_lab5

October 2, 2024

0.0.1

-

0.1 № 5

cc

0.1.1 :

:

: -01-21

0.2 2024

0.2.1 №25

25

Ozone Level Detection Data Set

: eighthr.data

: http://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection

: class (No 74)

– ˘ (SelectKBest)

:

-
-
- (degree=2)

- ROC-

1. UCI, .

[1]:
```
import numpy as np
```

[2]:
```
import pandas as pd
```

[3]:
```
import matplotlib.pyplot as plt
```

[4]:
```
import warnings
warnings.filterwarnings("ignore")
```

[5]:
```
#                  DataFrame
my_data = pd.read_csv( 'eighthr.data', header=None )
my_data
```

[5]:
```
               0    1    2    3    4    5    6    7    8    9  …    64  \
0       1/1/1998  0.8  1.8  2.4  2.1    2  2.1  1.5  1.7  1.9  …  0.15
1       1/2/1998  2.8  3.2  3.3  2.7  3.3  3.2  2.9  2.8  3.1  …  0.48
2       1/3/1998  2.9  2.8  2.6  2.1  2.2  2.5  2.5  2.7  2.2  …   0.6
3       1/4/1998  4.7  3.8  3.7  3.8  2.9  3.1  2.8  2.5  2.4  …  0.49
4       1/5/1998  2.6  2.1  1.6  1.4  0.9  1.5  1.2  1.4  1.3  …     ?
…            …  …  …  …  …  …  …  …  …  …  …  …
2529  12/27/2004  0.3  0.4  0.5  0.5  0.2  0.3  0.4  0.4  1.3  …  0.07
2530  12/28/2004    1  1.4  1.1  1.7  1.5  1.7  1.8  1.5  2.1  …  0.04
2531  12/29/2004  0.8  0.8  1.2  0.9  0.4  0.6  0.8  1.1  1.5  …  0.06
2532  12/30/2004  1.3  0.9  1.5  1.2  1.6  1.8  1.1    1  1.9  …  0.25
2533  12/31/2004  1.5  1.3  1.8  1.4  1.2  1.7  1.6  1.4  1.6  …  0.54

         65     66    67     68     69     70   71    72   73
0     10.67  -1.56  5795  -12.1   17.9  10330  -55     0  0.0
1      8.39   3.84  5805  14.05     29  10275  -55     0  0.0
2      6.94    9.8  5790   17.9   41.3  10235  -40     0  0.0
3      8.73  10.54  5775  31.15   51.7  10195  -40  2.08  0.0
4         ?      ?     ?      ?      ?      ?    ?  0.58  0.0
…     …    …   …    …    …    …   …   …   …
2529   7.93  -4.41  5800  -25.6   21.8  10295   65     0  0.0
2530   5.95  -1.14  5845  -19.4   19.1  10310   15     0  0.0
2531    7.8  -0.64  5845   -9.6   35.2  10275  -35     0  0.0
2532   7.72  -0.89  5845  -19.6   34.2  10245  -30  0.05  0.0
2533  13.07   9.15  5820   1.95  39.35  10220  -25     0  0.0

[2534 rows x 74 columns]
```

[6]:
```
my_data = my_data.rename(columns={73: "class"})
my_data
```

[6]:
```
                 0    1    2    3    4    5    6    7    8    9  …    64  \
0         1/1/1998  0.8  1.8  2.4  2.1    2  2.1  1.5  1.7  1.9  …  0.15
1         1/2/1998  2.8  3.2  3.3  2.7  3.3  3.2  2.9  2.8  3.1  …  0.48
```

```
2        1/3/1998  2.9  2.8  2.6  2.1  2.2  2.5  2.5  2.7  2.2  …   0.6
3        1/4/1998  4.7  3.8  3.7  3.8  2.9  3.1  2.8  2.5  2.4  …  0.49
4        1/5/1998  2.6  2.1  1.6  1.4  0.9  1.5  1.2  1.4  1.3  …     ?
…             …   …   …   …   …   …   …   …   …   …   …   …
2529   12/27/2004  0.3  0.4  0.5  0.5  0.2  0.3  0.4  0.4  1.3  …  0.07
2530   12/28/2004    1  1.4  1.1  1.7  1.5  1.7  1.8  1.5  2.1  …  0.04
2531   12/29/2004  0.8  0.8  1.2  0.9  0.4  0.6  0.8  1.1  1.5  …  0.06
2532   12/30/2004  1.3  0.9  1.5  1.2  1.6  1.8  1.1    1  1.9  …  0.25
2533   12/31/2004  1.5  1.3  1.8  1.4  1.2  1.7  1.6  1.4  1.6  …  0.54

          65     66    67     68     69     70   71    72 class
0      10.67  -1.56  5795  -12.1   17.9  10330  -55     0   0.0
1       8.39   3.84  5805  14.05     29  10275  -55     0   0.0
2       6.94    9.8  5790   17.9   41.3  10235  -40     0   0.0
3       8.73  10.54  5775  31.15   51.7  10195  -40  2.08   0.0
4          ?      ?     ?      ?      ?      ?    ?  0.58   0.0
…      …    …    …    …    …    …   …    …    …
2529    7.93  -4.41  5800  -25.6   21.8  10295   65     0   0.0
2530    5.95  -1.14  5845  -19.4   19.1  10310   15     0   0.0
2531     7.8  -0.64  5845   -9.6   35.2  10275  -35     0   0.0
2532    7.72  -0.89  5845  -19.6   34.2  10245  -30  0.05   0.0
2533   13.07   9.15  5820   1.95  39.35  10220  -25     0   0.0

[2534 rows x 74 columns]
```

2. , .
, ,
. ( ), .

[7]:
```
#

my_data['class'].isnull().sum(axis=0)
```

[7]:
```
0
```

[8]:
```
#

len(my_data['class'].unique())
```

[8]:
```
2
```

3. - , .
( ) . ,
.

[9]:
```
#
my_data.dtypes
```

```
[9]: 0         object
     1         object
     2         object
     3         object
     4         object
                ...
     69        object
     70        object
     71        object
     72        object
     class    float64
     Length: 74, dtype: object
```

```
[10]: #                    NaN
      my_data[:73] = my_data[:73].replace('?', np.nan)
```

```
[11]: my_data = my_data.drop(my_data.columns[0], axis=1)
      my_data.dtypes
```

```
[11]: 1         object
      2         object
      3         object
      4         object
      5         object
                 ...
      69        object
      70        object
      71        object
      72        object
      class    float64
      Length: 73, dtype: object
```

```
[12]: for col in my_data.columns[:73]:
          my_data[col] = pd.to_numeric(my_data[col], errors='coerce')

      #     NaN
      for col in my_data.columns[:73]:
          my_data[col] = my_data.groupby('class')[col].transform(lambda x: x.fillna(x.
        ↪mean()))
```

```
[13]: #                 :     object
      #         float
      my_data.dtypes
```

```
[13]: 1        float64
      2        float64
      3        float64
```

```
4        float64
5        float64
          …
69       float64
70       float64
71       float64
72       float64
class    float64
Length: 73, dtype: object
```

[14]:
```
#
#
my_data.isnull().sum(axis=0)
```

[14]:
```
1        0
2        0
3        0
4        0
5        0
        ..
69       0
70       0
71       0
72       0
class    0
Length: 73, dtype: int64
```

4. .

[15]:
```
y = my_data['class']
X = my_data.drop(columns='class')
X.shape, y.shape
```

[15]:
```
((2534, 72), (2534,))
```

[16]:
```
from sklearn import preprocessing
X1 = X.to_numpy()
X_scaled = preprocessing.scale(X1)
new_data = pd.DataFrame(X_scaled)
new_data
```

[16]:
```
            0         1         2         3         4         5         6   \
0    -0.706100  0.175988  0.729019  0.502307  0.420123  0.502524 -0.128527
1     0.966740  1.350171  1.500432  1.031026  1.572305  1.499570  1.151769
2     1.050382  1.014690  0.900444  0.502307  0.597382  0.865086  0.785970
3     2.555938  1.853393  1.843283  2.000345  1.217787  1.408930  1.060319
4     0.799456  0.427599  0.043319 -0.114532 -0.554800 -0.041319 -0.402876
…         …         …         …         …         …         …         …
```

```
2529 -1.124310 -0.998196 -0.899520 -0.907611 -1.175206 -1.129006 -1.134474
2530 -0.538816 -0.159493 -0.385244  0.149827 -0.023024  0.139962  0.145822
2531 -0.706100 -0.662715 -0.299532 -0.555132 -0.997947 -0.857084 -0.768675
2532 -0.287890 -0.578844 -0.042394 -0.290772  0.065605  0.230602 -0.494326
2533 -0.120606 -0.243363  0.214744 -0.114532 -0.288912  0.139962 -0.037077

             7         8         9  …        62        63        64  \
0    -0.319853 -0.575293 -0.479037  … -1.314553 -0.637391  0.086034
1     0.686889  0.500175  0.476524  … -1.050979  0.721427 -0.163600
2     0.595367 -0.306426 -0.305299  … -1.419982  1.215542 -0.322358
3     0.412323 -0.127181  0.215917  … -1.657199  0.762603 -0.126374
4    -0.594419 -1.113027 -1.260860  … -0.037286  0.026451  0.044503
…          …         …         …  …         …         …         …
2529 -1.509638 -1.113027 -0.565906  … -0.497475 -0.966801 -0.213964
2530 -0.502897 -0.396048 -0.392168  … -0.392045 -1.090330 -0.430752
2531 -0.868984 -0.933782 -1.173991  … -0.339331 -1.007977 -0.228198
2532 -0.960506 -0.575293 -0.739645  … -0.075757 -0.225628 -0.236957
2533 -0.594419 -0.844160  0.129048  … -0.365688  0.968484  0.348807

            65        66        67        68        69        70        71
0    -0.340272 -0.307405 -1.122041 -1.780072  3.223762 -1.582211 -0.282455
1     0.426052 -0.178248  0.175691 -0.766119  2.154203 -1.582211 -0.282455
2     1.271847 -0.371983  0.366752  0.357450  1.376342 -1.149766 -0.282455
3     1.376861 -0.565718  1.024303  1.307460  0.598481 -1.149766  1.296679
4     0.037422 -0.037417 -0.006215 -0.010517  0.025822  0.001992  0.157881
…          …         …         …         …         …         …         …
2529 -0.744721 -0.242826 -1.791998 -1.423818  2.543133  1.877344 -0.282455
2530 -0.280669  0.338379 -1.484314 -1.670455  2.834831  0.435863 -0.282455
2531 -0.209713  0.338379 -0.997975 -0.199767  2.154203 -1.005618 -0.282455
2532 -0.245191  0.338379 -1.494239 -0.291114  1.570807 -0.861470 -0.244495
2533  1.179604  0.015487 -0.424789  0.179323  1.084644 -0.717322 -0.282455

[2534 rows x 72 columns]
```

5. , , ,
10 . 25: – ˘ (SelectKBest)

```
[17]: from sklearn.feature_selection import SelectKBest, f_classif
      valid_columns = [col for col in new_data.columns if new_data[col].nunique() >␣
       ↪10 and col != 'class']

      #
      selector = SelectKBest(score_func=f_classif, k=2)
      X_new = selector.fit_transform(X, y)

      #
      selected_indices = selector.get_support(indices=True)
```

```
#
selected_features = X.columns[selected_indices]
print("          :", selected_features)
```

```
          : Index([42, 43], dtype='object')
```

