МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматики и Вычислительной техники** |
| Кафедра | **Информатики** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка комиссии: |  | | | Рейтинг: |  |
| Подписи членов комиссии: | | | | | |
|  | |  | Вишневская Е. А. | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | | | | | |
| (дата) | | | | | |
|  | |  |  | | |

**ДОМАШНЕЕ ЗАДАНИЕ №2**

|  |  |
| --- | --- |
| по дисциплине | Средства интеллектуального анализа данных и машинное |
| обучение | |

|  |  |
| --- | --- |
| на тему | Задача классификации |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| «К ЗАЩИТЕ» |  | ВЫПОЛНИЛ: |  |
|  |  | Студент группы | **АА-22-08** |
|  |  |  | (номер группы) |
| Доцент, к. ф. м. н.: Вишневская Е. А. |  | Сафуанов Артур Ришатович | |
| (должность, ученая степень; фамилия, и.о.) |  | (фамилия, имя, отчество) | |
|  |  |  | |
| (подпись) |  | (подпись) | |
|  |  |  | |
| (дата) |  | (дата) | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Москва, 20 | 25 |  |

Оглавление

[Введение 3](#_Toc167825323)

[Ход работы 4](#_Toc167825324)

[Список используемой литературы 38](#_Toc167825325)

**Введение**

В условиях стремительного роста объёмов данных и их разнообразия задача автоматизированного анализа информации становится одной из ключевых в современных информационных системах. Классификация, как одна из основных задач машинного обучения, позволяет автоматически распределять объекты по категориям на основе их признаков, что находит применение в медицине, финансах, маркетинге, компьютерном зрении и многих других областях.

Целью данной работы является изучение методов и инструментов интеллектуального анализа данных для решения задачи классификации, а также их практическое применение на реальных данных. В рамках исследования планируется рассмотреть ключевые алгоритмы классификации (например, логистическая регрессия, метод опорных векторов, деревья решений), провести их сравнительный анализ и оценить эффективность на основе метрик качества (точность, F1-мера, ROC-кривая).

Работа включает этапы предобработки данных, выбора признаков, обучения моделей, их валидации и интерпретации результатов. В качестве инструментов используются библиотеки Python, такие как scikit-learn, pandas и matplotlib, обеспечивающие гибкость и наглядность реализации.

Исследование демонстрирует, как современные методы машинного обучения позволяют преобразовывать сырые данные в ценную информацию для поддержки принятия решений. Результаты работы могут служить основой для разработки классификационных систем в прикладных проектах, а также для углублённого изучения особенностей работы алгоритмов в различных условиях.

**Ход работы**

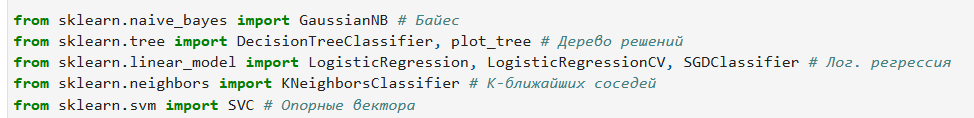
**Задание 1. Изучить алгоритмы Машинного обучения для решения задачи**

**классификации:**

* Наивный Байес
* Деревья Решений
* Логистическая Регрессия
* K-ближайших соседей
* Машины Опорных Векторов

Решение:

1. **Наивный Байес (Naive Bayes)**  
   *Основан на теореме Байеса* с предположением о независимости признаков друг от друга (отсюда «наивный»).
   * **Принцип работы**: вычисляет вероятность принадлежности объекта к классу, используя условные вероятности признаков.
   * **Преимущества**: высокая скорость, эффективность на больших данных, хорошая работа с текстовыми данными (например, фильтрация спама).
   * **Недостатки**: игнорирует зависимости между признаками, что может снижать точность.
2. **Деревья Решений (Decision Trees)**  
   *Иерархическая структура* с узлами (условия на признаки) и листьями (классы).
   * **Принцип работы**: рекурсивно разделяет данные на подмножества, максимизируя информативность (через энтропию или индекс Джини).
   * **Преимущества**: интерпретируемость, работа с нелинейными данными, отсутствие требований к масштабированию признаков.
   * **Недостатки**: склонность к переобучению, чувствительность к шумам.
3. **Логистическая Регрессия (Logistic Regression)**  
   *Линейный метод* для бинарной классификации, обобщаемый на многоклассовые задачи.
   * **Принцип работы**: оценивает вероятность принадлежности к классу через сигмоидную функцию (логит).
   * **Преимущества**: простота, скорость, интерпретируемость коэффициентов.
   * **Недостатки**: требует линейной разделимости данных, чувствителен к мультиколлинеарности.
4. **K-ближайших соседей (K-Nearest Neighbors, KNN)**  
   *Непараметрический алгоритм*, классифицирующий объекты на основе схожести с ближайшими примерами.
   * **Принцип работы**: выбирает класс, доминирующий среди K*K* ближайших соседей (расстояние вычисляется, например, через Евклидову метрику).
   * **Преимущества**: простота реализации, адаптивность к нелинейным данным.
   * **Недостатки**: высокая вычислительная сложность на больших данных, чувствительность к шумам и выбору K*K*.
5. **Машины Опорных Векторов (Support Vector Machines, SVM)**  
   *Метод поиска оптимальной гиперплоскости*, разделяющей классы с максимальным зазором.
   * **Принцип работы**: преобразует данные в пространство высшей размерности (через ядра: линейное, RBF, полиномиальное) для поиска разделяющей границы.
   * **Преимущества**: эффективность на данных с высокой размерностью, устойчивость к переобучению (при правильном выборе ядра).
   * **Недостатки**: сложность настройки, высокая вычислительная нагрузка для больших выборок.

Загрузку моделей можно произвести с помощью библиотеки scikit-learn:

**Задание 2. Изучить соответствующие библиотеки и инструменты для анализа данных в Python.**

Решение:

**Изучение библиотек**:

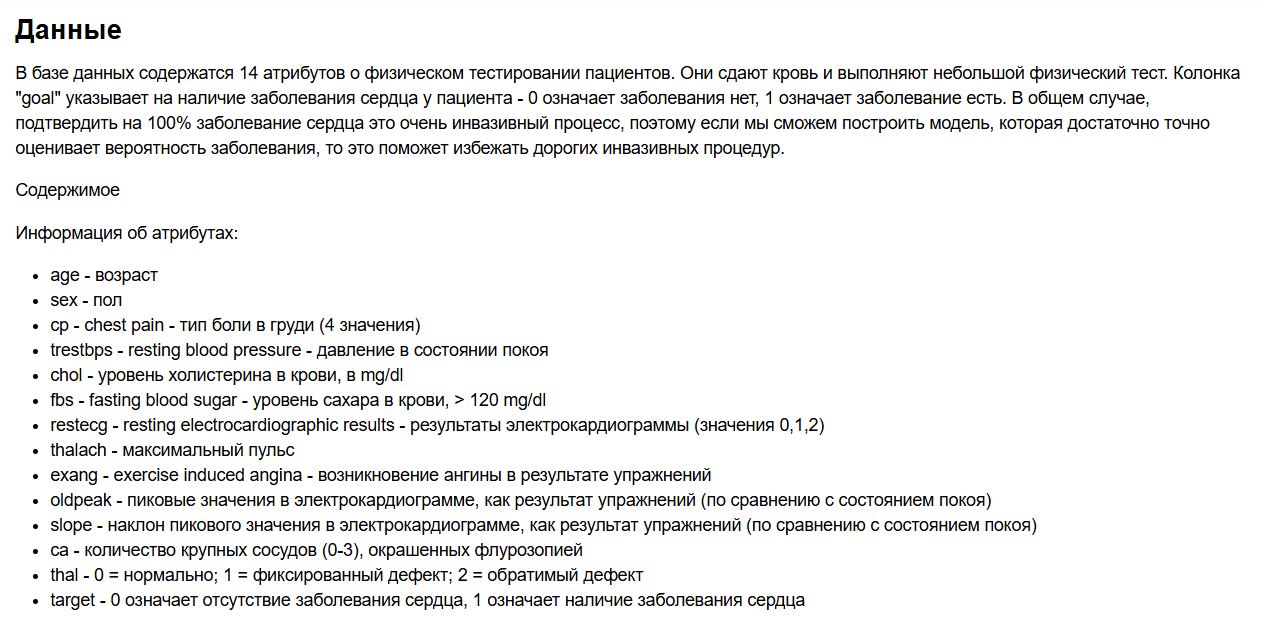
* **Pandas**: библиотека для работы с табличными данными (DataFrame). Позволяет загружать, очищать, фильтровать и агрегировать данные.
* **NumPy**: библиотека для работы с многомерными массивами и выполнения математических операций.
* **Matplotlib и Seaborn**: инструменты для визуализации данных (графики, гистограммы, диаграммы рассеяния).
* **Scikit-learn**: библиотека для машинного обучения, включающая реализацию алгоритмов классификации, регрессии, кластеризации, а также инструменты для предобработки данных и оценки моделей.

Вот все мои библиотеки и методы, которые я использовал в своей работе:



**Задание 3. Подобрать или сгенерировать данные с подходящей структурой для решения задачи классификации.**

Решение:

Данные я взял с сайта Kaggle, где можно найти множество датасетов для самых разных задач, вот их описание:

Источник: <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

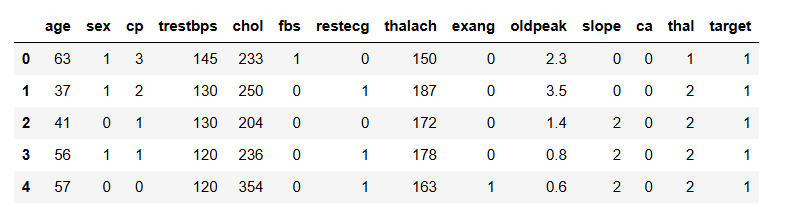
**Задание 4 – 5. Построить модели на основе указанных методов, используя соответствующие библиотеки Python.**

Решение:

Для начала загружаем данные в датафрейм и рассматриваем его:

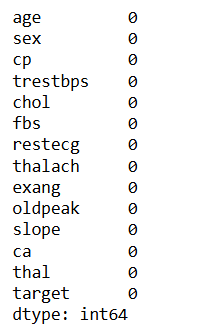
df = pd.read\_csv('heart.csv')

df.head()

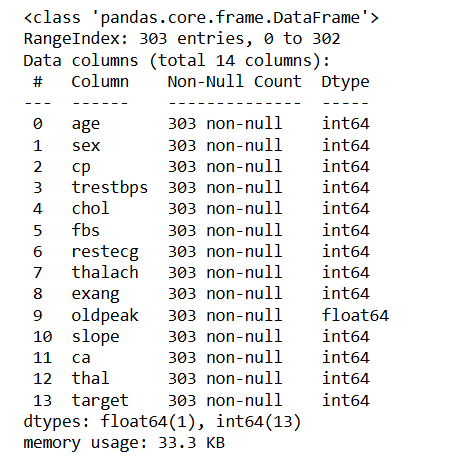


Проверяем на выбросы:

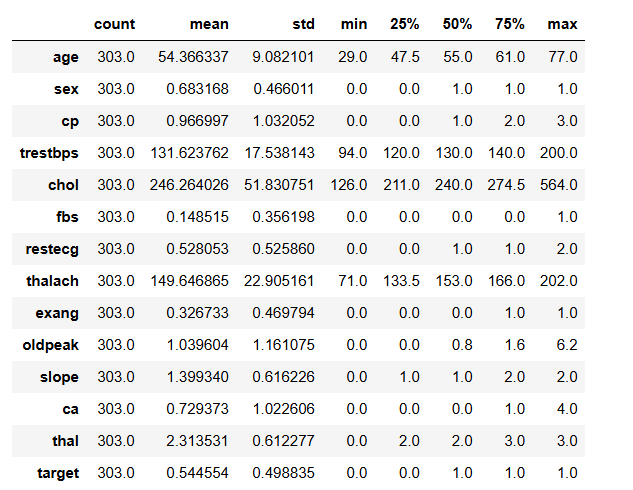
df.isnull().sum()



Информация о признаках датафрейма:



Сводка статистических показателей признаков нашего датафрейма:



Удаляем дубликаты:

df.drop(index = 163, inplace=True)

Сделаем анализ выбросов:

plt**.**figure(figsize**=**(12,8),dpi**=**200)

sns**.**boxplot(data**=**df**.**drop(['trestbps', 'chol', 'thalach', 'age', 'target', 'oldpeak', 'ca'], axis**=**1), fill**=**False, color**=**'orange', linewidth**=**1.75,

flierprops**=**{"marker": "x"})

plt**.**title('Распределения категориальных признаков', fontsize**=**20);

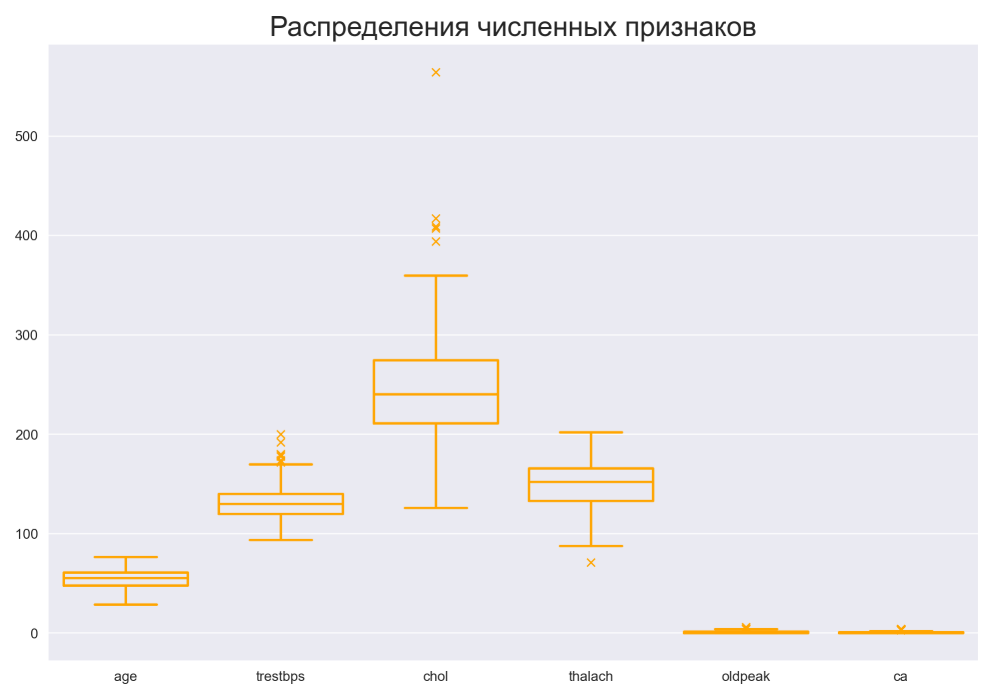


plt.figure(figsize=(12,8),dpi=200)

sns.boxplot(data=df.drop(['sex', 'cp', 'fbs', 'restecg', 'target', 'exang', 'slope', 'thal'], axis=1), fill=False, color='orange', linewidth=1.75,

flierprops={"marker": "x"})

plt.title('Распределения численных признаков', fontsize=20);

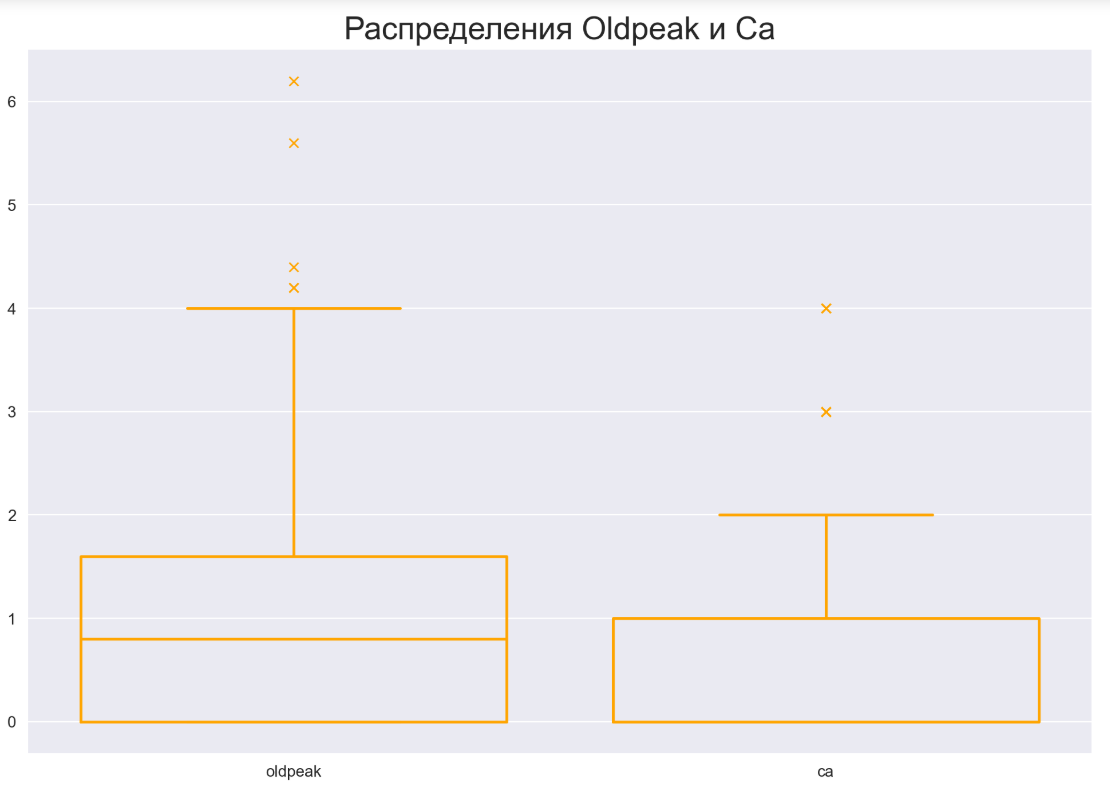


plt.figure(figsize=(12,8),dpi=200)

