РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

Дисциплина: Интеллектуальный анализ данных

Студент: Генералов Даниил

Группа: НПИбд-01-21

Москва 2024

---

1. Считайте заданный набор данных из репозитария UCI, включая указанный в индивидуальном задании столбец с метками классов.
2. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если столбец с метками классов содержит более двух классов, то объедините некоторые классы, чтобы получить набор для бинарной классификации с примерно равным количеством точек в положительном и отрицательном классах. Если один из классов является преобладающим (мажоритарным), то объедините все прочие классы в другой класс.
3. Если какие-либо числовые признаки в наборе были распознаны неверно, то преобразуйте их в числовые. Удалите из набора признаки с текстовыми (категориальными) значениями. Если в оставшихся числовых признаках имеются пропущенные значения, то замените их на средние значения для положительного и отрицательного классов.
4. Выполните стандартизацию признаков набора данных.
5. Используя метод отбора признаков, указанный в индивидуальном задании, определите и оставьте в наборе данных два наиболее значимых признака, принимающих более 10 различных значений.
6. Визуализируйте набора данных в виде точек на плоскости, отображая точки положительного и отрицательного классов разными цветами и

разными маркерами. В качестве подписей осей используйте названия признаков, согласно описания набора данных. В подписи рисунка укажите название набора данных. Создайте легенду набора данных.

7. Создайте модели классификации точек набора данных из двух признаков на базе классификаторов, указанных в индивидуальном задании. Используйте при обучении классификаторов разделение набора данных на обучающую и тестовую выборки в соотношении 70% на 30%.

8. Визуализируйте для каждого из классификаторов границу принятия решения, подписывая оси и рисунок и создавая легенду для меток классов набора данных в соответствии с требованиями п. 6.

9. Визуализируйте на одном рисунке кривые бинарной классификации, указанные в индивидуальном задании, для каждого из классификаторов, подписывая оси и рисунок. Используйте в качестве меток легенды для названия классификаторов.

10. Определите лучший из используемых методов бинарной классификации по показателю площади, ограниченной кривой из п. 9.

Вариант 7

Forest Fires Data Set

Название файла: forestfires.csv

Ссылка: http://archive.ics.uci.edu/ml/datasets/Forest+Fires

Класс: month (столбец No 3)

Метод отбора признаков – одномерный отбор признаков (SelectKBest)

Модели классификации:

- `наивный байесовский классификатор`
- `классификатор логистической регрессии`
- `классификатор логистической регрессии с полиномиальной зависимостью (degree=2)`

Кривая для визуализации - PR-кривая

1. открыть базу данных и прочитать значения

In [1]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
forest_fires = fetch_ucirepo(id=162)
```

In [2]:
```
forest_fires['data'].keys()
```

Out[2]:
```
dict_keys(['ids', 'features', 'targets', 'original', 'headers'])
```

In [3]:
```
forest_fires['data']['features']
```

Out[3]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0
515	1	4	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0
516	6	3	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0

517 rows × 12 columns

In [4]:
```
forest_fires['data']['features']['month'].unique()
```

Out[4]:
```
array(['mar', 'oct', 'aug', 'sep', 'apr', 'jun', 'jul', 'feb', 'jan',
       'dec', 'may', 'nov'], dtype=object)
```

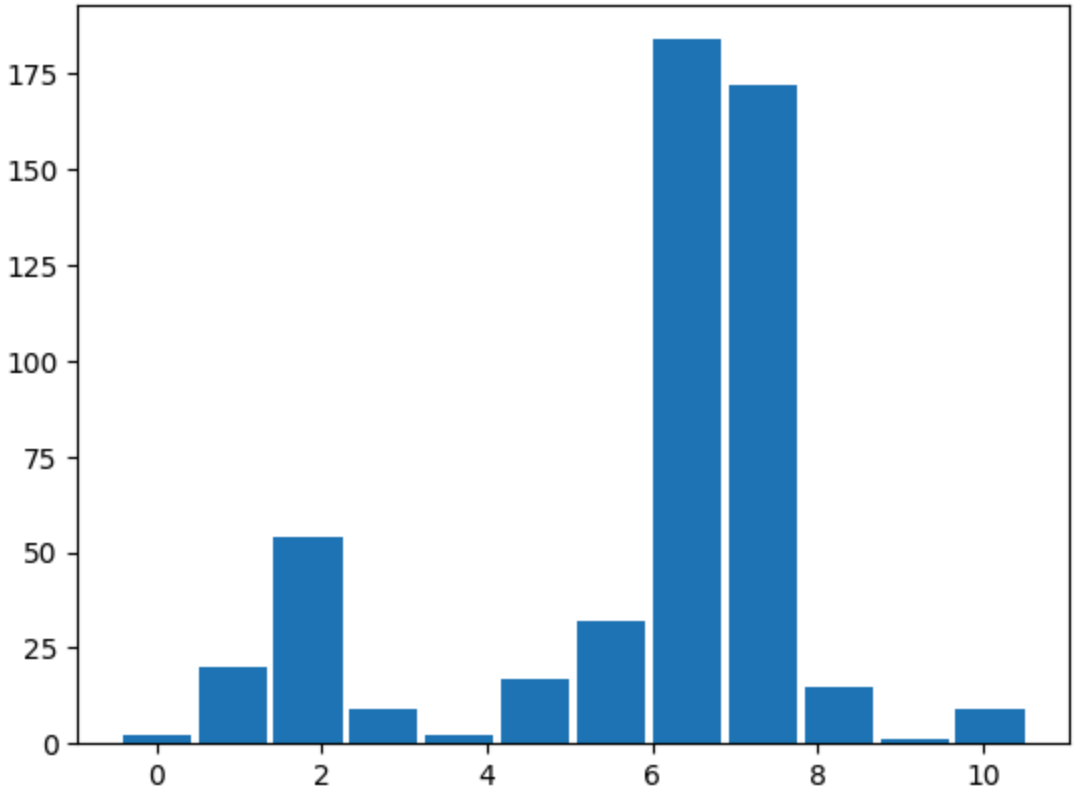
In [5]:
```
data = forest_fires['data']['features']
```

2. объединение классов

In [6]:
```
KLASS = 'month'

klassOrd = ['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', '
```

```
data[KLASS] = data[KLASS].apply(lambda x: klassOrd.index(x))

import matplotlib.pyplot as plt
counts, edges, bars = plt.hist(data[KLASS], bins=len(klassOrd), rwidth=0.9,
# for b in bars:
#     plt.bar_label(b)
```

```
/tmp/ipykernel_38578/963419700.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[KLASS] = data[KLASS].apply(lambda x: klassOrd.index(x))
```



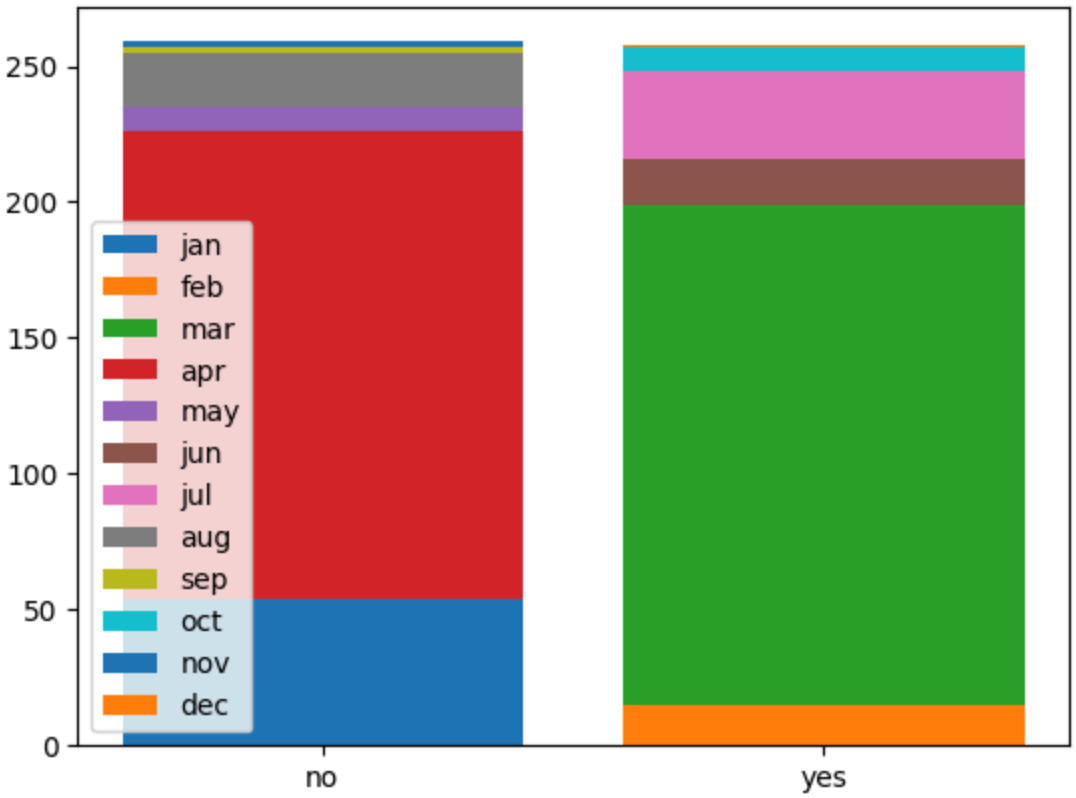
In [7]:
```
import random
import collections
hist = collections.Counter(data[KLASS])

best_score = float('inf')
best_yes = []
best_no = []
for _ in range(100000):
    yes = []
    no = []
    for k, v in hist.items():
        if random.random() < 0.5:
            yes.append(k)
        else:
            no.append(k)

    score = sum(hist[k] for k in yes) - sum(hist[k] for k in no)
```

```
    if abs(score) < abs(best_score):
        print(score)
        best_score = score
        best_yes = yes
        best_no = no

yes = best_yes
no = best_no
print(sum(hist[k] for k in yes), sum(hist[k] for k in no))
```

```
67
59
47
-7
-1
258 259
```

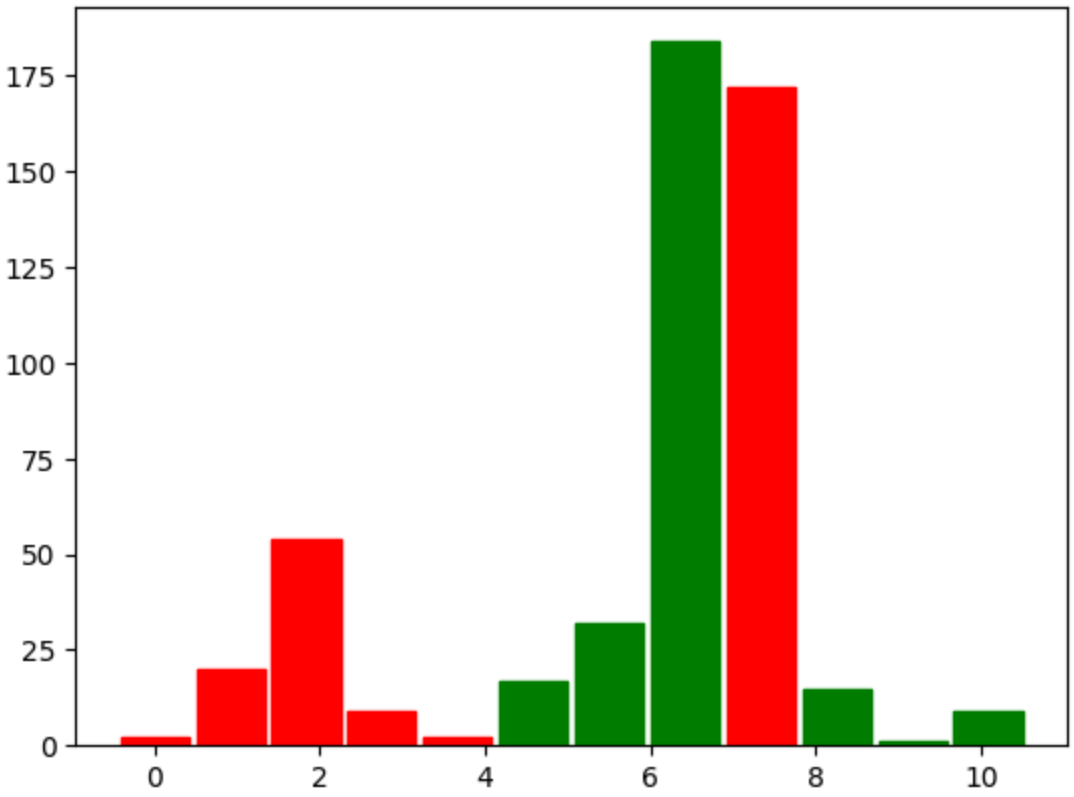
In [8]:
```
import numpy as np

yes_data = []
no_data = []
for k, v in hist.items():
    if k in yes:
        yes_data.append(v)
        no_data.append(0)
    else:
        no_data.append(v)
        yes_data.append(0)

ax = plt.axes()
bottom = np.zeros(2)
for k, y, n in zip(klassOrd, yes_data, no_data):
    if y:
        ax.bar('yes', y, bottom=bottom[0], label=k)
    else:
        ax.bar('no', n, bottom=bottom[1], label=k)
    bottom += np.array([y, n])
ax.legend()
```

Out[8]:
```
<matplotlib.legend.Legend at 0x7eff70d64590>
```



In [9]:
```
data['klass'] = data[KLASS].apply(lambda x: 1 if x in yes else 0)
```

In [10]:
```
import matplotlib.pyplot as plt
counts, edges, bars = plt.hist(data[KLASS], bins=len(klassOrd), rwidth=0.9,
for k, b in zip(klassOrd, bars):
    k = klassOrd.index(k)
    if k in yes:
        b.set_color('green')
    else:
        b.set_color('red')
```



3. удалить текстовые признаки

In [11]:
```
del data['month']
del data['day']
data
```

Out[11]:

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	klass
0	7	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0
1	7	4	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	1
2	7	4	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	1
3	8	6	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0
4	8	6	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...
512	4	3	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	1
513	2	4	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	1
514	7	4	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	1
515	1	4	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	1
516	6	3	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	1

517 rows × 11 columns

4. стандартизировать числовые признаки

In [12]:
```
klassData = data['klass']
data = (data - data.mean()) / data.std()
data['klass'] = klassData
data
```

Out[12]:

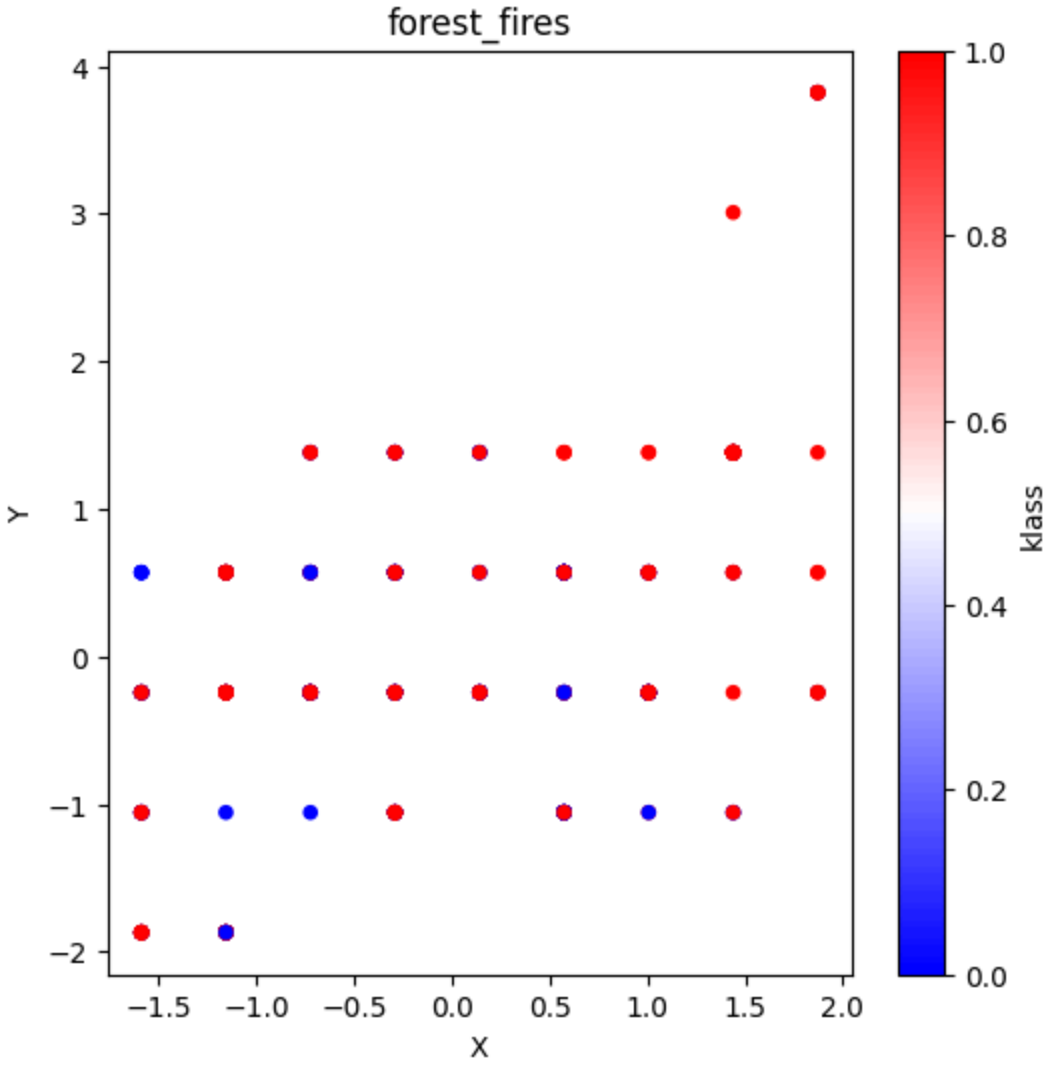
	X	Y	FFMC	DMC	DC	ISI	temp
0	1.007337	0.569309	-0.805180	-1.322045	-1.828706	-0.860113	-1.840857
1	1.007337	-0.243765	-0.008094	-1.178399	0.488418	-0.509195	-0.153130
2	1.007337	-0.243765	-0.008094	-1.048806	0.560173	-0.509195	-0.738668
3	1.439531	1.382383	0.191177	-1.211188	-1.896429	-0.004751	-1.823636
4	1.439531	1.382383	-0.243597	-0.930142	-1.796859	0.126843	-1.289763
...	...	...	...	...	...	...	...
512	-0.289244	-1.056839	-1.638496	-0.845829	0.474309	-1.561947	1.534597
513	-1.153631	-0.243765	-1.638496	-0.845829	0.474309	-1.561947	0.518517
514	1.007337	-0.243765	-1.638496	-0.845829	0.474309	-1.561947	0.397965
515	-1.585825	-0.243765	0.680298	0.548471	0.269122	0.499693	1.155720
516	0.575144	-1.056839	-2.018923	-1.684282	-1.778719	-1.737406	-1.220876

517 rows × 11 columns

5. визуализация

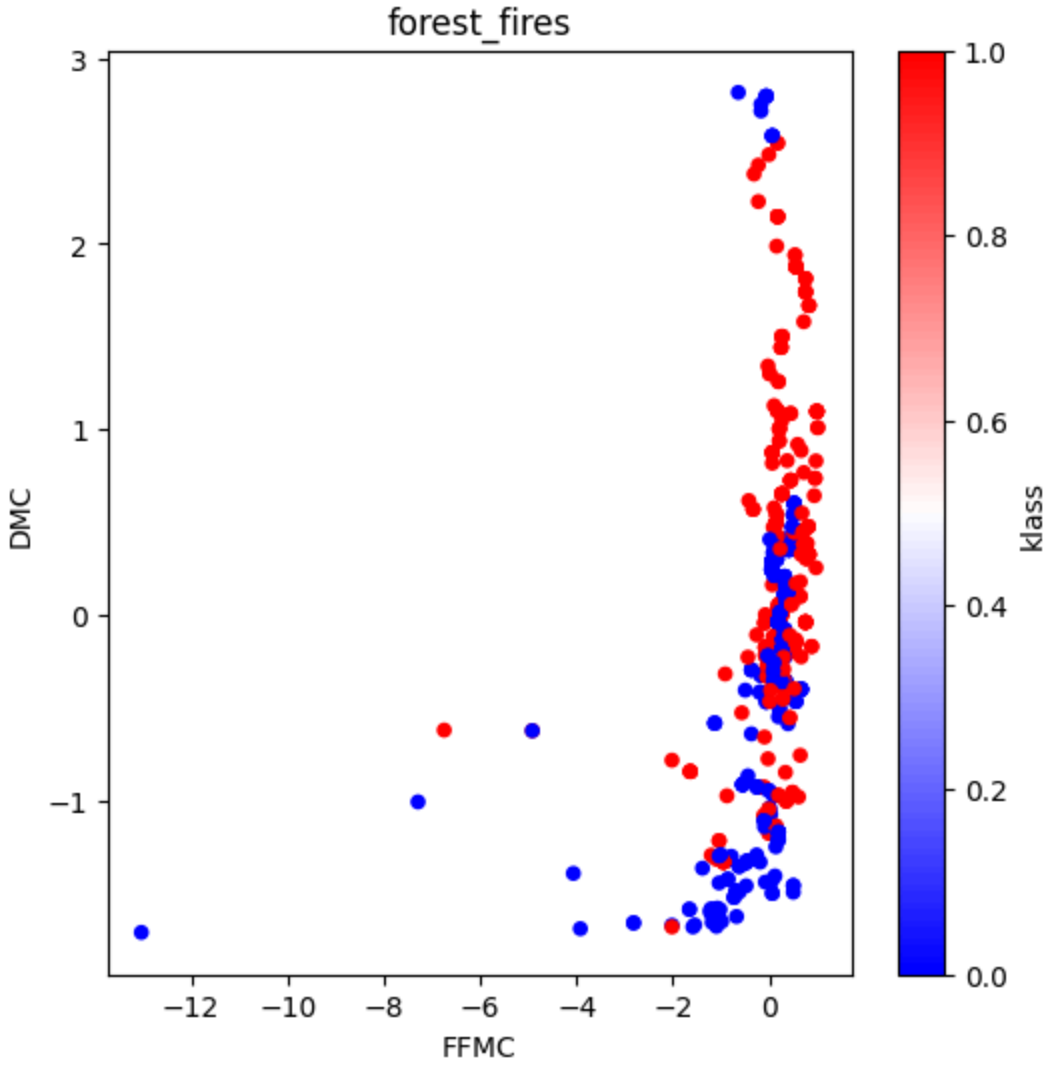
In [13]:
```
data.plot.scatter('X', 'Y', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[13]:
```
Text(0.5, 1.0, 'forest_fires')
```



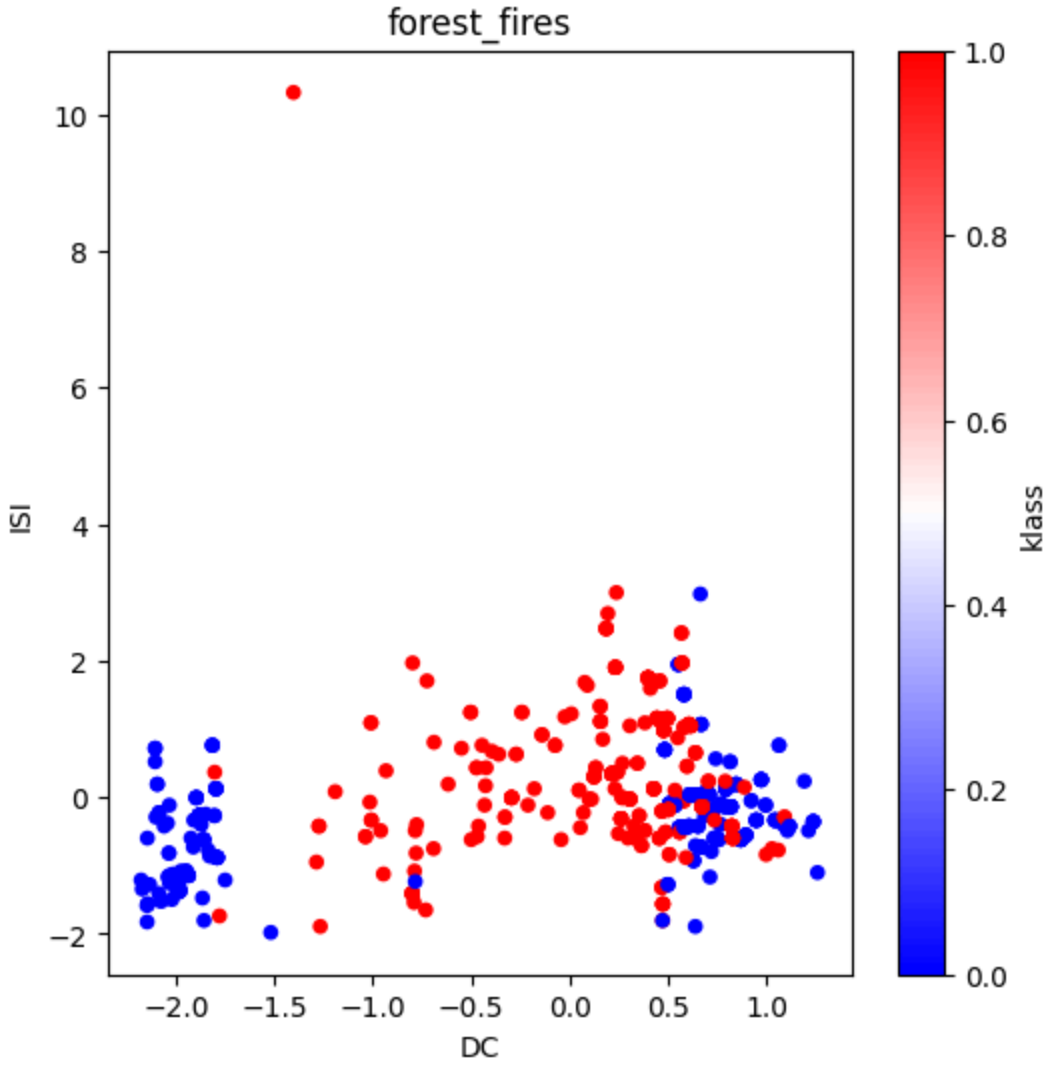
In [14]:
```
data.plot.scatter('FFMC', 'DMC', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[14]:
```
Text(0.5, 1.0, 'forest_fires')
```



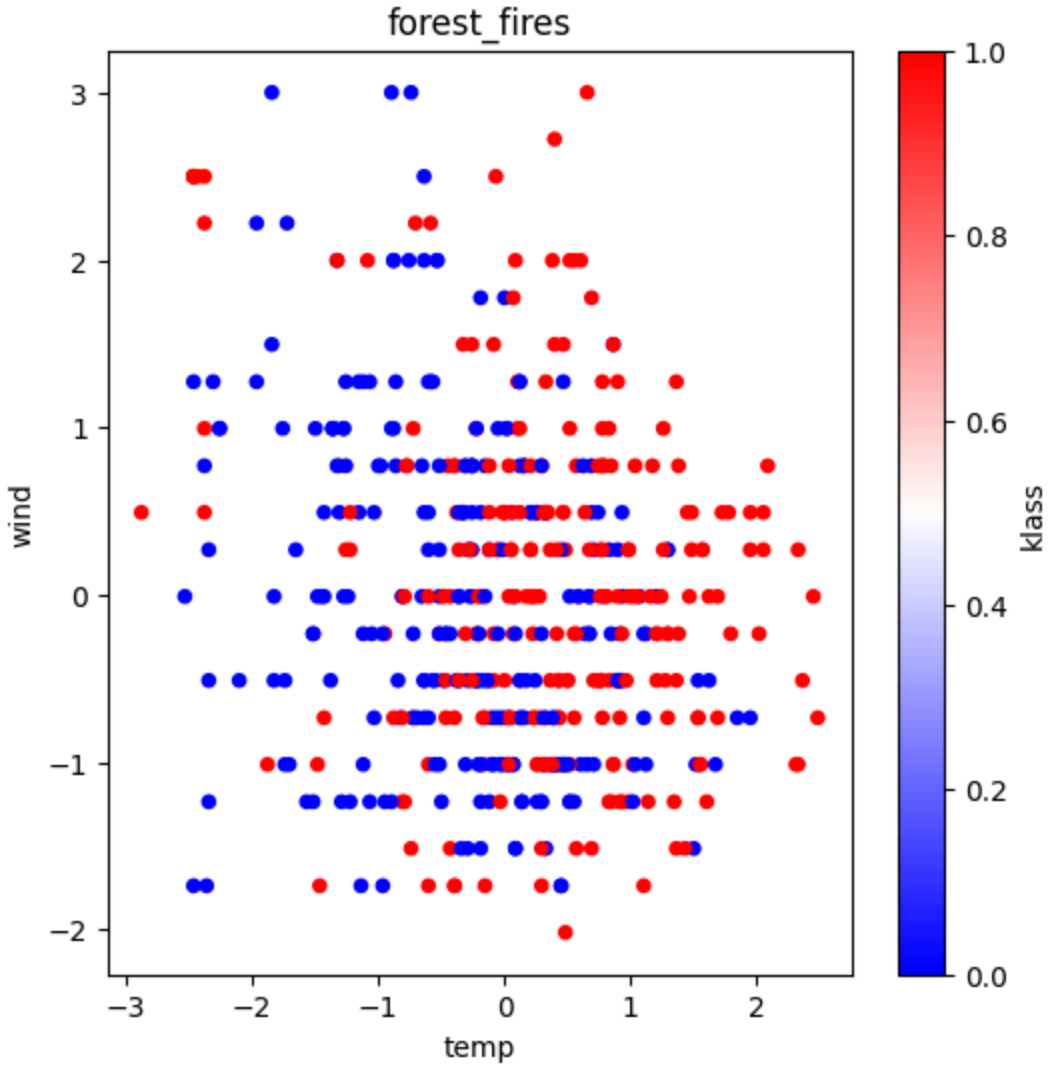
In [15]:
```
data.plot.scatter('DC', 'ISI', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[15]:
```
Text(0.5, 1.0, 'forest_fires')
```



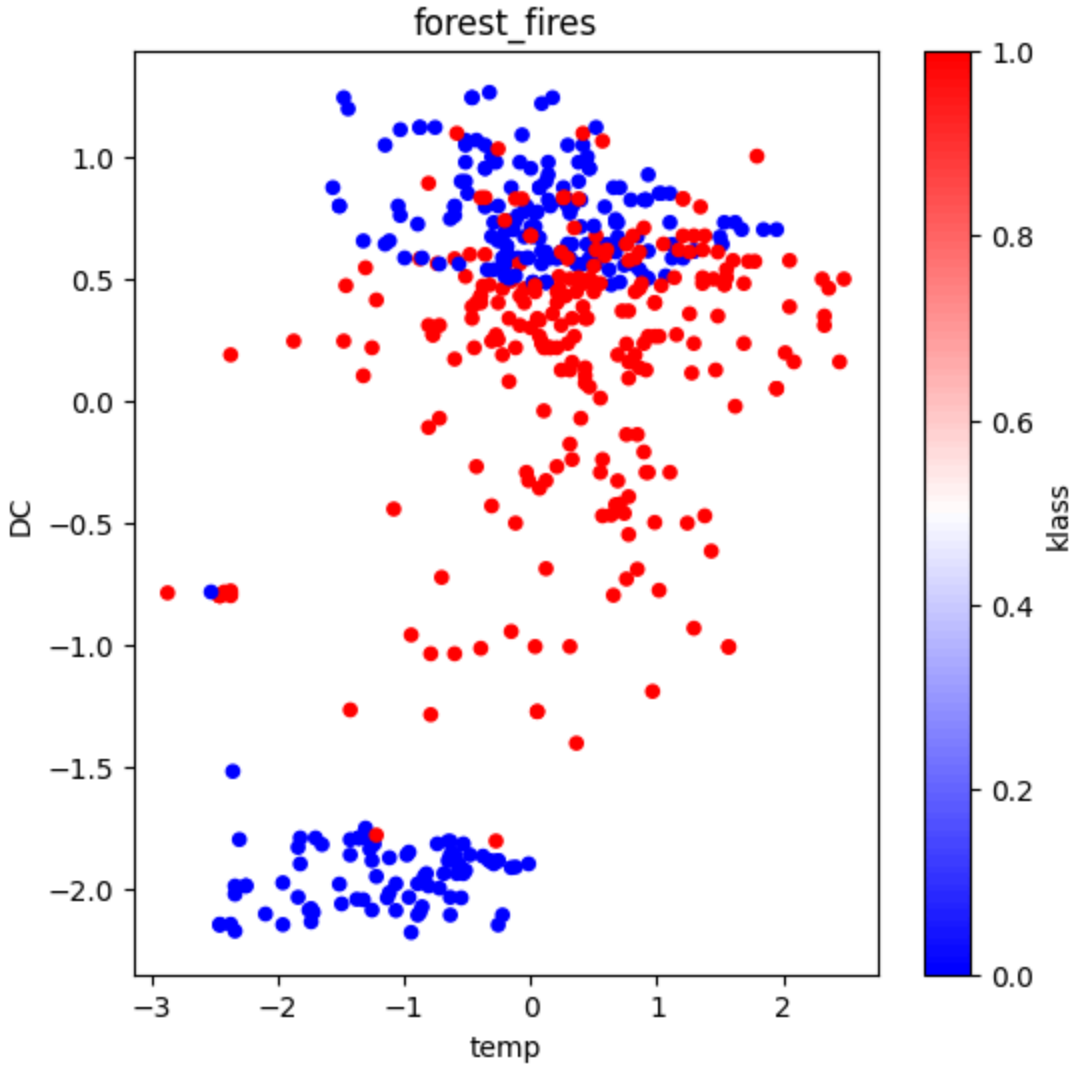
In [16]:
```
data.plot.scatter('temp', 'wind', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[16]:
```
Text(0.5, 1.0, 'forest_fires')
```



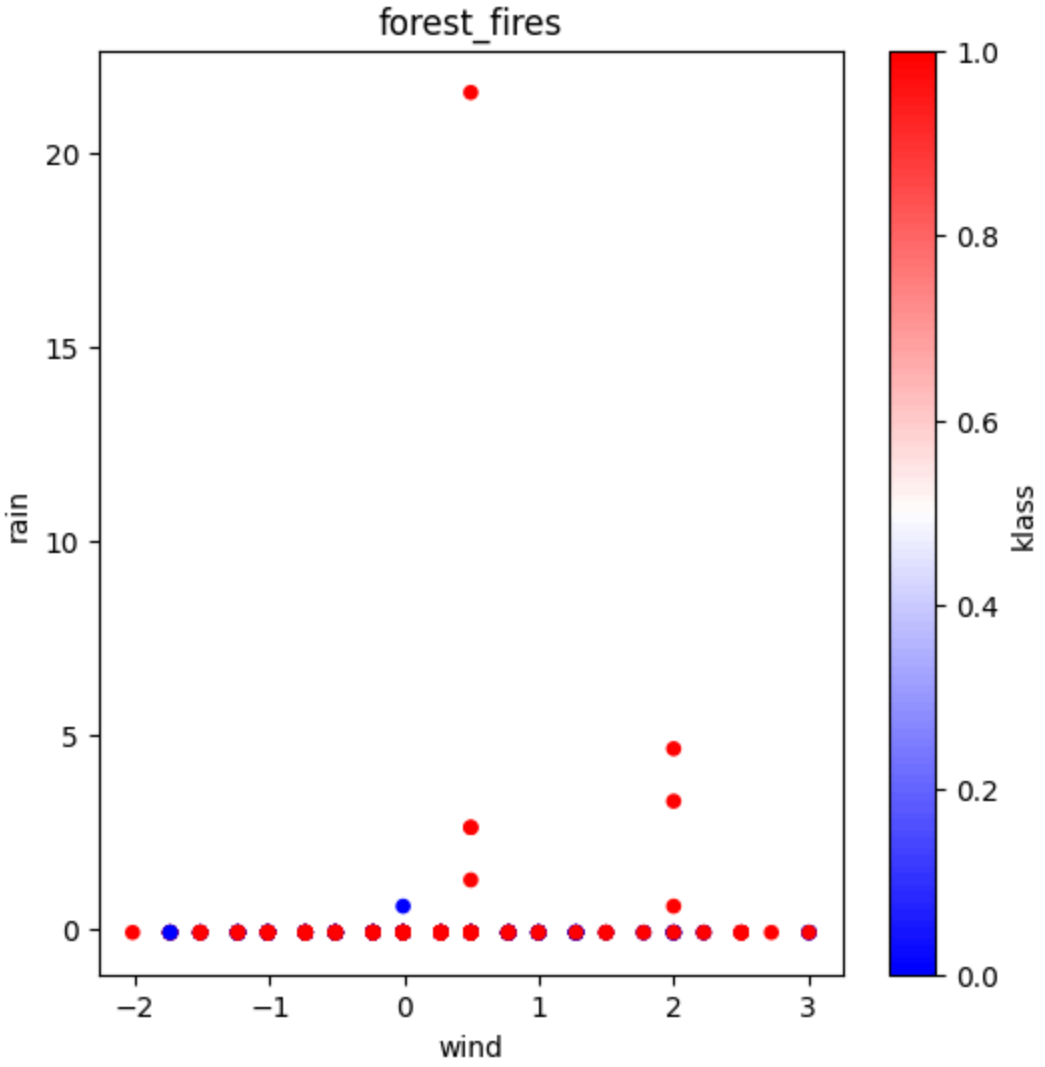
In [17]:
```
data.plot.scatter('temp', 'DC', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[17]:
```
Text(0.5, 1.0, 'forest_fires')
```



In [18]:
```
data.plot.scatter('wind', 'rain', c='klass', cmap='bwr', figsize=(6, 6))
plt.title('forest_fires')
```

Out[18]:
```
Text(0.5, 1.0, 'forest_fires')
```



7. модели классификации

In [19]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=0.3)
test
```

Out[19]:

	X	Y	FFMC	DMC	DC	ISI	temp
21	0.142950	-0.243765	0.209293	-0.505451	0.710939	0.039113	0.036309
48	-0.289244	-0.243765	-0.624024	-1.357957	-1.948029	-1.079436	-1.220876
209	1.439531	-1.056839	0.553489	-0.467978	0.553320	1.947227	0.742399
163	1.439531	1.382383	0.336102	0.159691	0.509783	-0.092481	-0.187573
91	1.439531	1.382383	0.191177	-1.172154	-1.883127	-0.267939	-0.256460
...	...	...	...	...	...	...	...
328	0.575144	0.569309	0.281755	-0.133846	0.820587	-0.136345	0.139639
321	-1.585825	-0.243765	0.390449	0.126903	0.949585	-0.333736	-0.359790
25	1.007337	-0.243765	0.136830	0.492262	0.215507	0.346166	-0.445899
89	0.575144	0.569309	0.191177	-1.172154	-1.883127	-0.267939	-0.256460
488	-0.289244	-0.243765	0.807107	0.475087	0.233244	1.903362	0.087974

156 rows × 11 columns

In [20]:
```
#train_X = train[['X', 'Y', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'wind']]
train_X = train[['DC','ISI']]
train_y = train['klass']
#test_X = test[['X', 'Y', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'wind']]
test_X = test[['DC','ISI']]
test_y = test['klass']
```

In [21]:
```
from sklearn.naive_bayes import GaussianNB
model_naive_bayes = GaussianNB()
model_naive_bayes.fit(train_X, train_y)

from sklearn.metrics import accuracy_score
print('Train accuracy: ', accuracy_score(train_y, model_naive_bayes.predict(
print('Test  accuracy: ', accuracy_score(test_y, model_naive_bayes.predict(t
```

```
Train accuracy:  0.6648199445983379
Test  accuracy:  0.6346153846153846
```

In [22]:
```
# logistic regression classifier
from sklearn.linear_model import LogisticRegression
model_logistic_regression = LogisticRegression()
model_logistic_regression.fit(train_X, train_y)

print('Train accuracy: ', accuracy_score(train_y, model_logistic_regression.
print('Test  accuracy: ', accuracy_score(test_y, model_logistic_regression.p
```

```
Train accuracy:  0.6094182825484764
Test  accuracy:  0.5641025641025641
```

In [23]:
```
# logistic regression classifier with polynomial degree=2

from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, interaction_only=False, include_bias=Fal
```

```
train_X_poly = poly.fit_transform(train_X)
test_X_poly = poly.transform(test_X)

model_logistic_regression_poly = LogisticRegression()
model_logistic_regression_poly.fit(train_X_poly, train_y)

print('Train accuracy: ', accuracy_score(train_y, model_logistic_regression_
print('Test  accuracy: ', accuracy_score(test_y, model_logistic_regression_p
```

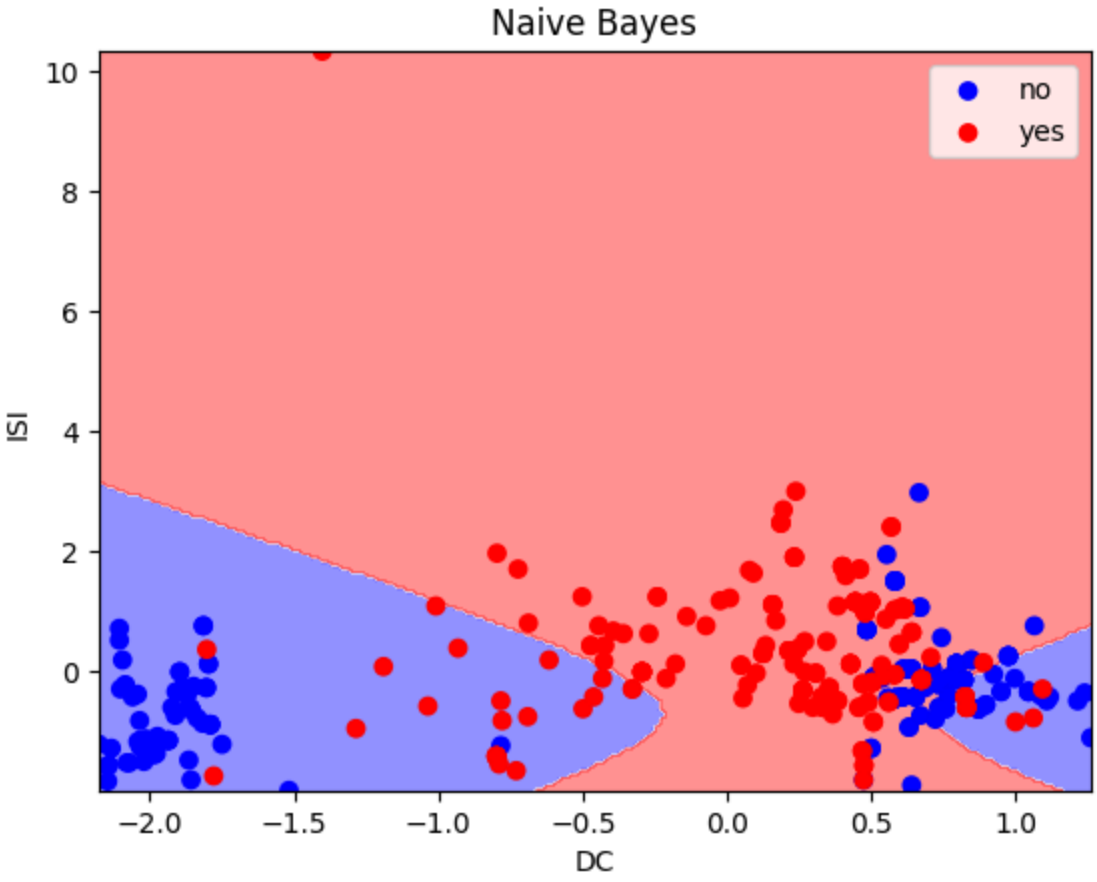
```
Train accuracy:  0.8393351800554016
Test  accuracy:  0.8269230769230769
```

8. границы принятия решений

In [24]:
```
from sklearn.inspection import DecisionBoundaryDisplay

xx0, xx1 = np.meshgrid(np.linspace(min(train_X.DC), max(train_X.DC), 200), r
grid = np.vstack([xx0.ravel(), xx1.ravel()]).T
y_pred = model_naive_bayes.predict(grid).reshape(xx0.shape)

disp = DecisionBoundaryDisplay(xx0=xx0, xx1=xx1, response=y_pred, xlabel='DC
disp.plot(alpha=0.5, cmap='bwr')

disp.ax_.scatter(train_X[train_y==0].DC, train_X[train_y==0].ISI, c=[0 for _
disp.ax_.scatter(train_X[train_y==1].DC, train_X[train_y==1].ISI, c=[1 for _


disp.ax_.legend()
disp.ax_.set_title('Naive Bayes')
```

```
/home/danya/.local/share/virtualenvs/rudn-year4-data-mining-Ym7BTVQk/lib/pyt
hon3.11/site-packages/sklearn/base.py:493: UserWarning: X does not have vali
d feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

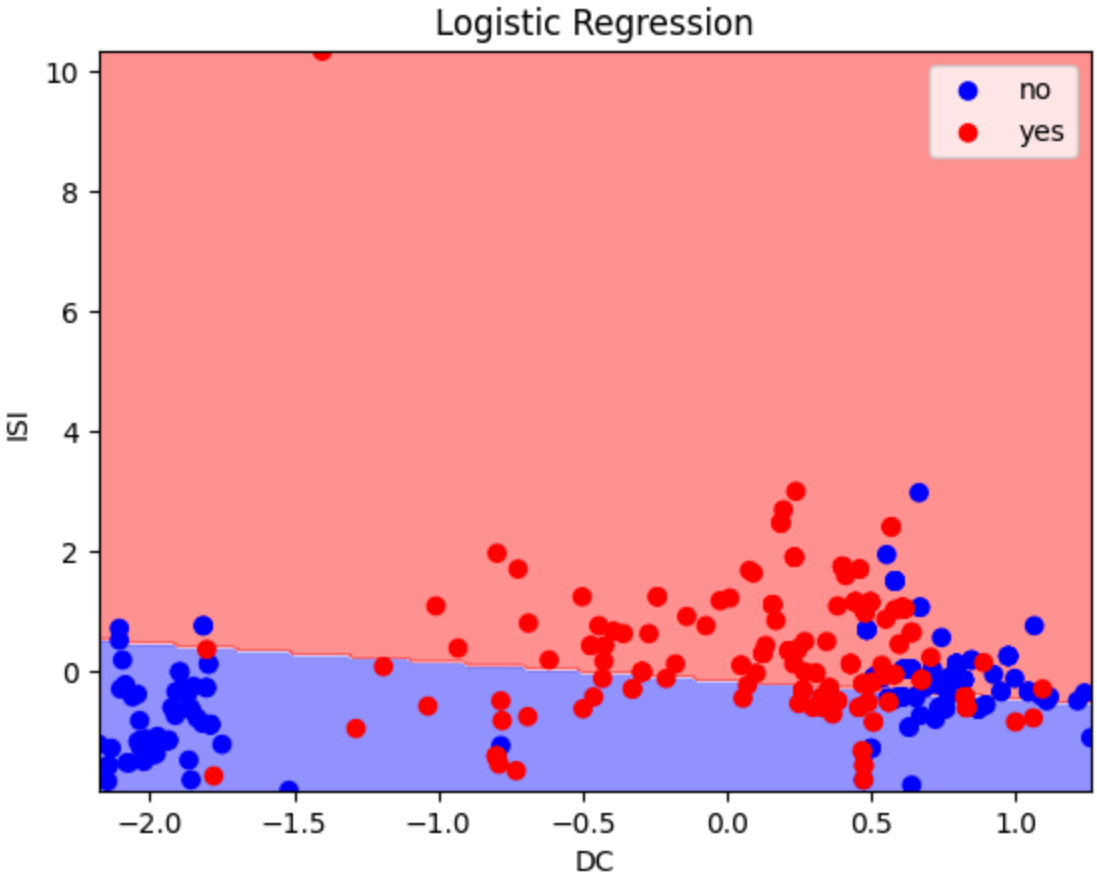
Out[24]:
```
Text(0.5, 1.0, 'Naive Bayes')
```



In [25]:
```
from sklearn.inspection import DecisionBoundaryDisplay

xx0, xx1 = np.meshgrid(np.linspace(min(train_X.DC), max(train_X.DC), 200), r
grid = np.vstack([xx0.ravel(), xx1.ravel()]).T
y_pred = model_logistic_regression.predict(grid).reshape(xx0.shape)

disp = DecisionBoundaryDisplay(xx0=xx0, xx1=xx1, response=y_pred, xlabel='DC
disp.plot(alpha=0.5, cmap='bwr')

disp.ax_.scatter(train_X[train_y==0].DC, train_X[train_y==0].ISI, c=[0 for _
disp.ax_.scatter(train_X[train_y==1].DC, train_X[train_y==1].ISI, c=[1 for _


disp.ax_.legend()
disp.ax_.set_title('Logistic Regression')
```

```
/home/danya/.local/share/virtualenvs/rudn-year4-data-mining-Ym7BTVQk/lib/pyt
hon3.11/site-packages/sklearn/base.py:493: UserWarning: X does not have vali
d feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

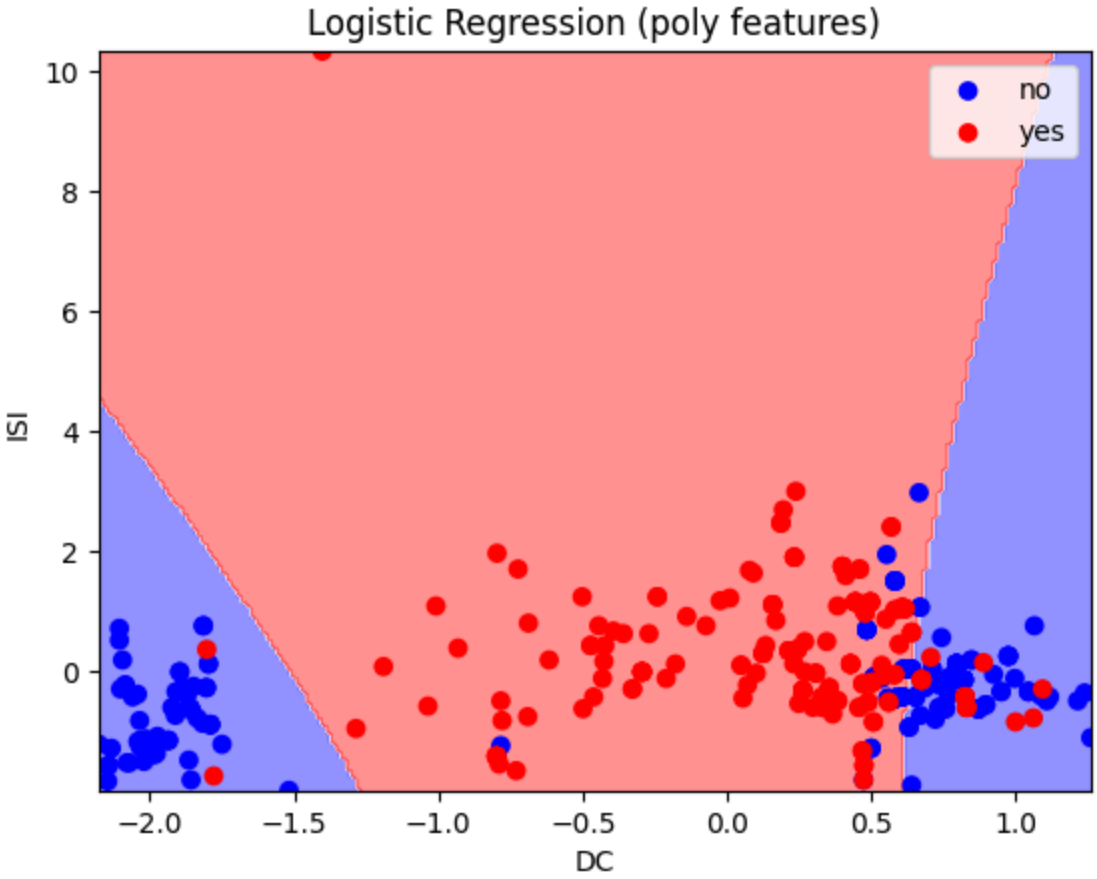
Out[25]:
```
Text(0.5, 1.0, 'Logistic Regression')
```



In [26]:
```
from sklearn.inspection import DecisionBoundaryDisplay

xx0, xx1 = np.meshgrid(np.linspace(min(train_X.DC), max(train_X.DC), 200), r
grid = np.vstack([xx0.ravel(), xx1.ravel()]).T
y_pred = model_logistic_regression_poly.predict(poly.transform(grid)).reshap

disp = DecisionBoundaryDisplay(xx0=xx0, xx1=xx1, response=y_pred, xlabel='DC
disp.plot(alpha=0.5, cmap='bwr')

disp.ax_.scatter(train_X[train_y==0].DC, train_X[train_y==0].ISI, c=[0 for _
disp.ax_.scatter(train_X[train_y==1].DC, train_X[train_y==1].ISI, c=[1 for _


disp.ax_.legend()
disp.ax_.set_title('Logistic Regression (poly features)')
```

```
/home/danya/.local/share/virtualenvs/rudn-year4-data-mining-Ym7BTVQk/lib/pyt
hon3.11/site-packages/sklearn/base.py:493: UserWarning: X does not have vali
d feature names, but PolynomialFeatures was fitted with feature names
  warnings.warn(
```

Out[26]:
```
Text(0.5, 1.0, 'Logistic Regression (poly features)')
```

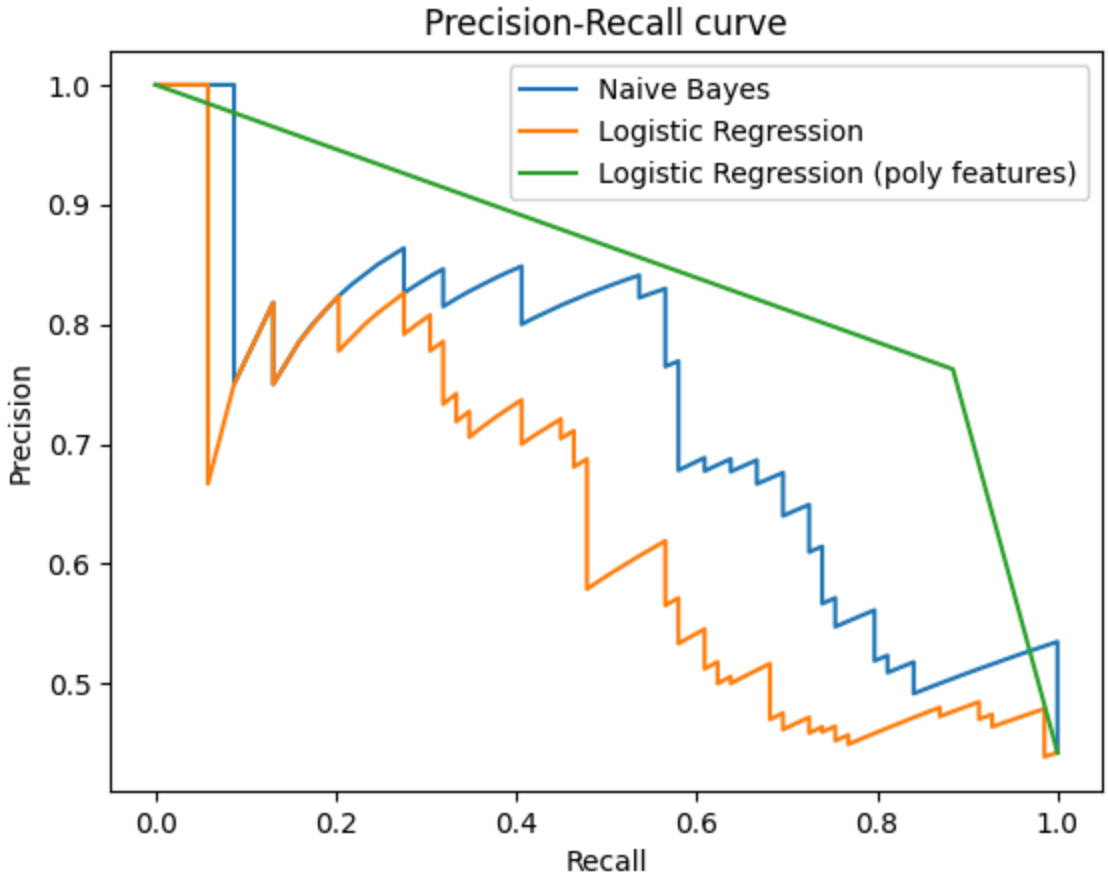


9. PR-кривые

In [27]:
```
from sklearn.metrics import precision_recall_curve
nb_prec, nb_recall, nb_thresh = precision_recall_curve(test_y, model_naive_b
lr_prec, lr_recall, lr_thresh = precision_recall_curve(test_y, model_logisti
lr_poly_prec, lr_poly_recall, lr_poly_thresh = precision_recall_curve(test_y
```

In [28]:
```
ax = plt.gca()
ax.plot(nb_recall, nb_prec, label='Naive Bayes')
ax.plot(lr_recall, lr_prec, label='Logistic Regression')
ax.plot(lr_poly_recall, lr_poly_prec, label='Logistic Regression (poly featu
ax.set_xlabel('Recall')
ax.set_ylabel('Precision')
ax.legend()
ax.set_title('Precision-Recall curve')
```

Out[28]:
```
Text(0.5, 1.0, 'Precision-Recall curve')
```



10. лучший классификатор

In [29]:
```
# calculate area under the curve for each model
from sklearn.metrics import auc
scores = [
    (auc(nb_recall, nb_prec), "Naive Bayes"),
    (auc(lr_recall, lr_prec), "Logistic Regression"),
    (auc(lr_poly_recall, lr_poly_prec), "Logistic Regression (poly features)")
]
scores.sort(reverse=True)

print("Scores:")
for i, score in enumerate(scores):
    print(f"{i+1}. {score[1]}: {score[0]}")
```

```
Scores:
1. Logistic Regression (poly features): 0.848920011148272
2. Naive Bayes: 0.733680679645162
3. Logistic Regression: 0.6384301403198843
```

In [ ]: