```
!pip install boruta
```

```
Requirement already satisfied: boruta in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.11/dis
Requirement already satisfied: scikit-learn>=0.17.1 in /usr/local/lib/python3
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3
```

```
from boruta import BorutaPy
import random
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import roc_auc_score, accuracy_score, confusion_matrix, prec
from sklearn.utils import resample
from sklearn.preprocessing import StandardScaler
from copy import deepcopy
from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold
```

```
df = pd.read_excel('/content/Параметры_для_многомерной_регрессии_Добычные_парамет
```

```
df
```

№	Скважины	Qж, м3/сут	Qн, т/сут	Qв, м3/сут	Обв, %	ГФ, м3/т	Кпрод	Кратность	кр
1	202_1_h	84.69	69.5960	0.25407	0.3	12.156	27.000000	31.631	
2	202_1_zbs	75.00	43.1250	2.25000	3.0	229.350	2.000000	32.640	
3	237_1_h	229.86	179.2793	11.49300	5.0	194.000	35.978370	29.693	
4	272_1_h	316.10	233.2430	3.47710	1.1	194.000	191.877689	35.421	
5	782_1_h	13.00	79.4892	0.78000	6.0	197.846	24.000000	25.391	
...	...	...	...	...	...	...	...	...	
271	657_53_h2	95.50	41.5549	44.88500	47.0	499.000	25.000000	62.755	
272	362_54_h	121.00	41.6239	70.30100	58.1	372.670	13.000000	59.706	
273	413_54_h	94.30	68.9410	10.37300	11.0	565.627	15.000000	93.852	
274	378_57_h	29.20	23.8533	0.14600	0.5	13312.000	2.200000	73.766	

```
print(df.columns.tolist())
```

```
['Скважины', 'Qж, м3/сут', 'Qн, т/сут', 'Qв, м3/сут', 'Обв, %', 'ГФ, м3/т', 'I
```

```
df = df.drop(columns=[
    "№", "Кпрод", "Qн, т/сут", "Qв, м3/сут", "Обв, %", "ГФ, м3/т",
    "Кратность ", "Выборка по кратности > 40", "Выборка по кратности > 70",
    "Снятие значений в рифее",

    # Геометрия коллектора
    "Общая проходка", "Длина коллектора, м", "Мощность коллектора, м",
    "Длина коллектора с поро>5%, м", "Длина глинистых участков, м", "Доля глинист

    # Все Средняя и Линейная по MD
    'Средняя ',"Средняя .1", "Линейная по MD", "Средняя .2", "Линейная по MD.1",
    "Средняя .3", "Линейная по MD.2", "Средняя .4", "Линейная по MD.3",
    "Средняя .5", "Линейная по MD.4", "Средняя .6", "Линейная по MD.5"
], errors="ignore")
```

```
df.reset_index(drop=True)
```

	Скважины	Qж, м3/сут	Качество коллектора	Акустический импеданс (PSTM)	RMS амплитуды	Расстояние от разлома R4 (интерпретатор)	Р с (ве
0	202_1_h	84.69	0.555	18329.34	1157.065	181.482	
1	202_1_zbs	75.00	0.571	19037.81	333.230	546.018	
2	237_1_h	229.86	0.639	17914.69	469.790	1298.976	
3	272_1_h	316.10	0.646	19384.03	1423.487	756.552	
4	782_1_h	13.00	0.625	17551.41	523.814	195.287	
...	...	...	...	...	...	...	
270	657_53_h2	95.50	0.617	18469.91	1110.697	891.204	
271	362_54_h	121.00	0.557	18127.75	1122.852	193.434	
272	413_54_h	94.30	0.630	17116.49	543.458	213.632	
273	378 57 h	29.20	0.547	19868.97	1128.926	1255.104	

```
df.describe()
```

	Qж, м3/сут	Качество коллектора	Акустический импеданс (PSTM)	RMS амплитуды	Расстояние от разлома R4 (интерпретатор)	Расст от ра (вероят разл
count	275.000000	275.000000	275.000000	275.000000	275.000000	275.
mean	118.440818	0.560625	18025.461964	1250.799927	784.099164	581.
std	104.659026	0.068420	1870.397750	1043.291816	551.062290	561.
min	1.000000	0.347000	11488.030000	81.338000	34.174000	13.
25%	34.245000	0.519000	17009.280000	543.910500	326.308000	174.
50%	86.400000	0.570000	18329.340000	927.013000	679.602000	376.

```
df = df.dropna()
```

```
df.isnull().sum()
```

	0
Скважины	0
Qж, м3/сут	0
Качество коллектора	0
Акустический импеданс (PSTM)	0
RMS амплитуды	0
Расстояние от разлома R4 (интерпретатор)	0
Расстояние от разлома (вероятность разломов)	0
Расстояние от разлома (интерпретатор + вероятности)	0
Расстояние от вреза	0
Расстояние от выклинивания толши	0
Глубина проводки (от эрозионной поверхности)	0
Толщина Б-Ro	0
Толщина R0-R4	0

dtype: int64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 275 entries, 1 to 275
Data columns (total 13 columns):
 #   Column                                          Non-Null Count  Dty
---  ------                                          --------------  --
```

```
 0   Скважины                                          275 non-null    obj
 1   Qж, м3/сут                                        275 non-null    flo
 2   Качество коллектора                               275 non-null    flo
 3   Акустический импеданс (PSTM)                      275 non-null    flo
 4   RMS амплитуды                                     275 non-null    flo
 5   Расстояние от разлома R4 (интерпретатор)          275 non-null    flo
 6   Расстояние от разлома (вероятность разломов)      275 non-null    flo
 7   Расстояние от разлома (интерпретатор + вероятности )  275 non-null  flo
 8   Расстояние от вреза                               275 non-null    flo
 9   Расстояние от выклинивания толши                  275 non-null    flo
 10  Глубина проводки (от эрозионной поверхности)      275 non-null    flo
 11  Толщина Б-Ro                                      275 non-null    flo
 12  Толщина R0-R4                                     275 non-null    flo
dtypes: float64(12), object(1)
memory usage: 30.1+ KB
```

```
print(df['Скважины'].nunique())
```

```
275
```

Т.к. все скважины уже усреднены и уникальны, то не будем делить их на трейновую и тестовую выборку

```
df = df.copy()
df['Qж_class'] = (df['Qж, м3/сут'] > 118).astype(int)
```

```
train_df, test_df = train_test_split(df, test_size=0.4, random_state=42)
```

Подбор признаков с помошью Boruta

```
X_boruta_q = df.drop(columns=['Qж_class', 'Qж, м3/сут', 'Скважины'])
y_boruta_q = df['Qж_class']
```

```
X_train_boruta_q, X_test_boruta_q, y_train_boruta_q, y_test_boruta_q = train_test
```

```
scaler_boruta_q = StandardScaler()
X_train_boruta_scaled = scaler_boruta_q.fit_transform(X_train_boruta_q)
X_test_boruta_scaled = scaler_boruta_q.transform(X_test_boruta_q)
```

```
rf_boruta_q = RandomForestClassifier(n_estimators=100, random_state=42, class_wei
boruta_selector_q = BorutaPy(
    rf_boruta_q,
    n_estimators='auto',
    alpha=0.05,
    max_iter=100,
    random_state=42
)
```

```
boruta_selector_q.fit(X_train_boruta_scaled, y_train_boruta_q.values)

# Получаем отобранные признаки
selected_features_q = X_boruta_q.columns[boruta_selector_q.support_].tolist()
print("Отобранные признаки Boruta:")
for feat in selected_features_q:
    print("-", feat)
```

```
Отобранные признаки Boruta:
- Толщина Б-Ro
```

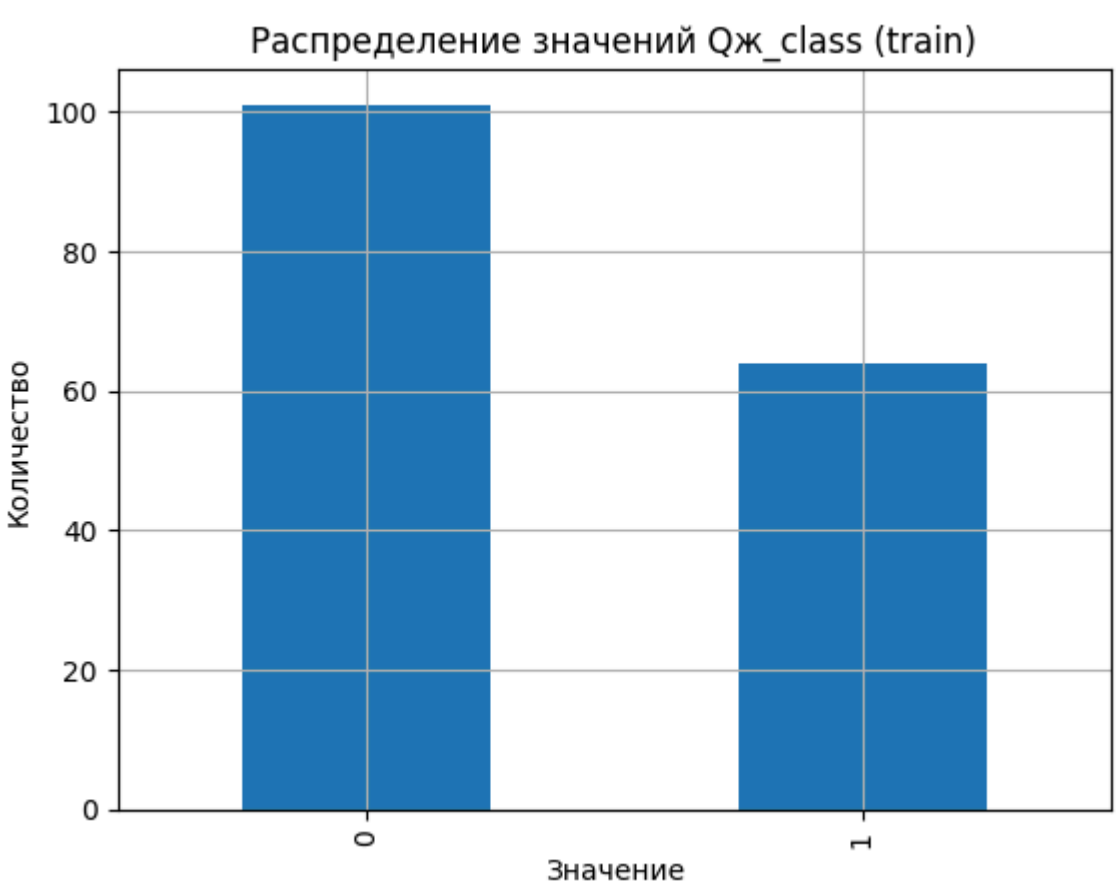
Борута не под этот датасет

Подбор признаков с помощью RFECV

```
# Распределение по классам в train
print(train_df['Qж_class'].value_counts())
```

```
Qж_class
0    101
1     64
Name: count, dtype: int64
```

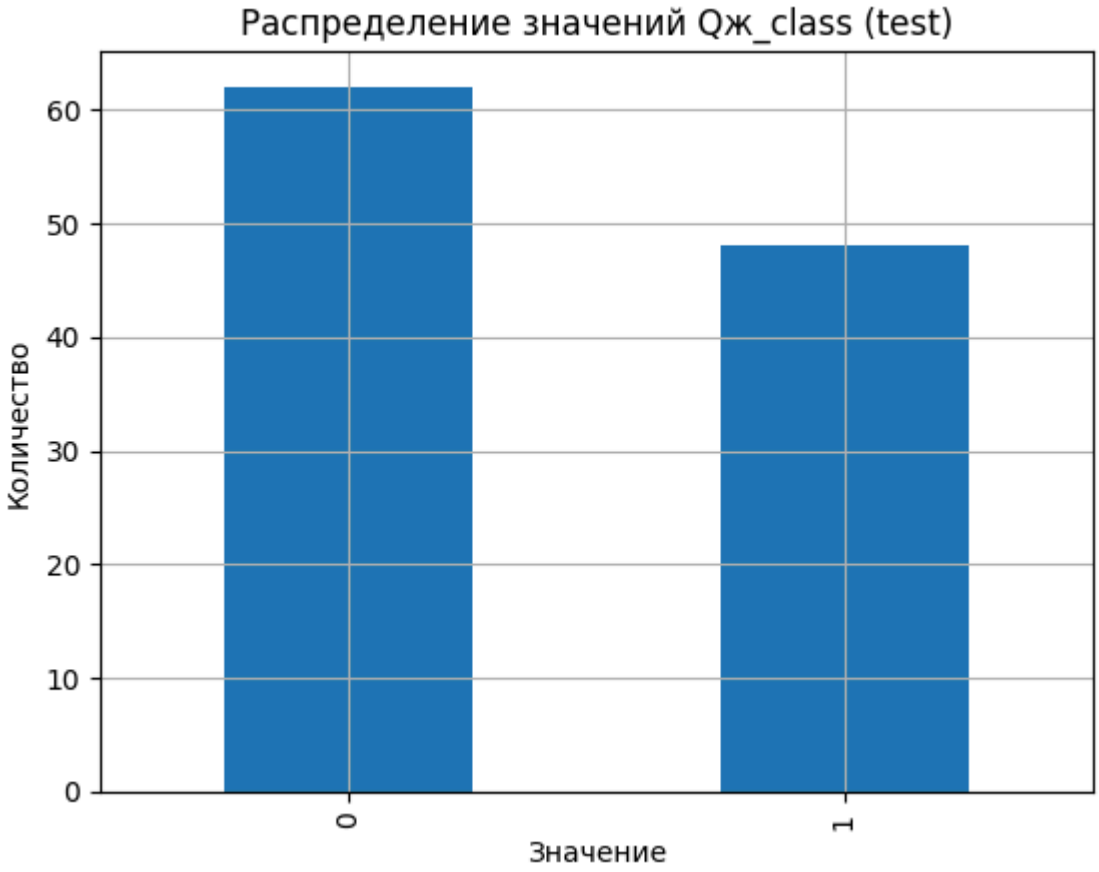
```
train_df['Qж_class'].value_counts().sort_index().plot(kind='bar')
plt.title("Распределение значений Qж_class (train)")
plt.xlabel("Значение")
plt.ylabel("Количество")
plt.xticks([0, 1], ['0', '1'])
plt.grid(True)
plt.show()
```




```
# Распределение по классам в test
print(test_df['Qж_class'].value_counts())
```

```
Qж_class
0    62
1    48
Name: count, dtype: int64
```

```
test_df['Qж_class'].value_counts().sort_index().plot(kind='bar')
plt.title("Распределение значений Qж_class (test)")
plt.xlabel("Значение")
plt.ylabel("Количество")
plt.xticks([0, 1], ['0', '1'])
plt.grid(True)
plt.show()
```




```
# Временные X и y только для RFECV
X_temp = df.drop(columns=['Скважины', 'Qж_class', 'Qж, м3/сут'])  # все признаки,
y_temp = df['Qж_class']                                  # целевая переменная

# Базовая модель
estimator_rfecv_q = RandomForestClassifier(n_estimators=30, max_depth=2, random_s

rfecv_q = RFECV(
    estimator=estimator_rfecv_q,
    step=1,
    cv=StratifiedKFold(3),
    scoring='f1',
    min_features_to_select=1,
)

# Обучение селектора
rfecv_q.fit(X_temp, y_temp)

# Получение отобранных признаков
selected_features_rfecv_q = X_temp.columns[rfecv_q.support_].tolist()
print("\nВыбранные признаки (RFECV):")
for feat in selected_features_rfecv_q:
    print("-", feat)
```

```
Выбранные признаки (RFECV):
- Качество коллектора
- Акустический импеданс (PSTM)
- RMS амплитуды
- Расстояние от разлома (вероятность разломов)
```

```
    - Расстояние от разлома (интерпретатор + вероятности )
    - Расстояние от вреза
    - Расстояние от выклинивания толши
    - Глубина проводки (от эрозионной поверхности)
    - Толщина Б-Ro
    - Толщина R0-R4
```

```
X_train = train_df[[
    'Качество коллектора',
    'Акустический импеданс (PSTM)',
    'RMS амплитуды',
    'Расстояние от разлома (вероятность разломов)',
    'Расстояние от разлома (интерпретатор + вероятности )',
    'Расстояние от вреза',
    'Расстояние от выклинивания толши',
    'Глубина проводки (от эрозионной поверхности)',
    'Толщина Б-Ro',
    'Толщина R0-R4'
]]
y_train = train_df['Qж_class']

X_test = test_df[[
    'Качество коллектора',
    'Акустический импеданс (PSTM)',
    'RMS амплитуды',
    'Расстояние от разлома (вероятность разломов)',
    'Расстояние от разлома (интерпретатор + вероятности )',
    'Расстояние от вреза',
    'Расстояние от выклинивания толши',
    'Глубина проводки (от эрозионной поверхности)',
    'Толщина Б-Ro',
    'Толщина R0-R4'
]]
y_test = test_df['Qж_class']
```

```
param_grid = {
    'n_estimators': [50, 100],         # Больше деревьев = стабильнее
    'max_depth': [2, 3],               # Жесткое ограничение на глубину (контроли
    'min_samples_split': [6, 10],      # Чем больше, тем деревья менее подвержены
    'min_samples_leaf': [4, 6]         # Минимум объектов в листе — тоже ключевой
}

# Инициализация и подбор по F1
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(rf, param_grid, cv=3, scoring='f1', n_jobs=-1)
grid_search.fit(X_train, y_train)

print("Лучшие параметры:", grid_search.best_params_)
```

```
Лучшие параметры: {'max_depth': 3, 'min_samples_leaf': 6, 'min_samples_split'
```

```
model = RandomForestClassifier(**grid_search.best_params_, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=3, min_samples_leaf=6, min_samples_split=6,
                       n_estimators=50, random_state=42)
```

```
print("Feature Importances:", model.feature_importances_)
```

```
Feature Importances: [0.11960531 0.0489722  0.06022939 0.11766236 0.11320264 0
 0.12251669 0.07454329 0.11942684 0.12693079]
```

```
y_pred_test = model.predict(X_test)
y_pred_train = model.predict(X_train)
```

```
print("Метрики для тестовой выборки:")
print("Accuracy:", accuracy_score(y_test, y_pred_test))
print("ROC AUC:", roc_auc_score(y_test, y_pred_test))
print("Precision:", precision_score(y_test, y_pred_test))
print("Recall:", recall_score(y_test, y_pred_test))
print("F1:", f1_score(y_test, y_pred_test))
```

```
Метрики для тестовой выборки:
Accuracy: 0.5636363636363636
ROC AUC: 0.5305779569892473
Precision: 0.5
Recall: 0.2708333333333333
F1: 0.35135135135135137
```

```
print("Метрики для обучающей выборки")
print("Accuracy:", accuracy_score (y_train, y_pred_train ))
print("ROC AUC:", roc_auc_score (y_train, y_pred_train ))
print("Precision:", precision_score (y_train, y_pred_train ))
print("Recall:", recall_score (y_train, y_pred_train ))
print("f1:", f1_score (y_train, y_pred_train ))
```

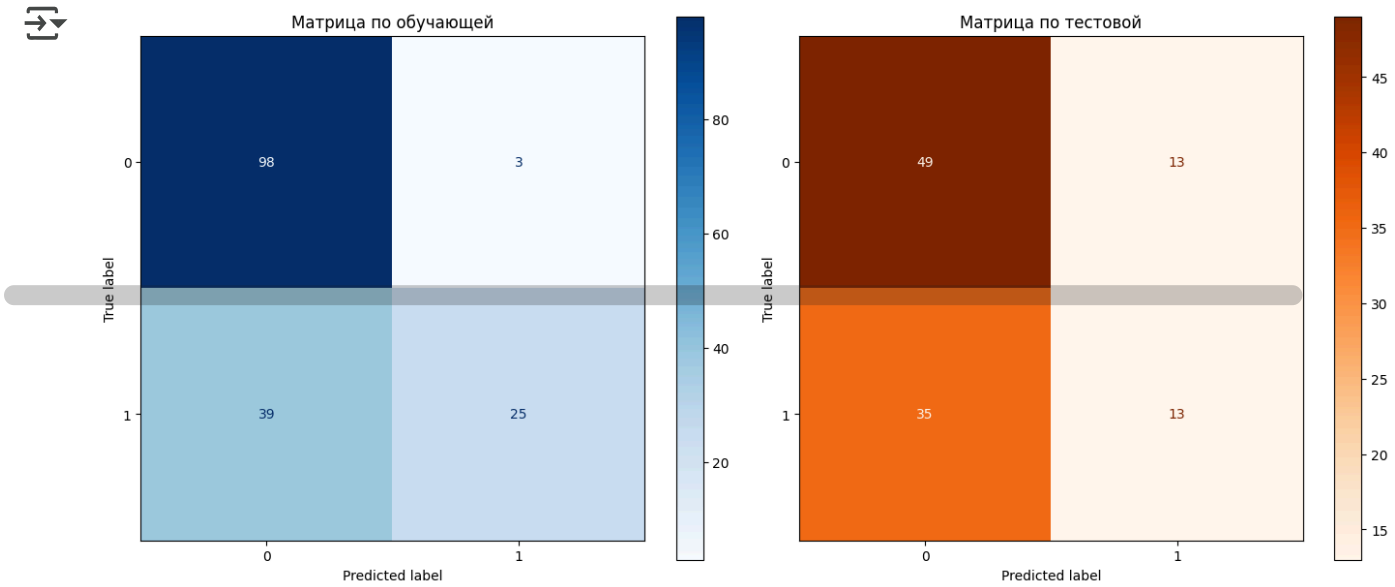
```
Метрики для обучающей выборки
Accuracy: 0.7454545454545455
ROC AUC: 0.6804610148514851
Precision: 0.8928571428571429
Recall: 0.390625
f1: 0.5434782608695652
```

