In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns


# Importing the dataset
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:, 4].values
```

In [2]:
```
head = dataset.head() # First 5 line
print("head",head)
tail = dataset.tail() # Last 5 line
print("tail",tail)
describe = dataset.describe() #summary statistics for numerical columns
print("describe",describe)
info = dataset.info() #index, datatype and memory information
print("info",info)
max = dataset.max() # Returns the highest value in each column
print("max",max)
min = dataset.min() # Returns the lowest value in each column
print("min",min)
median = dataset.median() # Returns the medians value in each column
print("median",median)
```

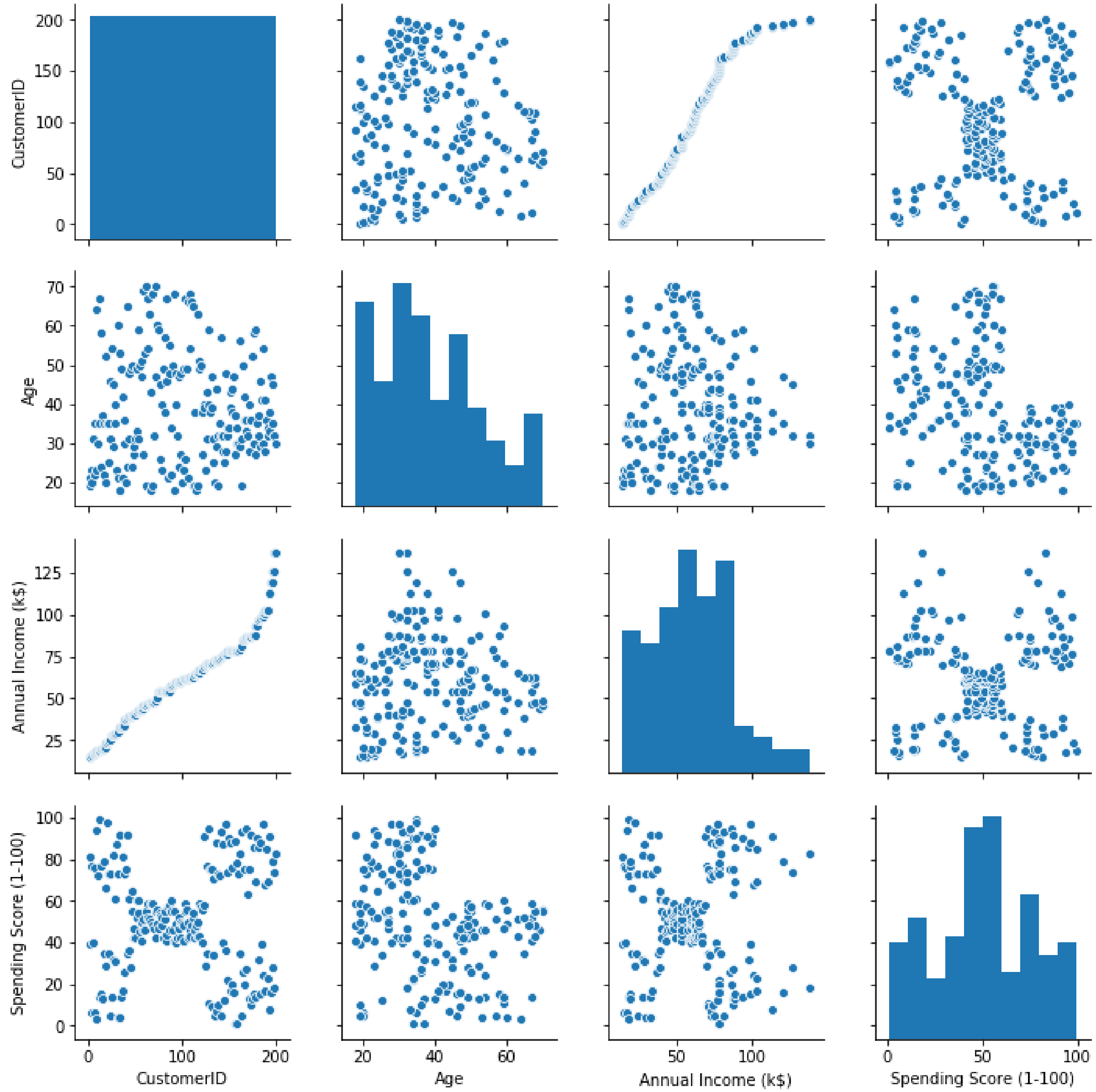
```
head    CustomerID   Genre   Age  Annual Income (k$)  Spending Score (1-
100)
0            1    Male   19                  15                      39
1            2    Male   21                  15                      81
2            3  Female   20                  16                       6
3            4  Female   23                  16                      77
4            5  Female   31                  17                      40
tail      CustomerID   Genre   Age  Annual Income (k$)  Spending Score
(1-100)
195         196  Female   35                 120                      7
```

