Курс “Python для DataScience”

Практическое задание

Тема “Обучение без учителя”

Задание 1

- Импортируйте библиотеки pandas, numpy и matplotlib.
- Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn.
- Создайте датафреймы X и y из этих данных.
- Разбейте эти датафреймы на тренировочные (X_train, y_train) и тестовые (X_test, y_test) с помощью функции train_test_split так, чтобы размер тестовой выборки составлял 20% от всех данных, при этом аргумент random_state должен быть равен 42.
- Масштабируйте данные с помощью StandardScaler.
- Постройте модель t-SNE на тренировочный данных с параметрами: n_components=2, learning_rate=250, random_state=42.
- Постройте диаграмму рассеяния на этих данных.

Задание 2

- С помощью KMeans разбейте данные из тренировочного набора на 3 кластера, используйте все признаки из датафрейма X_train.
- Параметр max_iter должен быть равен 100, random_state сделайте равным 42.
- Постройте еще раз диаграмму рассеяния на данных, полученных с помощью TSNE, и раскрасьте точки из разных кластеров разными цветами.
- Вычислите средние значения price и CRIM в разных кластерах.

*Задание 3

- Примените модель KMeans, построенную в предыдущем задании, к данным из тестового набора.
- Вычислите средние значения price и CRIM в разных кластерах на тестовых данных.

Задание 1

1.1 Импортируйте библиотеки pandas, numpy и matplotlib.

В [233]:
```
import numpy as np
import pandas as pd
#import matplotlib as mpl
import matplotlib.pyplot as plt
```

1.2 Загрузите "Boston House Prices dataset" из встроенных наборов данных библиотеки sklearn.

В [282]:
```
from sklearn.datasets import load_boston

boston = load_boston()
boston.keys()
```

Out[282]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

B [235]:
```
for line in boston.DESCR.split('\n'):
    print(line)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (https://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley,
 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Co
nference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

1.3 Создайте датафреймы X и y из этих данных.

B [236]:
```
data = boston.data

feature_names = boston.feature_names
feature_names
```

Out[236]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

B [237]:
```
X = pd.DataFrame(data, columns=feature_names)
y = boston.target
X.head()
```

Out[237]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

B [238]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

1.4 Разбейте эти датафреймы на тренировочные (X_train, y_train) и тестовые (X_test, y_test) с помощью функции train_test_split так, чтобы размер тестовой выборки составлял 20% от всех данных, при этом аргумент random_state должен быть равен 42.

B [240]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

B [241]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[241]:
```
((404, 13), (102, 13), (404,), (102,))
```

1.5 Масштабируйте данные с помощью StandardScaler.

B [242]:
```
from sklearn.preprocessing import StandardScaler
```

B [283]:
```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)  # Строим модель масштабирования
X_test_scaled = scaler.transform(X_test)  # Применяем модель стандартизации на тестовой выборке

X_train_scaled = pd.DataFrame(X_train_scaled, columns=feature_names)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=feature_names)
```

1.6 Постройте модель t-SNE на тренировочный данных с параметрами: n_components=2, learning_rate=250, random_state=42.

B [244]:
```
from sklearn.manifold import TSNE
```

B [246]:
```
# Применяем метод t-SNE (понижение размерность данных)
tsne = TSNE(n_components=2, learning_rate=250, random_state=42)
```

B [247]:
```
X_train_tsne = tsne.fit_transform(X_train_scaled)
```

B [248]:
```
X_train_tsne
```

