Machine Learning Individual Project

--- factor analysis of student math score

Instructor: Hassane Kone

Student Name： Diwei Zhu

Net ID: dz1397

May 2020

1 Introduction

My project is focusing on the student dataset from the UCI website. The student dataset includes student identity of its behavior and family backgrounds, as well as the score in different courses.

I chose to analyze the math course score. With the 30 provided factors, I'm trying to analyze which factors are affecting students' scores. After the analysis, I will try to build a classifier to identify student performance based on its background.

2 Problem Definition and Algorithm

Task definition

The task includes data preparation, Factor analysis, and selection, model setup, and validation.

Data preparation

For the data preparation part, the task is to loading data into python, checking for missing values. Plot the correlation of performance data. With the cleaned data, transform the non-numeric data into numbers.

Factor analysis and selection

For the student performance, find out the optimal number of clusters and use KMeans to do clustering. For factors, use RFE, correlation matrix, and variance threshold to determine the optimal number of features to choose and which feature to choose.

Model setup and validation

Use the clustered and labeled performance data and the converted feature data to fit the classify model. Record the performance of different model options. Use cross-validation to evaluate the best model.

Algorithm definition

RFE

RFE is a recursive feature elimination. Use a selected estimator to recursively calculate the importance of features an remove the feature with the least importance. The outcome of RFE is the importance score for features.

Linear regression

Linear regression models the relationship between variables and responses. The aim is to find out the coefficient that can best explain the relationship.

Logistic regression

Instead of a continuous response, a logistic regression aims to build the relationship between variables and a binominal response. The core of logistic regression is the sigmoid function, it transfers the linear relationship into an “s” curve result.

Perceptron

Perceptron is a method to find a line or hyperplane that separate two class of data. The core of the perceptron is the softmax function, which maximized the distance from the closest point of each class to the line.

KNN

KNN classify a point by the most popular class of its neighbors. KNN calculates the distance of the point to all other points by the sum of different variables.

Multinomial naïve Bayes

The basic idea of naïve Bayes is calculating the possibility of the appearance of features when a specific repose appears. By multiplying all the possibilities, we can have the possibility of different results in a set of variables. A multinominal naïve Bayes is focused on discrete features.

Decision tree

The decision tree is a set of conditions test in a tree-like graph, each leaf of the tree has a prediction of classes under conforming its condition. The core idea when building the tree is the build the condition that could separate most of the data.

3 Experimental evaluation

The experiment consists of four main parts: data pre-processing, feature selection, model setup, validation.

Methodology

Data processing

For data cleaning, simply check whether there are missing values and outliners. Remove the roles that contain those unusable data.

For performance parameter selection, use variance threshold, and correlation matrix to see the relationship between three scores and see how the period score is affecting the final score.

The features come with different formats. The numeric factor will use normalization. The non-numeric factors with only two options will be converted into (1,0). The rest of the factors will use the one-hot encoder to separate each option to independent columns.

Optimal cluster number

For the KMeans method, there is a variable called inertia that measures the sum of the distance from points to the cluster center in all clusters. As the K approaches the sample size, the inertia will finally drop to 0.

The way to find the best K is looking for a point that k and inertia both are small.

Optimal feature number

Since some of the parameters are not continuous, so we cannot transfer to numbers directly. Instead, we can use the one-hot encoder to transform into a series of binary columns representing every single option. After the conversion using the one-hot encoder, the parameter increased from 30 to 46. It will be better to test several numbers of features to select. The method is like determining the clusters. The aim is the find the smallest number of features that still have enough information for classification. the score os features are evaluated by the RFE. To reduce the bias, the rank of the feature will be the average rank from estimator of random forest, linear regression, logistic regression, and SVC.

Model evaluation

To find the best model for prediction, the model setup processing will setup serval models with the same dataset. The model will include SVC, linear regression, logistic regression, Perceptron, KNN, naïve Bayes, and Decision Tree.

There are two ways to evaluate the models: predicting the original data and cross-validation. A model is suitable for the data if it can reproduce the result of the original data. Cross-validation could see how well the model can perform with additional or future data.

Result

Performance factor selection

In the data cleaning process, the students with any of the scores that are 0 are removed. This is because only G2 and G3 have 0. However, the final score should be related to both period scores. Since G1 didn't have 0, those with G3 is 0 are regarded as an outliner.

There are three scores in the dataset: G1- first-period score, G2 – second-period score, G3 – the final score. The Pairplot graph and the correlation matrix shows those three scores are highly correlated. From the description of the dataset, it would be reasonable to only take the final score G3 as the performance variables.

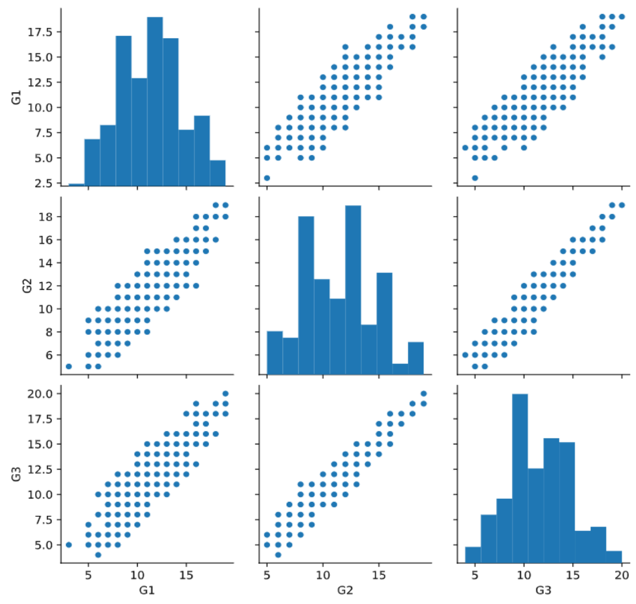


Figure 1 . Pair plot for three different scores

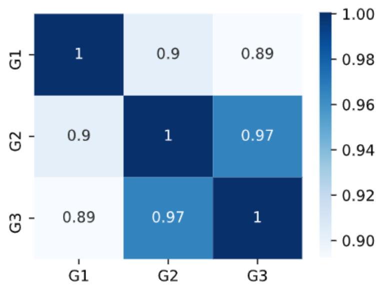


Figure 2. Correlation matrix of three scores

Optimal cluster number

I tried to enumerate the k number from 1 to 20 and see which number is more efficient. the result shows the third point is the “elbow point” of the graph. Before that point, the inertia decreases sharply as the k increase. After that point, the inertia is not changing at the same speed. So the optimal number of K is 3

