In [2]:
```
import pandas as pd
import numpy as np
import sklearn
```

In [3]:
```
dataset_path = "./student/student-mat.csv"
dataset = pd.read_csv(dataset_path, sep=";")
dataset.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freet
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]:
```
# number of column of dataset
len(dataset.columns)
```

Out[4]:
```
33
```

In [5]:
```
# 每行包含一些数值属性（第3、7、8、13、14、15、24、25、26、27、28、29、30列）和一些

# 将数据集中的数值属性分别存储在一个列表中，然后将这个列表存储在变量X中。


X = dataset.iloc[:, [2, 6, 7, 12, 13, 14, 23, 24, 25, 26, 27, 28, 29]].values

# 第33列包含数值形式的年终成绩。
Y = dataset.iloc[:, -1].values

# 使用最终成绩将数据分成四个部分，每个部分由一个四分位数确定（所以表示最低的四分之一
# 为了将数据分成四个部分，我们使用numpy.percentile函数，它返回数据中给定百分位数的值
# 例如，numpy.percentile(X, 25)返回X中25%的值。

split_1 = np.percentile(Y, 25)
split_2 = np.percentile(Y, 50)
split_3 = np.percentile(Y, 75)
split_4 = np.percentile(Y, 100)

print(split_1, split_2, split_3, split_4)
```

```
8.0 11.0 14.0 20.0
```

In [6]:
```
# 现在，使用数值特征和 k 均值算法对数据进行聚类。

# 数值特征聚类
from sklearn.cluster import KMeans
numerical_cluster = KMeans(n_clusters=4, random_state=0).fit(X)
# KNN聚类
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, numerical_cluster.labels_)
```

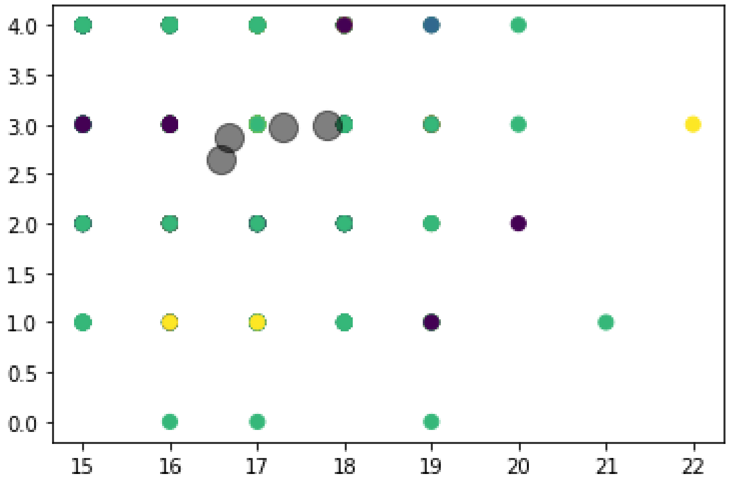
```
# 预测
knn.predict(X)
```

Out[6]:

```
array([0, 2, 0, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 3, 2, 2, 2,
       2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 3, 0, 2, 2,
       0, 0, 0, 2, 2, 2, 2, 2, 0, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 2, 2, 0, 2, 2, 2, 2, 1, 0, 0, 2, 2, 0, 2, 2, 0, 2, 2, 0, 2, 2,
       0, 3, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 3, 2, 2, 3, 2, 0, 0, 2, 0, 2,
       0, 2, 0, 0, 0, 2, 2, 2, 3, 0, 2, 0, 2, 3, 2, 2, 2, 2, 2, 0, 2, 2,
       0, 3, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2,
       2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 0, 2, 2, 1, 0, 0, 2, 2, 0, 2, 0, 2, 0, 0, 2, 2, 2, 0,
       3, 2, 2, 0, 2, 3, 0, 3, 0, 0, 0, 0, 0, 3, 2, 3, 0, 2, 3, 0, 2, 2,
       2, 2, 2, 2, 2, 3, 0, 2, 3, 0, 3, 2, 3, 2, 3, 0, 2, 3, 2, 2, 3, 2,
       2, 2, 2, 0, 2, 3, 0, 2, 2, 0, 2, 2, 2, 2, 0, 0, 0, 2, 3, 2, 2, 2,
       2, 3, 2, 0, 0, 2, 3, 2, 2, 2, 2, 0, 1, 3, 3, 0, 3, 3, 2, 2, 2, 2,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 2, 0, 2, 0, 3, 2, 2, 2, 3, 0, 2, 3,
       2, 3, 2, 3, 2, 3, 3, 1, 2, 0, 2, 2, 3, 0, 2, 2, 2, 2, 2, 0, 0, 2,
       2, 0, 2, 2, 2, 3, 0, 2, 0, 2, 2, 2, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2,
       0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 3,
       2, 2, 2, 2, 2, 3, 2, 0, 2, 2, 3, 2, 0, 2, 2, 2, 0, 2, 2, 2, 2])
```

In [7]:

```
# 展示聚类结果
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=numerical_cluster.labels_, s=50, cmap='viridis')
centers = numerical_cluster.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```



In [ ]: