Lab_5_1

May 31, 2021

:

[1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_absolute_error, mean_squared_error, ␣
 ↪median_absolute_error, r2_score
```

[2]:
```
#
data = pd.read_csv('letterdata.csv')
data.head()
```

[2]:
```
  letter  xbox  ybox  width  height  onpix  xbar  ybar  x2bar  y2bar  xybar  \
0      T     2     8      3       5      1     8    13      0      6      6
1      I     5    12      3       7      2    10     5      5      4     13
2      D     4    11      6       8      6    10     6      2      6     10
3      N     7    11      6       6      3     5     9      4      6      4
4      G     2     1      3       1      1     8     6      6      6      6

   x2ybar  xy2bar  xedge  xedgey  yedge  yedgex
0      10       8      0       8      0       8
1       3       9      2       8      4      10
2       3       7      3       7      3       9
3       4      10      6      10      2       8
4       5       9      1       7      5      10
```

[3]:
```
#
data.shape
```

[3]:
```
(20000, 17)
```

[4]:

```
#
data.dtypes
```

[4]:

```
letter     object
xbox        int64
ybox        int64
width       int64
height      int64
onpix       int64
xbar        int64
ybar        int64
x2bar       int64
y2bar       int64
xybar       int64
x2ybar      int64
xy2bar      int64
xedge       int64
xedgey      int64
yedge       int64
yedgex      int64
dtype: object
```

0.1

0.1.1

[5]:

```
data.isnull().sum()
```

[5]:

```
letter    0
xbox      0
ybox      0
width     0
height    0
onpix     0
xbar      0
ybar      0
x2bar     0
y2bar     0
xybar     0
x2ybar    0
xy2bar    0
xedge     0
xedgey    0
yedge     0
yedgex    0
dtype: int64
```

0.1.2

```
[6]: data.describe()
```

```
[6]:            xbox          ybox         width        height         onpix  \
     count  20000.000000  20000.000000  20000.000000  20000.00000  20000.000000
     mean       4.023550      7.035500      5.121850      5.37245      3.505850
     std        1.913212      3.304555      2.014573      2.26139      2.190458
     min        0.000000      0.000000      0.000000      0.00000      0.000000
     25%        3.000000      5.000000      4.000000      4.00000      2.000000
     50%        4.000000      7.000000      5.000000      6.00000      3.000000
     75%        5.000000      9.000000      6.000000      7.00000      5.000000
     max       15.000000     15.000000     15.000000     15.00000     15.000000

                    xbar          ybar         x2bar         y2bar         xybar  \
     count  20000.000000  20000.000000  20000.000000  20000.000000  20000.000000
     mean       6.897600      7.500450      4.628600      5.178650      8.282050
     std        2.026035      2.325354      2.699968      2.380823      2.488475
     min        0.000000      0.000000      0.000000      0.000000      0.000000
     25%        6.000000      6.000000      3.000000      4.000000      7.000000
     50%        7.000000      7.000000      4.000000      5.000000      8.000000
     75%        8.000000      9.000000      6.000000      7.000000     10.000000
     max       15.000000     15.000000     15.000000     15.000000     15.000000

                  x2ybar        xy2bar         xedge        xedgey         yedge  \
     count  20000.00000  20000.000000  20000.000000  20000.000000  20000.000000
     mean       6.45400      7.929000      3.046100      8.338850      3.691750
     std        2.63107      2.080619      2.332541      1.546722      2.567073
     min        0.00000      0.000000      0.000000      0.000000      0.000000
     25%        5.00000      7.000000      1.000000      8.000000      2.000000
     50%        6.00000      8.000000      3.000000      8.000000      3.000000
     75%        8.00000      9.000000      4.000000      9.000000      5.000000
     max       15.00000     15.000000     15.000000     15.000000     15.000000

                  yedgex
     count  20000.00000
     mean       7.80120
     std        1.61747
     min        0.00000
     25%        7.00000
     50%        8.00000
     75%        9.00000
     max       15.00000
```

0.1.3

```
[7]: from sklearn.preprocessing import LabelEncoder
```

```
[8]: data.head()
```

```
[8]:   letter  xbox  ybox  width  height  onpix  xbar  ybar  x2bar  y2bar  xybar  \
     0      T     2     8      3       5      1     8    13      0      6      6
     1      I     5    12      3       7      2    10     5      5      4     13
     2      D     4    11      6       8      6    10     6      2      6     10
     3      N     7    11      6       6      3     5     9      4      6      4
     4      G     2     1      3       1      1     8     6      6      6      6

        x2ybar  xy2bar  xedge  xedgey  yedge  yedgex
     0      10       8      0       8      0       8
     1       3       9      2       8      4      10
     2       3       7      3       7      3       9
     3       4      10      6      10      2       8
     4       5       9      1       7      5      10
```

```
[9]: data.describe()
```

```
[9]:                xbox          ybox         width       height         onpix  \
     count  20000.000000  20000.000000  20000.000000  20000.00000  20000.000000
     mean       4.023550      7.035500      5.121850      5.37245      3.505850
     std        1.913212      3.304555      2.014573      2.26139      2.190458
     min        0.000000      0.000000      0.000000      0.00000      0.000000
     25%        3.000000      5.000000      4.000000      4.00000      2.000000
     50%        4.000000      7.000000      5.000000      6.00000      3.000000
     75%        5.000000      9.000000      6.000000      7.00000      5.000000
     max       15.000000     15.000000     15.000000     15.00000     15.000000

                    xbar          ybar         x2bar         y2bar         xybar  \
     count  20000.000000  20000.000000  20000.000000  20000.000000  20000.000000
     mean       6.897600      7.500450      4.628600      5.178650      8.282050
     std        2.026035      2.325354      2.699968      2.380823      2.488475
     min        0.000000      0.000000      0.000000      0.000000      0.000000
     25%        6.000000      6.000000      3.000000      4.000000      7.000000
     50%        7.000000      7.000000      4.000000      5.000000      8.000000
     75%        8.000000      9.000000      6.000000      7.000000     10.000000
     max       15.000000     15.000000     15.000000     15.000000     15.000000

                 x2ybar        xy2bar         xedge        xedgey         yedge  \
     count  20000.00000  20000.000000  20000.000000  20000.000000  20000.000000
     mean       6.45400      7.929000      3.046100      8.338850      3.691750
     std        2.63107      2.080619      2.332541      1.546722      2.567073
     min        0.00000      0.000000      0.000000      0.000000      0.000000
     25%        5.00000      7.000000      1.000000      8.000000      2.000000
```

```
50%         6.00000       8.000000       3.000000       8.000000       3.000000
75%         8.00000       9.000000       4.000000       9.000000       5.000000
max        15.00000      15.000000      15.000000      15.000000      15.000000

             yedgex
count   20000.00000
mean        7.80120
std         1.61747
min         0.00000
25%         7.00000
50%         8.00000
75%         9.00000
max        15.00000
```

0.2 1.

train_test_split sklearn

[10]:
```python
from sklearn.model_selection import train_test_split
```

0.2.1 1.1.

,

:

[11]:
```python
X = data.drop(['width', 'letter', 'onpix', 'xbar',  'ybar',  'x2bar',  'y2bar',␣
 ↪ 'xybar',  'x2ybar',  'xy2bar',  'xedge',  'xedgey',  'yedge',  'yedgex'],␣
 ↪axis = 1)
Y = data.width
print('        :\n\n', X.head(), '\n\n         :\n\n', Y.head())
```

```
        :

    xbox  ybox  height
0      2     8       5
1      5    12       7
2      4    11       8
3      7    11       6
4      2     1       1

         :

 0    3
1    3
2    6
3    6
4    3
Name: width, dtype: int64
```

0.2.2 1.2.

: 10%

```
[12]: X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, random_state = 0,␣
      ↪test_size = 0.1)
      print('                    :\n\n',X_train.head(), \
            '\n\n                  :\n\n', X_test.head(), \
            '\n\n                   :\n\n', Y_train.head(), \
            '\n\n                   :\n\n', Y_test.head())
```

```
                    :

        xbox  ybox  height
17964      3     6       5
11632      2     1       1
10869      4     9       7
9179       4    10       8
8871       4     8       6

                   :

        xbox  ybox  height
19134      3     3       2
4981       3     5       4
16643      4     8       5
19117      5    10       7
5306       4     7       8

                    :

 17964    5
11632    2
10869    4
9179     5
8871     5
Name: width, dtype: int64

                    :

 19134    4
4981     6
16643    4
19117    7
5306     4
Name: width, dtype: int64
```

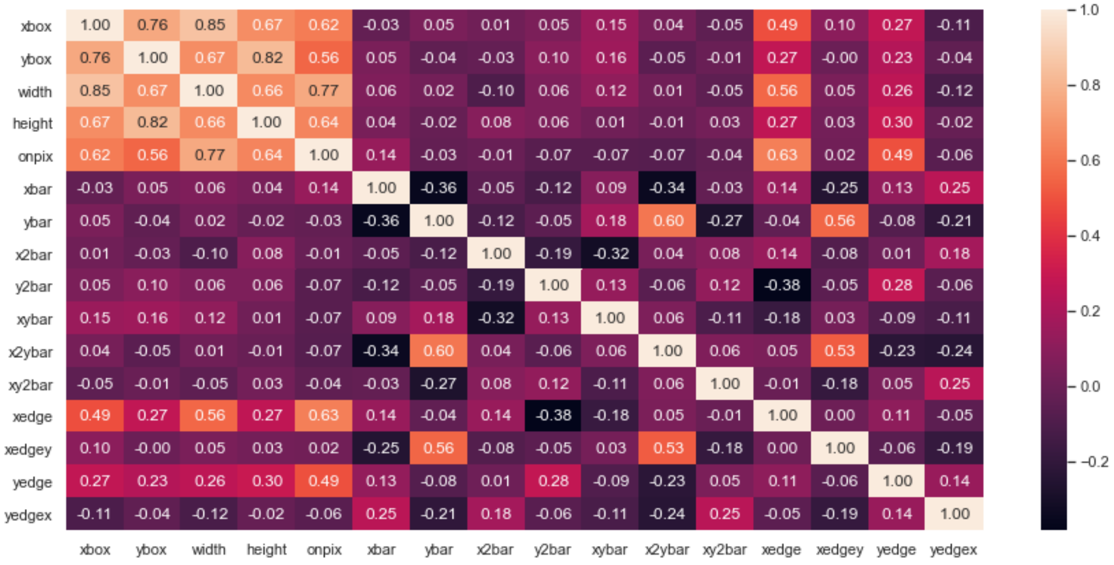
:

[13]:
```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(18000, 3)
(2000, 3)
(18000,)
(2000,)
```

[14]:
```
#
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data.corr(method='pearson'), ax=ax, annot=True, fmt='.2f')
```

[14]:
```
<AxesSubplot:>
```



	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xybar	x2ybar	xy2bar	xedge	xedgey	yedge	yedgex
xbox	1.00	0.76	0.85	0.67	0.62	-0.03	0.05	0.01	0.05	0.15	0.04	-0.05	0.49	0.10	0.27	-0.11
ybox	0.76	1.00	0.67	0.82	0.56	0.05	-0.04	-0.03	0.10	0.16	-0.05	-0.01	0.27	-0.00	0.23	-0.04
width	0.85	0.67	1.00	0.66	0.77	0.06	0.02	-0.10	0.06	0.12	0.01	-0.05	0.56	0.05	0.26	-0.12
height	0.67	0.82	0.66	1.00	0.64	0.04	-0.02	0.08	0.06	0.01	-0.01	0.03	0.27	0.03	0.30	-0.02
onpix	0.62	0.56	0.77	0.64	1.00	0.14	-0.03	-0.01	-0.07	-0.07	-0.07	-0.04	0.63	0.02	0.49	-0.06
xbar	-0.03	0.05	0.06	0.04	0.14	1.00	-0.36	-0.05	-0.12	0.09	-0.34	-0.03	0.14	-0.25	0.13	0.25
ybar	0.05	-0.04	0.02	-0.02	-0.03	-0.36	1.00	-0.12	-0.05	0.18	0.60	-0.27	-0.04	0.56	-0.08	-0.21
x2bar	0.01	-0.03	-0.10	0.08	-0.01	-0.05	-0.12	1.00	-0.19	-0.32	0.04	0.08	0.14	-0.08	0.01	0.18
y2bar	0.05	0.10	0.06	0.06	-0.07	-0.12	-0.05	-0.19	1.00	0.13	-0.06	0.12	-0.38	-0.05	0.28	-0.06
xybar	0.15	0.16	0.12	0.01	-0.07	0.09	0.18	-0.32	0.13	1.00	0.06	-0.11	-0.18	0.03	-0.09	-0.11
x2ybar	0.04	-0.05	0.01	-0.01	-0.07	-0.34	0.60	0.04	-0.06	0.06	1.00	0.06	0.05	0.53	-0.23	-0.24
xy2bar	-0.05	-0.01	-0.05	0.03	-0.04	-0.03	-0.27	0.08	0.12	-0.11	0.06	1.00	-0.01	-0.18	0.05	0.25
xedge	0.49	0.27	0.56	0.27	0.63	0.14	-0.04	0.14	-0.38	-0.18	0.05	-0.01	1.00	0.00	0.11	-0.05
xedgey	0.10	-0.00	0.05	0.03	0.02	-0.25	0.56	-0.08	-0.05	0.03	0.53	-0.18	0.00	1.00	-0.06	-0.19
yedge	0.27	0.23	0.26	0.30	0.49	0.13	-0.08	0.01	0.28	-0.09	-0.23	0.05	0.11	-0.06	1.00	0.14
yedgex	-0.11	-0.04	-0.12	-0.02	-0.06	0.25	-0.21	0.18	-0.06	-0.11	-0.24	0.25	-0.05	-0.19	0.14	1.00

0.3 1.

0.3.1 1.1.

5

RandomForestRegressor sklearn

[15]:
```
from sklearn.ensemble import RandomForestRegressor
```

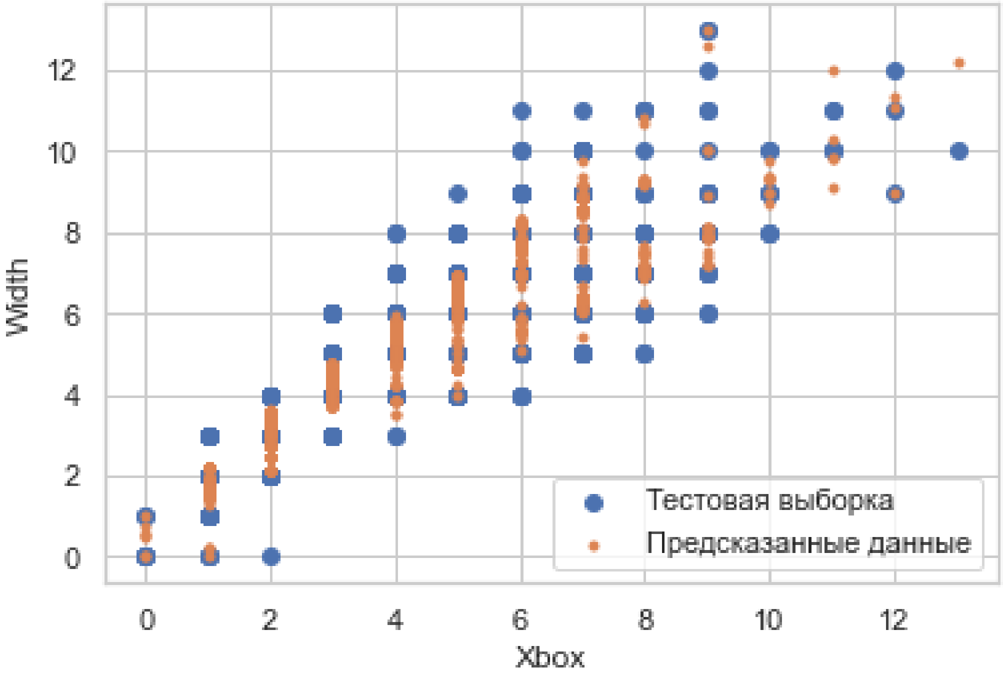
[16]:
```
#
forest_1 = RandomForestRegressor(n_estimators=5, oob_score=True,␣
 ↪random_state=10)
```

```
forest_1.fit(X, Y)
Y_predict = forest_1.predict(X_test)
```

[17]:
```
#
print('               :',  mean_absolute_error(Y_test, Y_predict))
print('                :', mean_squared_error(Y_test, Y_predict))
print('Median absolute error:',       median_absolute_error(Y_test, Y_predict))
print('              :',   r2_score(Y_test, Y_predict))
```

```
               : 0.6508392965288694
                : 0.6560432684278356
Median absolute error: 0.5365963465421117
              : 0.8338153947778956
```

[18]:
```
#
plt.scatter(X_test.xbox, Y_test,    marker = 'o', label = '         ')
plt.scatter(X_test.xbox, Y_predict, marker = '.', label = '           ')
plt.legend(loc = 'lower right')
plt.xlabel('Xbox')
plt.ylabel('Width')
plt.show()
```



-      .

0.3.2 1.2.

```
[19]: params2 = {
          'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
          'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0]
      }
```

```
[20]: grid_2 = GridSearchCV(estimator=RandomForestRegressor(oob_score=True,␣
       ↪random_state=10),
                            param_grid=params2,
                            scoring='neg_mean_squared_error',
                            cv=3,
                            n_jobs=-1)
      grid_2.fit(X, Y)
```

```
[20]: GridSearchCV(cv=3,
                   estimator=RandomForestRegressor(oob_score=True, random_state=10),
                   n_jobs=-1,
                   param_grid={'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0],
                               'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20,
                                                25, 50, 75, 100]},
                   scoring='neg_mean_squared_error')
```

```
[21]: #
      print('                                :', -grid_2.best_score_)
      print('                       :\n',              grid_2.best_params_)
```

```
                               : 0.6945457917385651
                      :
      {'max_features': 0.8, 'n_estimators': 75}
```

```
[22]: #
      forest_3 = RandomForestRegressor(n_estimators=75, max_features = 0.8,␣
       ↪oob_score=True, random_state=10)
      forest_3.fit(X, Y)
      Y_predict3 = forest_3.predict(X_test)
```

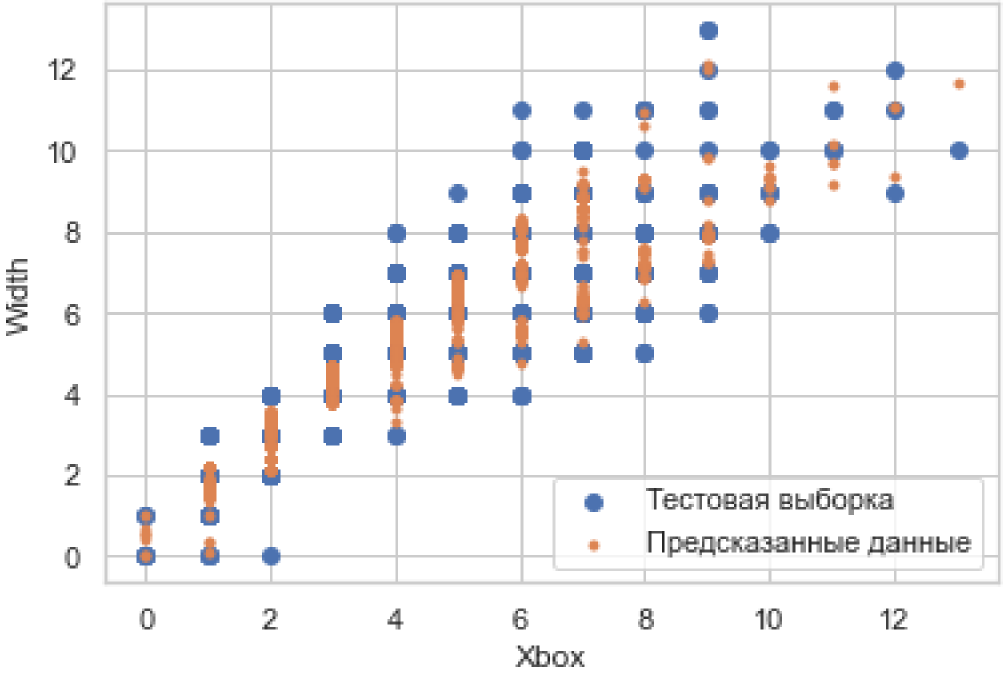
```
[23]: #
      print('                 :',   mean_absolute_error(Y_test, Y_predict3))
      print('                  :', mean_squared_error(Y_test, Y_predict3))
      print('Median absolute error:',       median_absolute_error(Y_test, Y_predict3))
      print('                :',    r2_score(Y_test, Y_predict3))
```

```
                  : 0.6506314147211842
                   : 0.6518545003531846
```

```
Median absolute error: 0.5241401178446781
              : 0.8348764662077129
```

[24]:

```
#
plt.scatter (X_test.xbox, Y_test,     marker = 'o', label = '         ')
plt.scatter (X_test.xbox, Y_predict3, marker = '.', label = '           ')
plt.legend (loc = 'lower right')
plt.xlabel ('Xbox')
plt.ylabel ('Width')
plt.show ()
```



0.4 2.

0.4.1 2.1.

5

GradientBoostingRegressor sklearn

[25]:

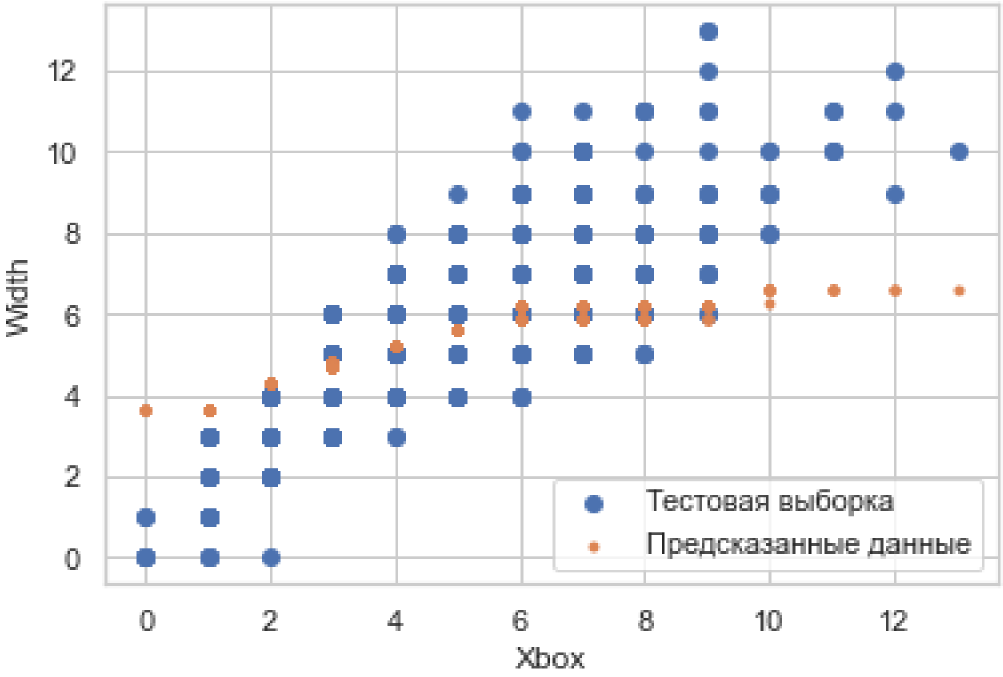
```
from sklearn.ensemble import GradientBoostingRegressor
```

[26]:
```
#
grad = GradientBoostingRegressor(n_estimators=5, random_state = 10)
grad.fit(X_train, Y_train)
Y_grad_pred = grad.predict(X_test)
```

[27]:
```
#
print('              :',   mean_absolute_error(Y_test, Y_grad_pred))
print('               :', mean_squared_error(Y_test, Y_grad_pred))
print('Median absolute error:',       median_absolute_error(Y_test,␣
 ↪Y_grad_pred))
print('             :',    r2_score(Y_test, Y_grad_pred))
```

```
              : 1.0978135621222465
               : 1.9918842979834392
Median absolute error: 0.8249463745494108
             : 0.49542885105466394
```

[28]:
```
#
plt.scatter (X_test.xbox, Y_test,      marker = 'o', label = '         ')
plt.scatter (X_test.xbox, Y_grad_pred, marker = '.', label = '           ')
plt.legend (loc = 'lower right')
plt.xlabel ('Xbox')
plt.ylabel ('Width')
plt.show ()
```



-      .

0.4.2 2.2.

[29]:
```
params = {
    'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
    'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0],
    'min_samples_leaf': [0.01, 0.04, 0.06, 0.08, 0.1]
}
```

[30]:
```
grid_gr = GridSearchCV(estimator=GradientBoostingRegressor(random_state=10),
                       param_grid=params,
                       scoring='neg_mean_squared_error',
                       cv=3,
                       n_jobs=-1)
grid_gr.fit(X, Y)
```

[30]:
```
GridSearchCV(cv=3, estimator=GradientBoostingRegressor(random_state=10),
             n_jobs=-1,
             param_grid={'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0],
                         'min_samples_leaf': [0.01, 0.04, 0.06, 0.08, 0.1],
                         'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20,
                                          25, 50, 75, 100]},
             scoring='neg_mean_squared_error')
```

[31]:
```
#
print('                          :', -grid_gr.best_score_)
print('                  :\n',           grid_gr.best_params_)
```

```
                          : 0.7589667032795534
                  :
{'max_features': 0.8, 'min_samples_leaf': 0.01, 'n_estimators': 100}
```

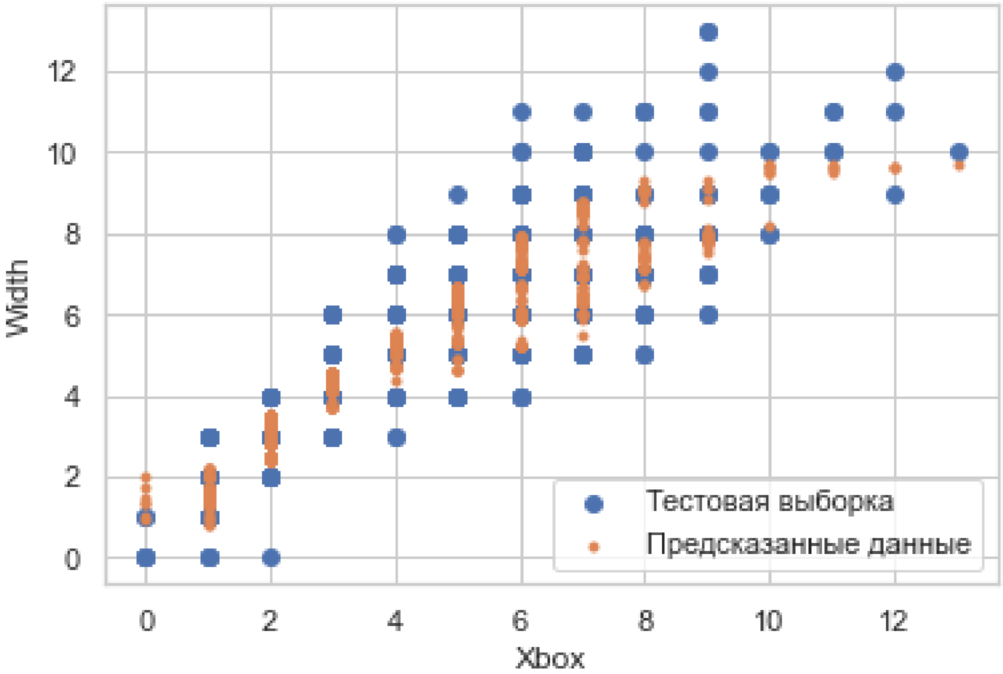
[32]:
```
#
grad1 = GradientBoostingRegressor(n_estimators=100, max_features = 0.8,␣
 ↪min_samples_leaf = 0.01, random_state = 10)
grad1.fit(X_train, Y_train)
Y_grad_pred1 = grad1.predict(X_test)
```

[33]:
```
#
print('              :',   mean_absolute_error(Y_test, Y_grad_pred1))
print('               :', mean_squared_error(Y_test, Y_grad_pred1))
print('Median absolute error:',       median_absolute_error(Y_test,␣
 ↪Y_grad_pred1))
```

```
print('             :',    r2_score(Y_test, Y_grad_pred1))
```

```
              : 0.691994537621164
               : 0.7446227771176452
Median absolute error: 0.5388123783932972
             : 0.8113770109230306
```

```
[34]: #
      plt.scatter (X_test.xbox, Y_test,       marker = 'o', label = '          ')
      plt.scatter (X_test.xbox, Y_grad_pred1, marker = '.', label = '            ')
      plt.legend (loc = 'lower right')
      plt.xlabel ('Xbox')
      plt.ylabel ('Width')
      plt.show()
```


Тестовая выборка
Предсказанные данные
Width
Xbox