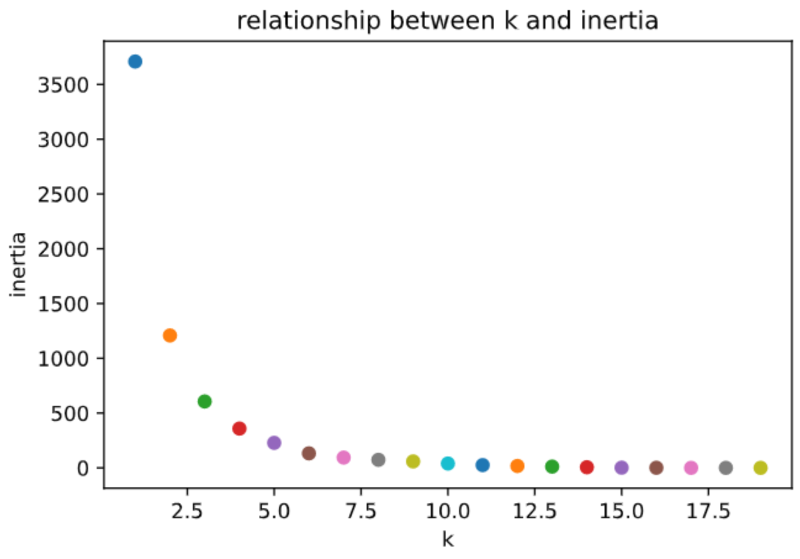


Figure 3. relationship between K and inertia

Optimal feature number and selection

The graph for the performance of different models under different feature shows that when the number of features increases, the performance will also increase. With the seven tested model, SVC and decision tree both show a significant trend. The “elbow point” for SVC is 15 and the decision tree is 20.

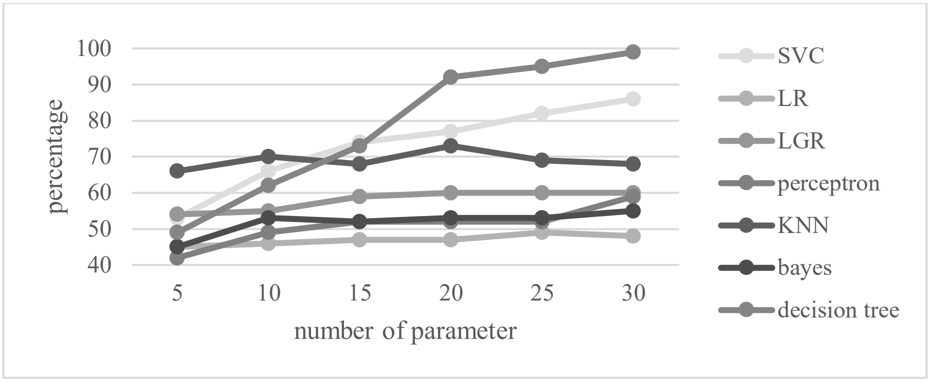


Figure 4. Feature number and model accuracy

The average rank of features shows staring at the 20th feature the average rank has a big increase, which means this feature's importance is not similar to the previous one. To eliminate the unimportant feature and limit the feature number, the optimal feature number should be 15.

The features that are selected to build the models are: 'absences', 'failures', 'Medu', 'schoolsup', 'goout', 'Walc', 'age', 'Mjob_other', 'studytime', 'Fjob_teacher', 'Dalc', 'Mjob_teacher', 'health', 'Fjob_other' and 'famsup'.

Model evaluation

Using 15 features for model setup, the SVC performance is similar to the Decision tree in original data. However, the SVC has the highest score in cross-validation, which means the SVC model can better adapt to new data. So the final model will be SVC.

Table1

Table of models and corresponding cross-validation score.

Model name	Cross validate score
SVC	0.52
LR	0.14
LGR	0.50
Perceptron	0.44
KNN	0.48
Bayes	0.48
Decision tree	0.44

Discussion

The result from the evaluation found out for the students' math course data set the optimal practice is to use 3 clusters to evaluate the students' performance and use the top 15 features to build the SVC model. It can have 74% accuracy on original data and can reach 0.52 in the cross-validation test score.

The average rank graph shows a steady increment in the ranking, which means no feature has a much higher relationship to the student performance. The score of a student is affected by many factors including family background, classmates, and the students themselves. In this project, we figure to find out a model that could quickly predict the performance using a small number of important features.

The model should only 74% on original data and 52% correction on new data. It is not a perfect model for high accuracy. But one reason for this is because of the size of the dataset. The dataset has 30 parameters, it needs 2^30 samples to cover all the situations, but it only has merely 390 rows. The ranking of the feature shows there is not a clear gap between features. The graph of the correlation between features and scores also shows the same idea. Almost all the features show some kinds of relationships to the final grade.

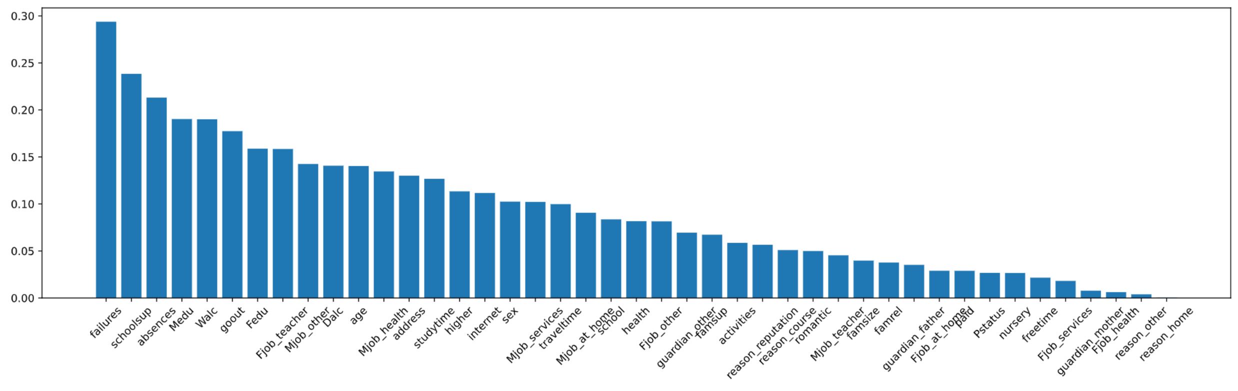


Figure 5. correlation of feature and the final score

4 Related work

Since the dataset is public on the internet, I have found some works from other people on Kaggle. I tried the go through other peoples' work to see how they analyze the data. Here I chose some highly voted notebook on Kaggle.

The first one (dipam7. ,2018)did a lot of plotting on features. He did more than calculating the correlations. He used the swarm plot on the correlation feature and can see the features' effect. He used linear regression to build the model, but instead of using the cross-validation, he use MSE to evaluate different linear models and find out the simple linear regression is the best.

The second one (hindelya. 2020) used the decision tree to do prediction .instead of clustering, he manually labels the student performance based on the percentage. The tree got 0.77 on validation. However, he didn't use the one-hot encoder for inconsistent features and also not limiting the tree depth, which might lead to overfitting.

the third one (Samuel joseph. ,2018)use ggplot for graph and did a lot of feature analysis. Which is great. She analysis the performance of different causes by gender. The relationship between feature contains much more information than simply analyzing the grade. She used linear regression to do the prediction, but she used MAE(mean absolute error) for the model evaluation.

The last one(skanderchaabini. ,2020) used two different methods for prediction: linear regression and classification. for the response value, he used the average score. For linear regression, he tried polynomial regression and multilinear regression and multilinear regression has the least R squared. He only used logistic regression for classification and the accuracy turns out to be 0.72

5 Future work

There are some points in my work that can be improved.

The first thing is to use a different clustering system. From the histogram of the score, the student performance is conformed to normal distribution. There is no clear difference between good or bad students. Kmeans might not be a perfect measure. It is possible to scale the score to 100 points and use the grading system used in school to give a letter grade for students.

The second improvement could be plotting for features with high relations to the scores. This could give us a general idea about whether to use classification or linear regression.

The third improvement could be more model options. From the others work, multilinear regression seems to have better performance than ordinal linear regression.

The last thing for improvement could be different ways to evaluate model performance. In my work, I simply use cross-validation for all the models. But it might be more accurate to use MSE or MAE for linear regression.

Conclusion

From the work and analysis about. I successfully built an SVC model than can have a 50% accuracy performance. I also find out the relationship between variables and determine the optimal number of clusters and features. Although there are some changes in method and numbers to the proposal, the result has proved the idea is feasible.

References

dipam7. (2018, September 14). Introduction to EDA and machine learning. Retrieved May 15, 2020, from Kaggle.com website: https://www.kaggle.com/dipam7/introduction-to-eda-and-machine-learning

hindelya. (2020, February 4). STUDENTS GRADE PREDICTION. Retrieved May 15, 2020, from Kaggle.com website: https://www.kaggle.com/hindelya/students-grade-prediction

samuelmjoseph. (2018, December 14). Student Grade Prediction using DecisionTree. Retrieved May 15, 2020, from Kaggle.com website: https://www.kaggle.com/samuelmjoseph/student-grade-prediction-using-decisiontree

skanderchaabini. (2020, May 5). Visualization, Regression and Classification. Retrieved May 15, 2020, from Kaggle.com website: https://www.kaggle.com/skanderchaabini/visualization-regression-and-classification

sklearn.feature_selection.RFE — scikit-learn 0.23.0 documentation. (2019). Retrieved May 14, 2020, from Scikit-learn.org website: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

sklearn.naive_bayes.MultinomialNB — scikit-learn 0.23.0 documentation. (2019). Retrieved May 14, 2020, from Scikit-learn.org website: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Diwei_zhu_Indproj_jupyternb

May 14, 2020

[1]:
```python
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import OneHotEncoder
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Perceptron
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_validate
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import train_test_split
```

[2]:
```python
data  = pd.read_csv('student/student-mat.csv',sep=";")
```

[3]:
```python
#data check
data.isna().values.any()
data.isnull().values.any()
data.head()
```

[3]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  …  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
```

```
0       4          3       4      1      1       3          6    5    6    6
1       5          3       3      1      1       3          4    5    5    6
2       4          3       2      2      3       3         10    7    8   10
3       3          2       2      1      1       5          2   15   14   15
4       4          3       2      1      2       5          4    6   10   10

[5 rows x 33 columns]
```

[4]:
```
#count the rows with 0 in scores
print(len(data[data.G1 == 0 ].index))
print(len(data[data.G2 == 0 ].index))
print(len(data[data.G3 == 0 ].index))
data = data[data.G3 !=0]
data.head()
```

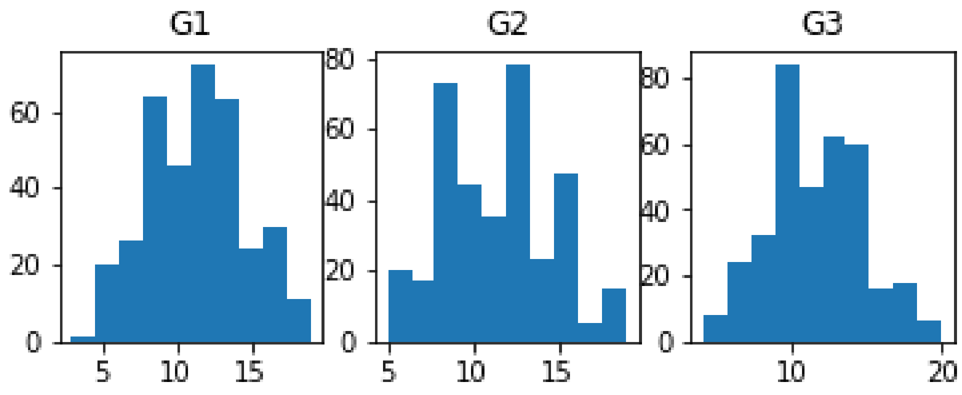
```
0
13
38
```

[4]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  …  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]
```

[5]:
```
#find out which score highly explain the final score


#linear regression
lr = LinearRegression(n_jobs=-1,fit_intercept=False)
lr.fit(data.iloc[:,-3:-2],data.iloc[:,-1])
print(lr.coef_,lr.intercept_,lr.score(data.iloc[:,-3:-2],data.iloc[:,-1]))
#coef shows that G2 has a big impact on G3

#histogram of grades
plt.figure(figsize=(6,2))
axg1 = plt.subplot(131)
```

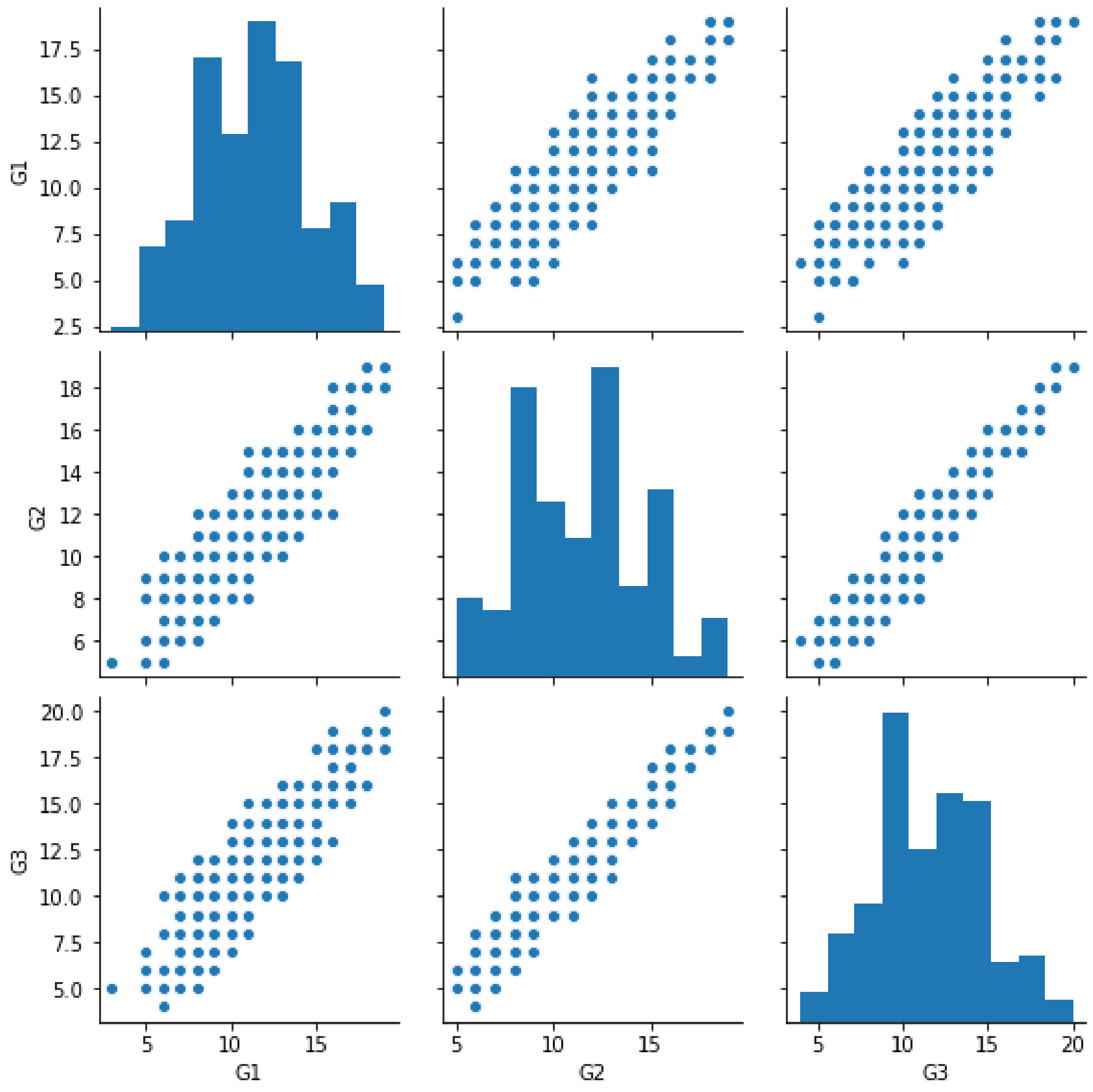
```
axg1.hist(data.iloc[:,-3])
axg1.title.set_text("G1")
axg2 = plt.subplot(132)
axg2.hist(data.iloc[:,-2])
axg2.title.set_text("G2")
axg3 = plt.subplot(133)
axg3.hist(data.iloc[:,-1])
axg3.title.set_text("G3")
plt.show()
#the grde is close to normal distribution

#using variance threshold
selector = VarianceThreshold()
# prefit object with df[cols]
selector.fit(data.iloc[:,-3:])
# check feature variances before selection
print(selector.variances_)
#conclusion: all 3 grade have big enough variance,G2 has bigger variance than G1

#pairplot for relationship
plt.figure(figsize=(4,3))
sns.pairplot(data=data,vars=["G1",'G2',"G3"])
plt.show()

plt.figure(figsize=(4,3))
cor = data.iloc[:,-3:].corr()
# visualize with Seaborn heat map, color map = Blues
sns.heatmap(cor, annot=True, cmap=plt.cm.Blues)
plt.show()
#although G2 has high relationship with 3, G1 cannot be ignore, G1 also has␣
↪high relationship with G2
#all two score have over 80% correlation with final grade, cannot take away any␣
↪of them
```

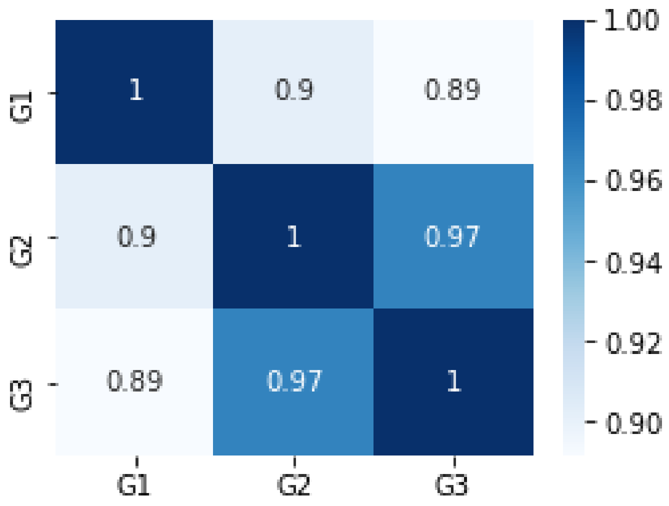
```
[1.01238971] 0.0 0.7785234530124574
```



```
[10.47110609  9.87704886 10.38948913]
```

```
<Figure size 288x216 with 0 Axes>
```





[6]:
```
#replace non numeric data to numbers

data_converted = data.copy(deep=1)
data_converted["school"].replace(['GP', 'MS'],[0,1],inplace = True)
data_converted["sex"].replace(['F', 'M'],[0,1],inplace = True)
data_converted["address"].replace(['U', 'R'],[0,1],inplace = True)
data_converted["famsize"].replace(['GT3', 'LE3'],[0,1],inplace = True)
data_converted["Pstatus"].replace(['A', 'T'],[0,1],inplace = True)
data_converted["schoolsup"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["famsup"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["paid"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["activities"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["nursery"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["higher"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["internet"].replace(['yes', 'no'],[0,1],inplace = True)
data_converted["romantic"].replace(['yes', 'no'],[0,1],inplace = True)

#use onehot encoder for features that are not continuous
enc = OneHotEncoder(sparse=False)
cats = ["Mjob","Fjob","reason","guardian"]
out_enc = enc.fit_transform(data_converted[cats])
new_cols = enc.get_feature_names(cats).tolist()
df_enc = pd.DataFrame(data = out_enc, columns = new_cols)
df_enc.index = data_converted.index
data_converted.drop(cats, axis=1, inplace=True)
data_converted = pd.concat([df_enc,data_converted], axis = 1)
print(data_converted.columns)

data_converted.head()
```

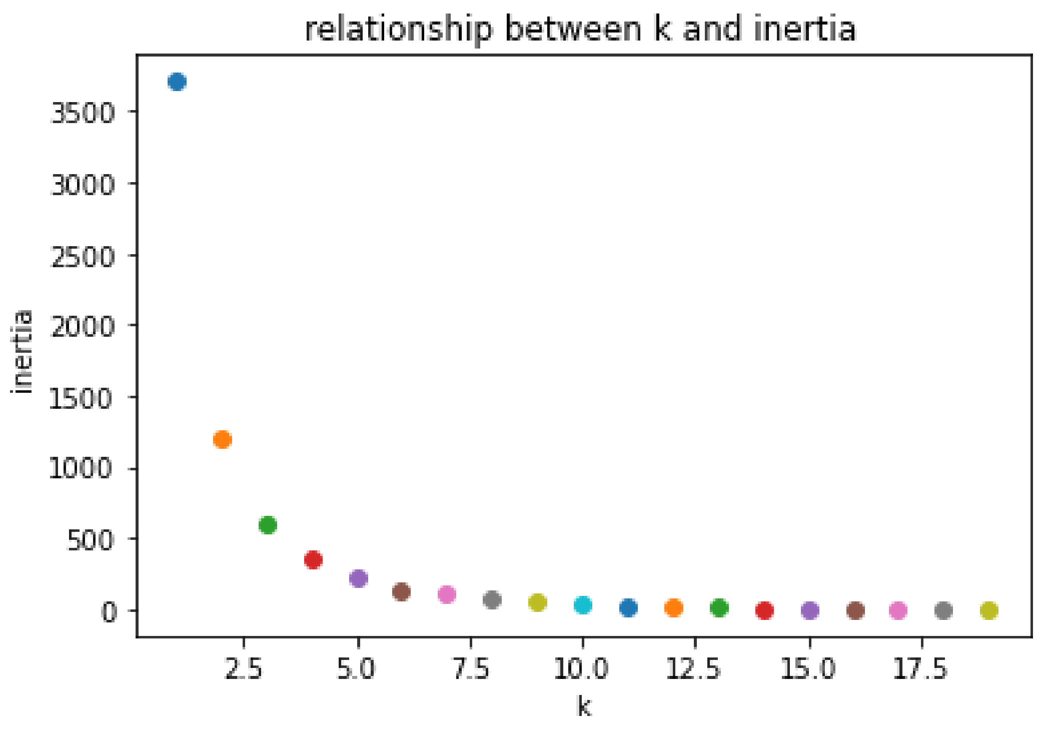
```
Index(['Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services',
       'Mjob_teacher', 'Fjob_at_home', 'Fjob_health', 'Fjob_other',
       'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home',
       'reason_other', 'reason_reputation', 'guardian_father',
       'guardian_mother', 'guardian_other', 'school', 'sex', 'age', 'address',
       'famsize', 'Pstatus', 'Medu', 'Fedu', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

[6]:
```
   Mjob_at_home  Mjob_health  Mjob_other  Mjob_services  Mjob_teacher  \
0           1.0          0.0         0.0            0.0           0.0
1           1.0          0.0         0.0            0.0           0.0
2           1.0          0.0         0.0            0.0           0.0
3           0.0          1.0         0.0            0.0           0.0
4           0.0          0.0         1.0            0.0           0.0

   Fjob_at_home  Fjob_health  Fjob_other  Fjob_services  Fjob_teacher  …  \
0           0.0          0.0         0.0            0.0           1.0  …
1           0.0          0.0         1.0            0.0           0.0  …
2           0.0          0.0         1.0            0.0           0.0  …
3           0.0          0.0         0.0            1.0           0.0  …
4           0.0          0.0         1.0            0.0           0.0  …

   famrel  freetime  goout  Dalc  Walc  health  absences  G1  G2  G3
0       4         3      4     1     1       3         6   5   6   6
1       5         3      3     1     1       3         4   5   5   6
2       4         3      2     2     3       3        10   7   8  10
3       3         2      2     1     1       5         2  15  14  15
4       4         3      2     1     2       5         4   6  10  10

[5 rows x 46 columns]
```

[7]:
```python
#find out how many cluster is suitable for clustering G3
for i in range(1,20):
    KM = KMeans(n_clusters=i, n_jobs= -1)
    KM.fit(data.iloc[:,-1].to_numpy().reshape(-1,1))
    plt.scatter(i,KM.inertia_)
plt.title("relationship between k and inertia")
plt.xlabel("k")
plt.ylabel("inertia")
plt.show()
#choose 3
```

```
<ipython-input-7-48e08526e8c3>:4: ConvergenceWarning: Number of distinct
clusters (17) found smaller than n_clusters (18). Possibly due to duplicate
points in X.
```

```
  KM.fit(data.iloc[:,-1].to_numpy().reshape(-1,1))
<ipython-input-7-48e08526e8c3>:4: ConvergenceWarning: Number of distinct
clusters (17) found smaller than n_clusters (19). Possibly due to duplicate
points in X.
  KM.fit(data.iloc[:,-1].to_numpy().reshape(-1,1))
```



[8]:
```
#all in one control panel
#group to cluster
group_to_cluster = 3
#variable to select
n_top = 15
#which score to use
score_to_use = "G3"
```

[9]:
```
#cluster by final score G3
KM = KMeans(n_clusters=group_to_cluster, n_jobs= -1,random_state = 80)
KM.fit(data[score_to_use].to_numpy().reshape(-1,1))
print(KM.inertia_)
grade_group = list()
for i,cen in  enumerate(KM.cluster_centers_):
    temp = list()
    temp.append(cen[0])
    temp.append(i)
    temp.append(list(KM.labels_).count(i))
    grade_group.append(temp)
grade_group = sorted(grade_group,key=lambda x:x[0],reverse=True)
#group center | label | count
print(grade_group)

#0-good;medium;k-bad
#kmean label - absolute label converter
kmean2absolute_dict = dict()

for i in range(len(grade_group)):
    kmean2absolute_dict.update({grade_group[i][1]:i})
label2insert = list([kmean2absolute_dict[x] for x in KM.labels_])
print(kmean2absolute_dict)

# insert the cluster label to dataframe as "perfromance" column
data_converted["performance"] = label2insert
```

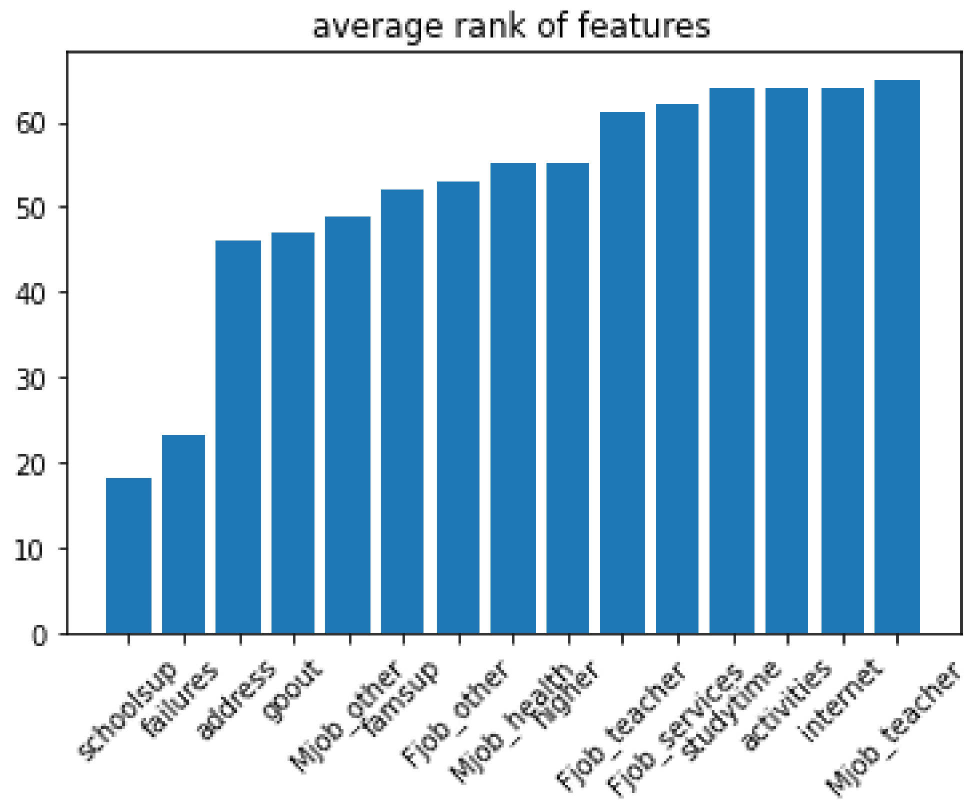
```
646.5
[[15.000000000000004, 1, 131], [10.500000000000004, 0, 162],
[7.0000000000000036, 2, 64]]
{1: 0, 0: 1, 2: 2}
```

[10]:
```
# set independent vars to X and dependent var to y
fX = data_converted.iloc[:,:-4]
fy = data_converted["performance"]

#use 4 different estimator and get the average
svc = SVC(kernel="linear", C=1,random_state=80)
rf = RandomForestClassifier(n_jobs=-1,random_state=80)
lr = LinearRegression(n_jobs=-1)
lg = LogisticRegression(n_jobs=-1)
rfe_lr = RFE(estimator=lr, n_features_to_select=1, step=1)
rfe_rf = RFE(estimator=rf, n_features_to_select=1, step=1)
rfe_svc = RFE(estimator=svc, n_features_to_select=1, step=1)
rfe_lg = RFE(estimator=lg, n_features_to_select=1, step=1)
rfe_lr.fit(fX, fy)
rfe_rf.fit(fX, fy)
rfe_svc.fit(fX, fy)
rfe_lg.fit(fX, fy)

#add up all the rank and find top 10 feature
combine_ranklist = list()
for i in range(len(rfe_lr.ranking_)):
    temp = list()
    temp.append(data_converted.columns[i])
    #print(data_converted.columns[i],rfe_lr.ranking_[i],rfe_rf.
 ↪ranking_[i],rfe_svc.ranking_[i])
```

```
    temp.append((rfe_lr.ranking_[i]+rfe_rf.ranking_[i]+rfe_svc.
↪ranking_[i]+rfe_lg.ranking_[i]))#
    #print(temp)
    combine_ranklist.append(temp)
combine_ranklist = sorted(combine_ranklist,key=lambda x :x[1])
#rank list
print(combine_ranklist)
feature_selected = list([x[0] for x in combine_ranklist[:n_top]])
#the column name of top10 feature
print(feature_selected)
```

```
[['schoolsup', 18], ['failures', 23], ['address', 46], ['goout', 47],
['Mjob_other', 49], ['famsup', 52], ['Fjob_other', 53], ['Mjob_health', 55],
['higher', 55], ['Fjob_teacher', 61], ['Fjob_services', 62], ['studytime', 64],
['activities', 64], ['internet', 64], ['Mjob_teacher', 65], ['Medu', 69],
['Fjob_at_home', 73], ['sex', 78], ['Dalc', 80], ['Walc', 83], ['Fjob_health',
88], ['Fedu', 89], ['romantic', 93], ['paid', 94], ['nursery', 97], ['health',
102], ['age', 103], ['famsize', 106], ['reason_course', 109], ['Pstatus', 110],
['guardian_mother', 117], ['Mjob_at_home', 120], ['Mjob_services', 121],
['reason_reputation', 121], ['traveltime', 123], ['freetime', 123], ['famrel',
126], ['absences', 127], ['reason_other', 128], ['guardian_father', 129],
['guardian_other', 129], ['reason_home', 134], ['school', 134]]
['schoolsup', 'failures', 'address', 'goout', 'Mjob_other', 'famsup',
'Fjob_other', 'Mjob_health', 'higher', 'Fjob_teacher', 'Fjob_services',
'studytime', 'activities', 'internet', 'Mjob_teacher']
```

[11]:
```
#sum of rank for the top ten feature

plt.bar([x for x in range(len(combine_ranklist[:n_top]))],[x[1] for x in␣
↪combine_ranklist[:n_top]],tick_label = [x[0] for x in combine_ranklist[:
↪n_top]])
plt.xticks(rotation = 45)
plt.title("average rank of features")
plt.show()
```

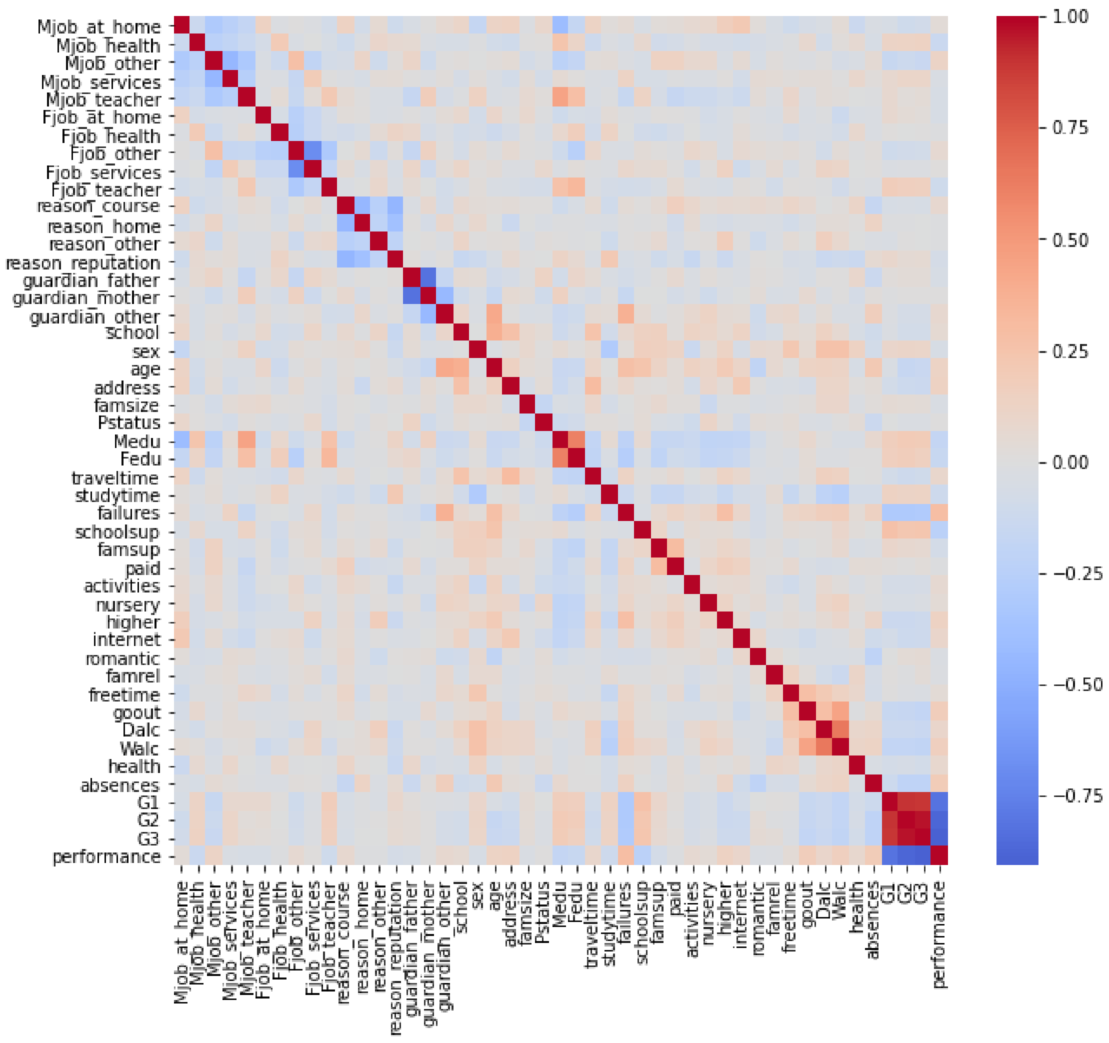


```
[12]: from sklearn.preprocessing import MinMaxScaler
      scaler = MinMaxScaler()
      # normalize the data and store in out_scaled numpy array
      out_scaled = scaler.fit_transform(data_converted.iloc[:,:-4])
      #convert the dataframe to normalized data
      data_converted[list(data_converted.iloc[:,:-4])] = out_scaled
      data_converted.iloc[:,:-4].head()
```

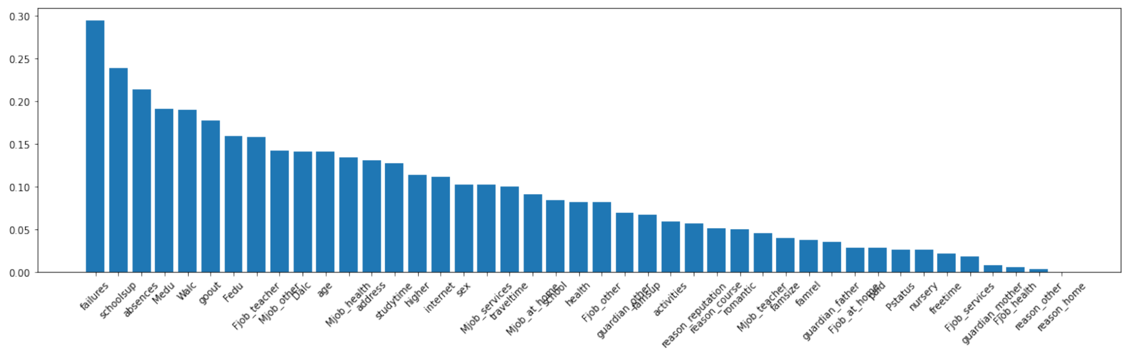
```
[12]:    Mjob_at_home  Mjob_health  Mjob_other  Mjob_services  Mjob_teacher  \
      0           1.0          0.0         0.0            0.0           0.0
      1           1.0          0.0         0.0            0.0           0.0
      2           1.0          0.0         0.0            0.0           0.0
      3           0.0          1.0         0.0            0.0           0.0
      4           0.0          0.0         1.0            0.0           0.0

         Fjob_at_home  Fjob_health  Fjob_other  Fjob_services  Fjob_teacher  …  \
      0           0.0          0.0         0.0            0.0           1.0  …
      1           0.0          0.0         1.0            0.0           0.0  …
      2           0.0          0.0         1.0            0.0           0.0  …
      3           0.0          0.0         0.0            1.0           0.0  …
      4           0.0          0.0         1.0            0.0           0.0  …
```

```
   higher  internet  romantic  famrel  freetime  goout  Dalc  Walc  health  \
0     0.0       1.0       1.0    0.75      0.50   0.75  0.00  0.00     0.5
1     0.0       0.0       1.0    1.00      0.50   0.50  0.00  0.00     0.5
2     0.0       0.0       1.0    0.75      0.50   0.25  0.25  0.50     0.5
3     0.0       0.0       0.0    0.50      0.25   0.25  0.00  0.00     1.0
4     0.0       1.0       1.0    0.75      0.50   0.25  0.00  0.25     1.0

   absences
0  0.080000
1  0.053333
2  0.133333
3  0.026667
4  0.053333

[5 rows x 43 columns]
```

[13]:
```
#feature correaltion
#since original dataframe has non numeric data, cannot use original datafram␣
 ↪for corr
f_cor = data_converted.corr()#.iloc[:,:-4]
# visualize with Seaborn heat map
plt.figure(figsize = (10,9))
sns.heatmap(f_cor, annot=False, cmap=plt.cm.coolwarm,center=0,xticklabels =␣
 ↪True, yticklabels = True,)
plt.show()
```



```
[14]: #plot yhe correlation of features to scores
      corr_dict =dict(f_cor["G3"])
      corr_dict = sorted(corr_dict.items(),key=lambda x : np.abs(x[1]),reverse= True )
      plt.figure(figsize=(20,5))
      plt.xticks(rotation = 45)
      plt.bar([x[0] for x in corr_dict[4:]],[np.abs(x[1]) for x in corr_dict[4:]])
      plt.show()
```



```
[15]: #build the traingin data
      x = data_converted[feature_selected]
      y = data_converted["performance"]
      x,y
```

```
[15]: (     schoolsup  failures  address  goout  Mjob_other  famsup  Fjob_other  \
      0          0.0  0.000000      0.0   0.75         0.0     1.0         0.0
      1          1.0  0.000000      0.0   0.50         0.0     0.0         1.0
      2          0.0  1.000000      0.0   0.25         0.0     1.0         1.0
      3          1.0  0.000000      0.0   0.25         0.0     0.0         0.0
      4          1.0  0.000000      0.0   0.25         1.0     0.0         1.0
      ..         …         …      …    …         …     …         …
      390        1.0  0.666667      0.0   0.75         0.0     0.0         0.0
      391        1.0  0.000000      0.0   1.00         0.0     1.0         0.0
      392        1.0  1.000000      1.0   0.50         1.0     1.0         1.0
      393        1.0  0.000000      1.0   0.00         0.0     1.0         1.0
      394        1.0  0.000000      0.0   0.50         1.0     1.0         0.0

           Mjob_health  higher  Fjob_teacher  Fjob_services  studytime  activities  \
      0            0.0     0.0           1.0            0.0   0.333333         1.0
      1            0.0     0.0           0.0            0.0   0.333333         1.0
      2            0.0     0.0           0.0            0.0   0.333333         1.0
      3            1.0     0.0           0.0            1.0   0.666667         0.0
      4            0.0     0.0           0.0            0.0   0.333333         1.0
      ..           …     …           …            …        …         …
      390          0.0     0.0           0.0            1.0   0.333333         1.0
      391          0.0     0.0           0.0            1.0   0.000000         1.0
      392          0.0     0.0           0.0            0.0   0.000000         1.0
      393          0.0     0.0           0.0            0.0   0.000000         1.0
      394          0.0     0.0           0.0            0.0   0.000000         1.0

           internet  Mjob_teacher
      0         1.0           0.0
      1         0.0           0.0
      2          0.0            0.0
      3          0.0            0.0
      4          1.0            0.0
      ..          …            …
      390        1.0            0.0
      391        0.0            0.0
      392        1.0            0.0
      393        0.0            0.0
      394        0.0            0.0

      [357 rows x 15 columns],
      0      2
      1      2
      2      1
      3      0
      4      1
            ..
      390    1
      391    0
      392    2
      393    1
      394    1
      Name: performance, Length: 357, dtype: int64)
```

[16]:
```
#predict on original data

clf = SVC(kernel="poly",random_state = 80)
scores = clf.fit(x,y)
print("SVC:", list(clf.predict(x) == y).count(1),list(clf.predict(x) == y).
 ↪count(1)/len(x))

clf_lr = LinearRegression()
scores = clf_lr.fit(x,y)
print("lr:", list([int(x) for x in clf_lr.predict(x)] == y).
 ↪count(1),list([int(x) for x in clf_lr.predict(x)] == y).count(1)/len(x))

clf_lgr = LogisticRegression(random_state = 80)
scores = clf_lgr.fit(x,y)
print("lgr:",list(clf_lgr.predict(x) == y).count(1),list(clf_lgr.predict(x) ==␣
 ↪y).count(1)/len(x))

clf_per = Perceptron(random_state = 80)
scores = clf_per.fit(x,y)
print("Perceptron:", list(clf_per.predict(x) == y).count(1),list(clf_per.
 ↪predict(x) == y).count(1)/len(x))
```

```
clf_knn = KNeighborsClassifier(n_neighbors=3,n_jobs=-1)
scores = clf_knn.fit(x,y)
print("knn:", list(clf_knn.predict(x) == y).count(1),list(clf_knn.predict(x) ==␣
 ↪y).count(1)/len(x))


clf_gnb = MultinomialNB()
scores = clf_gnb.fit(x,y)
print("bayes:", list(clf_gnb.predict(x) == y).count(1),list(clf_gnb.predict(x)␣
 ↪== y).count(1)/len(x))


clf_tree = DecisionTreeClassifier(random_state = 80,max_depth=n_top /2)
scores = clf_tree.fit(x,y)
print("decision tree:", list(clf_tree.predict(x) == y).count(1),list(clf_tree.
 ↪predict(x) == y).count(1)/len(x))
```

```
SVC: 262 0.7338935574229691
lr: 174 0.48739495798319327
lgr: 203 0.5686274509803921
Perceptron: 179 0.5014005602240896
knn: 247 0.6918767507002801
bayes: 171 0.4789915966386555
decision tree: 252 0.7058823529411765
```

[17]:
```
#shuffle cross validation

cv = ShuffleSplit(n_splits=100, test_size=0.33,random_state = 80)
clf = SVC(random_state = 80)
scores = cross_validate(clf, x, y, cv=cv,groups=[kmean2absolute_dict[x] for x ␣
 ↪in KM.labels_])
print("SVC:",scores["test_score"].mean())

clf_lr = LinearRegression()
scores = cross_validate(clf_lr, x, y, cv=cv,groups=[kmean2absolute_dict[x] for␣
 ↪x  in KM.labels_])
print("lr:",scores["test_score"].mean())

clf_lgr = LogisticRegression(random_state = 80)
scores = cross_validate(clf_lgr, x, y, cv=cv,groups=[kmean2absolute_dict[x] for␣
 ↪x  in KM.labels_])
print("lgr:",scores["test_score"].mean())

clf_per = Perceptron(random_state = 80)
scores = cross_validate(clf_per, x, y, cv=cv,groups=[kmean2absolute_dict[x] for␣
 ↪x  in KM.labels_],scoring="accuracy")
print("Perceptron:",scores["test_score"].mean())
```

```
clf_knn = KNeighborsClassifier(n_neighbors=3,n_jobs=-1)
scores = cross_validate(clf_knn, x, y, cv=cv,groups=[kmean2absolute_dict[x] for␣
 ↪x  in KM.labels_],scoring="accuracy")
print("knn:",scores["test_score"].mean())


clf_gnb = MultinomialNB()
scores = cross_validate(clf_gnb, x, y, cv=cv,groups=[kmean2absolute_dict[x] for␣
 ↪x  in KM.labels_])
print("bayes:",scores["test_score"].mean())


clf_tree = DecisionTreeClassifier(random_state = 80,max_depth=n_top /2)
scores = cross_validate(clf_tree, x, y, cv=cv,groups=[kmean2absolute_dict[x]␣
 ↪for x  in KM.labels_])
print("decision tree:",scores["test_score"].mean())
```

```
SVC: 0.48694915254237287
lr: 0.12813930672941326
lgr: 0.486186440677966
Perceptron: 0.42669491525423725
knn: 0.4597457627118644
bayes: 0.46593220338983043
decision tree: 0.4585593220338983
```