1	"5"	"5"	"7"	"15"	"6"
G2	"6"	"5"	"8"	"14"	"10"

In [42]:
```
#Do the one hot encolding
df_one = pd.get_dummies(df, columns = one_hot_encode_cols, drop_first=True)
df_one.describe().T
```

Out[42]:

	count	mean	std	min	25%	50%	75%	max
Medu	395.0	2.749367	1.094735	0.0	2.0	3.0	4.0	4.0
Fedu	395.0	2.521519	1.088201	0.0	2.0	2.0	3.0	4.0
traveltime	395.0	1.448101	0.697505	1.0	1.0	1.0	2.0	4.0
studytime	395.0	2.035443	0.839240	1.0	1.0	2.0	2.0	4.0
failures	395.0	0.334177	0.743651	0.0	0.0	0.0	0.0	3.0
famrel	395.0	3.944304	0.896659	1.0	4.0	4.0	5.0	5.0
freetime	395.0	3.235443	0.998862	1.0	3.0	3.0	4.0	5.0
goout	395.0	3.108861	1.113278	1.0	2.0	3.0	4.0	5.0
Dalc	395.0	1.481013	0.890741	1.0	1.0	1.0	2.0	5.0
Walc	395.0	2.291139	1.287897	1.0	1.0	2.0	3.0	5.0
health	395.0	3.554430	1.390303	1.0	3.0	4.0	5.0	5.0
absences	395.0	5.708861	8.003096	0.0	0.0	4.0	8.0	75.0
G3	395.0	10.415190	4.581443	0.0	8.0	11.0	14.0	20.0
school_"MS"	395.0	0.116456	0.321177	0.0	0.0	0.0	0.0	1.0
sex_"M"	395.0	0.473418	0.499926	0.0	0.0	0.0	1.0	1.0
age_16	395.0	0.263291	0.440978	0.0	0.0	0.0	1.0	1.0
age_17	395.0	0.248101	0.432459	0.0	0.0	0.0	0.0	1.0
age_18	395.0	0.207595	0.406099	0.0	0.0	0.0	0.0	1.0
age_19+	395.0	0.073418	0.261152	0.0	0.0	0.0	0.0	1.0
address_"U"	395.0	0.777215	0.416643	0.0	1.0	1.0	1.0	1.0
famsize_"LE3"	395.0	0.288608	0.453690	0.0	0.0	0.0	1.0	1.0

```
Pstatus_"T"      395.0   0.896203  0.305384  0.0  1.0   1.0   1.0   1.0
Mjob_"health"    395.0   0.086076  0.280832  0.0  0.0   0.0   0.0   1.0
Mjob_"other"     395.0   0.356962  0.479711  0.0  0.0   0.0   1.0   1.0
Mjob_"services"  395.0   0.260759  0.439606  0.0  0.0   0.0   1.0   1.0
Mjob_"teacher"   395.0   0.146835  0.354391  0.0  0.0   0.0   0.0   1.0
Fjob_"health"    395.0   0.045570  0.208814  0.0  0.0   0.0   0.0   1.0
Fjob_"other"     395.0   0.549367  0.498188  0.0  0.0   1.0   1.0   1.0
Fjob_"services"  395.0   0.281013  0.450064  0.0  0.0   0.0   1.0   1.0
Fjob_"teacher"   395.0   0.073418  0.261152  0.0  0.0   0.0   0.0   1.0
...                ...        ...       ...  ...  ...   ...   ...   ...
G1_"13"          395.0   0.083544  0.277054  0.0  0.0   0.0   0.0   1.0
G1_"14"          395.0   0.075949  0.265253  0.0  0.0   0.0   0.0   1.0
G1_"15"          395.0   0.060759  0.239192  0.0  0.0   0.0   0.0   1.0
G1_"16"          395.0   0.055696  0.229625  0.0  0.0   0.0   0.0   1.0
G1_"17"          395.0   0.020253  0.141044  0.0  0.0   0.0   0.0   1.0
G1_"18"          395.0   0.020253  0.141044  0.0  0.0   0.0   0.0   1.0
G1_"19"          395.0   0.007595  0.086927  0.0  0.0   0.0   0.0   1.0
G1_"3"           395.0   0.002532  0.050315  0.0  0.0   0.0   0.0   1.0
G1_"4"           395.0   0.002532  0.050315  0.0  0.0   0.0   0.0   1.0
G1_"5"           395.0   0.017722  0.132105  0.0  0.0   0.0   0.0   1.0
G1_"6"           395.0   0.060759  0.239192  0.0  0.0   0.0   0.0   1.0
G1_"7"           395.0   0.093671  0.291740  0.0  0.0   0.0   0.0   1.0
G1_"8"           395.0   0.103797  0.305384  0.0  0.0   0.0   0.0   1.0
G1_"9"           395.0   0.078481  0.269268  0.0  0.0   0.0   0.0   1.0
G2_"10"          395.0   0.116456  0.321177  0.0  0.0   0.0   0.0   1.0
G2_"11"          395.0   0.088608  0.284537  0.0  0.0   0.0   0.0   1.0
G2_"12"          395.0   0.103797  0.305384  0.0  0.0   0.0   0.0   1.0
G2_"13"          395.0   0.093671  0.291740  0.0  0.0   0.0   0.0   1.0
G2_"14"          395.0   0.058228  0.234471  0.0  0.0   0.0   0.0   1.0
G2_"15"          395.0   0.086076  0.280832  0.0  0.0   0.0   0.0   1.0
G2_"16"          395.0   0.032911  0.178631  0.0  0.0   0.0   0.0   1.0
G2_"17"          395.0   0.012658  0.111936  0.0  0.0   0.0   0.0   1.0
G2_"18"          395.0   0.030380  0.171848  0.0  0.0   0.0   0.0   1.0
G2_"19"          395.0   0.007595  0.086927  0.0  0.0   0.0   0.0   1.0
G2_"4"           395.0   0.002532  0.050315  0.0  0.0   0.0   0.0   1.0
G2_"5"           395.0   0.037975  0.191377  0.0  0.0   0.0   0.0   1.0
G2_"6"           395.0   0.035443  0.185131  0.0  0.0   0.0   0.0   1.0
G2_"7"           395.0   0.053165  0.224646  0.0  0.0   0.0   0.0   1.0
G2_"8"           395.0   0.081013  0.273201  0.0  0.0   0.0   0.0   1.0
G2_"9"           395.0   0.126582  0.332926  0.0  0.0   0.0   0.0   1.0

[75 rows x 8 columns]
```

```
In [43]: df_one.shape
```

```
Out[43]: (395, 75)
```

Now we have from the original data (df) two dataframes, both with numeric values: df_one

after one-hot encoding, and df_num before the rudimentary numeric data extracted. In overall, we have 395 x 73 = 28 835 feature values, which whom we can fit a model.

1.4 4. Key Findings and Insights, which synthesizes the results of Exploratory Data Analysis insightfully and actionable

As seen there was a dataset with data which need to be transformed that they can be used in a model. These transformings included for now summarization of some age-groups and one-hot encoding. The target values mean lays in the middle, so all students can be seen together as average. There are also some outliers (0 points for grade) in the dataset. The cleaning of these outliers just by the drop-function is an possible way to handle them, even so then there are a few less observations in the whole dataset. After cleaning the outliers the target values mean would probably rise, but for now it is good to show that there are also very weak students because often they need support.

1.5 5. Formulating at least 3 hypothesis about this data

1.5.1 First testing

H0: When I go into 100 individually other math classes in Portugal, I will not find the same number of students or even more as in this combined class

H1: When I go into 100 individually other math classes in Portugal, I will find exactly the same number or even more of students as in this combined class

1.5.2 Second testing

HO: Alcohol consum does lead to fewer grade points in the end

H1: Alcohol consum does not lead to fewer grade points in the end

1.5.3 Third testing

H0: Students who want to visit later on the university have better grade points than others

H1: Students who want to visit later on the university do not gave better grade points others

1.6 6. Conducting a formal significance test for one of the hypotheses and discuss the results

We will test the *first testing*:

H0: When I go into 100 individually other math classes in Portugal, I will not find the same number of students as in this class

H1: When I go into 100 individually other math classes in Portugal, I will find exactly the same number of students as in this class

So what is the probability to meet the exact same students number of 395 students in another two combined math class in Portugal.

```
In [48]: #the probability of meeting 395 or more students in other 100 classes
         prob = 1 - binom.cdf(394, 100, 0.5)

         print(str(round(prob*100, 1))+"%")
```

```
0.0%
```

The probability lays at 0.0 % so H0 is proven. I would need to visit more, maybe much more classes.

1.7 7. Suggestions for next steps in analyzing this data

Possible steps: - Comparing female and male students for grades - Comparing students which come from "educated" homes, so where parents have a university degree with the others - Comparing students where at least one parent is at home and not working with others - Comparing students which want to got to university with others - and so on...

1.8 8. A paragraph that summarizes the quality of this data set and a request for additional data if needed

The quality of the data is in overall very good, eg. no missing values. There could be more students observations to build a model.

Notebook created by Verena Dornauer for coursera IBM Machine Learning Professional Certificate