sns.boxplot(data=df.drop(['sex', 'cp', 'fbs', 'restecg', 'target', 'exang', 'slope', 'thal', 'age', 'trestbps', 'chol', 'thalach'], axis=1), fill=False, color='orange', linewidth=1.75,

flierprops={"marker": "x"})

plt.title('Распределения Oldpeak и Ca', fontsize=20);



Теперь обработаем выбросы в числовых столбцах с помощью теста IQR:

for col in ['trestbps', 'chol', 'thalach', 'age', 'oldpeak', 'ca']:

Q1 = np.percentile(df[col], 25)

Q3 = np.percentile(df[col], 75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

mean = df[(df[col] <= upper\_bound) & (df[col] >= lower\_bound)][col].mean()

df[col] = df[col].mask(df[col] > upper\_bound, mean)

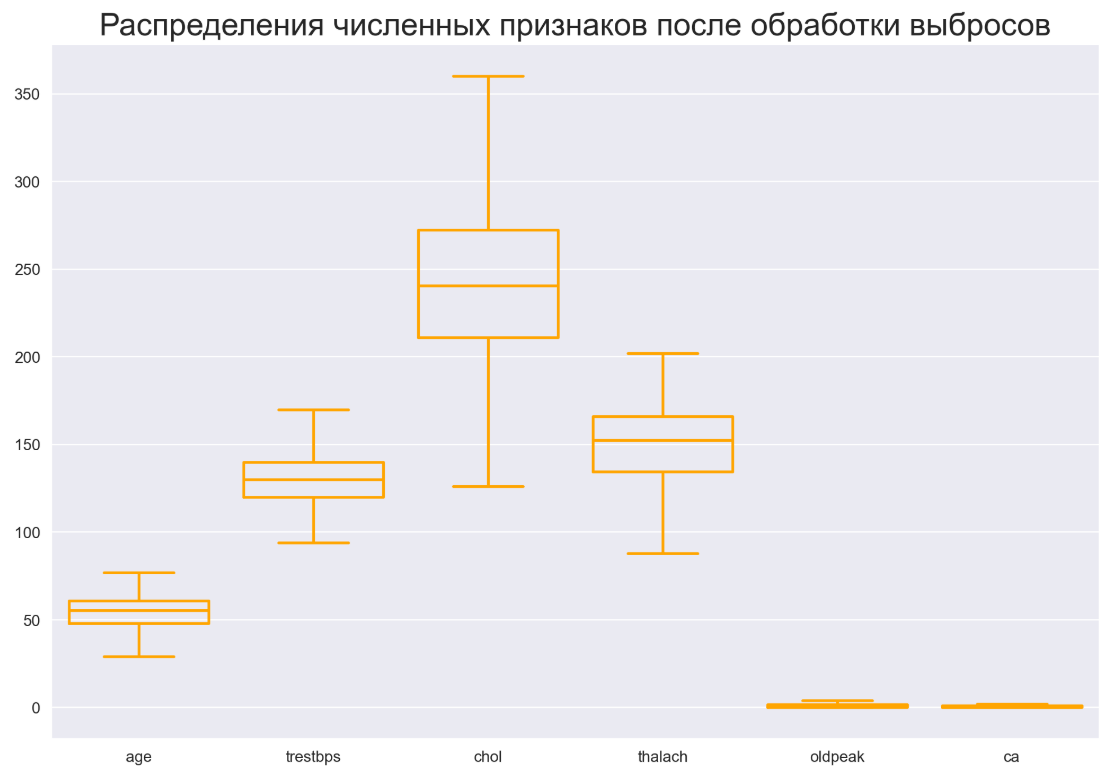
df[col] = df[col].mask(df[col] < lower\_bound, mean)

plt.figure(figsize=(12,8),dpi=200)

sns.boxplot(data=df.drop(['sex', 'cp', 'fbs', 'restecg', 'target', 'exang', 'slope', 'thal'], axis=1), fill=False, color='orange', linewidth=1.75,

flierprops={"marker": "x"})

plt.title('Распределения численных признаков после обработки выбросов', fontsize=20);



Перед применением моделей проведём некоторую визуализацию, чтобы лучше:

plt.figure(figsize=(12,6), dpi=200)

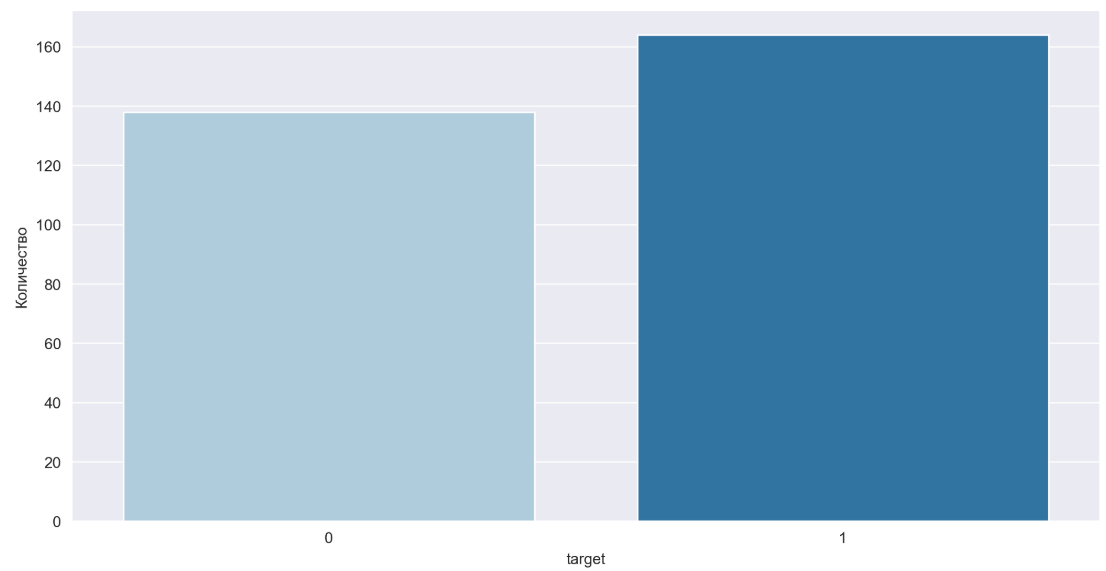
sns.countplot(data=df, x='target', palette='Paired');

plt.ylabel('Количество');

plt.figure(figsize=(12,6), dpi=200)

sns.countplot(data=df, x='target', palette='Paired');

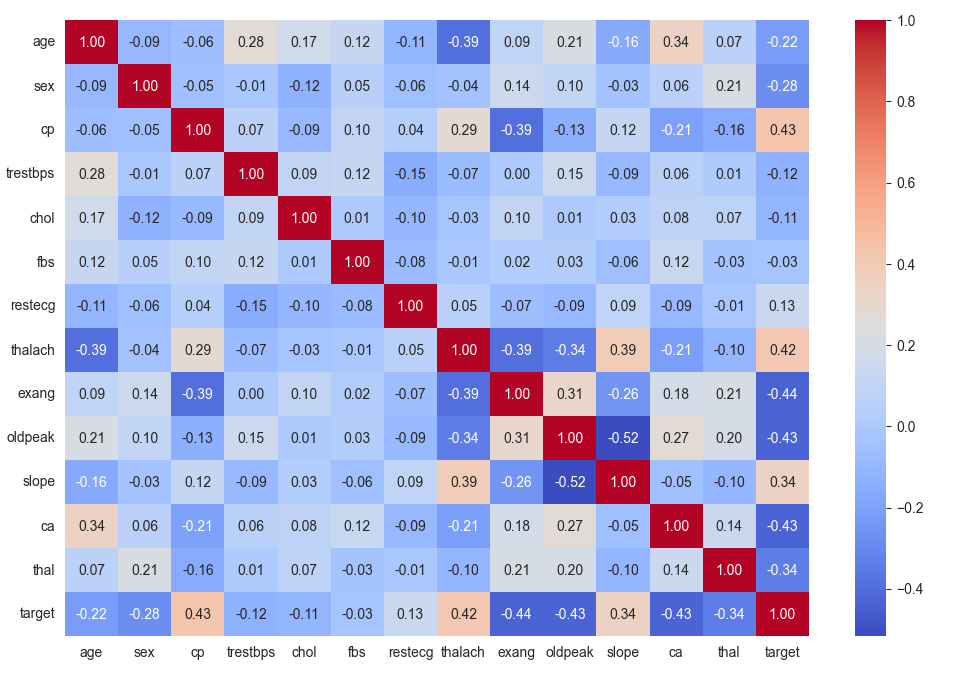
plt.ylabel('Количество');



Также рассмотрим тепловую карту корреляции признаков:

plt.figure(figsize=(12,8))

sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm', annot\_kws={'size':10});



Наконец, приступим к машинному обучению, разделив выборку на тестовую и обучающую:

X = df.drop('target', axis=1)

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=101)

Проведём масштабирование, чтобы повысить эффективность обучения:

scaler = StandardScaler()

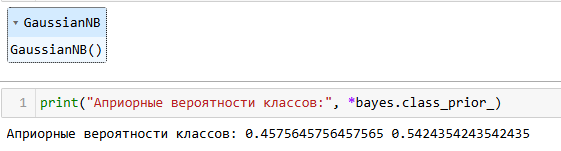
X\_train\_scale = scaler.fit\_transform(X\_train)

X\_test\_scale = scaler.transform(X\_test)

**Классификатор Байеса:**

bayes = GaussianNB()

bayes.fit(X\_train\_scale, y\_train)



Выведем средние значения и дисперсию для каждого класса в признаках, предварительно отмасштабировав данные обратно:

bayes\_teta = scaler.inverse\_transform(bayes.theta\_)

bayes\_var = scaler.inverse\_transform(bayes.var\_)

pd.DataFrame({'Средние значения класс 0': bayes.theta\_[0],

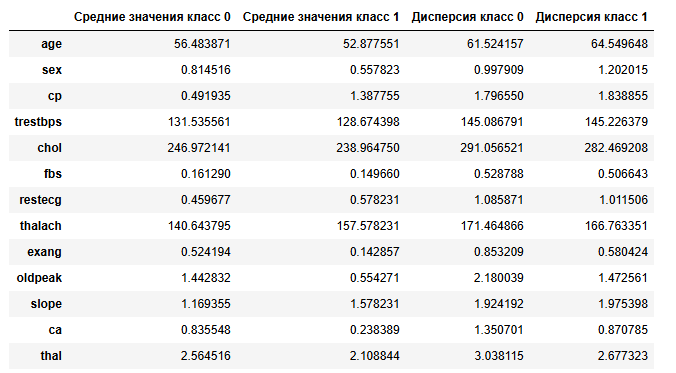
'Средние значения класс 1': bayes.theta\_[1],

'Дисперсия класс 0': bayes.var\_[0],

'Дисперсия класс 1': bayes.var\_[1]},

index = X.columns

)



Большое различие в средних значениях по классам означает большую значимость для задачи. Из этой таблицы можно заключить, что признаки: cp -тип боли в груди, oldpeak - давление в состоянии покоя, slope - наклон пикового значения в электрокардиограмме и ca - количество крупных сосудов сильно влияют на нашу задачу.

**Дерево решений:**

Запишем возможные гиперпараметры в словаре, среди которых выберем оптимальные во время кросс-валидации:

max\_depth = np.linspace(1, 10, 10).astype('int')

min\_samples\_split = np.linspace(1, 10, 10).astype('int')

ccp\_alpha = np.linspace(0, 1, 10)

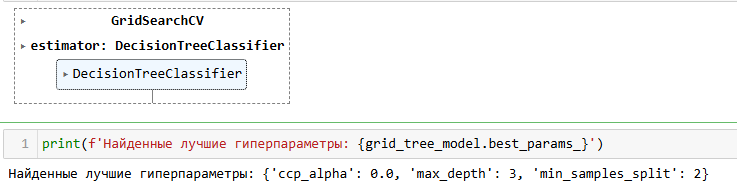
params = {'max\_depth': max\_depth, 'min\_samples\_split': min\_samples\_split, 'ccp\_alpha': ccp\_alpha}

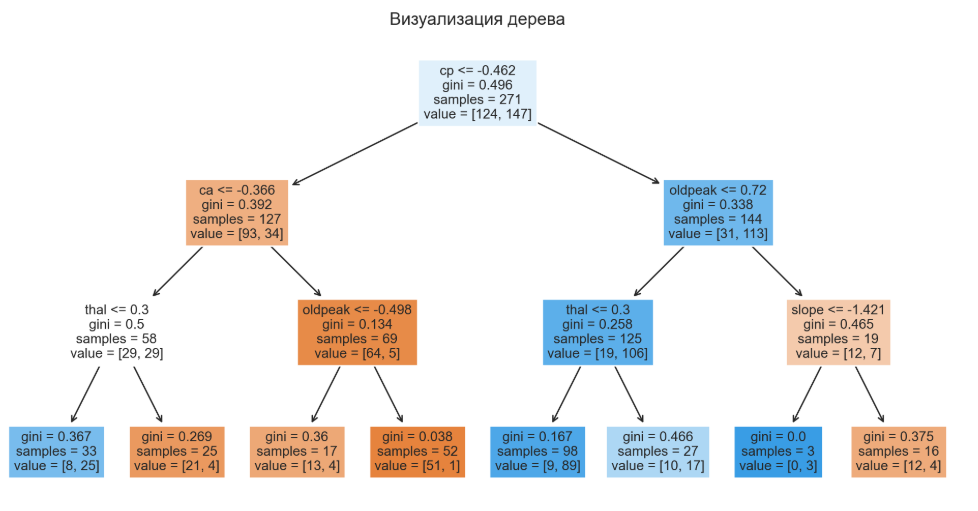
Проинициализируем и обучим модель:

tree = DecisionTreeClassifier(criterion='gini')

grid\_tree\_model = GridSearchCV(tree, param\_grid=params)

grid\_tree\_model.fit(X\_train\_scale, y\_train)



В библиотеки scikit-learn есть возможность визуализировать обученное дерево, поэтому не будем пренебрегать этим:

По направлению к листья данные становятся всё более однородными, что соответствует основному принципу обучения этого алгоритма – стремление к уменьшению неопределённости в узлах дерева. Конкретно в нашем случае, мы применяем метрику Gini для снижения неопределённости.

**Логистическая регрессия:**

Также определим гиперпараметры и обучим модель:

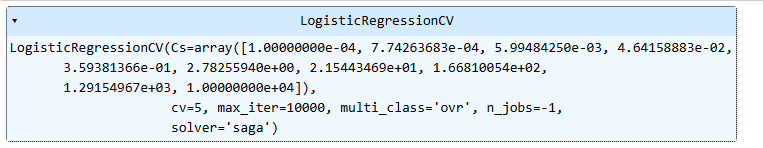
l1\_ratio = np.linspace(0, 1, 30)

C = np.logspace(-4, 4, num=10)

log\_cv\_model = LogisticRegressionCV(cv=5, Cs=C,

max\_iter=10000, solver='saga', multi\_class='ovr', n\_jobs=-1)

log\_cv\_model.fit(X\_train\_scale, y\_train)



Отсортируем коэффициенты уравнения лог. регрессии и выведем на графике:

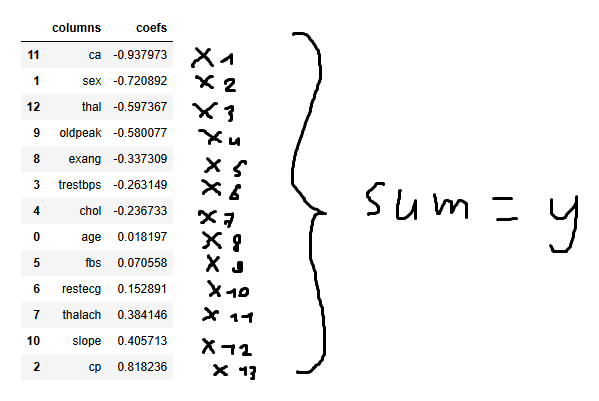
coefs = log\_cv\_model.coef\_[0]

columns = X\_train.columns

feature\_df = pd.DataFrame({'columns': columns, 'coefs': coefs})

feature\_df = feature\_df.sort\_values(by='coefs')

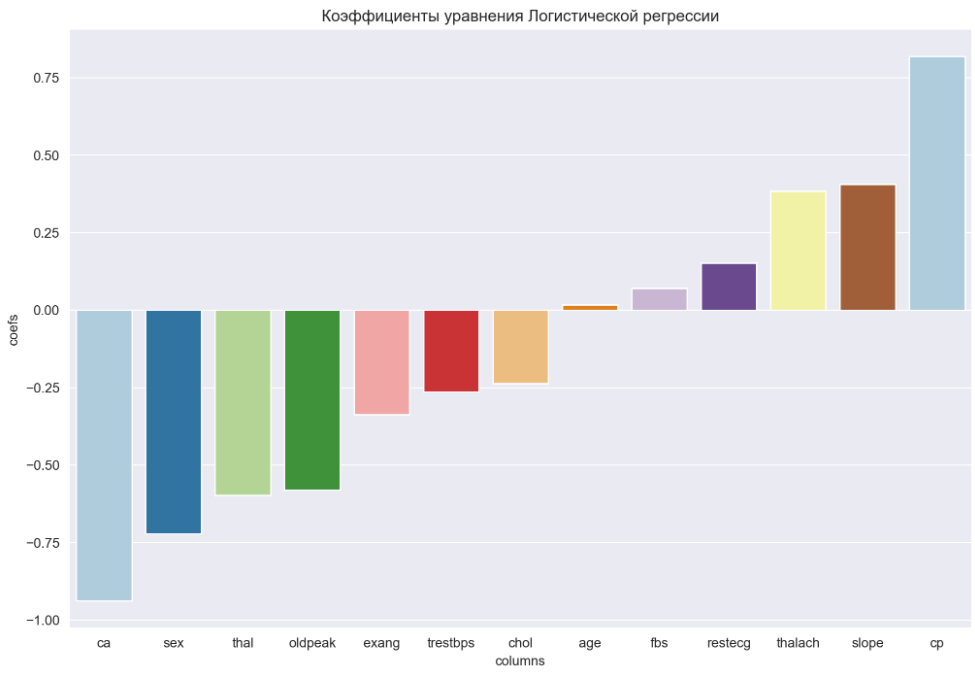
feature\_df



plt.figure(figsize=(12,8), dpi=200)

sns.barplot(x=feature\_df['columns'], y=feature\_df['coefs'], palette='Paired');

plt.title('Коэффициенты уравнения Логистической регрессии');



Благодаря этому графику можно определить, какие признаки не информативные, например, age или fbs - уровень сахара в крови не вносят большого вклада в задачу.

**К-ближайшие соседи:**

Здесь в качестве гиперпараметров выступают метрики расстояния и количество соседей:

p = [1, 2]

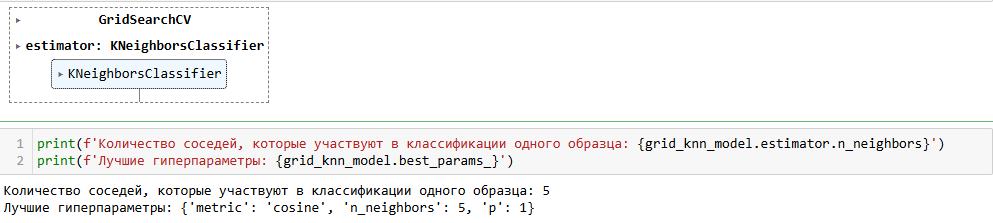
metrics = ['minkowsky', 'cosine']

n\_neighbors = [3, 4, 5, 6, 7]

params = {'n\_neighbors': n\_neighbors, 'p': p, 'metric': metrics}

knn\_model = KNeighborsClassifier(n\_jobs=-1)

grid\_knn\_model = GridSearchCV(knn\_model, param\_grid=params)

grid\_knn\_model.fit(X\_train\_scale, y\_train)

Алгоритм лучше решает задачу на 5 соседях с косинусным расстоянием между ними.

Так как алгоритм К-ближайших соседей относится к непараметрическим, то его достаточно сложно интерпретировать, тем более он относится к обучению без учителя, поэтому кроме как лучших гиперпараметров, я не привёл здесь никаких пояснений к модели. Да и явной стадии обучения у этого алгоритма нет, он просто рассматривает голоса соседей и выбирает большинство.

**SVM:**

Этот алгоритм требует тщательной настройки гиперпараметров, поэтому передадим в неё больше комбинаций:

kernel = ['linear', 'rbf']

C = np.linspace(1, 10, 20)

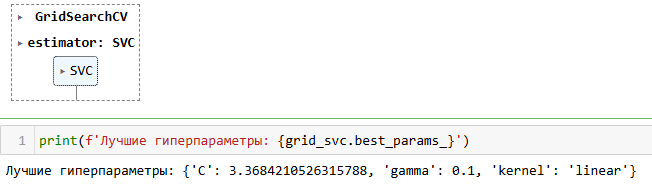
gamma = np.linspace(0.1, 100, 30)

params = {'kernel': kernel, 'C': C, 'gamma': gamma}

svc = SVC()

grid\_svc = GridSearchCV(svc, param\_grid=params)

grid\_svc.fit(X\_train\_scale, y\_train)

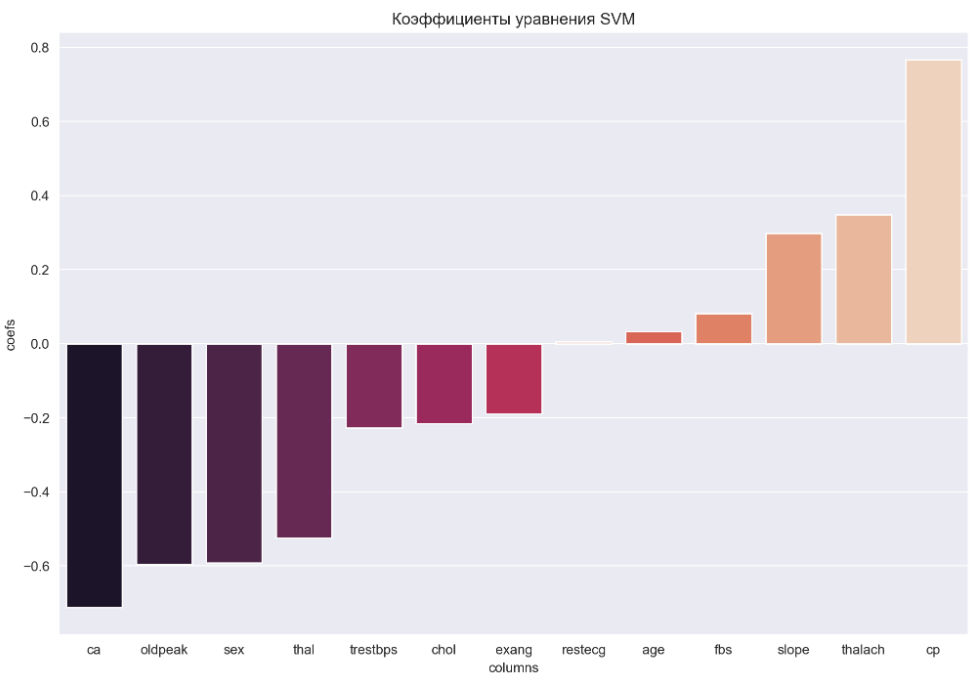


С – это параметр регуляризации, gamma отвечает за жёсткость области, в которой происходит обучение. Так как линейный SVM показал лучший результат, то мы можем вывести коэффициенты, полученные после обучения:

plt.figure(figsize=(12,8), dpi=200)

sns.barplot(x=feature\_df['columns'], y=feature\_df['coefs'], palette='rocket');

plt.title('Коэффициенты уравнения SVM');



Как можем видеть, всё те же age и fbs дают малый вклад в задачу. Можем сделать вывод, что их не обязательно использовать при обучении моделей.

**Задание 5-6. Оценить результаты моделирования, произвести настройку параметров модели. Визуализировать результаты.**

Решение:

Для оценки моделей я написал универсальную функцию, которая выводит все необходимые метрики и графики классификации:

def print\_all\_metrics(model, X\_test, y\_test, name):

preds = model.predict(X\_test)

# Conf matrix

print('Матрица ошибок:')

print(confusion\_matrix(y\_test, preds))

print()

print()

# Display conf matrix

fig, ax = plt.subplots(figsize=(12,8))

ConfusionMatrixDisplay.from\_estimator(model, X\_test, y\_test, cmap='rocket', ax=ax)

plt.grid(False)

plt.title(f'Матрица ошибок для {name}')

plt.show()

# Classification report

print('Отчёт по классификации:')

print(classification\_report(y\_test, preds))

print()

# PR-кривая

PrecisionRecallDisplay.from\_estimator(model, X\_test\_scale, y\_test);

plt.title(f'PR-кривая для {name}')

plt.show()

# ROC-кривая

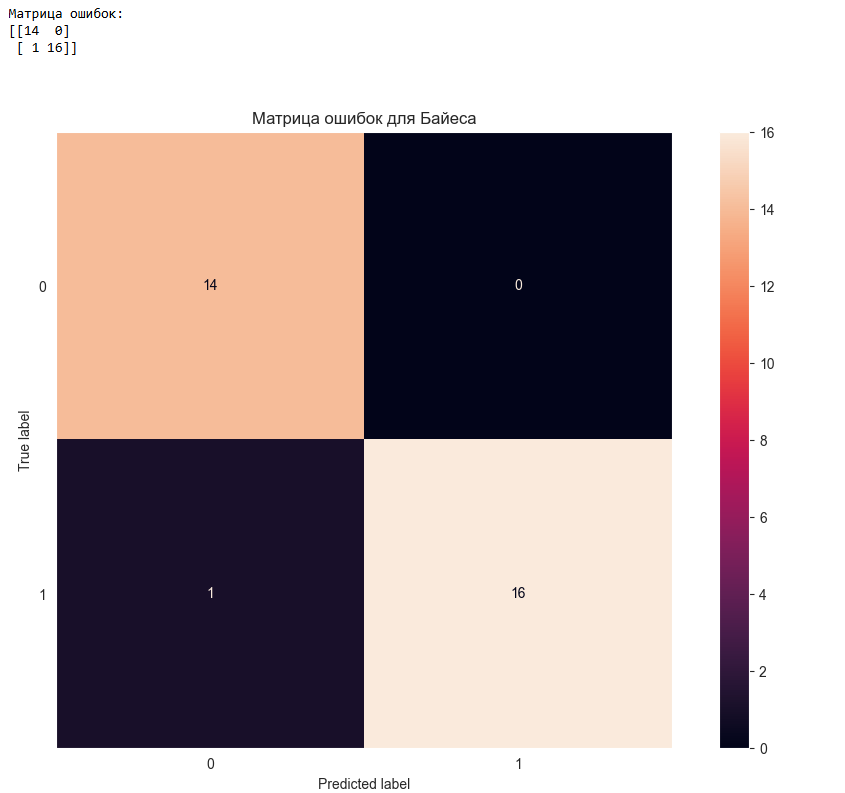
RocCurveDisplay.from\_estimator(model, X\_test\_scale, y\_test);

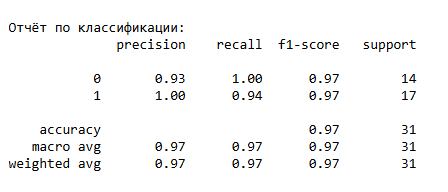
plt.title(f'ROC-кривая для {name}')

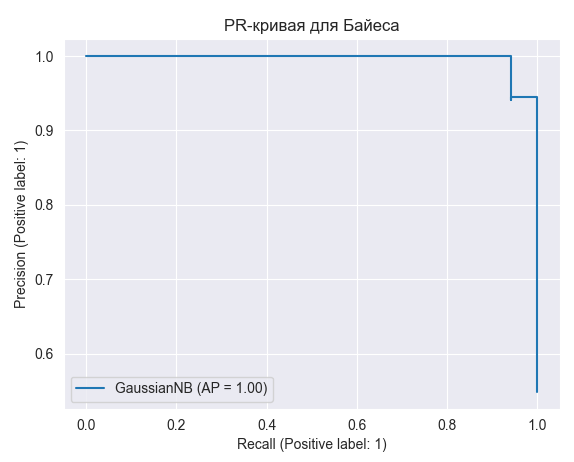
plt.show()

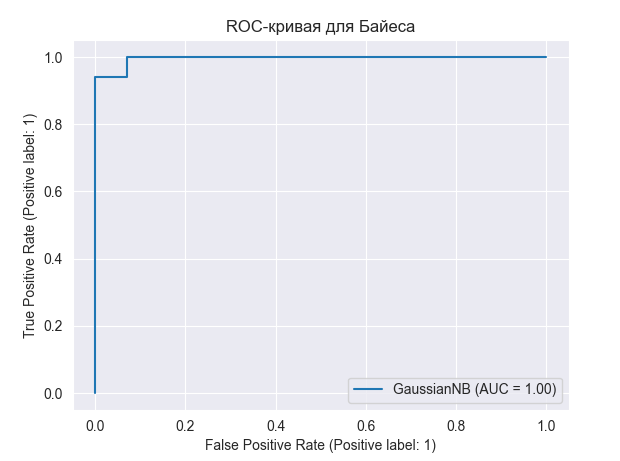
**Начнём с Байеса, и так дальше:**

print\_all\_metrics(bayes, X\_test\_scale, y\_test, 'Байеса')





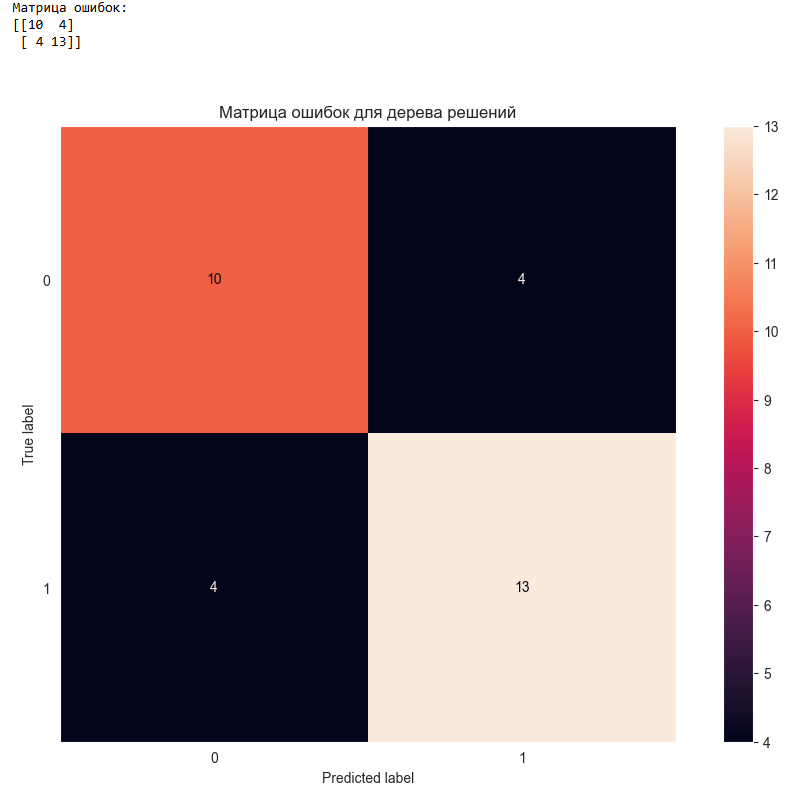


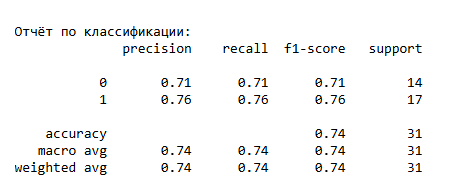


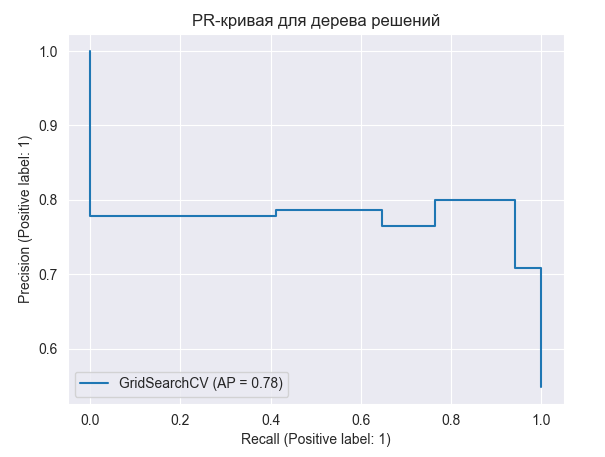
По увиденным метрикам, классификатор Байеса почти идеально справился с этой задачей.

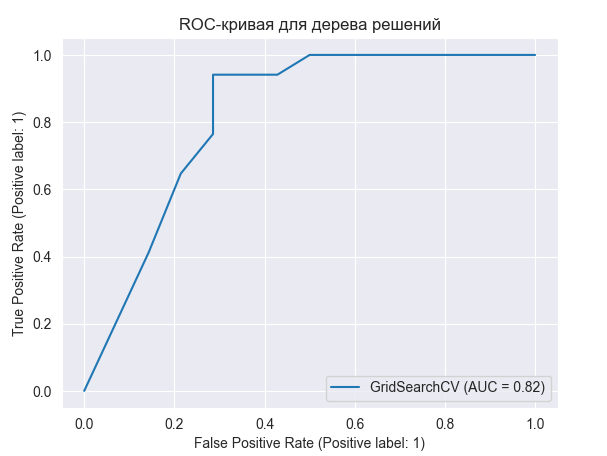
**Дерево решений:**

print\_all\_metrics(grid\_tree\_model, X\_test\_scale, y\_test, 'дерева решений')





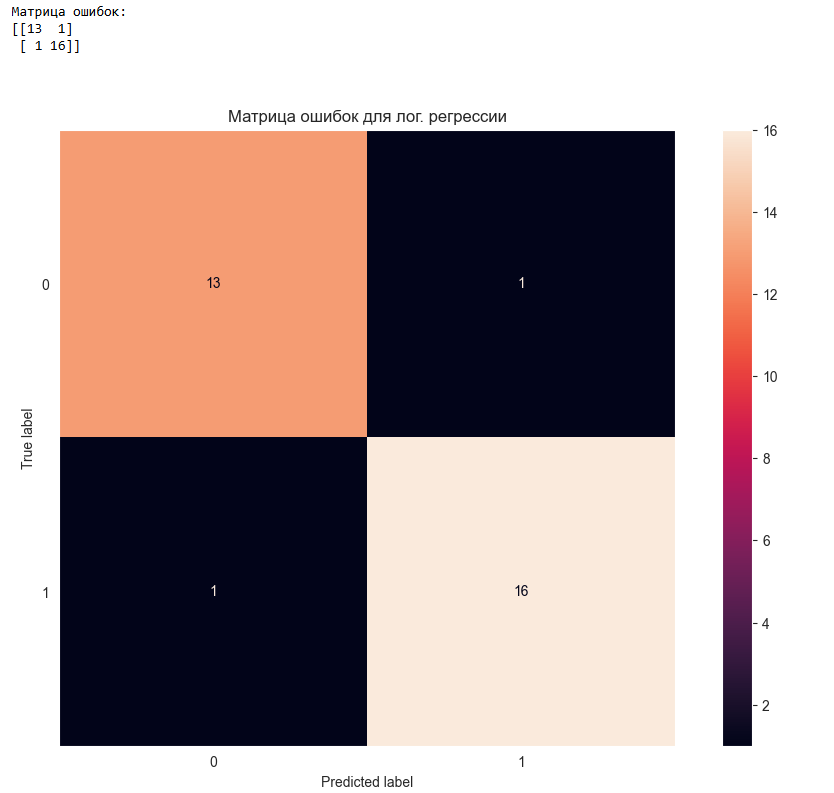


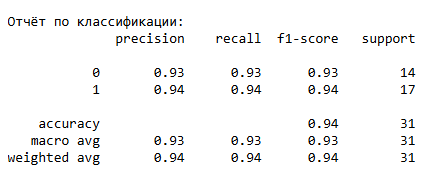


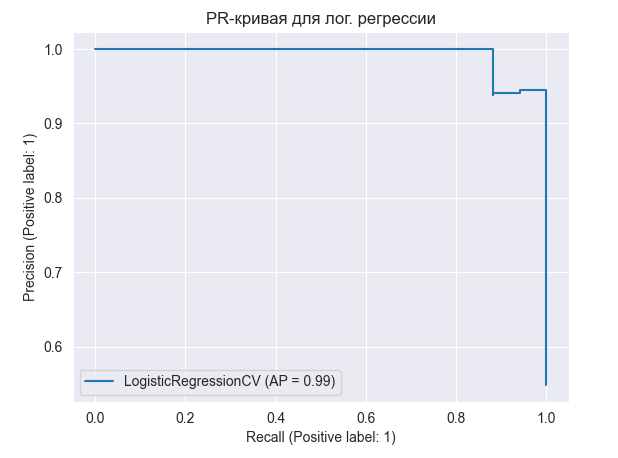
Дерево решений уже значительно хуже справилось с задачей, уступая классификатору Байеса на 0.2 по взвешенной f1-мере – это очень много.

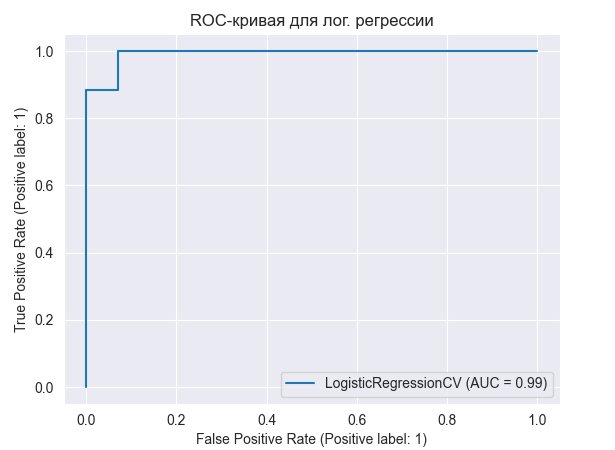
**Логистическая регрессия:**

print\_all\_metrics(log\_cv\_model, X\_test\_scale, y\_test, 'лог. регрессии')





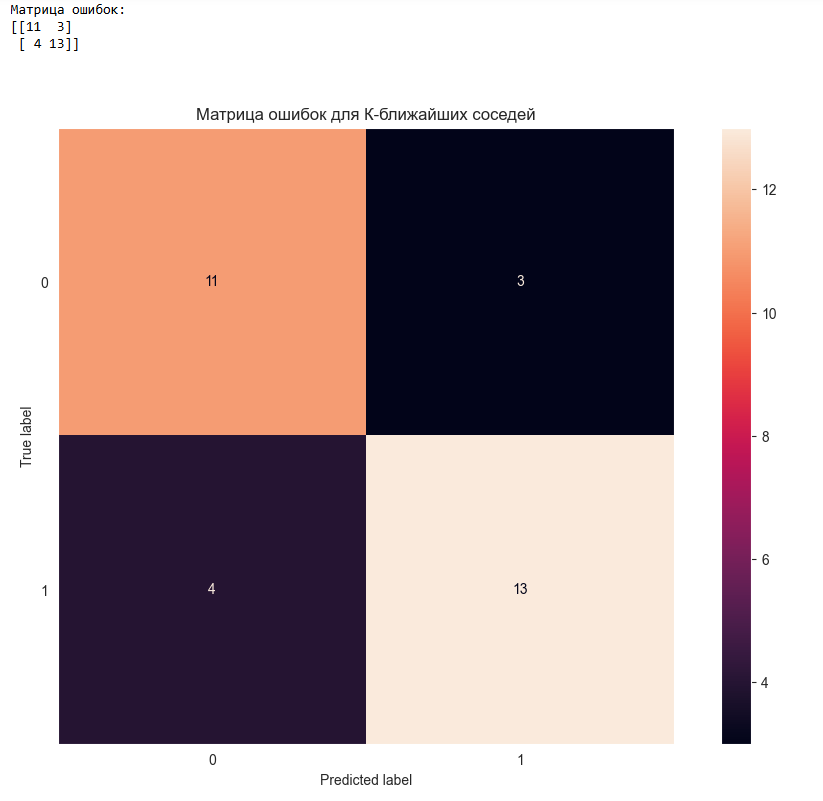


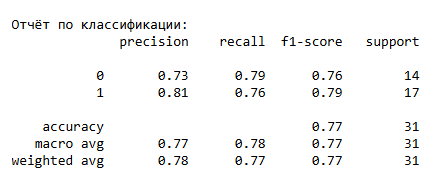


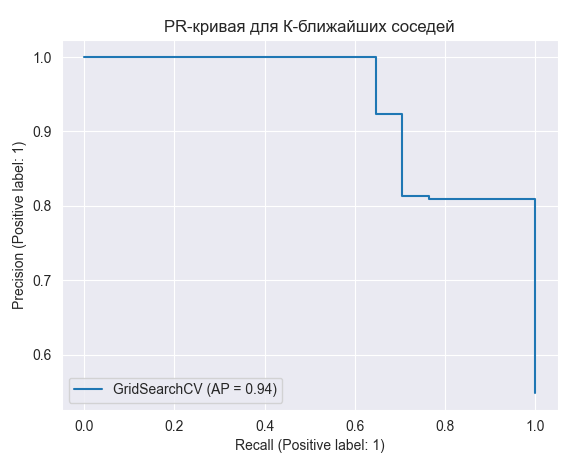
Лог. регрессия уже получше классифицирует данные, близко подобравшись к результатам Байеса.

**К-ближайших соседей:**

print\_all\_metrics(grid\_knn\_model, X\_test\_scale, y\_test, 'К-ближайших соседей')



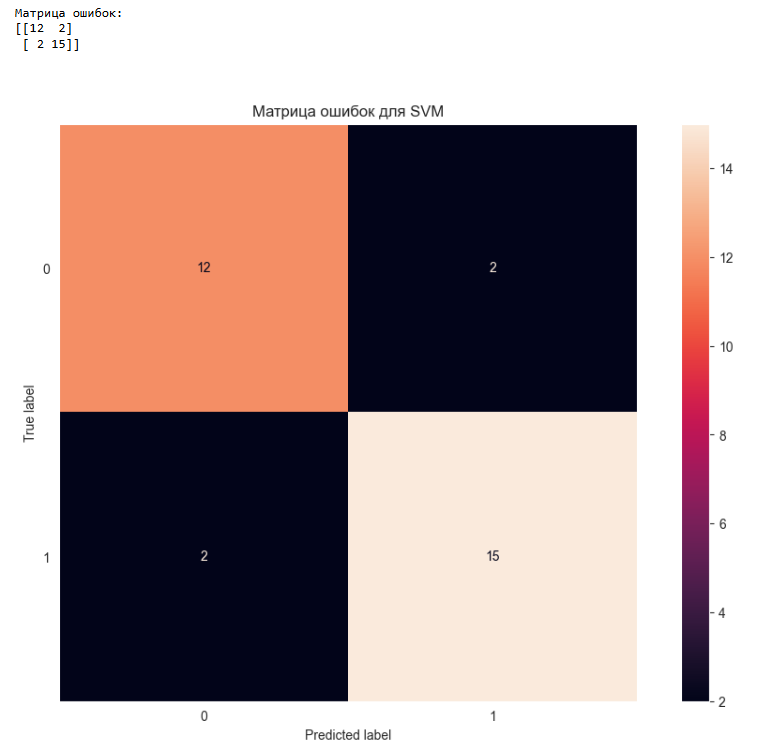


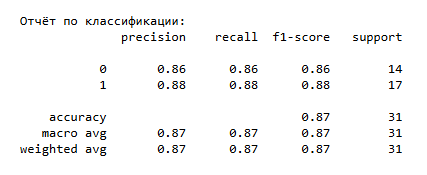


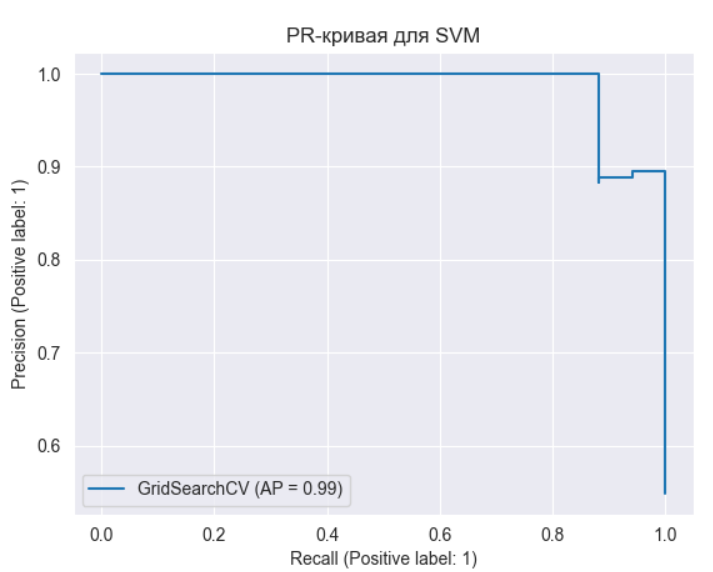


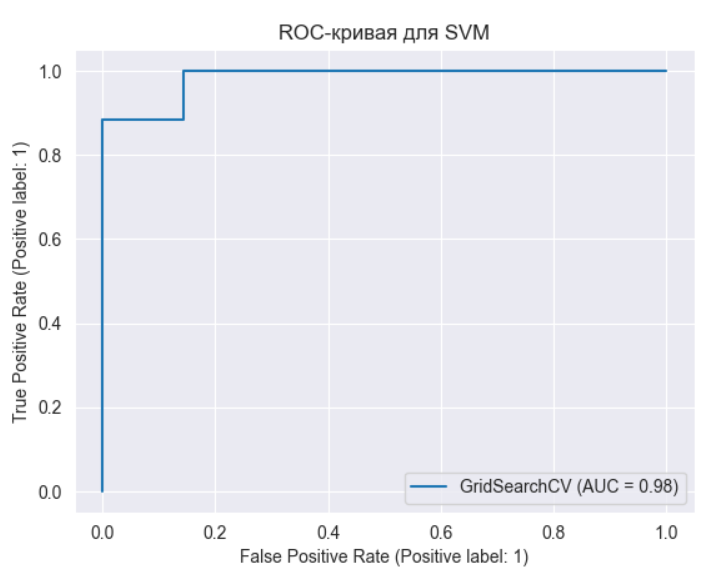
Метрики KNN тоже не радуют, но скорее всего это из-за того, что данный алгоритм очень чувствителен к объёму данных. В нашей выборке всего лишь 300 значений, а метод делает акцент именно на количестве: чем больше данных, тем лучше; поэтому здесь плохая классификация.

**SVM:**









В принципе, неплохие результаты для SVM. Сам алгоритм является достаточно сильным, тем более была произведена настройка гиперпараметров.

**Сравнение моделей:**

bayes = [0.97, 0.97, 0.97, 0.97, 0]

decision\_tree = [0.74, 0.74, 0.74, 0.74, 0]

log\_regr = [0.93, 0.93, 0.93, 0.94, 0]

knn = [0.77, 0.78, 0.77, 0.77, 1]

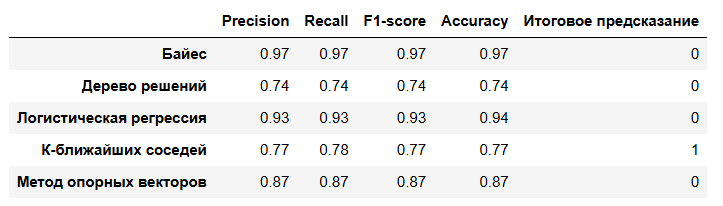
svm = [0.87, 0.87, 0.87, 0.87, 0]

pd.DataFrame(data=[bayes, decision\_tree, log\_regr, knn, svm],

columns=['Precision', 'Recall', 'F1-score', 'Accuracy', 'Итоговое предсказание'],

index=['Байес', 'Дерево решений', 'Логистическая регрессия', 'К-ближайших соседей', 'Метод опорных векторов']

)



Лучшим является Байес, а хуже всех справились дерево решений и knn. Итоговое предсказание проводилось по 1 случайному пациенту на введённых мною данных, как ни странно, К-ближайших соседей не смог справиться с задачей.

**Список используемой литературы**

1. McKinney, W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. — 2nd Edition. — O'Reilly Media, 2017. — 544 p.
2. Pedregosa, F., et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.
3. Waskom, M. L. seaborn: statistical data visualization // Journal of Open Source Software. — 2021. — Vol. 6, № 60. — P. 3021.
4. Hunter, J. D. Matplotlib: A 2D Graphics Environment // Computing in Science & Engineering. — 2007. — Vol. 9, № 3. — P. 90–95.
5. VanderPlas, J. Python Data Science Handbook: Essential Tools for Working with Data. — O'Reilly Media, 2016. — 548 p.