```
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

ConfusionMatrixDisplay.from_predictions(y_train, y_pred_train, ax=ax[0], cmap='Bl
ax[0].set_title('Матрица по обучающей')

ConfusionMatrixDisplay.from_predictions(y_test, y_pred_test, ax=ax[1], cmap='Oran
ax[1].set_title('Матрица по тестовой')
```

```
plt.tight_layout()
plt.show()
```

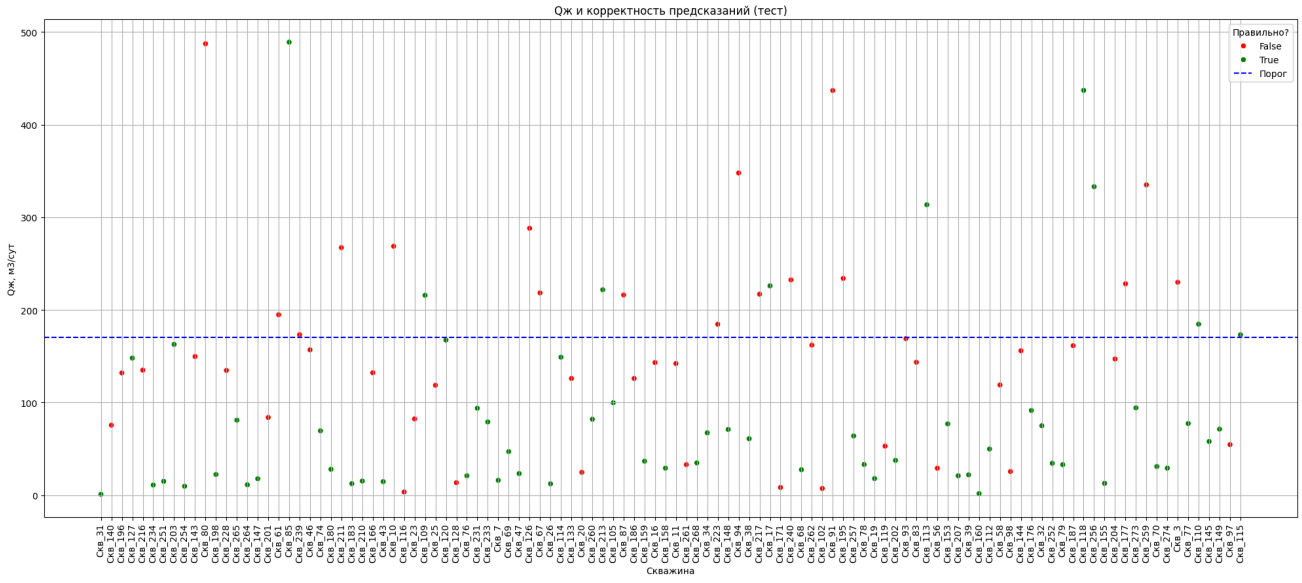



```
# Добавляем столбец "Скважина" в test_df
test_df = test_df.copy()
test_df['Скважина'] = ['Скв_' + str(i) for i in test_df.index]

# Создаем DataFrame с реальными и предсказанными классами
df_test_eval = test_df.copy()
df_test_eval['y_pred'] = y_pred_test
df_test_eval['Qж_class'] = y_test

# Строим график
plt.figure(figsize=(20, 9))
correct = df_test_eval['y_pred'] == df_test_eval['Qж_class']
sns.scatterplot(x='Скважина',
                y='Qж, м3/сут',
                hue=correct,
                data=df_test_eval,
                palette={True: 'green', False: 'red'})
plt.axhline(170.2, linestyle='--', color='blue', label='Порог')
```

```
plt.title('Qж и корректность предсказаний (тест)')
plt.xlabel('Скважина')
plt.ylabel('Qж, м3/сут')
plt.xticks(rotation=90)
plt.legend(title='Правильно?')
plt.grid(True)
plt.tight_layout()
plt.show()
```

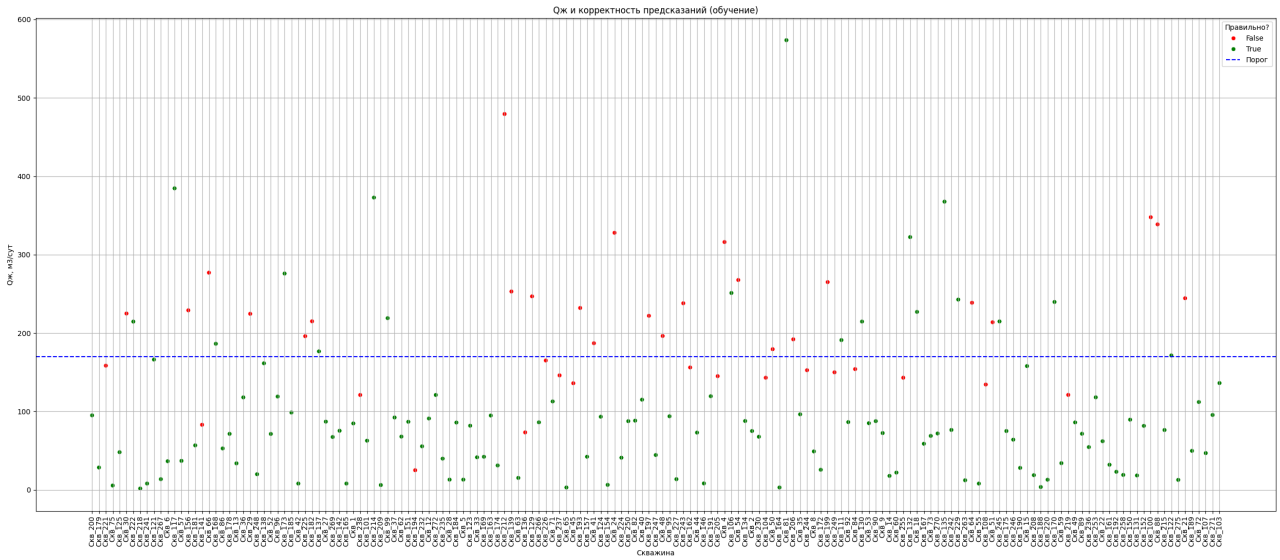



```
# Добавляем столбец "Скважина" в train_df
train_df = train_df.copy()
train_df['Скважина'] = ['Скв_' + str(i) for i in train_df.index]

# Создаем DataFrame с реальными и предсказанными классами
df_train_eval = train_df.copy()
```

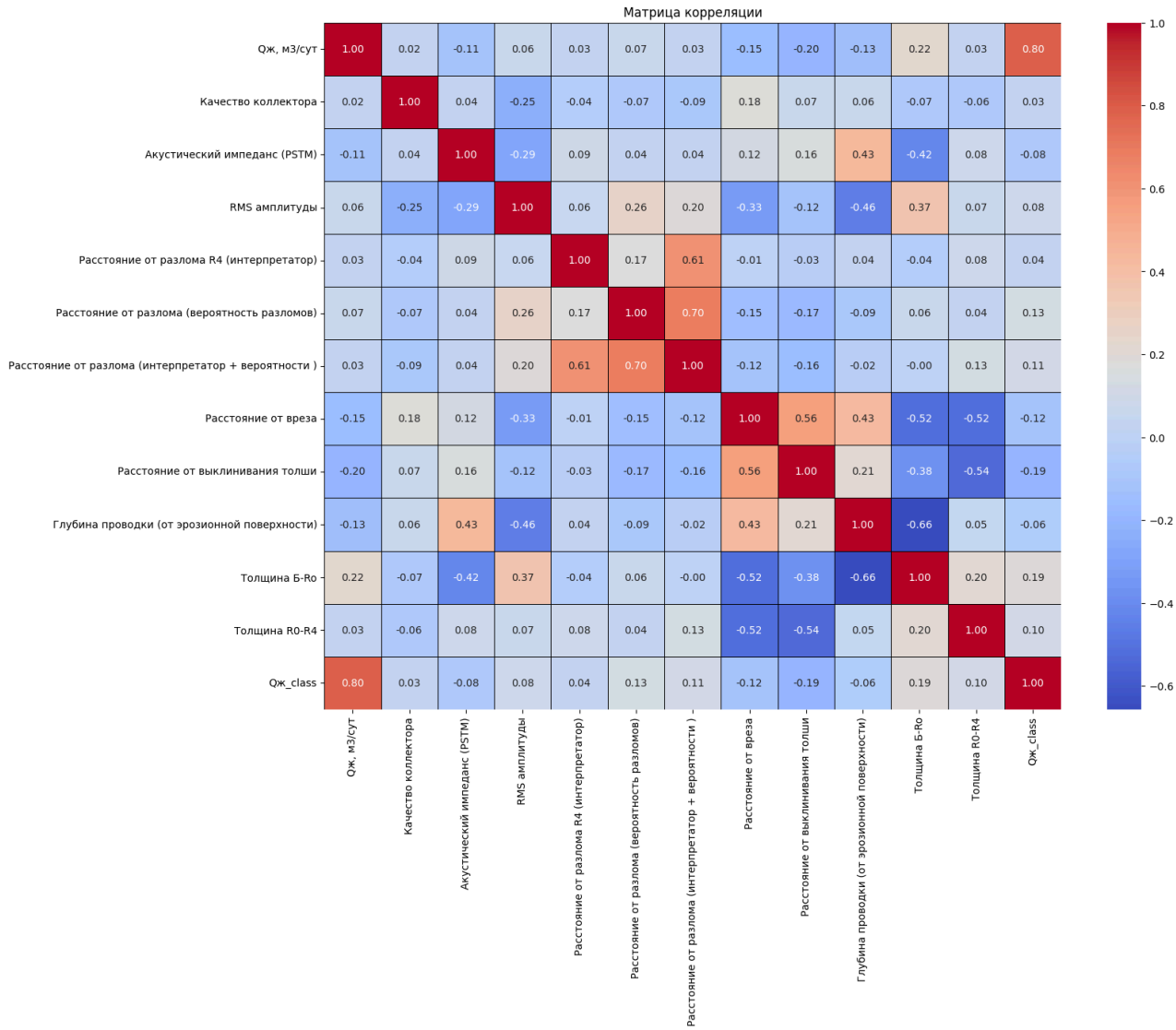
```
df_train_eval['y_pred'] = y_pred_train
df_train_eval['Qж_class'] = y_train

# Строим график
plt.figure(figsize=(25, 11))
correct = df_train_eval['y_pred'] == df_train_eval['Qж_class']
sns.scatterplot(x='Скважина',
                y='Qж, м3/сут',
                hue=correct,
                data=df_train_eval,
                palette={True: 'green', False: 'red'})
plt.axhline(170.2, linestyle='--', color='blue', label='Порог')
plt.title('Qж и корректность предсказаний (обучение)')
plt.xlabel('Скважина')
plt.ylabel('Qж, м3/сут')
plt.xticks(rotation=90)
plt.legend(title='Правильно?')
plt.grid(True)
plt.tight_layout()
plt.show()
```


Qж и корректность предсказаний (обучение)
Правильно?
False
True
Порог
Qж, м3/сут
Скважина

Ручной подбор признаков основываясь на матрице корреляции

```
plt.figure(figsize=(16, 12))
sns.heatmap(
    df.corr(numeric_only=True),
    annot=True,
    cmap='coolwarm',
    fmt='.2f',
    linecolor='black',
    linewidths=0.5
)
plt.title('Матрица корреляции')
plt.show()
```


Матрица корреляции
Qж, м3/сут
Качество коллектора
Акустический импеданс (PSTM)
RMS амплитуды
Расстояние от разлома R4 (интерпретатор)
Расстояние от разлома (вероятность разломов)
Расстояние от разлома (интерпретатор + вероятности)
Расстояние от вреза
Расстояние от выклинивания толши
Глубина проводки (от эрозионной поверхности)
Толщина Б-Ro
Толщина R0-R4
Qж_class
1.00 0.02 -0.11 0.06 0.03 0.07 0.03 -0.15 -0.20 -0.13 0.22 0.03 0.80
0.02 1.00 0.04 -0.25 -0.04 -0.07 -0.09 0.18 0.07 0.06 -0.07 -0.06 0.03
-0.11 0.04 1.00 -0.29 0.09 0.04 0.04 0.12 0.16 0.43 -0.42 0.08 -0.08
0.06 -0.25 -0.29 1.00 0.06 0.26 0.20 -0.33 -0.12 -0.46 0.37 0.07 0.08
0.03 -0.04 0.09 0.06 1.00 0.17 0.61 -0.01 -0.03 0.04 -0.04 0.08 0.04
0.07 -0.07 0.04 0.26 0.17 1.00 0.70 -0.15 -0.17 -0.09 0.06 0.04 0.13
0.03 -0.09 0.04 0.20 0.61 0.70 1.00 -0.12 -0.16 -0.02 -0.00 0.13 0.11
-0.15 0.18 0.12 -0.33 -0.01 -0.15 -0.12 1.00 0.56 0.43 -0.52 -0.52 -0.12
-0.20 0.07 0.16 -0.12 -0.03 -0.17 -0.16 0.56 1.00 0.21 -0.38 -0.54 -0.19
-0.13 0.06 0.43 -0.46 0.04 -0.09 -0.02 0.43 0.21 1.00 -0.66 0.05 -0.06
0.22 -0.07 -0.42 0.37 -0.04 0.06 -0.00 -0.52 -0.38 -0.66 1.00 0.20 0.19
0.03 -0.06 0.08 0.07 0.08 0.04 0.13 -0.52 -0.54 0.05 0.20 1.00 0.10
0.80 0.03 -0.08 0.08 0.04 0.13 0.11 -0.12 -0.19 -0.06 0.19 0.10 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
X_train_corr = train_df[['Расстояние от вреза', 'Расстояние от выклинивания толши
y_train_corr = train_df['Qж_class']

X_test_corr = test_df[['Расстояние от вреза', 'Расстояние от выклинивания толши',
y_test_corr = test_df['Qж_class']
```

```
param_grid_corr = {
    'n_estimators': [30],
    'max_depth': [2, 3, 5],
    'min_samples_split': [4, 6],
    'min_samples_leaf': [2, 3]
}

rf_corr = RandomForestClassifier(random_state=42)
grid_search_corr = GridSearchCV(rf_corr, param_grid_corr, cv=3, scoring='f1', n_j
grid_search_corr.fit(X_train_corr, y_train_corr)

print("Лучшие параметры (corr):", grid_search_corr.best_params_)
```

```
Лучшие параметры (corr): {'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_
```

```
model_corr = RandomForestClassifier(**grid_search_corr.best_params_, random_state
model_corr.fit(X_train_corr, y_train_corr)
```

```
RandomForestClassifier(max_depth=5, min_samples_leaf=2, min_samples_split=4,
                       n_estimators=30, random_state=42)
```

```
print("Feature Importances (corr):", model_corr.feature_importances_)
```

```
Feature Importances (corr): [0.29405632 0.36218708 0.3437566 ]
```

```
y_pred_test_corr = model_corr.predict(X_test_corr)
y_pred_train_corr = model_corr.predict(X_train_corr)
```

```
print("Метрики для тестовой выборки (corr):")
print("Accuracy:", accuracy_score(y_test_corr, y_pred_test_corr))
print("ROC AUC:", roc_auc_score(y_test_corr, y_pred_test_corr))
print("Precision:", precision_score(y_test_corr, y_pred_test_corr))
print("Recall:", recall_score(y_test_corr, y_pred_test_corr))
print("F1:", f1_score(y_test_corr, y_pred_test_corr))
```

```
Метрики для тестовой выборки (corr):
Accuracy: 0.6727272727272727
ROC AUC: 0.6485215053763441
Precision: 0.6875
Recall: 0.4583333333333333
F1: 0.55
```

```
print("Метрики для обучающей выборки (corr):")
print("Accuracy:", accuracy_score(y_train_corr, y_pred_train_corr))
print("ROC AUC:", roc_auc_score(y_train_corr, y_pred_train_corr))
print("Precision:", precision_score(y_train_corr, y_pred_train_corr))
print("Recall:", recall_score(y_train_corr, y_pred_train_corr))
print("F1:", f1_score(y_train_corr, y_pred_train_corr))
```

