Алёшин Александр Денисович ИУ5-63Б

РК №2 по ТМО по теме "Методы построения моделей машинного обучения"

Вариант 1

Задание: для заданного набора данных постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных).Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Методы: 1-Дерево решений, 2-Случайный лес

По заданию буду использовать Dataset "boston".

In [16]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
%matplotlib inline
sns.set(style="ticks")
```

In [45]:
```
from sklearn.datasets import load_boston
boston = load_boston()
data = pd.read_csv('housing_data1.txt', sep="\s+|\t+|\s+\t+|\t+\s+", engine = 'pytho
```

In [26]:
```
data.head()
```

Out[26]:

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

In [27]:
```
# определим тип данных
type(boston)
```

Out[27]:
```
sklearn.utils.Bunch
```

In [28]:
```
boston.keys()
```

Out[28]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

In [29]:
```
# название столбцов возьмем из boston.feature_names
boston_df = pd.DataFrame(boston.data, columns = boston.feature_names)

# выведем первые пять районов с помощью функции head()
boston_df.head()
```

Out[29]:

| | 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 |

In [30]:
```
# теперь добавим в таблицу целевую переменную и назовем ее MEDV
boston_df['MEDV'] = boston.target

# снова воспользуемся функцией head()
boston_df.head()
```

Out[30]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | |

In [31]:
```
# посмотрим с каким типом переменных нам предстоит работать
# для этого есть метод .info()
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 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
```

```
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [32]:
```
boston_df.describe().round(2)
```

Out[32]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.00 | 506.( |
| **mean** | 3.61 | 11.36 | 11.14 | 0.07 | 0.55 | 6.28 | 68.57 | 3.80 | 9.55 | 408.24 | 18.46 | 356.( |
| **std** | 8.60 | 23.32 | 6.86 | 0.25 | 0.12 | 0.70 | 28.15 | 2.11 | 8.71 | 168.54 | 2.16 | 91.2 |
| **min** | 0.01 | 0.00 | 0.46 | 0.00 | 0.38 | 3.56 | 2.90 | 1.13 | 1.00 | 187.00 | 12.60 | 0.3 |
| **25%** | 0.08 | 0.00 | 5.19 | 0.00 | 0.45 | 5.89 | 45.02 | 2.10 | 4.00 | 279.00 | 17.40 | 375.3 |
| **50%** | 0.26 | 0.00 | 9.69 | 0.00 | 0.54 | 6.21 | 77.50 | 3.21 | 5.00 | 330.00 | 19.05 | 391.4 |
| **75%** | 3.68 | 12.50 | 18.10 | 0.00 | 0.62 | 6.62 | 94.07 | 5.19 | 24.00 | 666.00 | 20.20 | 396.2 |
| **max** | 88.98 | 100.00 | 27.74 | 1.00 | 0.87 | 8.78 | 100.00 | 12.13 | 24.00 | 711.00 | 22.00 | 396.9 |

In [33]:
```
boston_df.isnull().sum()
```

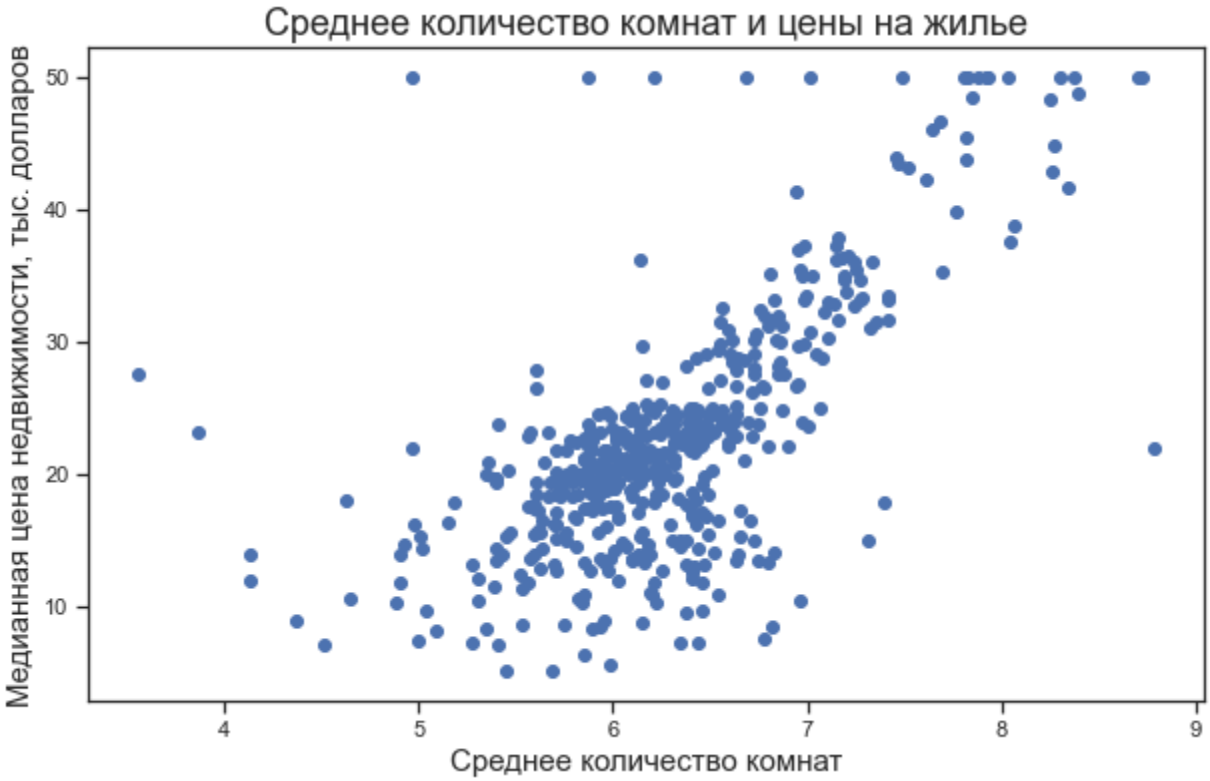
Out[33]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [34]:
```
# подготовим данные (поместим столбцы датафрейма в переменные)
x1 = boston_df['LSTAT']
x2 = boston_df['RM']
y = boston_df['MEDV']
# зададим размер и построим первый график
plt.figure(figsize = (10,6))
plt.scatter(x1, y)

# добавим подписи
plt.xlabel('Процент населения с низким социальным статусом', fontsize = 15)
plt.ylabel('Медианная цена недвижимости, тыс. долларов', fontsize = 15)
plt.title('Социальный статус населения и цены на жилье', fontsize = 18)
```

Out[34]:
```
Text(0.5, 1.0, 'Социальный статус населения и цены на жилье')
```


In [35]:
```
# зададим размер и построим второй график
plt.figure(figsize = (10,6))
plt.scatter(x2, y)

# добавим подписи
plt.xlabel('Среднее количество комнат', fontsize = 15)
plt.ylabel('Медианная цена недвижимости, тыс. долларов', fontsize = 15)
plt.title('Среднее количество комнат и цены на жилье', fontsize = 18)
```

Out[35]:
```
Text(0.5, 1.0, 'Среднее количество комнат и цены на жилье')
```




Поместим наши признаки в переменную X, а цены на жилье в переменную y.

In [37]:
```
X = boston_df[['RM', 'LSTAT', 'PTRATIO', 'TAX', 'INDUS']]
y = boston_df['MEDV']
```

Теперь, когда мы загрузили, обработали и исследовали данные, а также отобрали наиболее значимые признаки, мы готовы к обучению модели. Вначале разобьем данные на обучающую и тестовую выборки.

In [39]:
```
from sklearn.model_selection import train_test_split

# размер тестовой выборки составит 30%
# также зададим точку отсчета для воспроизводимости
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 42)
```

In [40]:
```
# размерность обучающей
print(X_train.shape, y_train.shape)

# и тестовой выборки
print(X_test.shape, y_test.shape)
```

```
(354, 5) (354,)
(152, 5) (152,)
```

In [41]:
```
# из набора линейных моделей библиотеки sklearn импортируем линейную регрессию
from sklearn.linear_model import LinearRegression

# создадим объект этого класса и запишем в переменную model
model = LinearRegression()

# обучим нашу модель
model.fit(X_train, y_train)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
# на основе нескольких независимых переменных (X) предскажем цену на жилье (y)
y_pred = model.predict(X_test)

# выведем первые пять значений с помощью диапазона индексов
print(y_pred[:5])
```

```
[26.62981059 31.10008241 16.95701338 25.59771173 18.09307064]
```

Осталось оценить качество модели. Посчитаем среднеквадратическую ошибку.

In [43]:
```
# импортируем модуль метрик
from sklearn import metrics

# выведем корень среднеквадратической ошибки
# сравним тестовые и прогнозные значения цен на жилье
print('Root Mean Squared Error (RMSE):', np.sqrt(metrics.mean_squared_error(y_test,
```

```
Root Mean Squared Error (RMSE): 5.107447670220914
```

Также рассчитаем новый критерий качества — коэффициент детерминации (R2 или R-квадрат). R2 показывает, какая доля изменчивости целевой переменной объясняется с помощью нашей модели.

In [44]:
```
print('R2:', np.round(metrics.r2_score(y_test, y_pred), 2))
```

```
R2: 0.65
```