[18]:
```
#     ,                10
len(new_data[42].unique())
```

[18]:
```
339
```

[19]:
```
#     ,                10
len(new_data[43].unique())
```

[19]:
```
331
```

[20]:
```
#       2
data2 = new_data[[42,43]]
```

6. ,

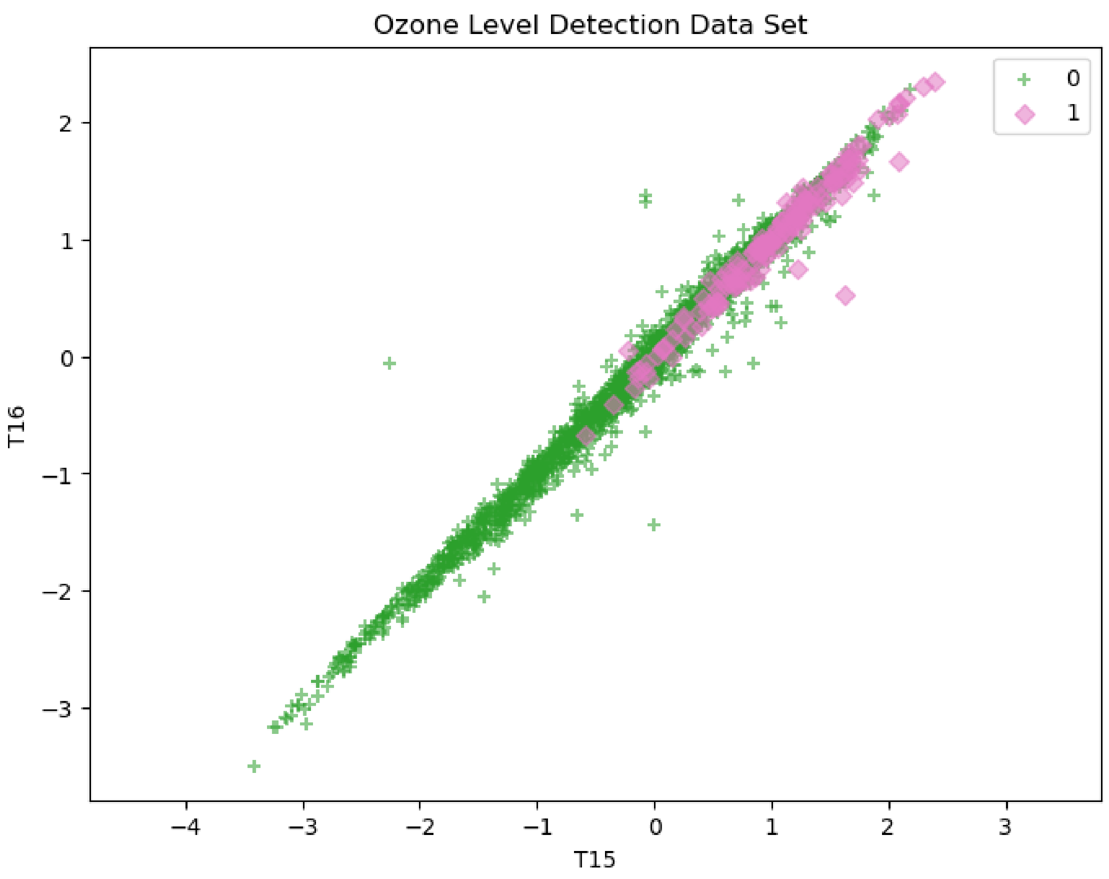
. , .

. .

[21]:
```
y_int = y.astype(int)
```

[22]:
```
plt.figure(figsize=(8, 6))
colors = ['tab:green', 'tab:pink']
markers = ['+', 'D']


for clr in y_int.unique():
    xx = data2[42].loc[y==clr]
    yy = data2[43].loc[y==clr]
    plt.scatter(xx, yy, c = colors[clr], label=clr,
                marker=markers[clr], alpha=0.55)
plt.axis('equal')
plt.title('Ozone Level Detection Data Set ')
plt.xlabel('T15')
plt.ylabel('T16')
plt.legend()
```

[22]:
```
<matplotlib.legend.Legend at 0x2283d3b0610>
```



7. , . **70%** **30%.** 3:

:

-
-
- (degree=2)

[23]:
```
#
def train_test_split(X, y, test_ratio=0.2, seed=None):
    """returns X_train, X_test, y_train, y_test"""
    assert X.shape[0] == y.shape[0], \
        "the size of X must be equal to the size of y"
    assert 0.0 <= test_ratio <= 1.0, \
        "test_ration must be valid"

    if seed:
        np.random.seed(seed)
    shuffled_indexes = np.random.permutation(len(X))

    test_size = int(len(X) * test_ratio)
    test_indexes = shuffled_indexes[:test_size]
    train_indexes = shuffled_indexes[test_size:]

    X_train = X[train_indexes]
    y_train = y[train_indexes]

    X_test = X[test_indexes]
    y_test = y[test_indexes]

    return X_train, X_test, y_train, y_test
```

[24]:
```
X_train, X_test, y_train, y_test = train_test_split(data2.to_numpy(), y, 0.3)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

[24]:
```
((1774, 2), (1774,), (760, 2), (760,))
```

[25]:
```
#
from sklearn.naive_bayes import GaussianNB

nbc = GaussianNB()
nbc.fit(X_train, y_train)
y_pred_nbc = nbc.predict(X_test)
```

[26]:
```
from sklearn.linear_model import LogisticRegression

lg = LogisticRegression()
lg.fit(X_train, y_train)
y_pred_jg = lg.predict(X_test)
```

[27]:
```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

#
lgp = LogisticRegression()

#
lgp.fit(X_train_poly, y_train)

#
y_pred_lpg = lgp.predict(X_test_poly)
```

8. ,

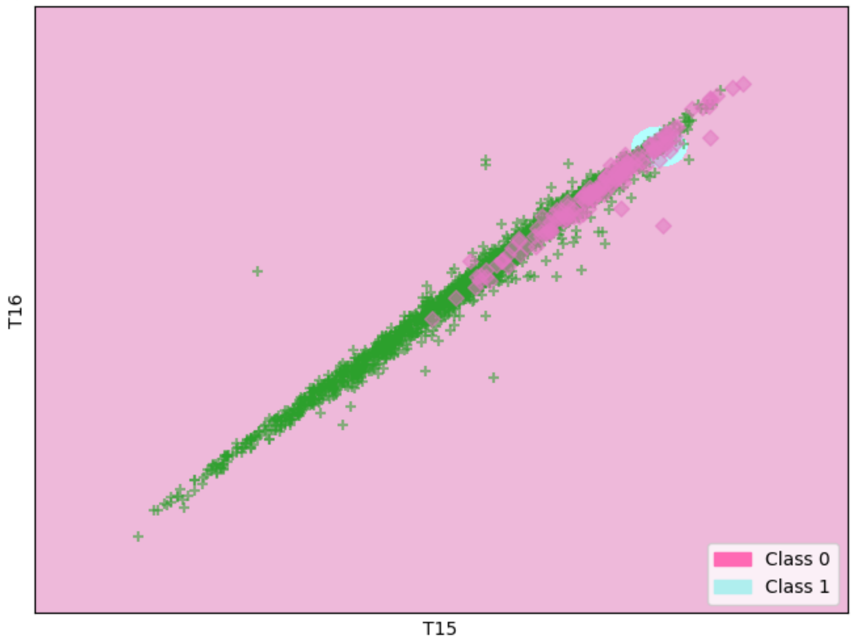
. **6.**

[28]:
```python
data2_np = data2.to_numpy()
```

[29]:
```python
import matplotlib.patches as mpatches
from matplotlib.colors import ListedColormap
```

[31]:
```python
h = 0.01
x_min, x_max = data2_np[:, 0].min() - 1, data2_np[:, 0].max() + 1
y_min, y_max = data2_np[:, 1].min() - 1, data2_np[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = nbc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

cmap_decision = ListedColormap(['#C71585', '#00FFFF'])

#
plt.figure(figsize=(8, 6))
plt.imshow(Z, interpolation="nearest", extent=(xx.min(), xx.max(), yy.min(), yy.
 ↪max()), aspect="auto", origin="lower", cmap=cmap_decision, alpha=0.3)

colors = ['tab:green', 'tab:pink']
markers = ['+', 'D']


for clr in y_int.unique():
    xx = data2[42].loc[y==clr]
    yy = data2[43].loc[y==clr]
    plt.scatter(xx, yy, c = colors[clr], label=clr,
                marker=markers[clr], alpha=0.55)

legend_class0 = mpatches.Patch(color='#FF69B4', label='Class 0')
legend_class1 = mpatches.Patch(color='#AFEEEE', label='Class 1')

plt.legend(handles=[legend_class0, legend_class1], loc='lower right')

plt.title("                                      Ozone level detection")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xlabel('T15')  #        X
plt.ylabel('T16')  #        Y
plt.xticks(())
plt.yticks(())

plt.show()
```

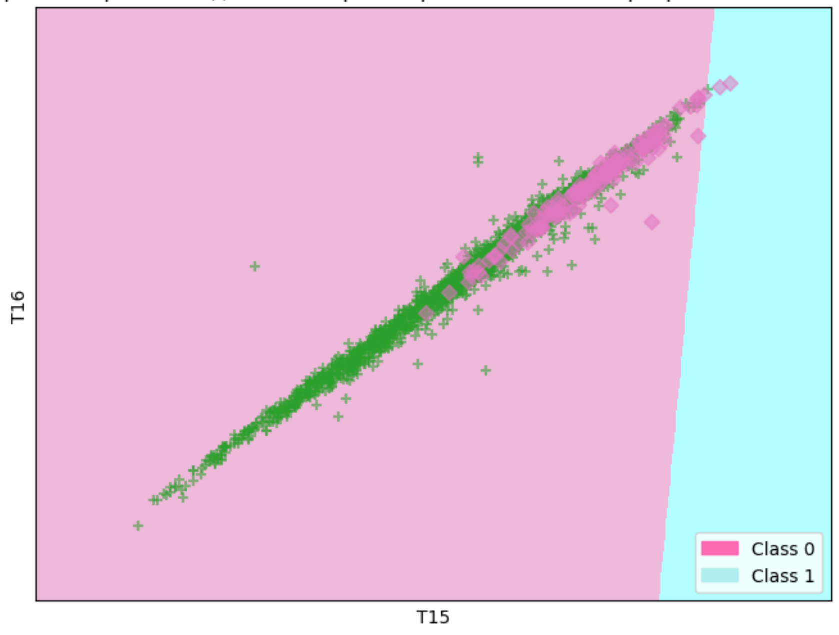
Границы принятия решений для наивного байесовского классификатора Ozone level detection



```
[34]: h = 0.01
x_min, x_max = data2_np[:, 0].min() - 1, data2_np[:, 0].max() + 1
y_min, y_max = data2_np[:, 1].min() - 1, data2_np[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = lg.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

cmap_decision = ListedColormap(['#C71585', '#00FFFF'])

#
plt.figure(figsize=(8, 6))
plt.imshow(Z, interpolation="nearest", extent=(xx.min(), xx.max(), yy.min(), yy.
  ↪max()), aspect="auto", origin="lower", cmap=cmap_decision, alpha=0.3)

colors = ['tab:green', 'tab:pink']
markers = ['+', 'D']

for clr in y_int.unique():
    xx = data2[42].loc[y==clr]
    yy = data2[43].loc[y==clr]
    plt.scatter(xx, yy, c = colors[clr], label=clr,
                marker=markers[clr], alpha=0.55)

legend_class0 = mpatches.Patch(color='#FF69B4', label='Class 0')
legend_class1 = mpatches.Patch(color='#AFEEEE', label='Class 1')
```

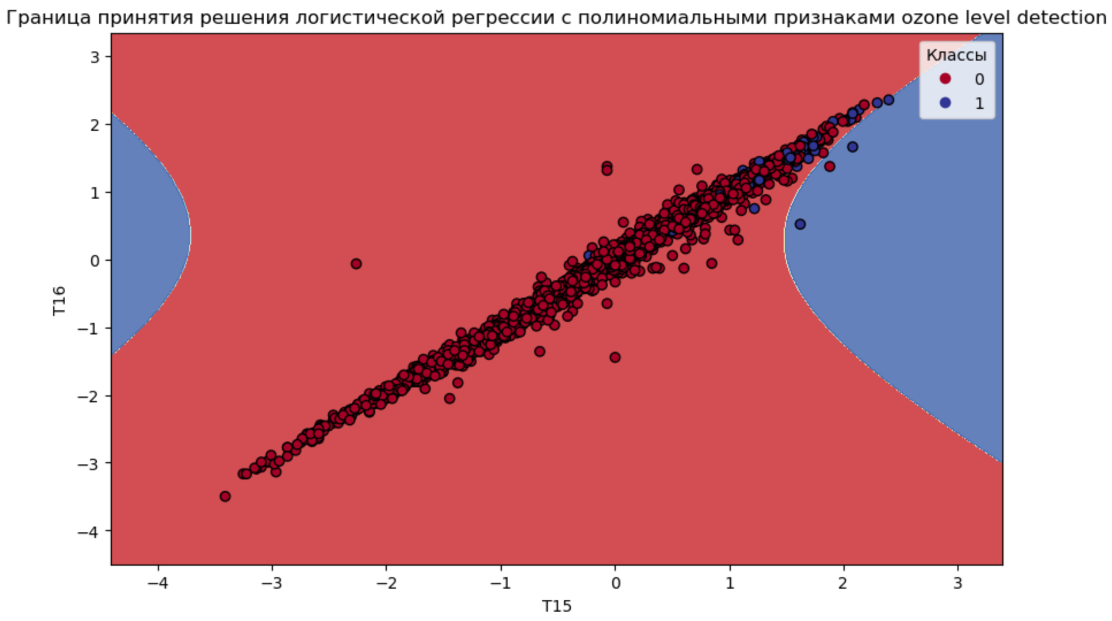
```
plt.legend(handles=[legend_class0, legend_class1], loc='lower right')

plt.title("                                        Ozone level detection")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xlabel('T15')
plt.ylabel('T16')
plt.xticks(())
plt.yticks(())

plt.show()
```

Границы принятия решений для классификатора логистической регрессии Ozone level detection



[35]:
```
x_min, x_max = data2_np[:, 0].min() - 1, data2_np[:, 0].max() + 1
y_min, y_max = data2_np[:, 1].min() - 1, data2_np[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))

#
Z = lgp.predict(poly.transform(np.c_[xx.ravel(), yy.ravel()]))
Z = Z.reshape(xx.shape)

#
plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.RdYlBu)
```

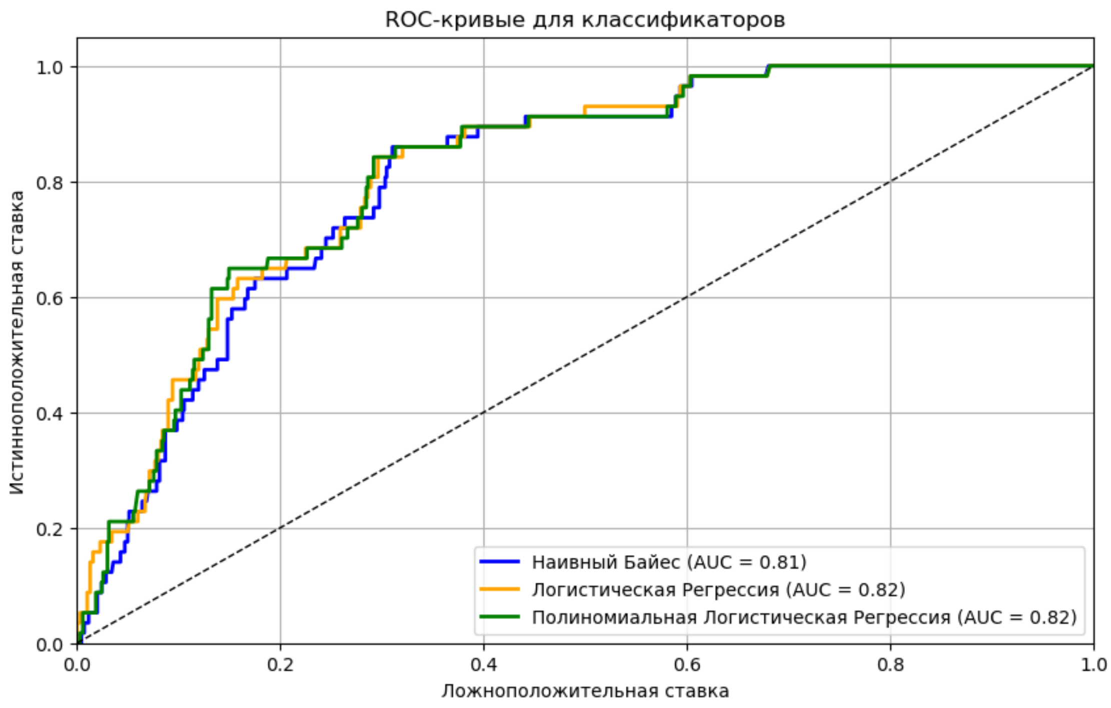
```
#
scatter = plt.scatter(data2_np[:, 0], data2_np[:, 1], c=y, edgecolors='k',␣
 ↪marker='o', cmap=plt.cm.RdYlBu)

#
plt.title("                                  ozone level detection")
plt.xlabel("T15")
plt.ylabel("T16")
plt.legend(*scatter.legend_elements(), title="   ")
plt.show()
```



9. , , ,
. . 3: - ROC-

[36]:
```
from sklearn.metrics import roc_curve, auc
y_prob_nbc = nbc.predict_proba(X_test)[:, 1]
y_prob_jg = lg.predict_proba(X_test)[:, 1]
y_prob_lpg = lgp.predict_proba(X_test_poly)[:, 1]

#       ROC-
fpr_nbc, tpr_nbc, _ = roc_curve(y_test, y_prob_nbc)
fpr_jg, tpr_jg, _ = roc_curve(y_test, y_prob_jg)
fpr_lpg, tpr_lpg, _ = roc_curve(y_test, y_prob_lpg)

#       AUC (           )
```

```
roc_auc_nbc = auc(fpr_nbc, tpr_nbc)
roc_auc_jg = auc(fpr_jg, tpr_jg)
roc_auc_lpg = auc(fpr_lpg, tpr_lpg)

#      ROC-
plt.figure(figsize=(10, 6))
plt.plot(fpr_nbc, tpr_nbc, color='blue', lw=2, label='         (AUC = {:.2f})'.
 ↪format(roc_auc_nbc))
plt.plot(fpr_jg, tpr_jg, color='orange', lw=2, label='              (AUC = {:.
 ↪2f})'.format(roc_auc_jg))
plt.plot(fpr_lpg, tpr_lpg, color='green', lw=2, label='                      ␣
 ↪(AUC = {:.2f})'.format(roc_auc_lpg))

#
plt.plot([0, 1], [0, 1], color='black', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('              ')
plt.ylabel('               ')
plt.title('ROC-               ')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```



10.   ,   . 9.

```
[37]: #                  AUC
best_auc = max(roc_auc_nbc, roc_auc_jg, roc_auc_lpg)

if best_auc == roc_auc_nbc:
    best_model = "                   "
elif best_auc == roc_auc_jg:
    best_model = "             "
else:
    best_model = "                              "

print(f"       :   AUC: {best_model}")
```

```
       :   AUC:
```