K-Means кластеризация с Python

Этот ноутбук служит лишь справочным материалом для видео-лекции и чтения.

Используемый метод

K-Means кластеризация – это алгоритм обучения без учителя, который группирует данные на основе их схожести. Обучение без учителя означает, что нет заранее определенного результата, а алгоритм просто пытается найти закономерности в данных.

В K-Means кластеризации необходимо указать количество кластеров, в которые должны быть сгруппированы данные. Алгоритм случайным образом присваивает каждое наблюдение определенному кластеру и находит центроид каждого кластера. Затем алгоритм выполняет итерации по двум этапам: перераспределяет точки данных в кластер, чей центроид находится ближе всего, и вычисляет новый центроид каждого кластера. Эти два шага повторяются до тех пор, пока внутрикластерная вариация не может быть уменьшена далее.

Внутрикластерная вариация рассчитывается как сумма евклидовых расстояний между точками данных и их соответствующими центроидами.

Импорт библиотек

In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Создание данных

In [23]:
```
from sklearn.datasets import make_blobs
```

In [42]:
```
# Create Data
data = make_blobs(n_samples=200, n_features=2,
                           centers=4, cluster_std=1.8,random_state=101)
```

Визуализация данных

In [43]:
```
plt.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x11ab34d30>
```

Создание кластеров

In [48]:
```
from sklearn.cluster import KMeans
```

In [49]:
```
kmeans = KMeans(n_clusters=4)
```

In [50]:
```
kmeans.fit(data[0])
```

Out[50]:
```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=4, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
```

In [51]:
```
kmeans.cluster_centers_
```

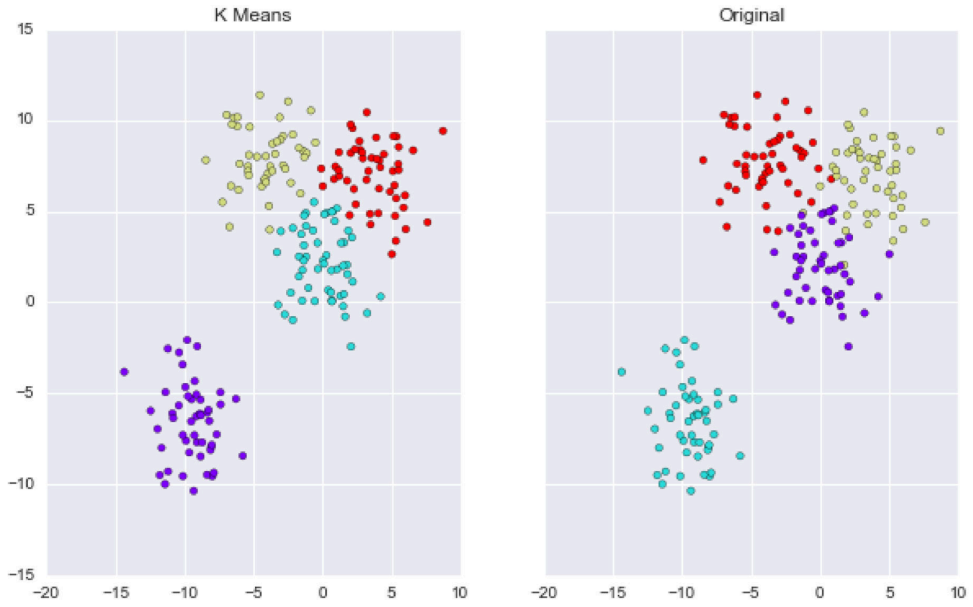
Out[51]:
```
array([[-4.13591321,  7.95389851],
       [-9.46941837, -6.56081545],
       [-0.0123077 ,  2.13407664],
       [ 3.71749226,  7.01388735]])
```

In [55]:
```
kmeans.labels_
```

Out[55]:
```
array([2, 3, 1, 3, 3, 0, 3, 1, 3, 1, 2, 1, 3, 3, 2, 1, 3, 1, 0, 2, 0, 1, 1,
       0, 2, 0, 0, 1, 3, 3, 2, 0, 3, 1, 1, 2, 0, 0, 0, 1, 0, 2, 2, 2, 1, 3,
       2, 1, 0, 1, 1, 2, 3, 1, 0, 2, 1, 1, 2, 3, 0, 3, 0, 2, 3, 1, 0, 3, 3,
       0, 3, 1, 0, 1, 0, 3, 3, 1, 2, 1, 1, 0, 3, 0, 1, 1, 1, 2, 1, 0, 0, 0,
       0, 1, 1, 0, 3, 2, 0, 3, 1, 0, 1, 1, 3, 1, 0, 3, 0, 0, 3, 2, 2, 3, 0,
       3, 2, 2, 3, 2, 1, 2, 1, 2, 1, 3, 2, 1, 0, 2, 2, 2, 1, 0, 0, 2, 3, 2,
       3, 1, 0, 3, 0, 2, 2, 3, 1, 0, 2, 2, 2, 2, 1, 3, 1, 2, 3, 3, 3, 1, 3,
       1, 1, 2, 0, 2, 1, 3, 2, 1, 3, 1, 2, 3, 1, 2, 3, 3, 0, 3, 2, 0, 0, 2,
       0, 0, 0, 0, 0, 1, 0, 3, 3, 2, 0, 1, 3, 3, 0, 1], dtype=int32)
```

In [69]:
```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(10,6))
ax1.set_title('K Means')
ax1.scatter(data[0][:,0],data[0][:,1],c=kmeans.labels_,cmap='rainbow')
ax2.set_title("Original")
ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

Out[69]:
```
<matplotlib.collections.PathCollection at 0x11da01be0>
```



Следует отметить, что цвета не имеют значения при сравнении двух графиков.
Отличная работа!