```
9
196         197  Female   45                 126                      2
8
197         198    Male   32                 126                      7
4
198         199    Male   32                 137                      1
8
199         200    Male   30                 137                      8
3
describe        CustomerID         Age  Annual Income (k$)  Spending Sc
ore (1-100)
count  200.000000  200.000000          200.000000                200.0000
00
mean   100.500000   38.850000           60.560000                 50.2000
00
std     57.879185   13.969007           26.264721                 25.8235
22
min      1.000000   18.000000           15.000000                  1.0000
00
25%     50.750000   28.750000           41.500000                 34.7500
00
50%    100.500000   36.000000           61.500000                 50.0000
00
75%    150.250000   49.000000           78.000000                 73.0000
00
max    200.000000   70.000000          137.000000                 99.0000
00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID                200 non-null int64
Genre                     200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
info None
max CustomerID                 200
```

```
Genre                         Male
Age                             70
Annual Income (k$)             137
Spending Score (1-100)          99
dtype: object
min CustomerID                       1
Genre                         Female
Age                               18
Annual Income (k$)                15
Spending Score (1-100)             1
dtype: object
median CustomerID                  100.5
Age                            36.0
Annual Income (k$)             61.5
Spending Score (1-100)         50.0
dtype: float64
```

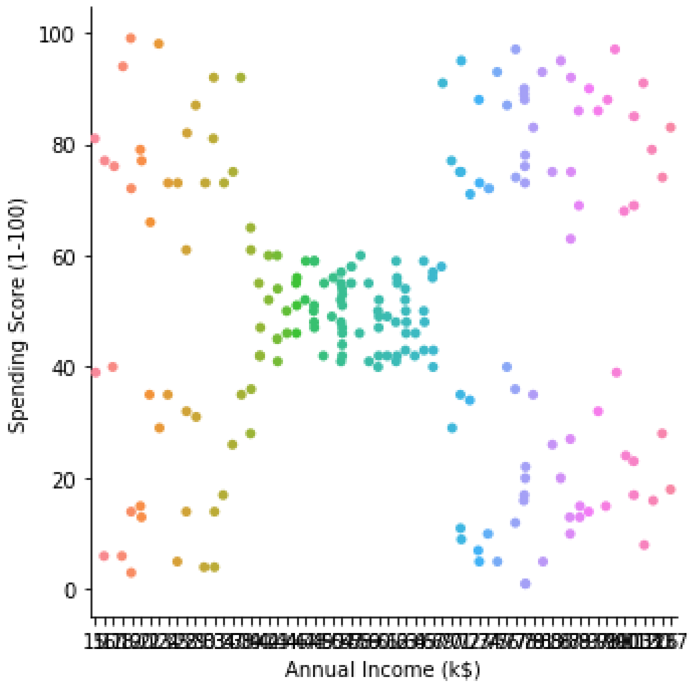
In [3]:
```
sns.pairplot(dataset)
```

Out[3]:
```
<seaborn.axisgrid.PairGrid at 0x294374b79b0>
```



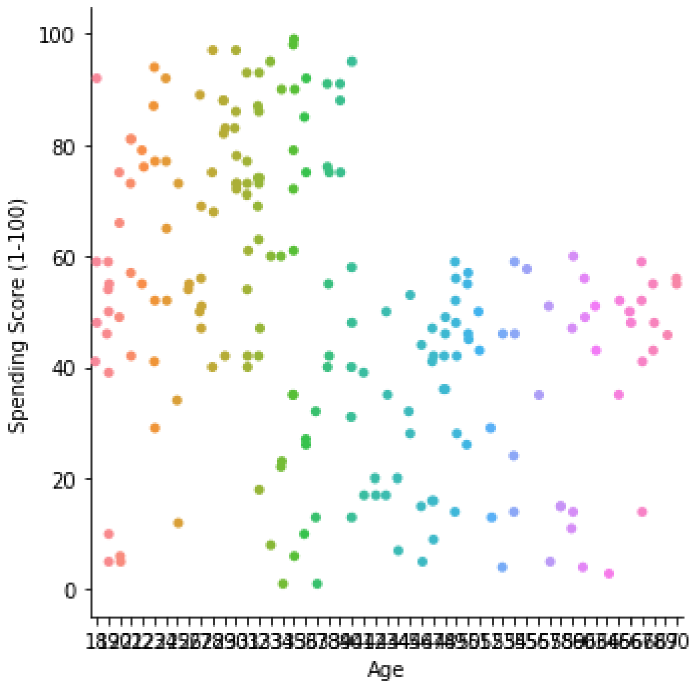

In [4]:
```
sns.catplot(x="Annual Income (k$)", y="Spending Score (1-100)", data=da
taset)
```

Out[4]: <seaborn.axisgrid.FacetGrid at 0x2943c74da90>



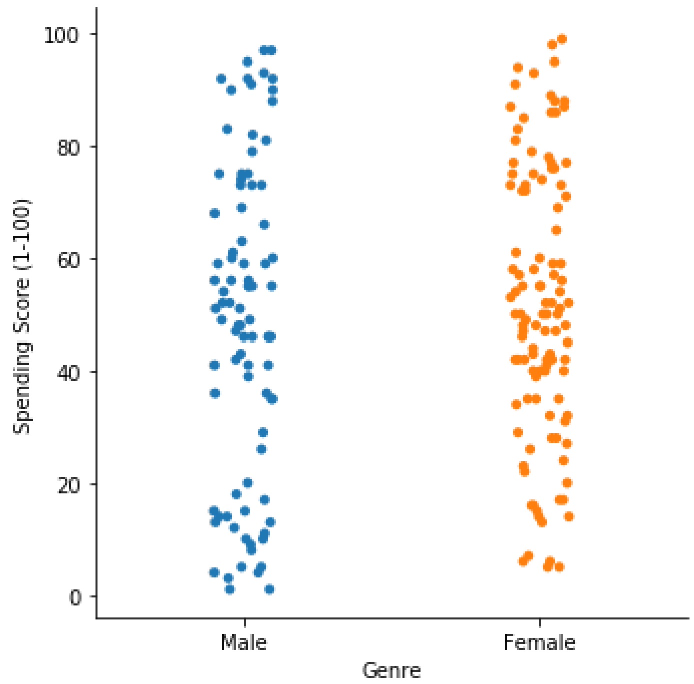

In [5]:
```
sns.catplot(x="Age", y="Spending Score (1-100)", data=dataset)
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x2943cb19400>




```
In [6]: sns.catplot(x="Genre", y="Spending Score (1-100)", data=dataset)
```

```
Out[6]: <seaborn.axisgrid.FacetGrid at 0x2943cc868d0>
```




In [7]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

X = np.expand_dims(X, axis=1)
y = np.expand_dims(y, axis=1)

columns = ["Age","Annual Income (k$)","Spending Score (1-100)"]

for col in columns:
    X = dataset.iloc[:, [2,3]].values
    y = dataset.iloc[:, 4].values
    X_train, X_test, y_train, y_test = train_test_split(X,y,random_stat
e=42)
    reg = DecisionTreeRegressor()
    reg.fit(X_train,y_train)
```

```
    print('Score for {} as dependent variable is {}'.format(col,reg.sco
re(X_test,y_test)))
```

```
Score for Age as dependent variable is -0.4557775726717237
Score for Annual Income (k$) as dependent variable is -0.45458904421257
95
Score for Spending Score (1-100) as dependent variable is -0.4619664101
482672
```

In [8]:
```
##########           PCA

from sklearn.decomposition import PCA
pca = PCA(n_components=3)
pca.fit(dataset.iloc[:, [2,3,4]].values)
pca.explained_variance_ratio_
```

Out[8]:
```
array([0.45125272, 0.44098465, 0.10776263])
```

In [9]:
```
# Sum of explained variances of the first two components is 89%
pca.explained_variance_ratio_[0]+pca.explained_variance_ratio_[1]
```

Out[9]:
```
0.8922373735506914
```

In [10]:
```
print(pca.components_)
```

```
[[-0.1889742   0.58864102  0.7859965 ]
 [ 0.1309652   0.80837573 -0.57391358]
 [ 0.97320957  0.00551667  0.22985365]]
```

In [11]:
```
dimensions = ['Dimension {}'.format(i) for i in range(1,len(pca.compone
nts_)+1)]

components = pd.DataFrame(pca.components_,columns=["Age","Annual Income
(k$)","Spending Score (1-100)"])
components.index = dimensions

variance = pd.DataFrame(pca.explained_variance_ratio_, columns=['Explai
ned Variance'])
```

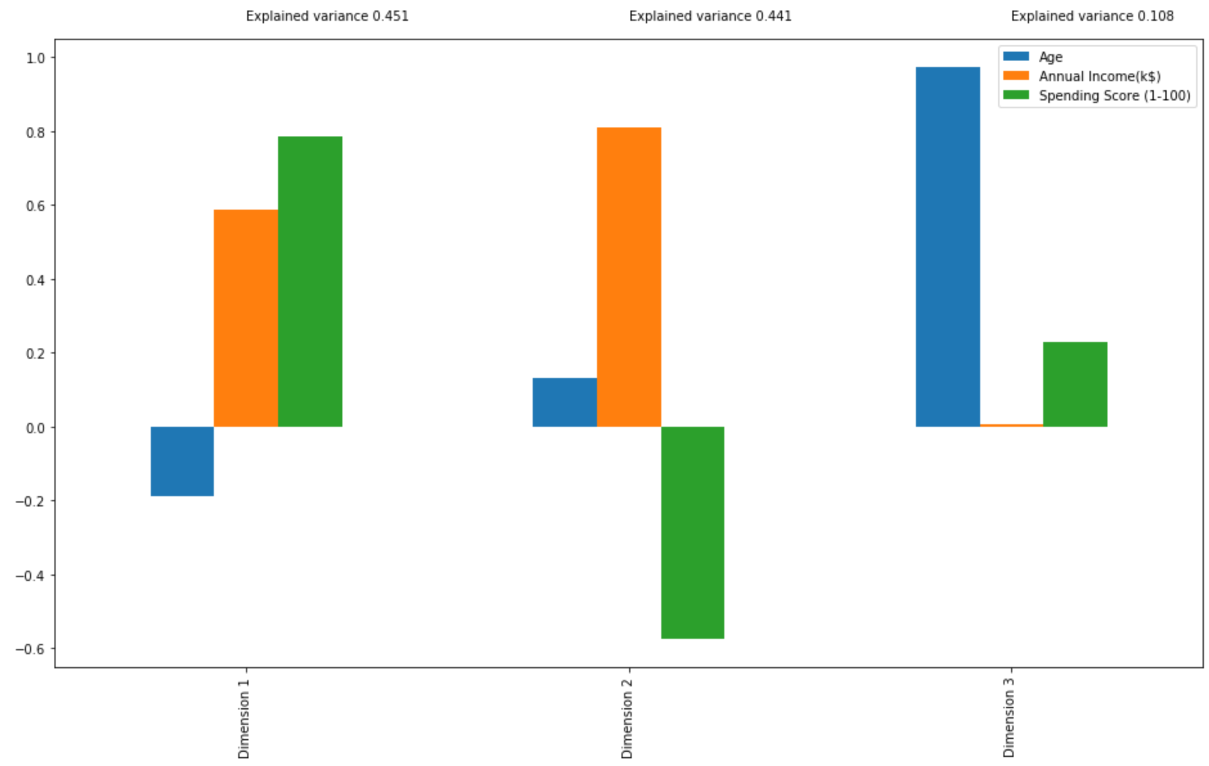
```
variance.index = dimensions

pd.concat([variance,components], axis=1)
```

Out[11]:

| | Explained Variance | Age | Annual Income(k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **Dimension 1** | 0.451253 | -0.188974 | 0.588641 | 0.785997 |
| **Dimension 2** | 0.440985 | 0.130965 | 0.808376 | -0.573914 |
| **Dimension 3** | 0.107763 | 0.973210 | 0.005517 | 0.229854 |

In [12]:

```
fig, ax = plt.subplots(figsize=(16,9))
components.plot(kind='bar', ax=ax)
ax.set_xticklabels(dimensions)
for i,variance in enumerate(pca.explained_variance_ratio_):
    ax.text(i,ax.get_ylim()[1]+0.05,'Explained variance {}'.format(np.r
ound(variance,3)))
plt.show()
```




In [13]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(dataset.iloc[:, [2,3,4]].values)
pca.explained_variance_ratio_
```

Out[13]:
```
array([0.45125272, 0.44098465])
```

In [14]:
```
transformed_data = pca.transform(dataset.iloc[:, [2,3,4]].values)
transformed_data = pd.DataFrame(transformed_data,columns=['Dimension 1'
,'Dimension 2'])
transformed_data.head()
```

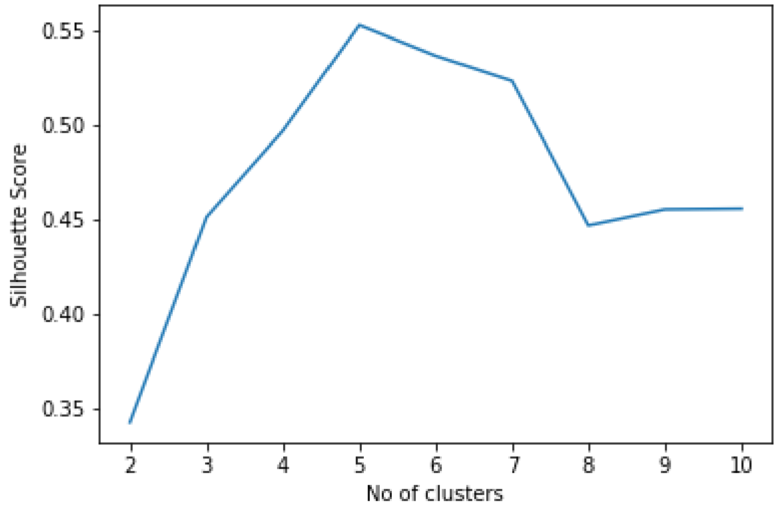
Out[14]:

| | Dimension 1 | Dimension 2 |
|---|---|---|
| **0** | -31.870508 | -33.001425 |

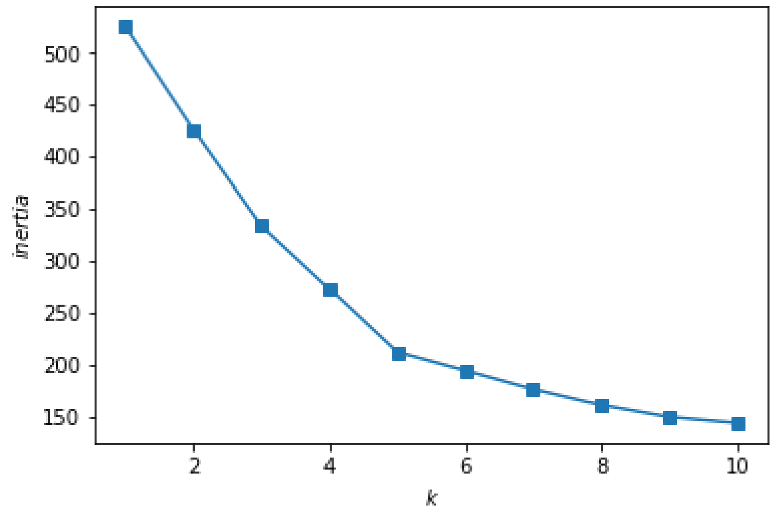
| | Dimension 1 | Dimension 2 |
|---|---|---|
| 1 | 0.763397 | -56.843865 |
| 2 | -57.408726 | -13.122936 |
| 3 | -2.169896 | -53.477905 |
| 4 | -32.174920 | -30.387005 |

In [15]:

```
# Use silhouette score to find the ideal number of clusters.
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

no_of_clusters= range(2,11)
kmeans = [KMeans(n_clusters=i) for i in no_of_clusters]
score = [silhouette_score(transformed_data,kmeans[i].fit(transformed_da
ta).predict(transformed_data),metric='euclidean') for i in range(len(km
eans))]
plt.plot(no_of_clusters,score)
plt.xlabel('No of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



In [16]:
```python
# Use elbow method to find the ideal number of clusters
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42).fit(transformed_data
)
    inertia.append(np.sqrt(kmeans.inertia_))

plt.plot(range(1, 11), inertia, marker='s');
plt.xlabel('$k$')
plt.ylabel('$inertia$');
```




In [17]:
```python
model = KMeans(n_clusters=5)
model.fit(transformed_data)
```

Out[17]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [18]:
```python
# Prediction on the entire data
all_predictions = model.predict(transformed_data)
print(all_predictions)
print(dataset.iloc[:, [2,3,4]].values)
```

```
[1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
4 1
 4 1 4 1 4 1 2 1 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2
 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 0 3 0 3 2 3 0 3 0 3 0 3 0 3 2 3 0 3
0 3
 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0
3 0
 3 0 3 0 3 0 3 0 3 0 3 0 3 0 3]
[[ 19  15  39]
 [ 21  15  81]
 [ 20  16   6]
 [ 23  16  77]
 [ 31  17  40]
 [ 22  17  76]
 [ 35  18   6]
 [ 23  18  94]
 [ 64  19   3]
 [ 30  19  72]
 [ 67  19  14]
 [ 35  19  99]
 [ 58  20  15]
 [ 24  20  77]
 [ 37  20  13]
 [ 22  20  79]
 [ 35  21  35]
 [ 20  21  66]
 [ 52  23  29]
 [ 35  23  98]
 [ 35  24  35]
 [ 25  24  73]
 [ 46  25   5]
 [ 31  25  73]
 [ 54  28  14]
 [ 29  28  82]
 [ 45  28  32]
 [ 35  28  61]
 [ 40  29  31]
```

```
[ 23  29  87]
[ 60  30   4]
[ 21  30  73]
[ 53  33   4]
[ 18  33  92]
[ 49  33  14]
[ 21  33  81]
[ 42  34  17]
[ 30  34  73]
[ 36  37  26]
[ 20  37  75]
[ 65  38  35]
[ 24  38  92]
[ 48  39  36]
[ 31  39  61]
[ 49  39  28]
[ 24  39  65]
[ 50  40  55]
[ 27  40  47]
[ 29  40  42]
[ 31  40  42]
[ 49  42  52]
[ 33  42  60]
[ 31  43  54]
[ 59  43  60]
[ 50  43  45]
[ 47  43  41]
[ 51  44  50]
[ 69  44  46]
[ 27  46  51]
[ 53  46  46]
[ 70  46  56]
[ 19  46  55]
[ 67  47  52]
[ 54  47  59]
[ 63  48  51]
[ 18  48  59]
[ 43  48  50]

[ 68  48  48]
```

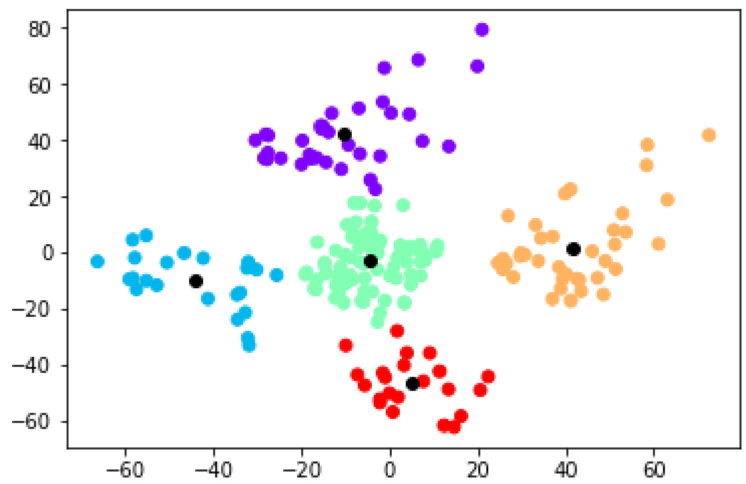
```
[ 19  48  59]
[ 32  48  47]
[ 70  49  55]
[ 47  49  42]
[ 60  50  49]
[ 60  50  56]
[ 59  54  47]
[ 26  54  54]
[ 45  54  53]
[ 40  54  48]
[ 23  54  52]
[ 49  54  42]
[ 57  54  51]
[ 38  54  55]
[ 67  54  41]
[ 46  54  44]
[ 21  54  57]
[ 48  54  46]
[ 55  57  58]
[ 22  57  55]
[ 34  58  60]
[ 50  58  46]
[ 68  59  55]
[ 18  59  41]
[ 48  60  49]
[ 40  60  40]
[ 32  60  42]
[ 24  60  52]
[ 47  60  47]
[ 27  60  50]
[ 48  61  42]
[ 20  61  49]
[ 23  62  41]
[ 49  62  48]
[ 67  62  59]
[ 26  62  55]
[ 49  62  56]
[ 21  62  42]

[ 66  63  50]
```

```
[ 54  63  46]
[ 68  63  43]
[ 66  63  48]
[ 65  63  52]
[ 19  63  54]
[ 38  64  42]
[ 19  64  46]
[ 18  65  48]
[ 19  65  50]
[ 63  65  43]
[ 49  65  59]
[ 51  67  43]
[ 50  67  57]
[ 27  67  56]
[ 38  67  40]
[ 40  69  58]
[ 39  69  91]
[ 23  70  29]
[ 31  70  77]
[ 43  71  35]
[ 40  71  95]
[ 59  71  11]
[ 38  71  75]
[ 47  71   9]
[ 39  71  75]
[ 25  72  34]
[ 31  72  71]
[ 20  73   5]
[ 29  73  88]
[ 44  73   7]
[ 32  73  73]
[ 19  74  10]
[ 35  74  72]
[ 57  75   5]
[ 32  75  93]
[ 28  76  40]
[ 32  76  87]
[ 25  77  12]

[ 28  77  97]
```

```
[ 48  77  36]
[ 32  77  74]
[ 34  78  22]
[ 34  78  90]
[ 43  78  17]
[ 39  78  88]
[ 44  78  20]
[ 38  78  76]
[ 47  78  16]
[ 27  78  89]
[ 37  78   1]
[ 30  78  78]
[ 34  78   1]
[ 30  78  73]
[ 56  79  35]
[ 29  79  83]
[ 19  81   5]
[ 31  81  93]
[ 50  85  26]
[ 36  85  75]
[ 42  86  20]
[ 33  86  95]
[ 36  87  27]
[ 32  87  63]
[ 40  87  13]
[ 28  87  75]
[ 36  87  10]
[ 36  87  92]
[ 52  88  13]
[ 30  88  86]
[ 58  88  15]
[ 27  88  69]
[ 59  93  14]
[ 35  93  90]
[ 37  97  32]
[ 32  97  86]
[ 46  98  15]
[ 29  98  88]

[ 41  99  39]
```

```
 [ 30  99  97]
 [ 54 101  24]
 [ 28 101  68]
 [ 41 103  17]
 [ 36 103  85]
 [ 34 103  23]
 [ 32 103  69]
 [ 33 113   8]
 [ 38 113  91]
 [ 47 120  16]
 [ 35 120  79]
 [ 45 126  28]
 [ 32 126  74]
 [ 32 137  18]
 [ 30 137  83]]
```

In [19]:
```
plt.scatter(transformed_data.iloc[:,0],transformed_data.iloc[:,1],c=mod
el.labels_,cmap='rainbow')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],col
or='black')
plt.show()
```




In [20]:
```
from scipy.cluster.hierarchy import linkage, dendrogram
```

```
dataset.pop('Genre').values
```

Out[20]:
```
array(['Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Femal
e',
       'Female', 'Male', 'Female', 'Male', 'Female', 'Female', 'Femal
e',
       'Male', 'Male', 'Female', 'Male', 'Male', 'Female', 'Male', 'Mal
e',
       'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female',
       'Female', 'Male', 'Female', 'Male', 'Male', 'Female', 'Female',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Male', 'Mal
e',
       'Female', 'Female', 'Female', 'Female', 'Female', 'Female',
       'Female', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male',
       'Female', 'Male', 'Female', 'Male', 'Male', 'Male', 'Female',
       'Female', 'Male', 'Male', 'Female', 'Female', 'Male', 'Female',
       'Male', 'Female', 'Female', 'Female', 'Male', 'Male', 'Female',
       'Male', 'Female', 'Female', 'Male', 'Male', 'Male', 'Female',
       'Female', 'Male', 'Female', 'Female', 'Female', 'Female', 'Femal
e',
       'Male', 'Male', 'Female', 'Female', 'Male', 'Female', 'Female',
       'Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Male',
       'Female', 'Female', 'Male', 'Male', 'Male', 'Male', 'Female',
       'Female', 'Male', 'Female', 'Female', 'Female', 'Female', 'Femal
e',
       'Female', 'Male', 'Female', 'Female', 'Male', 'Female', 'Femal
e',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Femal
e',
       'Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Female',
       'Male', 'Female', 'Female', 'Male', 'Male', 'Male', 'Female',
       'Female', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female',
       'Female', 'Male', 'Female', 'Male', 'Female', 'Female', 'Femal
e',
       'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Female',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Mal
e',
       'Male', 'Male', 'Male', 'Female', 'Female', 'Male', 'Female',
       'Female', 'Male', 'Female', 'Male', 'Female', 'Female', 'Femal
e',
```

```
       'Female', 'Male', 'Female', 'Female', 'Female', 'Female', 'Mal
e',
       'Male', 'Male'], dtype=object)
```

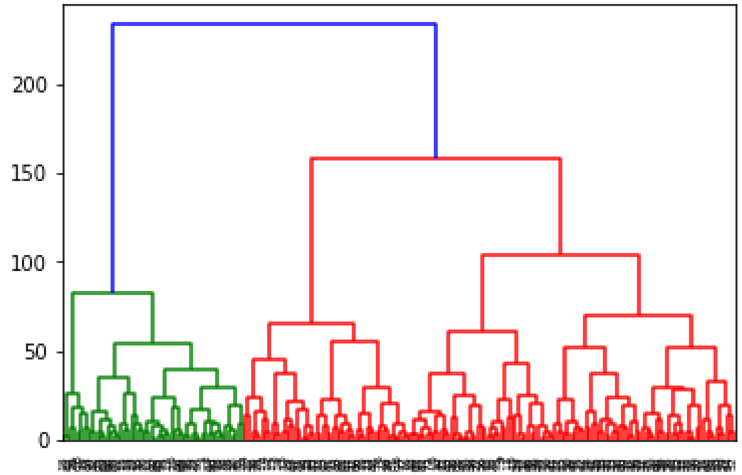
In [21]:
```python
varieties = dataset.pop('Spending Score (1-100)').values
print(varieties.shape)
```

```
(200,)
```

In [22]:
```python
samples = dataset.values
print(samples.shape)
```

```
(200, 3)
```

In [23]:
```python
mergings = linkage(samples, method='complete')

dendrogram(mergings,
           labels=varieties,
           leaf_rotation=90,
           leaf_font_size=6,
           )
plt.show()
```



```
In [ ]:
```