Student Performance Data Set Project

Nuriye Merve TATLIDİL – 150212002

1-Introduction

What can affect a student's education life? What can a family provide for a high school student to succeed in their lessons? Do long studies at the desk bring benefits or extra-curricular activities? As data scientists, we need to ask a variety of questions for the issues we focus on.We should use domain information, statistics and algorithmic programming together in our projects.

2- General View of the Data

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)

31) G1 - first period grade (numeric: from 0 to 20)

31) G2 - second period grade (numeric: from 0 to 20)

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

**The G3 attribute in the data set shows the final grades of the students.I've made a classification between notes so I can easily apply Classification algorithms.Accordingly :

15-20 : A

10-15 : B

5-10 : C

0-5 : D

3-Importing Library

Since I will use the dataset as a dataframe, I first imported the pandas library. Then I added the dataset to my project. I will make some changes on the attributes of my dataset, so I saved the dataset with the first version name because I want to show the first and final version of the dataset.

```
import pandas as pd
df = pd.read_csv("../data/student-mat-firstversion.csv", sep=';')
df
#Firstly I imported the pandas library and then I read the comma separated excel file
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	letter_grade
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	C
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	C
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	B
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1	1	5	2	15	14	15	A
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	B
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	4	4	5	4	11	9	9	9	C
391	MS	M	17	U	LE3	T	3	1	services	services	...	4	5	3	4	2	3	14	16	16	A
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	3	3	3	3	3	10	8	7	C
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	1	3	4	5	0	11	12	10	B
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	2	3	3	3	5	5	8	9	9	C

395 rows × 34 columns

I showed some important values of the attributes in the data set. I applied the transposition because I thought it looked more understandable when I received the transposition.

```
In [8]: df.describe().T #I showed some important values of the dataset
```

Out[8]:

	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
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
G1	395.0	10.908861	3.319195	3.0	8.0	11.0	13.0	19.0
G2	395.0	10.713924	3.761505	0.0	9.0	11.0	13.0	19.0
G3	395.0	10.415190	4.581443	0.0	8.0	11.0	14.0	20.0

It is important for the application of algorithms that the types of our data are categorical or numerical.For this reason, I looked at how many numerical and categorical variables there are because we would convert categorical variables into numerical variables in the next steps.We can also see the size of the data set in this way.

```
: df.info() #I showed info data type, column and row number, memory usage information
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 34 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 int64
G2              395 non-null int64
G3              395 non-null int64
letter_grade    395 non-null object
dtypes: int64(16), object(18)
memory usage: 105.0+ KB
```

4-Pre-Processing Step

There are many steps in the data preprocessing phase. One of them is data cleaning. In this phase, we check the missing, noisy and outlier data.

First, I check for missing data.

Is there any missing value?

```
In [11]: df.isnull().values.any()
```

```
Out[11]: False
```

We can show the outlier data with the boxplot. The Value Determination also tells how much the middle values are spreading. I showed them because I want to experience these stages.

Is there any outlier data ?

```
In [12]: cn = df.select_dtypes(include = ['int64']) # I have listed the continuous variables
```

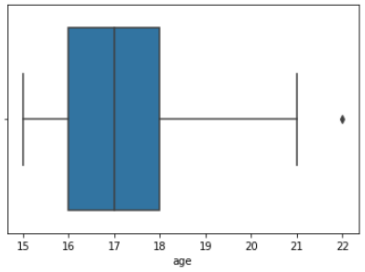
```
In [13]: cn
```

Out[13]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	18	4	4	2	2	0	4	3	4	1	1	3	6	5	6	6
1	17	1	1	1	2	0	5	3	3	1	1	3	4	5	5	6
2	15	1	1	1	2	3	4	3	2	2	3	3	10	7	8	10
3	15	4	2	1	3	0	3	2	2	1	1	5	2	15	14	15
4	16	3	3	1	2	0	4	3	2	1	2	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	20	2	2	1	2	2	5	5	4	4	5	4	11	9	9	9
391	17	3	1	2	1	0	2	4	5	3	4	2	3	14	16	16
392	21	1	1	1	1	3	5	5	3	3	3	3	3	10	8	7
393	18	3	2	3	1	0	4	4	1	3	4	5	0	11	12	10
394	19	1	1	1	1	0	3	2	3	3	3	5	5	8	9	9

395 rows × 16 columns

```
[15]: import seaborn as sns
      df_age = df["age"]
      sns.boxplot(x = df_age) ; #Firstly I imported the seaborn library then I assigned df age to df_age and I showed
      #df_age with boxplot
```




```
[16]: Q1 = df_age.quantile(0.25) #first interquartile range
      Q3 = df_age.quantile(0.75) #third interquartile range
      IQR = Q3-Q1   #IQR tells how far the middle values spread.And this is its formula
```

```
[17]: Q1
```

```
t[17]: 16.0
```

```
[18]: Q3
```

```
t[18]: 18.0
```

```
[19]: IQR
```

```
t[19]: 2.0
```

```
In [20]: low_limit = Q1 - 1.5*IQR #low limit formula
         high_limit = Q3 + 1.5*IQR #high limit formula
```

```
In [21]: low_limit
```

```
Out[21]: 13.0
```

```
In [22]: high_limit
```

```
Out[22]: 21.0
```

```
In [23]: outliers_age = (df_age < low_limit) | (df_age > high_limit) #I assigned the range I set to the outliers_age variable
```

```
In [24]: df_age[outliers_age] #I will find the value outside the range
```

```
Out[24]: 247    22
         Name: age, dtype: int64
```

```
In [25]: type(df_age) #I am checking the type as I will do the operation on df_age
```

```
Out[25]: pandas.core.series.Series
```

```
In [26]: df_age = pd.DataFrame(df_age) #I converted the df_age dataframe
```

We can extract the Outlier data or fill it with mean.I did not apply because there is no such thing in the project requirements.

```
In [27]: type(df_age) #I am checking the type
Out[27]: pandas.core.frame.DataFrame

In [28]: df_age.shape #pd_age has 395 rows and 1 column
Out[28]: (395, 1)

In [29]: #clean_age = df_age[~((df_age < (low_limit)) | (df_age > (high_limit))).any(axis = 1)]
         #I indicate that I want to take those who do not meet the condition with the tilda sign.outlier data deleted
         #clean_age.shape #data set 1 decreased

In [30]: #df_age.mean()
         #df_age[outliers_age] = df_age.mean()
         #df_age[outliers_age]
         #deleting data from the data set is not much favored as it will also destroy other useful attributes.
         #so when we find the value that is outlier, we can replace it with the mean.
```

5-Variable Transformation Step

My dataset had 2 variable attributes and attributes with more than 2 variables.I applied 0-1 conversion to 2 variable attributes.I applied a One-Hot transformation to attributes that have more than 2 variables.Because with the One-Hot transform, instead of directly giving the variables numbers like 1,2,3, it gives the re-formed attribute a value of 1.Thus, no categorical variable is superior to the others.For example, we have two attributes in our data set that include types of professions. the" teacher "variable has no advantage over the" health " variable.So I think it makes sense to apply One Hot transformation.

One Hot Transformation

```
In [68]: df["reason"].value_counts() #I looked at how many variables the data has
Out[68]: course        145
         home          109
         reputation    105
         other          36
         Name: reason, dtype: int64

In [69]: columns = ['reason','Mjob','Fjob','guardian']
         df2 = pd.concat([df, pd.get_dummies(df[columns])], axis=1).drop(columns, axis = 1)
         df2
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	traveltime	studytime	...	Mjob_services	Mjob_teacher	Fjob_at_home	Fjob_health	Fjob_other	Fjob_ser
0	GP	F	18	U	GT3	A	4	4	2	2	...	0	0	0	0	0	
1	GP	F	17	U	GT3	T	1	1	1	2	...	0	0	0	0	1	
2	GP	F	15	U	LE3	T	1	1	1	2	...	0	0	0	0	1	
3	GP	F	15	U	GT3	T	4	2	1	3	...	0	0	0	0	0	
4	GP	F	16	U	GT3	T	3	3	1	2	...	0	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
390	MS	M	20	U	LE3	A	2	2	1	2	...	1	0	0	0	0	
391	MS	M	17	U	LE3	T	3	1	2	1	...	1	0	0	0	0	
392	MS	M	21	R	GT3	T	1	1	1	1	...	0	0	0	0	1	
393	MS	M	18	R	LE3	T	3	2	3	1	...	1	0	0	0	1	
394	MS	M	19	U	LE3	T	1	1	1	1	...	0	0	1	0	0	

0-1 Transformation

```
In [82]: from sklearn.preprocessing import LabelEncoder
         lbe = LabelEncoder()
         clms = ['school','sex','address','famsize','Pstatus','schoolsup','famsup','paid','activities','nursery','higher','internet','rom
         for i in clms:
             df2[i] = lbe.fit_transform(df2[i])

In [83]: df2
```

traveltime	studytime	...	Mjob_services	Mjob_teacher	Fjob_at_home	Fjob_health	Fjob_other	Fjob_services	Fjob_teacher	guardian_father	guardian_mother	guardian_other
2	2	...	0	0	0	0	0	0	1	0	1	0
1	2	...	0	0	0	0	1	0	0	1	0	0
1	2	...	0	0	0	0	1	0	0	0	1	0
1	3	...	0	0	0	0	0	1	0	0	1	0
1	2	...	0	0	0	0	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1	2	...	1	0	0	0	0	1	0	0	0	1
2	1	...	1	0	0	0	0	1	0	0	1	0
1	1	...	0	0	0	0	1	0	0	0	0	1
3	1	...	1	0	0	0	1	0	0	0	1	0
1	1	...	0	0	1	0	0	0	0	1	0	0

school : GP = 0 MS = 1

sex : F = 0 M = 1

address : U = 1 R = 0

famsize : GT3 = 0 LE3 = 1

Pstatus : A = 0 T = 1

schoolsup : Yes = 1 No = 0

famsup : Yes = 1 No = 0

paid : Yes = 1 No = 0

activities : Yes = 1 No = 0

nursery : Yes = 1 No = 0

higher : Yes = 1 No = 0

internet : Yes = 1 No = 0

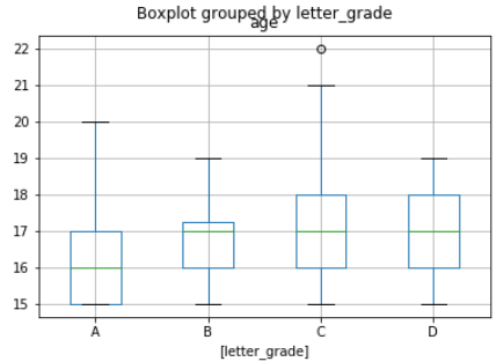
romantic : Yes = 1 No = 0

6-Showing Box Plot

I showed the age and traveltime qualities together with grade i on the box plot chart.I had the opportunity to see outliers against the Boxplot chart.

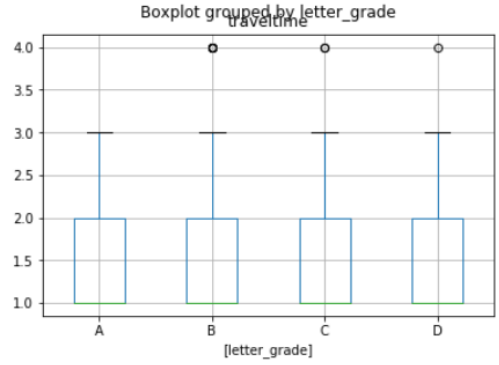
```
df2.boxplot(column=["age"], by=["letter_grade"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28fce0a0588>
```




```
df2.boxplot(column=["traveltime"], by=["letter_grade"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28fce04f948>
```

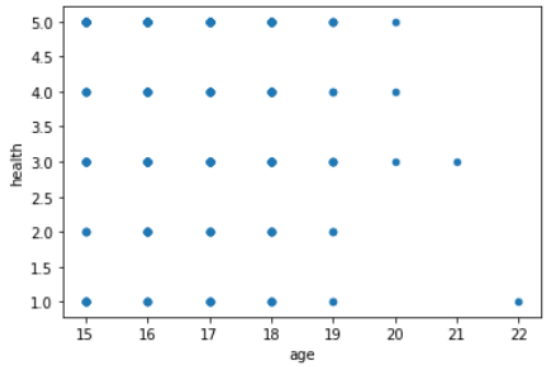


7-Showing Scatter Plot

Scatter diagram is used to show and interpret the relationship between two variables.Scatter's meaning is distribution.

```
df2.plot.scatter(x='age', y='health')
```

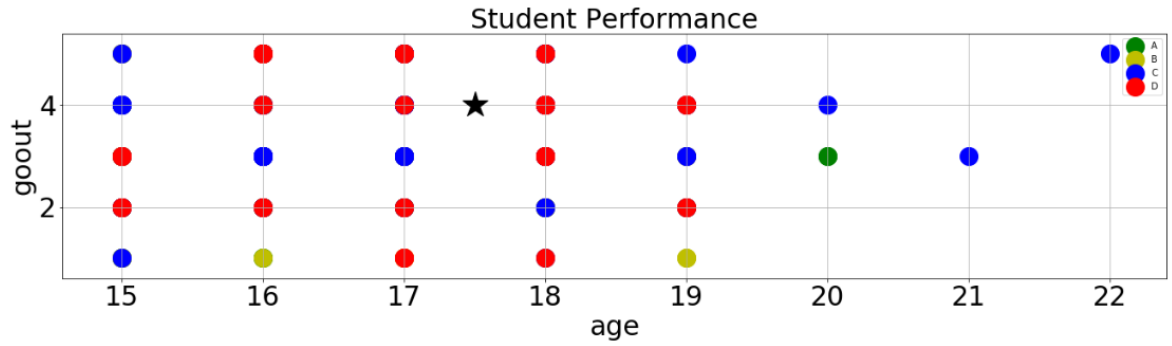
```
<matplotlib.axes._subplots.AxesSubplot at 0x28fcd718788>
```



I used goout, age and letter_grade attributes in the chart below.The value of all classes appears on the graph.Red values (letter note: D) are high on the chart.Red (letter note : D) and blue values (letter grade: C) are between 16-18 years old.This made me think that failure in this age range was more than the attribute of the goout.

```
import matplotlib.pyplot as plt
plt.figure(figsize = (max(df['age']), max(df['goout'])))
plt.xlabel('age', fontsize = 30)
plt.ylabel('goout', fontsize = 30)
plt.title('Student Performance', fontsize = 30)
plt.tick_params(axis='both', labelsize=30)
targets = ['A', 'B','C','D']
colors = ['g','y','b','r']
for target, color in zip(targets,colors):
    indices = df['letter_grade'] == target
    plt.scatter(df.loc[indices, 'age']
                , df.loc[indices, 'goout']
                , c = color
                , s = 400)
plt.legend(targets, fontsize=10)
plt.grid()
plt.plot(17.5, 4, color='black', marker='*', markersize=30)
```

```
[<matplotlib.lines.Line2D at 0x28fce293a88>]
```



It made me think that if I think of the marker (star,*) value as k = 5, it will have a red value.

8-Algorithm Step

I have previously imported the pandas, seaborn and matplotlib libraries. I also import the Numpy, Standard scaler, train-test, confusion_matrix, accuracy_score and KNeighbors libraries.

```
import numpy as np
from sklearn.preprocessing import  StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.neighbors import KNeighborsClassifier
#I imported the necessary libraries
```

```
df2
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	traveltime	studytime	...	Mjob_services	Mjob_teacher	Fjob_at_home	Fjob_health	Fjob_o
0	0	0	18	1	0	0	4	4	2	2	...	0	0	0	0	
1	0	0	17	1	0	1	1	1	1	2	...	0	0	0	0	
2	0	0	15	1	1	1	1	1	1	2	...	0	0	0	0	
3	0	0	15	1	0	1	4	2	1	3	...	0	0	0	0	
4	0	0	16	1	0	1	3	3	1	2	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
390	1	1	20	1	1	0	2	2	1	2	...	1	0	0	0	
391	1	1	17	1	1	1	3	1	2	1	...	1	0	0	0	
392	1	1	21	0	0	1	1	1	1	1	...	0	0	0	0	
393	1	1	18	0	1	1	3	2	3	1	...	1	0	0	0	
394	1	1	19	1	1	1	1	1	1	1	...	0	0	1	0	

1-)Without Standard Scaler

letter_grade = label of my data set

so I put it in the variable y, the remaining attributes in the variable x. I have separated the data set as 70% train and 30% test.

- I'm generating an object from the KNeighborsClassifier class

- n_neighbors: k value

- metric: the formula for calculating distance between values (minkowski,euclid..)

I train the machine with the train data set. Then I gave him the test dataset and got him to guess the letter_grade.

```
y = df2["letter_grade"]
x = df2.drop(["letter_grade"], axis = 1)
#I put it in the variable y, the remaining attributes in the variable x
```

Split Data

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.30,random_state=6)
#I have separated the data set as 70% train and 30% test.
```

k = 3

```
knn = KNeighborsClassifier(n_neighbors=3,metric='euclidean')
knn.fit(x_train,y_train)
#I created object knn. I set it as k = 3. And I used euclidean distance.I trained the machine with the train.
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
y_pred1 = knn.predict(x_test)
# I gave the test dataset and got him to guess the letter grade
```

```
A1 = accuracy_score(y_test,y_pred1)
#I looked at success
```

```
A1
```

```
0.7899159663865546
```

Accuracy_score refers to the correct calculation.I calculated the Complexity with Confusion_matrix. The reason for the complexity matrix to be 4x4 is that I have 4 classes. The diagonal of the matrix expresses the truth.

```
|: print(classification_report(y_test,y_pred1))
   #I showed Precision, recall, F1 score and support.
```

```
              precision    recall  f1-score   support

           A       0.76      0.76      0.76        21
           B       0.75      0.84      0.79        57
           C       0.81      0.61      0.69        28
           D       1.00      1.00      1.00        13

    accuracy                           0.79       119
   macro avg       0.83      0.80      0.81       119
weighted avg       0.79      0.79      0.79       119
```

```
|: confusion_matrix(y_test,y_pred1)
   #Confusion matrix of data set with 4 classes
```

```
|: array([[16,  5,  0,  0],
        [ 5, 48,  4,  0],
        [ 0, 11, 17,  0],
        [ 0,  0,  0, 13]], dtype=int64)
```

k = 5

```
|: knn1 = KNeighborsClassifier(n_neighbors=5,metric='euclidean')
   knn1.fit(x_train,y_train)
   #I created object knn1 I set it as k = 5. And I used euclidean distance.I trained the machine with the train.
```

```
|: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                      metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                      weights='uniform')
```

```
|: y_pred2 = knn1.predict(x_test)
   # I gave the test dataset and got him to guess the letter grade
```

```
|: A2 = accuracy_score(y_test,y_pred2)
   #I looked at success
```

```
0]: A2
```

```
0]: 0.8823529411764706
```

```
3]: print(classification_report(y_test,y_pred2))
   #I showed Precision, recall, F1 score and support.
```

```
              precision    recall  f1-score   support

           A       0.86      0.86      0.86        21
           B       0.85      0.91      0.88        57
           C       0.92      0.79      0.85        28
           D       1.00      1.00      1.00        13

    accuracy                           0.88       119
   macro avg       0.91      0.89      0.90       119
weighted avg       0.88      0.88      0.88       119
```

```
4]: confusion_matrix(y_test,y_pred2)
   #Confusion matrix of data set with 4 classes
```

```
4]: array([[18,  3,  0,  0],
        [ 3, 52,  2,  0],
        [ 0,  6, 22,  0],
        [ 0,  0,  0, 13]], dtype=int64)
```

k = 7

```
1]: knn2 = KNeighborsClassifier(n_neighbors=7,metric='euclidean')
   knn2.fit(x_train,y_train)
   ##I created object knn2 I set it as k = 7. And I used euclidean distance.I trained the machine with the train.
```

```
1]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                      metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                      weights='uniform')
```

```
2]: y_pred3 = knn2.predict(x_test)
    # I gave the test dataset and got him to guess the letter grade

3]: A3 = accuracy_score(y_test,y_pred3)
    #I looked at success

4]: A3

4]: 0.8823529411764706

5]: print(classification_report(y_test,y_pred3))
    #I showed Precision, recall, F1 score and support.
```

	precision	recall	f1-score	support
A	0.86	0.86	0.86	21
B	0.83	0.95	0.89	57
C	1.00	0.71	0.83	28
D	1.00	1.00	1.00	13
accuracy			0.88	119
macro avg	0.92	0.88	0.89	119
weighted avg	0.89	0.88	0.88	119

```
6]: confusion_matrix(y_test,y_pred3)
    #Confusion matrix of data set with 4 classes

6]: array([[18,  3,  0,  0],
           [ 3, 54,  0,  0],
           [ 0,  8, 20,  0],
           [ 0,  0,  0, 13]], dtype=int64)
```

k = 9

```
|: knn3 = KNeighborsClassifier(n_neighbors=9,metric='euclidean')
   knn3.fit(x_train,y_train)
   #I created object knn3 I set it as k = 9 And I used euclidean distance.I trained the machine with the train.

|: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                        metric_params=None, n_jobs=None, n_neighbors=9, p=2,
                        weights='uniform')

|: y_pred4 = knn3.predict(x_test)
   # I gave the test dataset and got him to guess the letter grade

|: A4 = accuracy_score(y_test,y_pred4)
   #I looked at success

|: A4

|: 0.8907563025210085

|: print(classification_report(y_test,y_pred4))
   #I showed Precision, recall, F1 score and support.
```

	precision	recall	f1-score	support
A	0.89	0.81	0.85	21
B	0.83	0.96	0.89	57
C	1.00	0.75	0.86	28
D	1.00	1.00	1.00	13
accuracy			0.89	119
macro avg	0.93	0.88	0.90	119
weighted avg	0.90	0.89	0.89	119

```
|: confusion_matrix(y_test,y_pred4)
   #Confusion matrix of data set with 4 classes
```

```
array([[17,  4,  0,  0],
       [ 2, 55,  0,  0],
       [ 0,  7, 21,  0],
       [ 0,  0,  0, 13]], dtype=int64)
```

2-)With Standard Scaler

Whether the data is normally distributed or not is an important factor in the operation of some algorithms. The direction of the data may affect the model performance. There are some methods to normalize these values and reduce dominance. Standardization: It is a method where the average value takes the value of 0 and the standard deviation takes the value of 1 and the distribution approaches the normal. According to the method, we subtract the average value from the value we have, then divide it by the variance value.

2-With standard scaler

Standard Scaler

```
X = StandardScaler().fit_transform(x)
#I standardized the attributes I put in x value
```

```
z = df2["letter_grade"]
#I put the letter_grade value, which I set as label, to the z variable to avoid confusion
```

Split Data

```
X_train, X_test, z_train, z_test = train_test_split(X,z,test_size=0.30,random_state=6)
#I separated the efficiency as 30 tests and 70 train
```

k = 3 (with standard scaler)

```
knn4 = KNeighborsClassifier(n_neighbors=3,metric='euclidean')
knn4.fit(X_train,z_train)
#I created object knn4 I set it as k = 3. And I used euclidean distance.I trained the machine with the train.
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
z_pred1 = knn4.predict(X_test)
# I gave the test dataset and got him to guess the letter grade
```

```
A5 = accuracy_score(z_test,z_pred1)
#I looked at success
```

```
A5
```

```
0.5126050420168067
```

```
print(classification_report(z_test,z_pred1))
#I showed Precision, recall, F1 score and support.
```

```
              precision    recall  f1-score   support

           A       0.40      0.48      0.43        21
           B       0.53      0.68      0.60        57
           C       0.53      0.32      0.40        28
           D       0.75      0.23      0.35        13

    accuracy                           0.51       119
   macro avg       0.55      0.43      0.45       119
weighted avg       0.53      0.51      0.50       119
```

```
confusion_matrix(z_test,z_pred1)
#Confusion matrix of data set with 4 classes
```

```
array([[10, 11,  0,  0],
       [13, 39,  4,  1],
       [ 1, 18,  9,  0],
       [ 1,  5,  4,  3]], dtype=int64)
```

I have standardized my data which I have determined as the attributes I put in x value.I put the letter_grade value, which I set as label, to the z variable to avoid confusion.

	K =3	K = 5	K = 7	K = 9	Standart Scaler K = 3
Accuracy	0.79	0.88	0.88	0.89	0.51

9- Where did I have difficulty?

I guess I was very excited because it was my first data science project, which I literally did. While trying to do everything perfectly, I think that I was distracted in some parts and my time was short for more important parts. When we did the standard scaler in class, we did it in the algorithm section.I always tried at the beginning of the project because I was confused.That brought about some problems for me.My data set didn't fit directly into classification. So I decided to classify it with letter notes.I thought I should apply the standard scaler to the letter note.But the letter grade was categorically variable.That's why the code was giving an error in this section. I knew where the problem was, but I didn't know how to solve it.And then I remembered that we split X and y in class.That problem is solved.I've wasted a lot of time on this part. I researched the transformation process a lot. I had no problems about this. It was very enjoyable to draw the graphics. I had a little problem with the algorithm.My friends and I had some doubts about whether we would write the code ourselves or whether we would use the sklearn package. I had the biggest problem with separating the data.Training,validation and testing required from us in the document was to separate. I've done the separation.But then I realized that I was lacking in the training,validation and testing phase of showing these for x and y and the process.So I tore it up in the form of normal shredding (training & testing))