```
[-10.733389  ,  -5.5205164 ],
[-18.339699  , -23.839216  ],
[ -3.087924  ,  -8.492775  ],
[  8.2998    ,   2.3935862 ],
[ -1.5845332 , -12.5177965 ],
[ -1.7503096 ,  13.745355  ],
[-17.836658  ,  -8.000917  ],
[ -8.456023  ,   7.1432576 ],
[ -3.9704554 ,   1.7679943 ],
[  5.3078537 ,   6.6208763 ],
[ 35.63743   ,  13.454849  ],
[-11.5760975 ,  -6.726915  ],
[-17.734854  , -25.460443  ],
[-15.124907  , -22.501612  ],
[ -9.148589  ,   6.324595  ],
[ -8.659377  ,   6.1306043 ],
[-16.324066  , -23.121286  ],
[-17.558323  , -22.254694  ],
[  6.48041   ,  -1.2868649 ],
[ 31.174524  ,  17.798105  ],
```

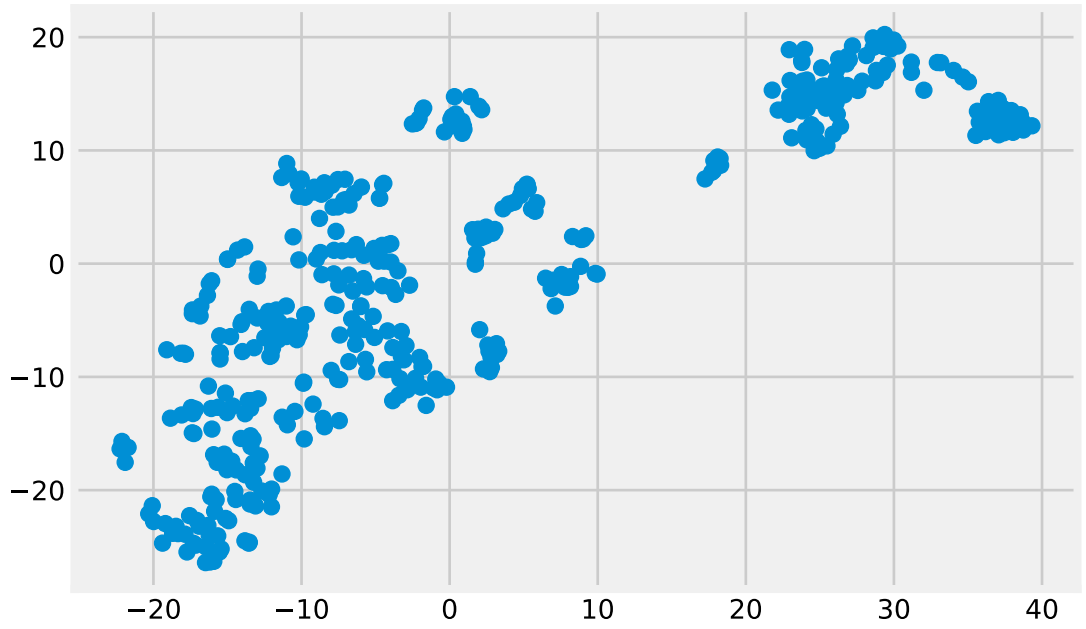
B [249]:
```
print('До\t{}'.format(X_train_scaled.shape))
print('После\t{}'.format(X_train_tsne.shape))
# Количество признаков уменьшилось до 2
```

```
До      (404, 13)
После   (404, 2)
```

1.7 Постройте диаграмму рассеяния на этих данных.

B [284]:
```
# Строим 2d диаграмму рассеивания
plt.style.use('fivethirtyeight')
%config InlineBackend.figure_format = 'svg'
%matplotlib inline

plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1])
plt.show()
```



Задание 2

2.1 С помощью KMeans разбейте данные из тренировочного набора на 3 кластера, используйте все признаки из датафрейма X_train.

Параметр max_iter должен быть равен 100, random_state сделайте равным 42.

B [251]:
```
from sklearn.cluster import KMeans
```

B [252]:
```
kmeans = KMeans(n_clusters=3, max_iter=100, random_state=42)
```

B [253]:
```
labels_train = kmeans.fit_predict(X_train_scaled)
#labels_train
```

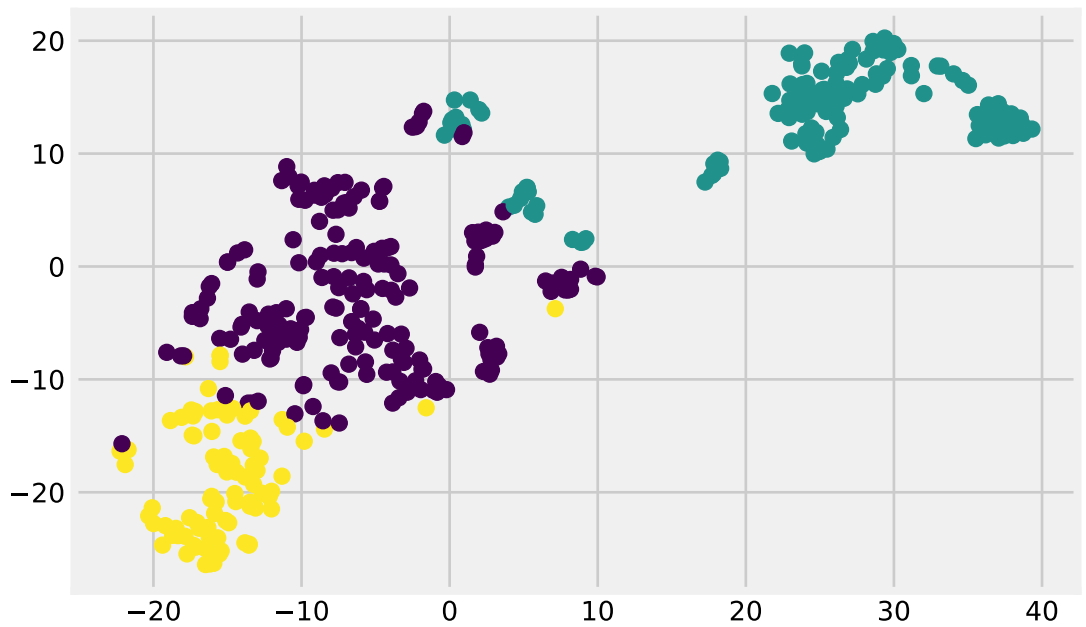
B [254]:
```
pd.value_counts(labels_train)
```

Out[254]:
```
0    191
1    127
2     86
dtype: int64
```

2.2 Постройте еще раз диаграмму рассеяния на данных, полученных с помощью TSNE, и раскрасьте точки из разных кластеров разными цветами.

B [280]:
```
plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1], c=labels_train)
#plt.text(-15, 15, 'Кластер 0')
#plt.text(-10, -25, 'Кластер 1')
#plt.text(20, -2, 'Кластер 2')

plt.show()
```



Мы получили разбиение на 3 кластера.

B [195]:
```
# Получаем метки кластеров для тестовой выборки
labels_test = kmeans.predict(X_test_scaled)  # Предсказание
# label_test
```

2.3 Вычислите средние значения price и CRIM в разных кластерах.

B [143]:
```
y_train.mean()  # Среднее значение по тренировочной выборке
```

Out[143]:
```
22.796534653465343
```

B [276]:
```
# Среднее значение цены недвижимости, по тренировочной выборке
print('Среднее значение цены:')
print('Кластер 0: {}'.format(y_train[labels_train == 0].mean()))  # Среднее значение по 0 - кластеру
print('Кластер 1: {}'.format(y_train[labels_train == 1].mean()))  # Среднее значение по 1 - кластеру
print('Кластер 2: {}'.format(y_train[labels_train == 2].mean()))  # Среднее значение по 2 - кластеру
```

```
Среднее значение цены:
Кластер 0: 24.958115183246072
Кластер 1: 16.165354330708663
Кластер 2: 27.78837209302326
```

Видно, что в кластер-2 попала более дорогая недвижимость, в кластер-0 средняя недвижимость, в кластер-1 самая дешёвая.

B [275]:
```
# Среднее значение по тренировочной выборке (CRIM - количество преступлений на душу населения
print('Среднее значение CRIM:')
print('Кластер 0: {}'.format(X_train.loc[labels_train == 0, 'CRIM'].mean()))  # Среднее значение по 0 - кластеру
print('Кластер 1: {}'.format(X_train.loc[labels_train == 1, 'CRIM'].mean()))  # Среднее значение по 1 - кластеру
print('Кластер 2: {}'.format(X_train.loc[labels_train == 2, 'CRIM'].mean()))  # Среднее значение по 2 - кластеру
```

```
Среднее значение CRIM:
Кластер 0: 0.42166020942408367
Кластер 1: 10.797028425196853
Кластер 2: 0.07356558139534886
```

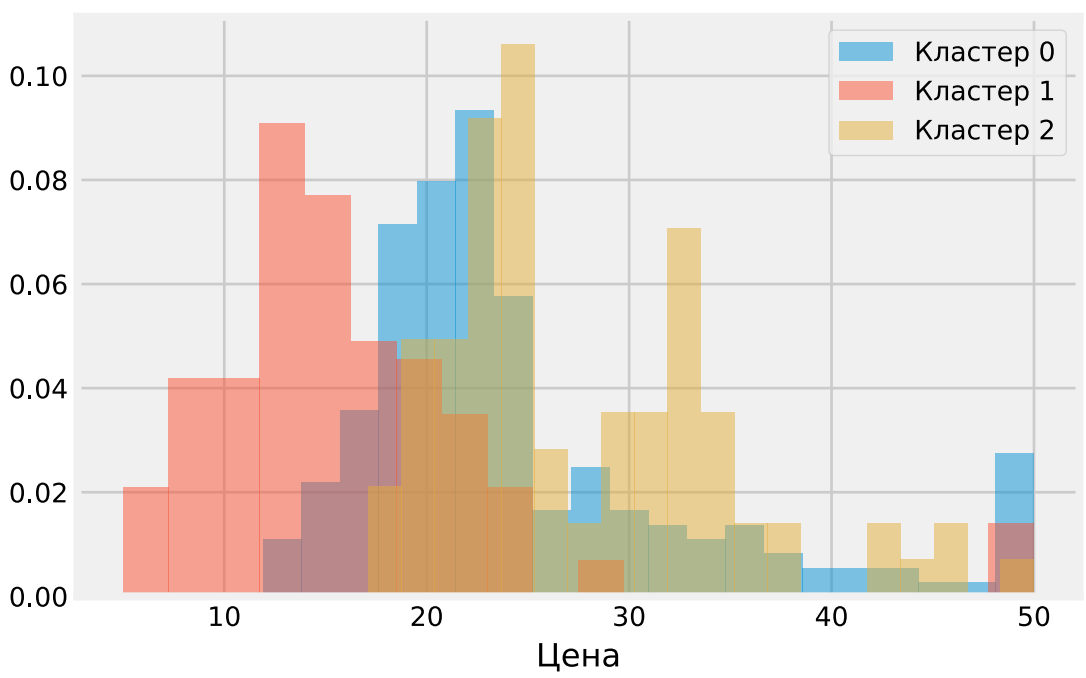
Видно, что значение CRIM значительно выше для кластера 1.

В [176]:
```
# Построим гистограммы распределения цены по каждому кластеру

plt.hist(y_train[labels_train == 0], bins=20, density=True, alpha=0.5)
plt.hist(y_train[labels_train == 1], bins=20, density=True, alpha=0.5)
plt.hist(y_train[labels_train == 2], bins=20, density=True, alpha=0.5)

plt.legend(['Кластер 0', 'Кластер 1', 'Кластер 2'])
plt.xlabel('Цена')

plt.show()
```



Гистограмма также отражает замеченную тенденцию.

*Задание 3

3.1 Примените модель KMeans, построенную в предыдущем задании, к данным из тестового набора.

В [179]:
```
from sklearn.cluster import KMeans
```

В [180]:
```
kmeans = KMeans(n_clusters=3, max_iter=100, random_state=42)
```

В [213]:
```
# Получаем метки кластеров для тестовой выборки
labels_test = kmeans.predict(X_test_scaled)
labels_test
```

Out[213]:
```
array([0, 2, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 2, 1, 2, 2,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 0,
       0, 1, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1, 1, 2, 0, 0, 2, 2, 0, 0, 2, 0,
       1, 2, 2, 1, 0, 1, 1, 2, 0, 1, 0, 2, 1, 1, 2, 1, 2, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0])
```

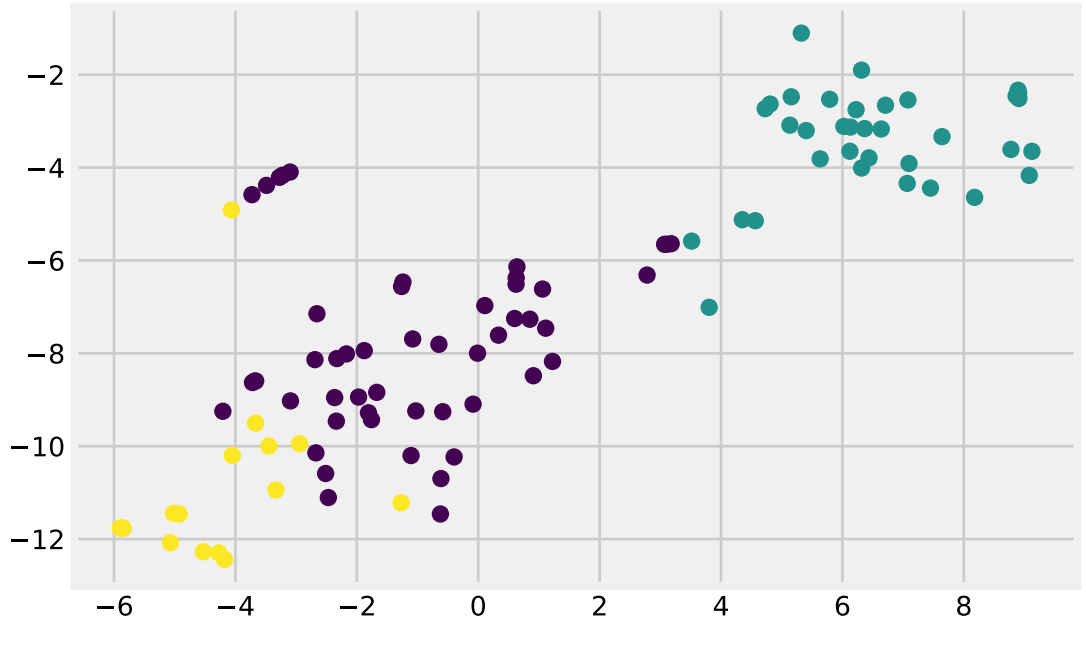
В [214]:
```
pd.value_counts(labels_test)
```

Out[214]:
```
0    51
1    35
2    16
dtype: int64
```

В [219]:
```
X_test_tsne = tsne.fit_transform(X_test_scaled)
# X_test_tsne
```

B [279]:
```
#plt.scatter(X_test_tsne[:, 0], X_test_tsne[:, 1], c=labels_test)
plt.scatter(X_test_tsne[:, 0], X_test_tsne[:, 1], c=labels_test)
#plt.text(-2, -5, 'Кластер 0')
#plt.text(-4, -12, 'Кластер 1')
#plt.text(6, -6, 'Кластер 2')

plt.show()
```



3.2 Вычислите средние значения price и CRIM в разных кластерах на тестовых данных.

B [264]:
```
y_test.mean()  # Среднее значение по тренировочной выборке
```

Out[264]:
```
21.488235294117644
```

B [277]:
```
# Среднее значение цены недвижимости, по тестовой выборке
print('Среднее значение цены:')
print('Кластер 0: {}'.format(y_test[labels_test == 0].mean()))  # Среднее значение по 0 - кластеру
print('Кластер 1: {}'.format(y_test[labels_test == 1].mean()))  # Среднее значение по 1 - кластеру
print('Кластер 2: {}'.format(y_test[labels_test == 2].mean()))  # Среднее значение по 2 - кластеру
```

```
Среднее значение цены:
Кластер 0: 21.860784313725492
Кластер 1: 16.43714285714286
Кластер 2: 31.35
```

Как и в тестовой выборке, видно, что кластер-2 попала более дорогая недвижимость, в кластер-0 средняя недвижимость, в кластер-1 самая дешёвая.

B [278]:
```
# Среднее значение по тренировочной выборке (CRIM - количество преступлений на душу населения)
print('Среднее значение CRIM:')
print('Кластер 0: {}'.format(X_test.loc[labels_test == 0, 'CRIM'].mean()))  # Среднее значение по 0 - кластеру
print('Кластер 1: {}'.format(X_test.loc[labels_test == 1, 'CRIM'].mean()))  # Среднее значение по 1 - кластеру
print('Кластер 2: {}'.format(X_test.loc[labels_test == 2, 'CRIM'].mean()))  # Среднее значение по 2 - кластеру
```

```
Среднее значение CRIM:
Кластер 0: 0.26607882352941176
Кластер 1: 10.165531142857143
Кластер 2: 0.062060000000000004
```

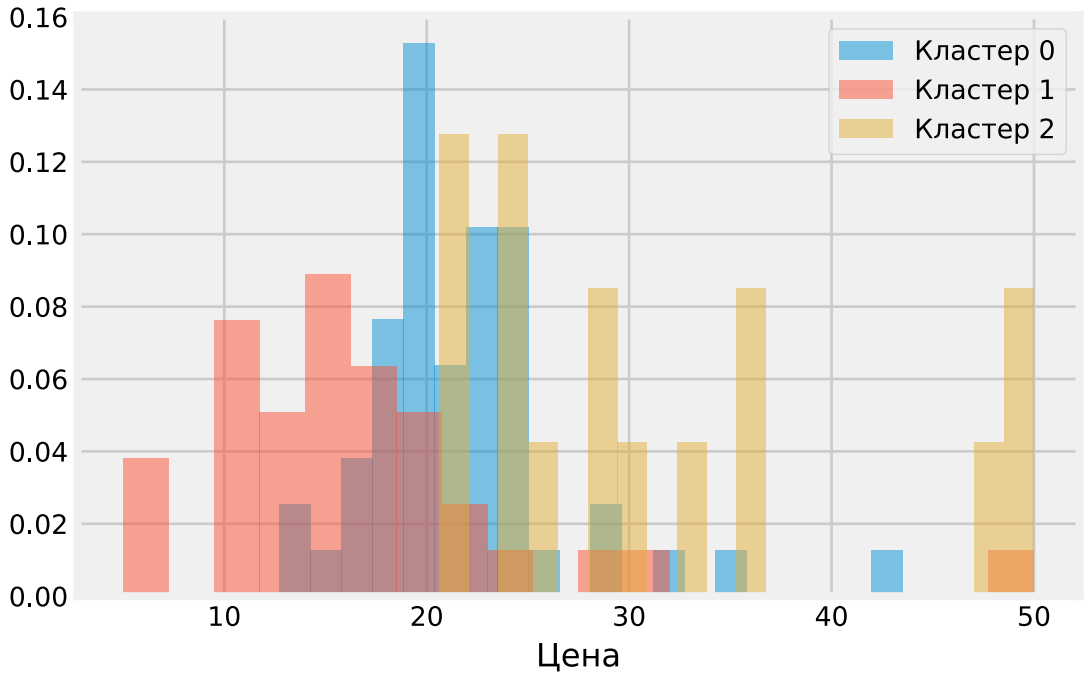
Видно, что значение CRIM значительно выше для кластера 1, как и в тестовой выборке .

In [272]:
```
# Построим гистограммы распределения цены по каждому кластеру

plt.hist(y_test[labels_test == 0], bins=20, density=True, alpha=0.5)
plt.hist(y_test[labels_test == 1], bins=20, density=True, alpha=0.5)
plt.hist(y_test[labels_test == 2], bins=20, density=True, alpha=0.5)

plt.legend(['Кластер 0', 'Кластер 1', 'Кластер 2'])
plt.xlabel('Цена')

plt.show()
```



Гистограмма также отражает замеченную тенденцию.

In [ ]: