1 线性模型

数据小鱼Rexa
CSDN：https://blog.csdn.net/qq_38395376?spm=1011.2124.3001.5343
Blibli：https://space.bilibili.com/283181288
Github：https://github.com/Rexa-Yu

1.1 简单的线性模型

In [221]:

```
import pandas as pd
import seaborn as sns
tips=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\seaborn-data-master
tips.head()
```

Out[221]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [222]:

```
# 问题研究：研究total_bill 对tip的影响
from sklearn import linear_model
lr=linear_model.LinearRegression()
print(type(tips["total_bill"]))
```

```
<class 'pandas.core.series.Series'>
```

In [223]:

```
tips["total_bill"]
```

Out[223]:

```
0      16.99
1      10.34
2      21.01
3      23.68
4      24.59
       ...
239    29.03
240    27.18
241    22.67
242    17.82
243    18.78
Name: total_bill, Length: 244, dtype: float64
```

In [263]:

```
# 由于sklearn的模型传入的数据需要numpy类型，所以我们需要转换类型
tips["total_bill"].values.reshape(-1,1)
```

Out[263]:

```
array([[16.99],
       [10.34],
       [21.01],
       [23.68],
       [24.59],
       [25.29],
       [ 8.77],
       [26.88],
       [15.04],
       [14.78],
       [10.27],
       [35.26],
       [15.42],
       [18.43],
       [14.83],
       [21.58],
       [10.33],
       [16.29],
```

In [225]:

```
# 拟合
predicted=lr.fit(X=tips["total_bill"].values.reshape(-1,1),y=tips["tip"].values.resh
```

In [226]:

```
# 获取系数
predicted.coef_
```

Out[226]:

```
array([[0.10502452]])
```

In [227]:

```
# 获取截距
predicted.intercept_
```

Out[227]:

```
array([0.92026961])
```

1.2 多元线性回归

In [228]:

```
# 导入多个变量无需重置X，因此不用reshape
#  Series可以重置为数组，可以使用xxxxx.values.reshape(-1,1)
predicted=lr.fit(X=tips[["total_bill","size"]],y=tips["tip"].values.reshape(-1,1))
```

In [229]:

```
predicted.coef_
```

Out[229]:

```
array([[0.09271334, 0.19259779]])
```

In [230]:

```
predicted.intercept_
```

Out[230]:

```
array([0.66894474])
```

1.3 使用get_dummies将分类变量换成虚拟变量

In [231]:

```
tips
```

Out[231]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [232]:

```
tips_dummy=pd.get_dummies(tips[["total_bill","size","sex","smoker","day","time"]],dr
tips_dummy
```

Out[232]:

	total_bill	size	sex_Male	smoker_Yes	day_Sat	day_Sun	day_Thur	time_Lunch
0	16.99	2	0	0	0	1	0	0
1	10.34	3	1	0	0	1	0	0
2	21.01	3	1	0	0	1	0	0
3	23.68	2	1	0	0	1	0	0
4	24.59	4	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...
239	29.03	3	1	0	1	0	0	0
240	27.18	2	0	1	1	0	0	0
241	22.67	2	1	1	1	0	0	0
242	17.82	2	1	0	1	0	0	0
243	18.78	2	0	0	0	0	1	0

244 rows × 8 columns

In [233]:

```
# 将虚拟变量带入模型
predicted=lr.fit(X=tips_dummy,y=tips["tip"].values.reshape(-1,1))
```

In [234]:

```
predicted.coef_
```

Out[234]:

```
array([[ 0.09448701,  0.175992  , -0.03244094, -0.08640832, -0.1214583
8,
        -0.02548066, -0.1622592 ,  0.0681286 ]])
```

In [235]:

```
predicted.intercept_
```

Out[235]:

```
array([0.80381728])
```

1.4 广义的线性模型（逻辑回归）

1.5 逻辑回归

In [236]:

```
# 当因变量是2值类的时候，可以使用逻辑回归
import pandas as pd
acs=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/acs_ny.csv")
acs.head()
```

Out[236]:

	Acres	FamilyIncome	FamilyType	NumBedrooms	NumChildren	NumPeople	NumRooms	Nur
0	1-10	150	Married	4	1	3	9	de
1	1-10	180	Female Head	3	2	4	6	de
2	1-10	280	Female Head	4	0	2	8	de
3	1-10	330	Female Head	2	1	2	4	de
4	1-10	330	Male Head	3	1	2	5	at

In [237]:

```
# 创造二值变量
acs["ge150k"]=pd.cut(acs["FamilyIncome"],[0,15000,acs["FamilyIncome"].max()],labels=
# cut函数切割，第一个参数是所要切的数据集，第二个是按照什么区间切，例如：0-15000，15000-最大，la
acs["ge150k"]
```

Out[237]:

```
0        0
1        0
2        0
3        0
4        0
        ..
22740    1
22741    1
22742    1
22743    1
22744    1
Name: ge150k, Length: 22745, dtype: category
Categories (2, int64): [0 < 1]
```

In [238]:

```
acs["ge150k_i"]=acs["ge150k"].astype(int)
```

In [239]:

```
acs
```

Out[239]:

	Acres	FamilyIncome	FamilyType	NumBedrooms	NumChildren	NumPeople	NumRooms	NumUnits	Num
0	1-10	150	Married	4	1	3	9	Single detached	
1	1-10	180	Female Head	3	2	4	6	Single detached	
2	1-10	280	Female Head	4	0	2	8	Single detached	
3	1-10	330	Female Head	2	1	2	4	Single detached	
4	1-10	330	Male Head	3	1	2	5	Single attached	
...	...	...	...	...	...	...	...	...	
22740	10+	565000	Married	5	3	5	10	Single detached	

In [240]:

```
# 进行虚拟变量化
predictors=pd.get_dummies(acs[["HouseCosts","NumWorkers","OwnRent","NumBedrooms","Fa
predictors
```

Out[240]:

	HouseCosts	NumWorkers	NumBedrooms	OwnRent_Outright	OwnRent_Rented	FamilyTy
0	1800	0	4	0	0	
1	850	0	3	0	1	
2	2600	1	4	0	0	
3	1800	0	2	0	1	
4	860	0	3	0	0	
...	...	...	...	...	...	
22740	1700	2	5	0	0	
22741	1300	2	4	0	0	
22742	410	3	4	0	0	
22743	1600	3	3	0	0	
22744	6500	2	4	0	0	

22745 rows × 7 columns

In [241]:

```
from sklearn import linear_model
lr=linear_model.LogisticRegression()
result = lr.fit(X=predictors,y=acs["ge150k_i"].values.reshape(-1,1).ravel())
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:762: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as show
n in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic
-regression (https://scikit-learn.org/stable/modules/linear_model.html
#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [242]:

```
result.coef_
```

Out[242]:

```
array([[ 3.26317321e-04,  1.41682531e+00,  1.16007943e-01,
         2.38272067e-01, -1.53203820e+00,  3.78355122e-01,
         1.34553573e+00]])
```

In [243]:

```
result.intercept_
```

Out[243]:

```
array([0.41135342])
```

2 聚类

2.1 Kmeans

In [244]:

```
import pandas as pd
wine=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/wine.csv")
wine.head()
```

Out[244]:

	Cultivar	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Pro:
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	

In [245]:

```
# 从表中看Cultival属性过于密集，即需要删除
wine=wine.drop("Cultivar",axis=1)
wine.head()
```

Out[245]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyan
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1

In [246]:

```
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=3,random_state=42).fit(wine)
print(kmeans)
```

```
KMeans(n_clusters=3, random_state=42)
```

In [247]:

```
import numpy as np
print(np.unique(kmeans.labels_,return_counts=True))
```

```
(array([0, 1, 2]), array([62, 47, 69], dtype=int64))
```

In [248]:

```
# 聚类制造标签
kmeans_3=pd.DataFrame(kmeans.labels_,columns=["cluster"])
print(kmeans_3)
```

```
     cluster
0          1
1          1
2          1
3          1
4          0
..       ...
173        0
174        0
175        0
176        0
177        2

[178 rows x 1 columns]
```

In [249]:

```
# 使用PCA降维
from sklearn.decomposition import PCA
# n_components = 主成分个数
pca=PCA(n_components=2).fit(wine)
```

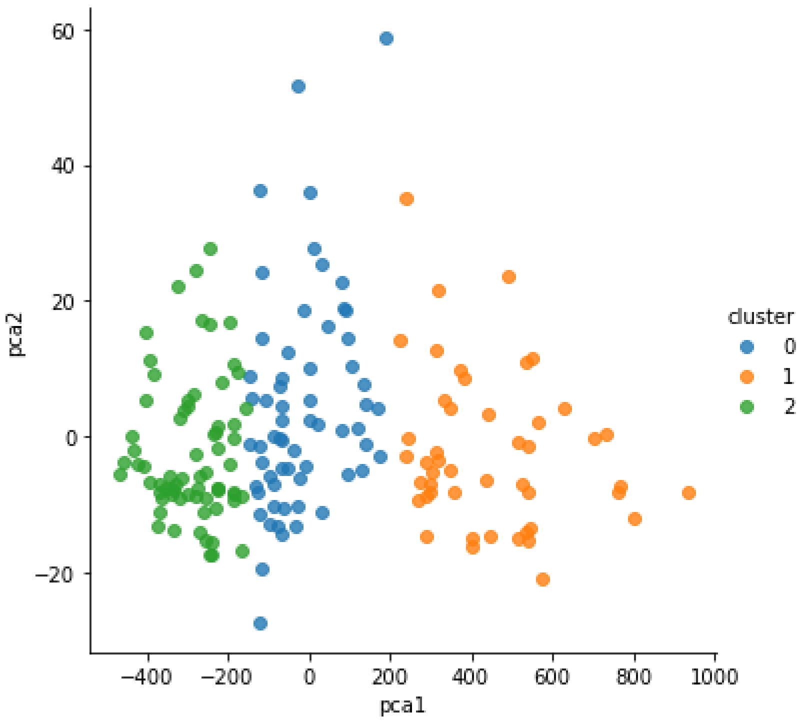
In [250]:

```
# 数据迁移
pca_trans=pca.transform(wine)
# 给投影命名
pca_trans_df=pd.DataFrame(pca_trans,columns=["pca1","pca2"])
# 连接数据
kmeans_3=pd.concat([kmeans_3,pca_trans_df],axis=1)
print(kmeans_3.head())
```

```
   cluster        pca1       pca2
0        1  318.562979  21.492131
1        1  303.097420  -5.364718
2        1  438.061133  -6.537309
3        1  733.240139   0.192729
4        0  -11.571428  18.489995
```

In [251]:

```
# 作图
import seaborn as sns
import matplotlib.pyplot as plt
fig=sns.lmplot(x="pca1",y="pca2",data=kmeans_3,hue="cluster",fit_reg=False)
plt.show()
```



In [252]:

```python
# 加入cultivar属性
wine_all=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/wine.csv")
wine.head()
```

Out[252]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyan
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1

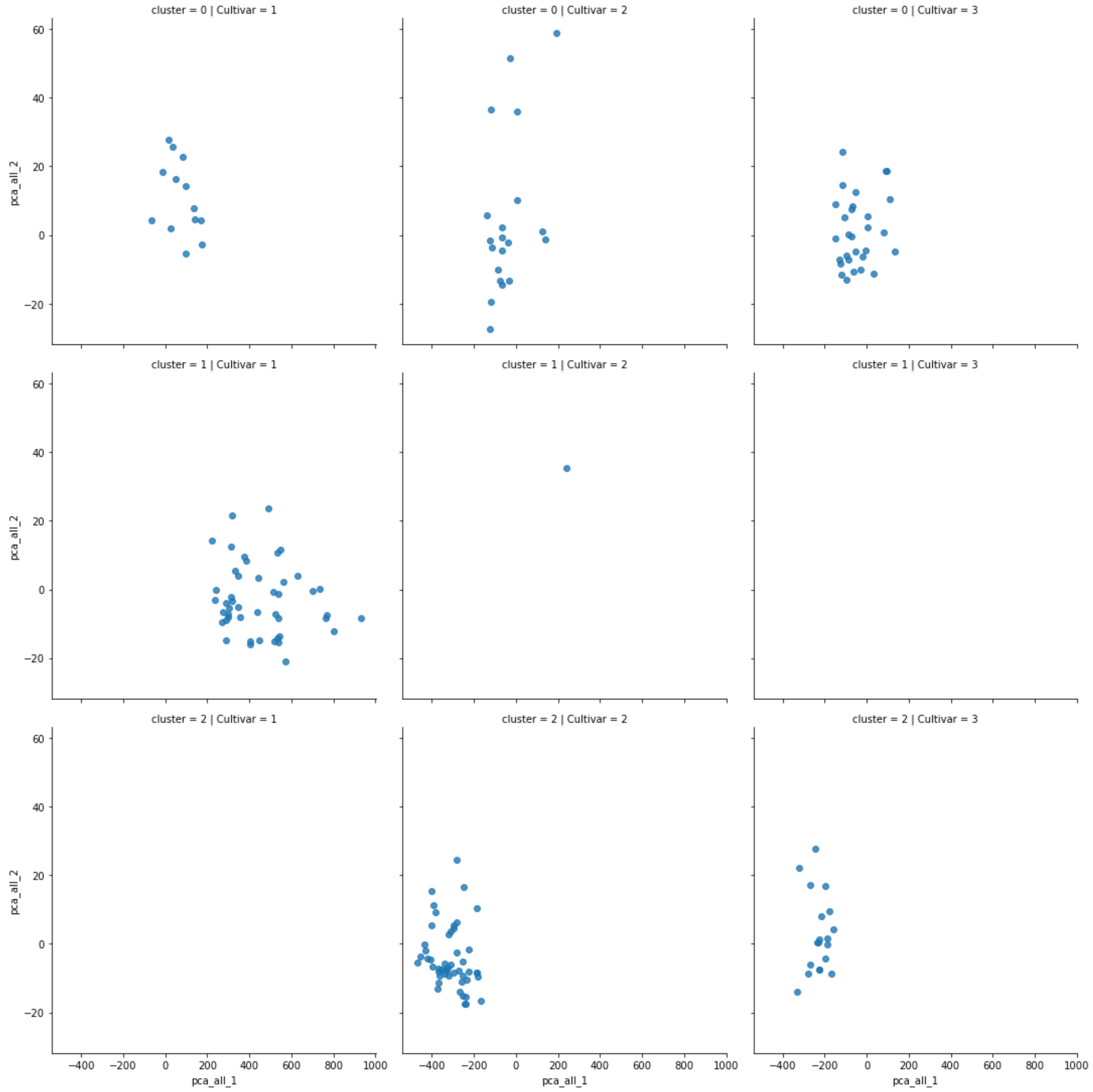
In [253]:

```python
pca_all=PCA(n_components=2).fit(wine_all)
pca_all_trans=pca_all.transform(wine_all)
pca_all_trans_df=pd.DataFrame(pca_all_trans,columns=["pca_all_1","pca_all_2"])
print(pca_all_trans)
kmeans_3=pd.concat([kmeans_3,pca_all_trans_df,wine_all["Cultivar"]],axis=1)
kmeans_3
```

```
[[ 3.18564055e+02  2.14907729e+01]
 [ 3.03098514e+02 -5.36608268e+00]
 [ 4.38062063e+02 -6.53798613e+00]
 [ 7.33240711e+02  1.93319951e-01]
 [-1.15699516e+01  1.84872549e+01]
 [ 7.03231800e+02 -3.31723191e-01]
 [ 5.42972384e+02 -1.35191971e+01]
 [ 5.48402656e+02  1.14491475e+01]
 [ 2.98037964e+02 -8.18149185e+00]
 [ 2.98050654e+02 -7.10283985e+00]
 [ 7.63080247e+02 -8.33356923e+00]
 [ 5.32944043e+02 -1.42878810e+01]
 [ 5.72835177e+02 -2.10050612e+01]
 [ 4.02926332e+02 -1.61035066e+01]
 [ 8.00053884e+02 -1.21175637e+01]
 [ 5.63246356e+02  2.21467383e+00]
 [ 5.33380465e+02  1.08044932e+01]
 [ 3.83318588e+02  8.47646622e+00]
 [ 9.33118716e+02 -8.35296527e+00]
```

In [254]:

```
with sns.plotting_context(font_scale=5):
    fig=sns.lmplot(x="pca_all_1",y="pca_all_2",data=kmeans_3,row="cluster",col="Cult
plt.show()
```



In [255]:

```
# 查看交叉表的频率计数
print(pd.crosstab(kmeans_3["cluster"],kmeans_3["Cultivar"],margins=True))
```

```
Cultivar   1   2   3  All
cluster
0         13  20  29   62
1         46   1   0   47
2          0  50  19   69
All       59  71  48  178
```

2.2 层次聚类

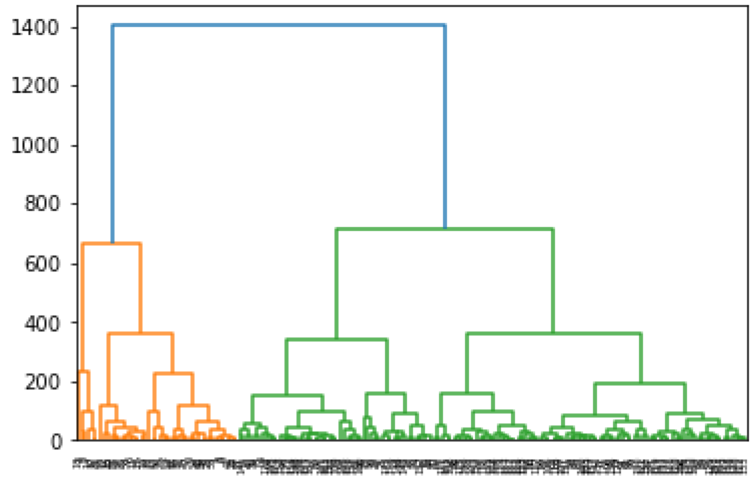
2.2.1 最长距离法

In [256]:

```
from scipy.cluster import hierarchy
wine=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/wine.csv")
wine=wine.drop("Cultivar",axis=1)
```

In [257]:

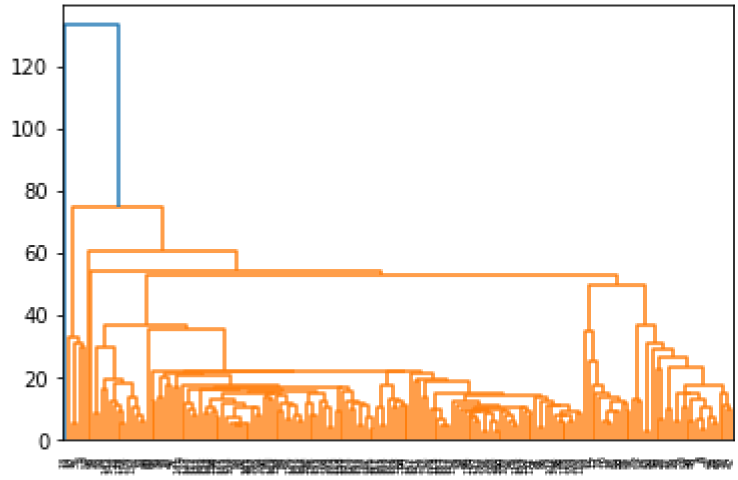
```
wine_complete=hierarchy.complete(wine)
fig=plt.figure()
dn=hierarchy.dendrogram(wine_complete)
plt.show()
```



2.2.2 最短距离法

In [258]:

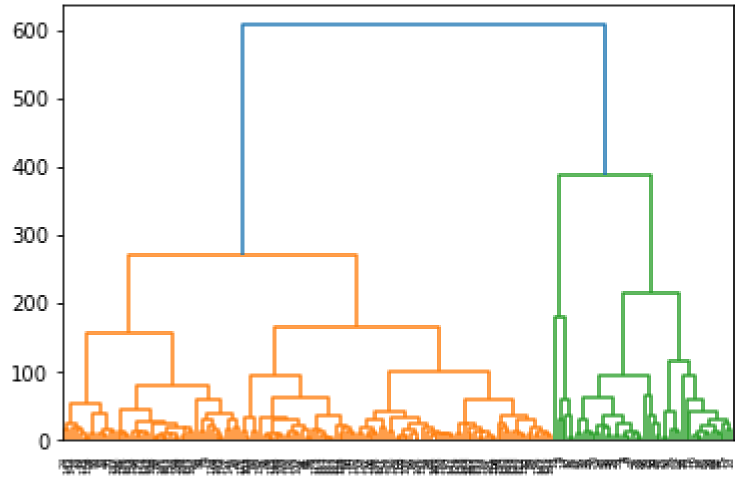
```
wine_single=hierarchy.single(wine)
fig=plt.figure()
dn=hierarchy.dendrogram(wine_single)
plt.show()
```



2.2.3 平均距离法

In [259]:

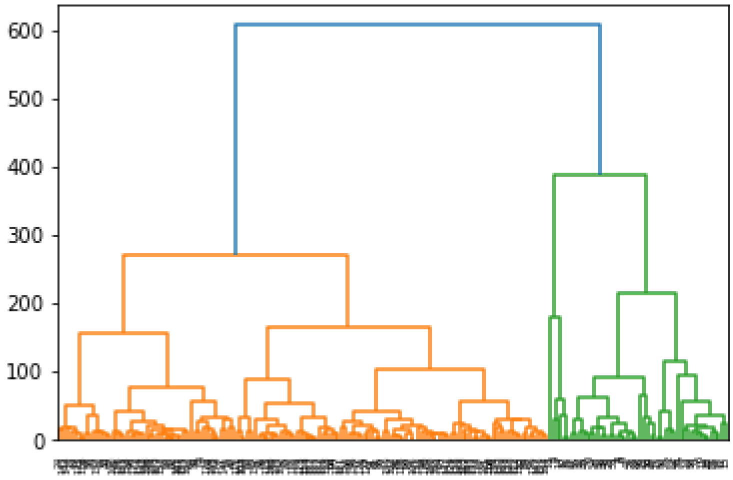
```
wine_average=hierarchy.average(wine)
fig=plt.figure()
dn=hierarchy.dendrogram(wine_average)
plt.show()
```



2.2.4 重心法

In [260]:

```
wine_centroid=hierarchy.centroid(wine)
fig=plt.figure()
dn=hierarchy.dendrogram(wine_centroid)
plt.show()
```



2.2.5 手动阈值法

In [261]:

```
wine_complete=hierarchy.complete(wine)
fig=plt.figure()
# 设置默认的matlab阈值
dn=hierarchy.dendrogram(wine_complete,color_threshold=0.7*max(wine_complete[:,2]),ak
plt.show()
```