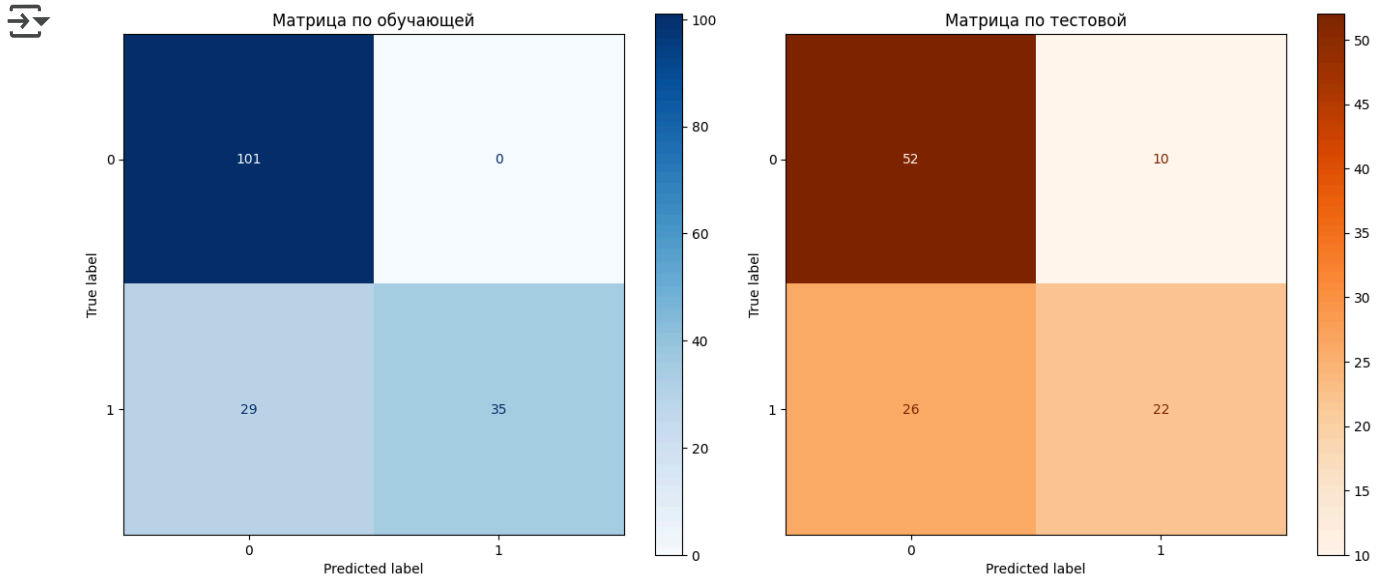
```
Метрики для обучающей выборки (corr):
Accuracy: 0.8242424242424242
ROC AUC: 0.7734375
Precision: 1.0
Recall: 0.546875
F1: 0.7070707070707071
```

```
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

ConfusionMatrixDisplay.from_predictions(y_train_corr, y_pred_train_corr, ax=ax[0]
ax[0].set_title('Матрица по обучающей')

ConfusionMatrixDisplay.from_predictions(y_test_corr, y_pred_test_corr, ax=ax[1],
ax[1].set_title('Матрица по тестовой')

plt.tight_layout()
plt.show()
```




```
test_df = test_df.copy()
test_df['Скважина'] = ['Скв_' + str(i) for i in test_df.index]

df_test_eval = test_df.copy()
df_test_eval['y_pred'] = y_pred_test_corr
df_test_eval['Qж_class'] = y_test_corr

plt.figure(figsize=(20, 9))
correct = df_test_eval['y_pred'] == df_test_eval['Qж_class']
sns.scatterplot(x='Скважина',
                y='Qж, м3/сут',
                hue=correct,
                data=df_test_eval,
                palette={True: 'green', False: 'red'})
plt.axhline(170.2, linestyle='--', color='blue', label='Порог')
plt.title('Qж и корректность предсказаний (тест)')
```