In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
df_mat = pd.read_csv('student-mat.csv',sep = ';')
```

In [3]:

```
df_mat.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	fr
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	

5 rows × 33 columns

In [4]:

```
df_mat.drop('school',axis = 1, inplace = True)
```

In [5]:

```
df_mat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 32 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   sex         395 non-null    object
 1   age         395 non-null    int64 
 2   address     395 non-null    object
 3   famsize     395 non-null    object
 4   Pstatus     395 non-null    object
 5   Medu        395 non-null    int64 
 6   Fedu        395 non-null    int64 
 7   Mjob        395 non-null    object
 8   Fjob        395 non-null    object
 9   reason      395 non-null    object
 10  guardian    395 non-null    object
 11  traveltime  395 non-null    int64 
 12  studytime   395 non-null    int64 
 13  failures    395 non-null    int64 
 14  schoolsup   395 non-null    object
 15  famsup      395 non-null    object
 16  paid        395 non-null    object
 17  activities  395 non-null    object
 18  nursery     395 non-null    object
 19  higher      395 non-null    object
 20  internet    395 non-null    object
 21  romantic    395 non-null    object
 22  famrel      395 non-null    int64 
 23  freetime    395 non-null    int64 
 24  goout       395 non-null    int64 
 25  Dalc        395 non-null    int64 
 26  Walc        395 non-null    int64 
 27  health      395 non-null    int64 
 28  absences    395 non-null    int64 
 29  G1          395 non-null    int64 
 30  G2          395 non-null    int64 
 31  G3          395 non-null    int64 
dtypes: int64(16), object(16)
memory usage: 98.9+ KB
```

In [52]:

```
df_mat.shape
```

Out[52]:

```
(395, 34)
```

In [6]:

```
df_mat.head()
```

Out[6]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	famrel	fr
0	F	18	U	GT3	A	4	4	at_home	teacher	course	...	4	
1	F	17	U	GT3	T	1	1	at_home	other	course	...	5	
2	F	15	U	LE3	T	1	1	at_home	other	other	...	4	
3	F	15	U	GT3	T	4	2	health	services	home	...	3	
4	F	16	U	GT3	T	3	3	other	other	home	...	4	

5 rows × 32 columns

In [7]:

```
# our dataset has some Non categorical Values let's change it:
#  categorize age into 3 classes:
def age(age):
    new_age=[]
    for i in age:
        if(i < 17):
            i=0
        elif (i < 19):
            i=1
        else:
            i=2
        new_age.append(i)
    return new_age
df_mat['age']=age(df_mat['age'])
df_mat.head()
```

Out[7]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	famrel	fr
0	F	1	U	GT3	A	4	4	at_home	teacher	course	...	4	
1	F	1	U	GT3	T	1	1	at_home	other	course	...	5	
2	F	0	U	LE3	T	1	1	at_home	other	other	...	4	
3	F	0	U	GT3	T	4	2	health	services	home	...	3	
4	F	0	U	GT3	T	3	3	other	other	home	...	4	

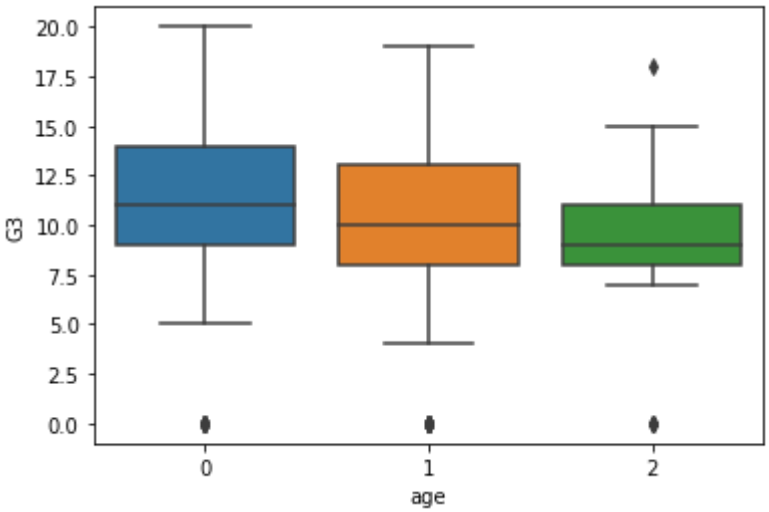
5 rows × 32 columns

In [8]:

```
sns.boxplot(x='age',y='G3',data=df_mat)
```

Out[8]:

```
<AxesSubplot:xlabel='age', ylabel='G3'>
```



Students of Age Less than 17 has higher median of scoring whereas students of age> 19 has lowest mean.

In [9]:

```
# comparing Performance of male and female students
df_mat['sex'].value_counts
```

Out[9]:

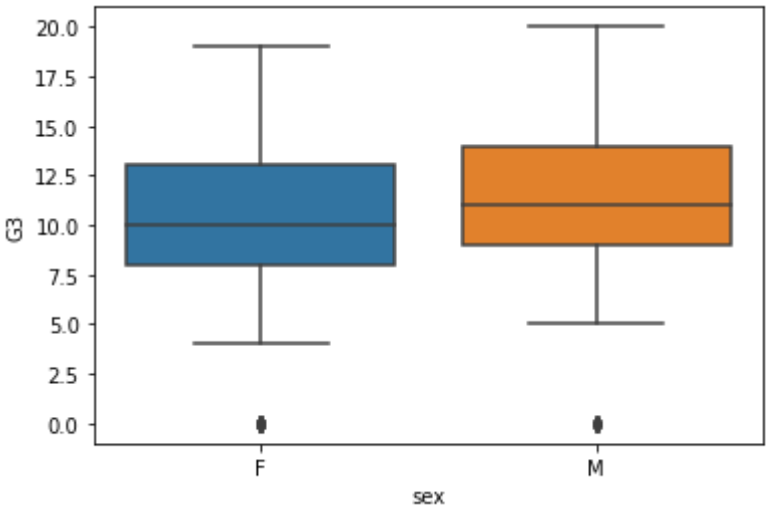
```
<bound method IndexOpsMixin.value_counts of 0      F
1      F
2      F
3      F
4      F
      ..
390    M
391    M
392    M
393    M
394    M
Name: sex, Length: 395, dtype: object>
```

In [10]:

```
sns.boxplot(x ='sex', y ='G3', data = df_mat)
```

Out[10]:

```
<AxesSubplot:xlabel='sex', ylabel='G3'>
```



Male students have better performance in school than the female students. Since the median score of boys is more than girls and the maximum marks also

In [11]:

```
df_mat['address'].value_counts()
```

Out[11]:

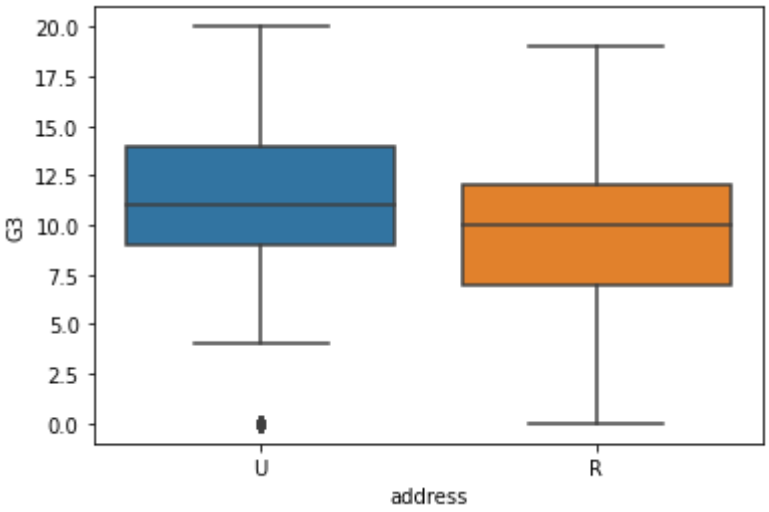
```
U    307
R     88
Name: address, dtype: int64
```

In [12]:

```
sns.boxplot('address','G3',data=df_mat)
```

Out[12]:

```
<AxesSubplot:xlabel='address', ylabel='G3'>
```



Students living in Urban Areas has better performance than students coming from Rural Areas.

In [13]:

```
df_mat['famsize'].value_counts()
```

Out[13]:

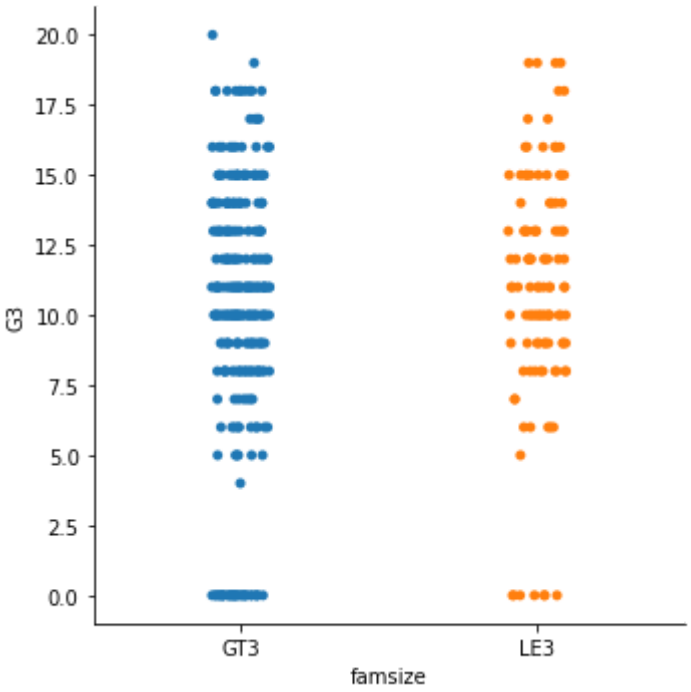
```
GT3    281
LE3    114
Name: famsize, dtype: int64
```

In [14]:

```
sns.catplot(x="famsize", y="G3", data=df_mat)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x18013b40d88>
```

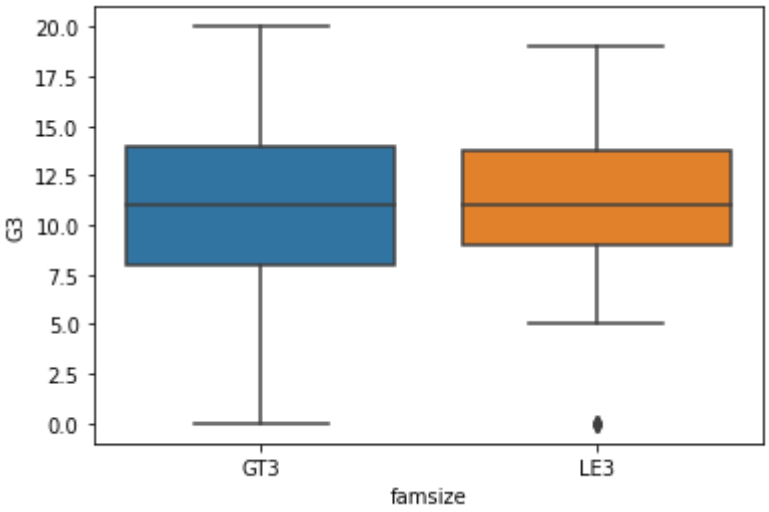


In [15]:

```
sns.boxplot(x = 'famsize', y = 'G3', data = df_mat)
```

Out[15]:

```
<AxesSubplot:xlabel='famsize', ylabel='G3'>
```



since family size doesn't tell much about this. I will drop this column.

In [16]:

```
df_mat.drop('famsize', axis = 1)
```

Out[16]:

	sex	age	address	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	...	famrel
0	F	1	U	A	4	4	at_home	teacher	course	mother	...	4
1	F	1	U	T	1	1	at_home	other	course	father	...	5
2	F	0	U	T	1	1	at_home	other	other	mother	...	4
3	F	0	U	T	4	2	health	services	home	mother	...	3
4	F	0	U	T	3	3	other	other	home	father	...	4
...	...	...	...	...	...	...	...	...	...	...	...	...
390	M	2	U	A	2	2	services	services	course	other	...	5
391	M	1	U	T	3	1	services	services	course	mother	...	2
392	M	2	R	T	1	1	other	other	course	other	...	5
393	M	1	R	T	3	2	services	other	course	mother	...	4
394	M	2	U	T	1	1	other	at_home	course	father	...	3

395 rows × 31 columns

In [17]:

```
df_mat['Pstatus'].value_counts()
```

Out[17]:

```
T    354
A     41
Name: Pstatus, dtype: int64
```

In [ ]:

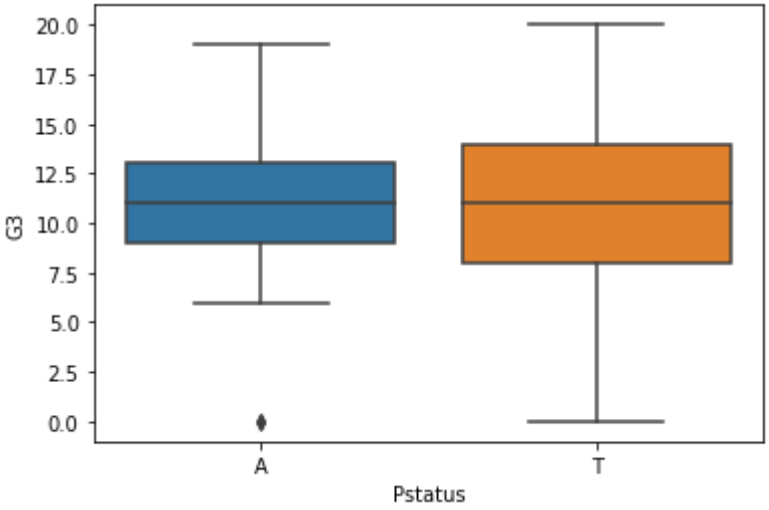
```

```

In [19]:

```
sns.boxplot("Pstatus","G3", data = df_mat)
```

Out[19]:

```
<AxesSubplot:xlabel='Pstatus', ylabel='G3'>
```



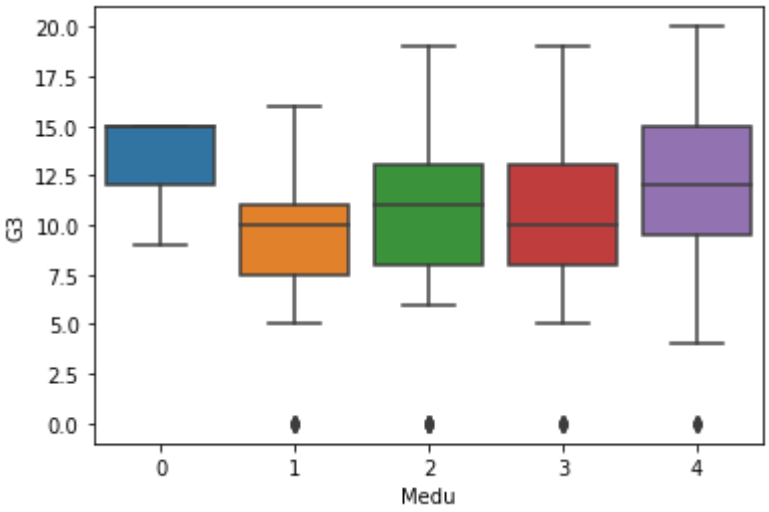
Students whose parents lived togethar tend to perfor better

In [20]:

```
sns.boxplot(x='Medu',y='G3',data=df_mat)
```

Out[20]:

```
<AxesSubplot:xlabel='Medu', ylabel='G3'>
```



Students whose mother has been more educated tend to perform better

In [21]:

```
df_mat['Fedu'].value_counts()
```

Out[21]:

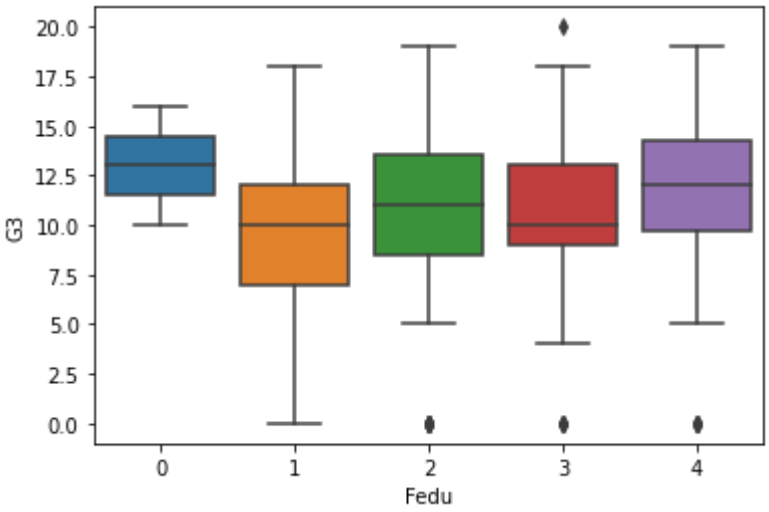
```
2    115
3    100
4     96
1     82
0      2
Name: Fedu, dtype: int64
```

In [22]:

```
sns.boxplot(x='Fedu',y='G3',data=df_mat)
```

Out[22]:

```
<AxesSubplot:xlabel='Fedu', ylabel='G3'>
```



Students whose fathers have completed higher education have slightly better chance of performing well in exams.

In [23]:

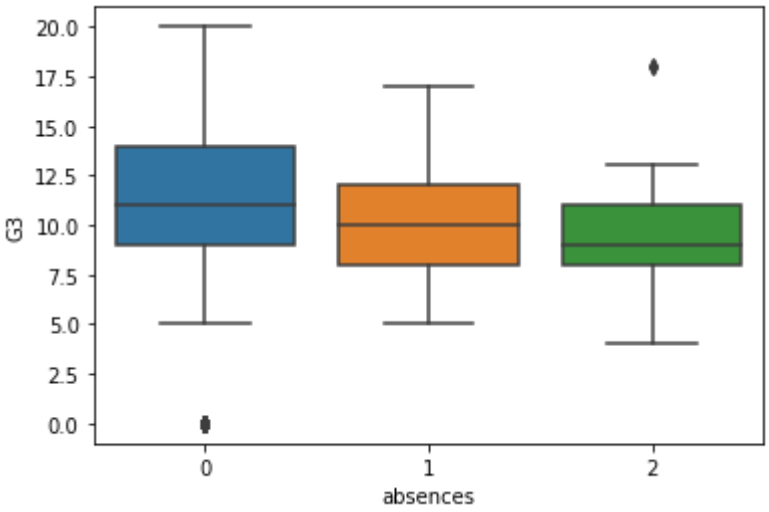
```
def absences(n):
    new=[]
    for i in n:
        if (i <= 10):
            i=0
        elif(i <= 20):
            i=1
        else:
            i=2
        new.append(i)
    return new
df_mat['absences']=absences(df_mat['absences'])
```

In [24]:

```
sns.boxplot("absences","G3", data = df_mat)
```

Out[24]:

```
<AxesSubplot:xlabel='absences', ylabel='G3'>
```



Students who has lower absents tends to perform better

In [25]:

```
# Now creating a new data field named as Grade:
# the grade will be the average of g1 , g2 and g3
# if average lies below 8 the student will fail else pass
df_mat['GradeAvg'] = (df_mat['G1'] + df_mat['G2'] + df_mat['G3'] ) /3
df_mat.head()
```

Out[25]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	freetim
0	F	1	U	GT3	A	4	4	at_home	teacher	course	...	
1	F	1	U	GT3	T	1	1	at_home	other	course	...	
2	F	0	U	LE3	T	1	1	at_home	other	other	...	
3	F	0	U	GT3	T	4	2	health	services	home	...	
4	F	0	U	GT3	T	3	3	other	other	home	...	

5 rows × 33 columns

In [26]:

```
# Creating a new column for pass or fail
def result(df):
    new_res = []
    for i in df:
        if i < 8:
            i = 0
        else:
            i = 1
        new_res.append(i)
    return new_res
df_mat['result'] = result(df_mat['GradeAvg'])
```

In [27]:

```
df_mat.head()
```

Out[27]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	goout	Da
0	F	1	U	GT3	A	4	4	at_home	teacher	course	...	4	
1	F	1	U	GT3	T	1	1	at_home	other	course	...	3	
2	F	0	U	LE3	T	1	1	at_home	other	other	...	2	
3	F	0	U	GT3	T	4	2	health	services	home	...	2	
4	F	0	U	GT3	T	3	3	other	other	home	...	2	

5 rows × 34 columns

In [28]:

```
df_mat['sex'] = df_mat['sex'].map({'M': 0, 'F': 1})
df_mat['address'] = df_mat['address'].map({'U': 0, 'R': 1})
df_mat['internet'] = df_mat['internet'].map({'no': 0, 'yes': 1})
df_mat['famsize'] = df_mat['famsize'].map({'LE3':0, 'GT3': 1})
df_mat['Pstatus'] = df_mat['Pstatus'].map({'A':0, 'T':'1'})
df_mat['Mjob'] = df_mat['Mjob'].map({'at_home':0, 'health':1, 'teacher':3,'services':4,'oth
df_mat['Fjob'] = df_mat['Fjob'].map({'at_home':0, 'health':1, 'teacher':3,'services':4,'oth
df_mat['reason'] = df_mat['reason'].map({'reputation':0, 'course':1,'home':2,'other':3})
df_mat['guardian'] = df_mat['guardian'].map({'father':0, 'mother':1,'other':2})

d= {'yes':0, 'no':1}
df_mat['schoolsup'] = df_mat['schoolsup'].map(d)
df_mat['famsup'] = df_mat['famsup'].map(d)
df_mat['paid'] = df_mat['paid'].map(d)
df_mat['activities'] = df_mat['activities'].map(d)
df_mat['nursery'] = df_mat['nursery'].map(d)
df_mat['higher'] = df_mat['higher'].map(d)
df_mat['romantic'] = df_mat['romantic'].map(d)
df_mat.head()
```

Out[28]:

	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	...	goout	Dalc	W
0	1	1	0	1	0	4	4	0	3	1	...	4	1	
1	1	1	0	1	1	1	1	0	5	1	...	3	1	
2	1	0	0	0	1	1	1	0	5	3	...	2	2	
3	1	0	0	1	1	4	2	1	4	2	...	2	1	
4	1	0	0	1	1	3	3	5	5	2	...	2	1	

5 rows × 34 columns

In [29]:

```
pd.isnull(df_mat).sum()
```

Out[29]:

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

In [73]:

```
mport KNeighborsClassifier
tion import train_test_split
, y_test = train_test_split(df_mat.drop(['G1','G2','G3','GradeAvg'],axis=1), df_mat['result'
er(n_neighbors=7)

)

ch model has not seen before
```

Out[73]:

```
0.8181818181818182
```

In [80]:

```
# applying Logistic regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
X_train, X_test, y_train, y_test = train_test_split(df_mat.drop(['G1','G2','G3','GradeAvg']
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[80]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [81]:

```
model.score(X_test,y_test)
```

Out[81]:

```
0.9711191335740073
```

In [ ]: