Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники.

Исследовательский проект по предмету «Математическая статистика»

Реализация и сравнительный анализ методов классификации для диагностики рака молочной железы: статистический подход vs машинное обучение

Выполнили:

Горюнов Семён	Статистические методы и теория
Вавилина Екатерина	Машинное обучение и теория
Медведева Даниэла	Предобработка данных

Группа: P3230

Санкт-Петербург
2025

1. Постановка задачи 2
1.1 Общая характеристика задачи 2
1.2 Описание данных 2
1.3 Математическое описание задачи 2
1.4 Описание используемых методов 2
2. Теория 3
2.1 Логистическая регрессия 3
2.2 Линейный дискриминантный анализ (LDA) 4
2.3 Квадратичный дискриминантный анализ (QDA) 5
2.4 Случайный лес (Random Forest) 6
2.5 Метод опорных векторов (SVM) 7
3. Используемые программные средства 9
4. Результаты 10
4.1 Предварительный анализ и обработка данных 10
4.2 Статистические методы 15
4.2.1 Логистическая регрессия 15
4.2.2 Линейный дискриминантный анализ (LDA) 16
4.2.3 Квадратичный дискриминантный анализ (QDA) 18
4.3 Машинное обучение 19
4.3.1 Случайный лес (Random Forest) 19
4.3.2 Метод опорных векторов (SVM) 21
5. Обсуждение и выводы 23
5.1 Сравнение методов статистического анализа 23
5.2 Сравнение методов машинного обучения 24
5.3 Общее сравнение 25
5.4 Дальнейшие исследования 27

1. Постановка задачи

1.1 Общая характеристика задачи

Цель исследования - сравнение эффективности и применимости статистических методов и методов машинного обучения в задаче бинарной классификации для предсказания типа опухоли молочной железы. Для этого проанализируем ключевые метрики качества классификации: точность и матрицы ошибок. Результаты позволят определить наиболее точные модели, подходящие для практического использования.

1.2 Описание данных

В рамках данного исследования мы классифицируем тип опухоли молочной железы на основе набора числовых характеристик, полученных из цифрового изображения клеток при тонкоигольной аспирационной биопсии (FNA).

Выборка состоит из 569 наблюдений, каждое из которых описывается 30 числовыми признаками, такими как радиус, текстура, периметр, площадь, компактность и т.д. Все данные взяты из открытого датасета Breast Cancer Wisconsin (Diagnostic) Data Set.

Датасет: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data

1.3 Математическое описание задачи

Для каждого из оцениваемых методов требуется построить функцию отображения, предсказывающую класс нового объекта на основе его признаков с минимальной ошибкой:

$$f: X \to Y$$

где $X \subset \Re^d$ - пространство признаков размерности d;
Y - пространство классов {Доброкачественные (B), Злокачественные (M)}.

1.4 Описание используемых методов

Статистические методы: Линейный дискриминантный анализ (LDA), квадратичный дискриминантный анализ (QDA), логистическая регрессия.

Методы машинного обучения: Метод опорных векторов (SVM), Алгоритм случайного леса (Random Forest).

2. Теория

2.1 Логистическая регрессия

Логистическая регрессия - это метод, используемый для задач бинарной классификации, на основе набора входных признаков. Она основана на предположении, что вероятность принадлежности объекта к классу 1 может быть выражена через сигмоидную функцию, которая преобразует линейную комбинацию признаков в значение от 0 до 1.

Иначе говоря, для набора признаков объекта $x^T = (x_1, x_2, \dots x_n)$ логистическая регрессия вычисляет линейную комбинацию:

$$z = b + w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n = w * x + b$$

где $w = (w_1, w_2, \cdots, w_n)$ - вектор весов признаков;

b - свободный член.

Полученное значение z преобразуется с помощью сигмоидальной функции:

$$P(y = 1 \mid x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

которая возвращает вероятность принадлежности объекта к классу 1. На основе этой вероятности принимается решение:

- если $\sigma(z) \geq 0.5$, то объект относится к классу 1;
- иначе — к классу 0.

Граница между классами, или разделяющая гиперплоскость , определяется уравнением $w * x + b = 0$

Для получения весов линейной комбинации необходимо решить задачу оптимизации логарифмической функции потерь, которая измеряет, насколько предсказанные вероятности отличаются от истинных меток:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

где N - число объектов;

y_i - истинная метка (0 или 1);

$\hat{y}_i$ - предсказанная вероятность для класса 1.

Для минимизации функции потерь используется градиентный спуск и рассчитывается градиент функции потерь:

$$\frac{\partial Loss}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)x_{ij}$$

$$\frac{\partial Loss}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)$$

Обновление коэффициентов происходит следующим образом:

$$w_j \leftarrow w_j - \eta * \frac{\partial Loss}{\partial w_j}, b \leftarrow b - \eta * \frac{\partial Loss}{\partial b}$$

где η - скорость обучения (размер шага, который делает модель).

2.2 Линейный дискриминантный анализ (LDA)

Линейный дискриминантный анализ - это метод, используемый для задач классификации и снижения размерности без потери способности к разделению классов. Он ищет линейные комбинации признаков, которые обеспечивают максимальное разделение между классами и минимизирует разброс внутри каждого класса. Метод полагает, что данные каждого класса подчиняются многомерному нормальному распределению и что ковариационные матрицы всех классов одинаковы.

Пусть:

X - матрица размером (n, p), где n - число объектов, p - число признаков.

y - вектор меток классов, где $y_i \in \{1, 2, \cdots, C\}$

Для каждого класса вычисляется среднее значение признаков:

$$\mu_c = \frac{1}{n_c}\sum_{i \in c} x_i$$

где n_c - количество объектов класса c.

Также вычисляется общее среднее значение всех объектов: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Метод строится на двух ключевых матрицах:

- внутриклассового разброса $S_W = \sum_{c=1}^{C} \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T$
- межклассового разброса $S_B = \sum_{c=1}^{C} n_c(\mu_c - \mu)(\mu_c - \mu)^T$

Целью LDA является поиск такого собственного вектора весов $w \in \Re^p$ матрицы $S_W^{-1}S_B$, который максимизирует критерий Фишера:

$$J(w) = \frac{w^T S_W w}{w^T S_B w}$$

Количество ненулевых собственных векторов ограничено $min(C-1, p)$, эти векторы определяют направления, на которые проецируются данные.

Для задачи бинарной классификации (C = 2), классы разделяются гиперплоскостью $w * x + b = 0$:

- если $w * x + b \geq 0$, точка принадлежит к классу 2,
- иначе - к классу 1

Вектор весов и свободный член вычисляются как:

$$w = S_W^{-1}(\mu_2 - \mu_1)$$

$$b = -\frac{1}{2}(\mu_1^T S_W^{-1}\mu_1 - \mu_2^T S_W^{-1}\mu_2) + ln(\frac{\pi_2}{\pi_1})$$

где $\pi_c = \frac{n_c}{n}$ - априорная плотность класса c.

2.3 Квадратичный дискриминантный анализ (QDA)

Квадратичный дискриминантный анализ - это статистический метод, используемый для задач классификации. QDA является обобщением метода линейного дискриминантного анализа и отличается тем, что позволяет каждому классу иметь свою собственную ковариационную матрицу.

Метод использует формулу Байеса, которая показывает апостериорную вероятность принадлежности объекта x к классу c:

$$P(c|x) \sim p(x|c) \cdot \pi_c$$

где $p(x|c)$ - плотность многомерного нормального распределения;

$\pi_c = \frac{n_c}{n}$ - априорная плотность класса c.

Логарифмируя апостериорную плотность, приходим к дискриминантной функции:

$$\delta_c(x) = -\frac{1}{2}ln|\Sigma_c| - \frac{1}{2}(x-\mu_c)^T\Sigma_c^{-1}(x-\mu_c) + ln(\pi_c)$$

где x - вектор признаков объекта;

μ_c - среднее значение признаков для класса c;

Σ_c - ковариационная матрица класса c;

$|\Sigma_c|$ - определитель ковариационной матрицы.

Объект классифицируется как класс c, для которого $\delta_c(x)$ максимальна.

2.4 Случайный лес (Random Forest)

Метод случайного леса — это алгоритм машинного обучения, предназначенный для решения задач классификации и регрессии. Основная идея метода заключается в построении множества деревьев решений, каждое из которых обучается на собственной случайной подвыборке исходных данных.

Ключевая особенность метода - bootstrap-выборки для обучения отдельных деревьев. При формировании обучающей выборки из n значений выбирается n элементов с возвращением. Т.е. некоторые элементы могут попасть в подвыборку несколько раз, другие - не попасть вообще.

Каждое дерево решений строится рекурсивно, начиная с корня. При этом на каждом шаге разделения узла учитываются не все признаки, а лишь случайно выбранный поднабор (обычно $\sqrt{p}$ признаков). Это снижает корреляцию между деревьями и повышает устойчивость модели к переобучению.

Для определения наилучшего разбиения в узлах деревьев используется индекс Джини - мера неоднородности выборки в узле.

Индекс до разбиения рассчитывается по формуле:

$$Gini_{до} = 1 - \sum_{k=1}^{K} p_k^2$$

где K - количество классов;

p_k - доля объектов класса k в текущей группе объектов y.

После разбиения узла на две ветви аналогично рассчитывается $Gini_{лев}$ и $Gini_{прав}$ для левой и правой ветви соответственно.

Алгоритм выбирает разбиение, которое максимизирует уменьшение неопределенности, рассчитанное по формуле:

$$Gain = Gini_{до} - (\frac{n_{лев}}{n} Gini_{лев} + \frac{n_{прав}}{n} Gini_{прав}) \rightarrow max$$

где n - количество объектов в узле до разбиения;

$n_{лев}$ и $n_{прав}$ - количество объектов, попавших в левую и правую ветви соответственно.

После того как все деревья обучены, предсказание для нового объекта формируется одним из следующих способов:

- Голосование большинством (hard voting) - выбирается класс, который в качестве ответа определили большинство деревьев.
- Взвешенное голосование (soft voting) - класс выбирается с учетом вероятностей, которые вычисляются как средние доли голосов за каждый класс по всем деревьям.

2.5 Метод опорных векторов (SVM)

Метод опорных векторов (Support Vector Machine, SVM) — это алгоритм машинного обучения, предназначенный для задач бинарной классификации. Основная идея метода - нахождение оптимальной разделяющей гиперплоскости, которая максимизирует зазор между классами. Зазор определяется как расстояние между гиперплоскостью и ближайшими к ней точками данных — опорными векторами, принадлежащими разным классам.

Для линейно неразделимых данных SVM использует kernel trick, который позволяет отобразить данные в пространство более высокой размерности, где они становятся линейно разделимыми. Одним из популярных ядер является RBF-ядро (Radial Basis Function):

$$K(x_1, x_2) = exp(-\gamma \cdot ||x_1 - x_2||^2)$$

где γ — параметр, контролирующий "ширину" ядра;

x_1, x_2 - два объекта, которые являются векторами признаков.

Большое значение γ делает ядро более узким, усиливая локальные различия между объектами, а малое γ увеличивает обобщающую способность модели, делая ядро более широким.

При обучении также задается степень жесткости C. Большее значение параметра означает, что модель старается минимизировать ошибки даже за счёт риска переобучения. Меньшее значение делает модель более устойчивой к выбросам, но может быть недообученной.

Для поиска оптимальных коэффициентов Лагранжа SVM решает задачу вида:

$$max_a(\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)), 0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

где α_i — коэффициенты Лагранжа, определяющие опорные вектора;

$y_i \in \{-1, 1\}$ - значения, соответствующие двум классам.

После нахождения α_i SVM использует следующую функцию для предсказания класса нового объекта x:

$$f(x) = \sum_i a_i y_i K(x_i, x) + b, b = \frac{1}{|S|}\sum_{i \in S} (y_i - \sum_{j=1}^{n} a_j y_j K(x_i, x_j))$$

где S - это множество индексов i, соответствующих опорным векторам в наборе данных.

Функция $f(x)$ возвращает числовое значение, которое интерпретируется для классификации: если $f(x) \geq 0$, объект классифицируется как класс 1, если $f(x) < 0$, то как класс 0.

3. Используемые программные средства

Для реализации проекта использовалась облачная среда разработки Google Colab с поддержкой Jupyter Notebook. Язык программирования — Python (версия 3.11). Основные библиотеки:

- NumPy и Pandas - для обработки и анализа данных.
- Scikit-learn - для реализации алгоритмов машинного обучения (RandomForestClassifier, SVC, StandardScaler, train_test_split, метрики accuracy_score, confusion_matrix).
- Seaborn и Matplotlib - для визуализации данных и построения корреляционных матриц.
- Collections (Counter) - для подсчета классов и голосования в кастомной реализации Random Forest.
- CVXPY - для решения задачи оптимизации в кастомной реализации SVM с RBF-ядром.

Ссылка на проект:

https://colab.research.google.com/drive/188E9HxHCqooYBiK83WMQRmBYu9nnpc2N?usp=sharing

В качестве источника данных использовался открытый датасет, взятый с платформы Kaggle https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

4. Результаты

Со всеми графиками и реализациями функций, описанными в данном разделе, можно ознакомится на Google Colab.

4.1 Предварительный анализ и обработка данных

Перед тем, как реализовывать и сравнивать методы бинарной классификации, мы провели предварительный анализ и подготовку данных.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980

Рис. 4.1.1 - Фрагмент таблицы значений признаков для первых 5 элементов датасета

На рисунке 4.1.1 приведен фрагмент таблицы значений признаков для первых 5 элементов датасета. Колонка diagnosis содержит целевой класс: B - доброкачественная опухоль или M - злокачественная опухоль.

В задаче бинарной классификации:

1. id - не используется, т.к. является уникальным параметром.
2. Unnamed: 32 - не используется, т.к. не содержит данных.

Признаки, с которыми мы работали в дальнейшем: *radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean, radius_se, texture_se, perimeter_se, area_se, smoothness_se, compactness_se, concavity_se, concave points_se, symmetry_se, fractal_dimension_se, radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst, fractal_dimension_worst*.

Дополнительно мы проверили, что в оставшихся элементах датасета отсутствуют NaN значения или экстремальные выбросы. На рисунке 4.1.2 приведен фрагмент таблицы, содержащей набор статистических характеристик для каждого из признаков:

- count: Количество ненулевых значений для каждого признака.
- mean: Среднее арифметическое значение признака по всем объектам.

- std: Стандартное отклонение.
- min: Минимальное значение признака в наборе данных.
- 25%: Значение, ниже которого находится 25% данных.
- 50%: Медиана.
- 75%: Значение, ниже которого находится 75% данных.
- max: Максимальное значение признака в наборе данных.

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800

Рис. 4.1.2 - Фрагмент таблицы значений статистических характеристик для признаков

Дальнейший поиск и очистка данных от выбросов не производилась, т.к. мы не обладаем достаточными медицинскими знаниями для их определения. Таким образом мы предполагаем, что в используемом датасете нет выбросов.

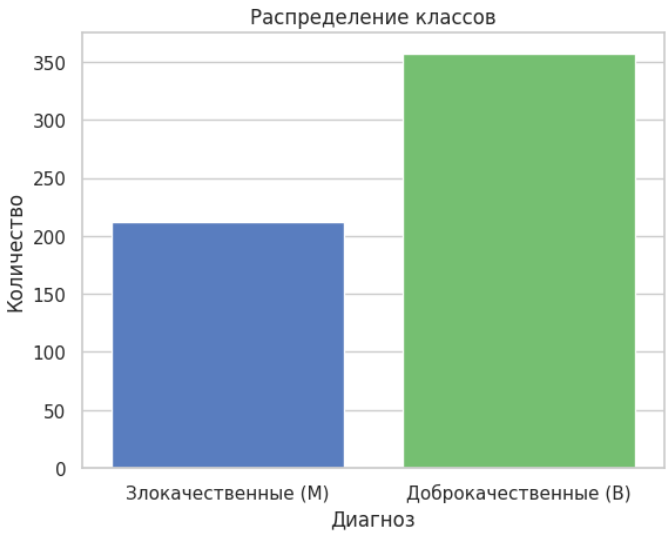

Рис. 4.1.3 - Гистограмма распределения данных по классам

Гистограмма распределения по классам (рис. 4.1.3) показывает, что злокачественные образования составляют около 40% от общего числа (212 против 357 доброкачественных), что достаточно для анализа.

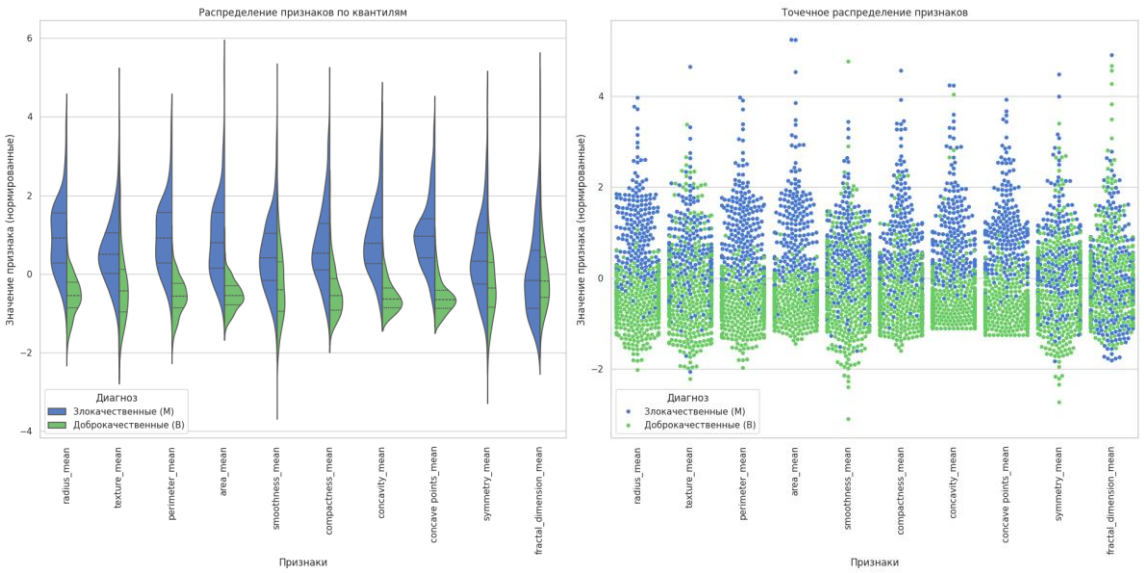

Рис. 4.1.4 - Скрипичная и роевая диаграммы для первых 10 признаков

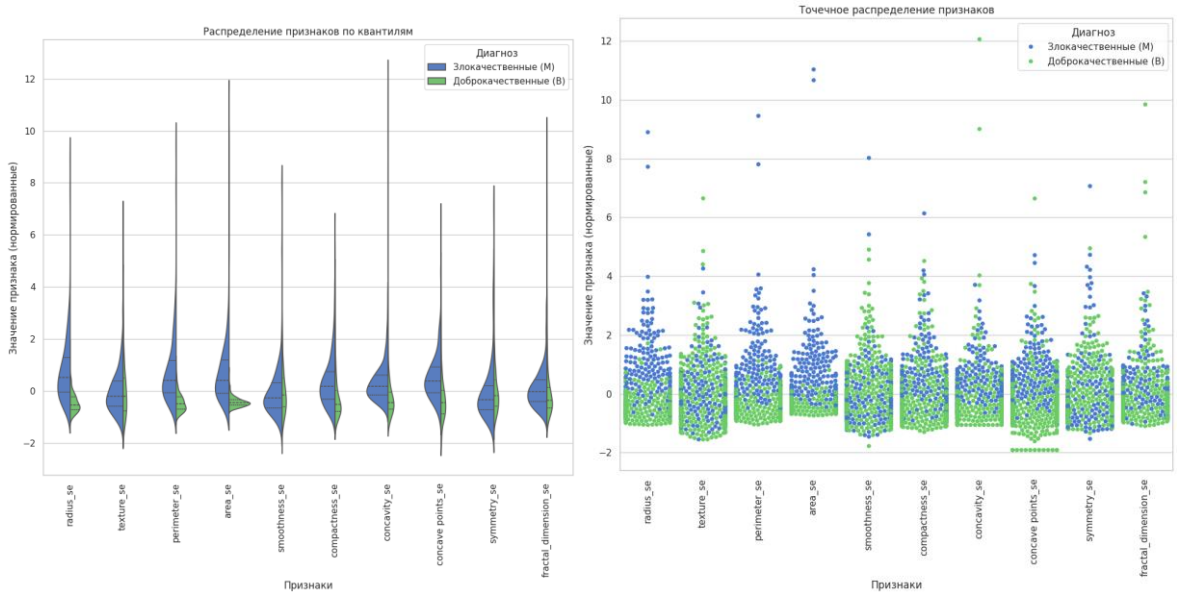

рис. 4.1.5 - Скрипичная и роевая диаграммы для признаков 11 - 20

Далее мы перешли к анализу распределения данных в каждом классе. Для наглядности были построены скрипичная и роевая диаграммы по каждому из нормированных признаков (см. рисунки 4.1.4 - 4.1.6).

- Скрипичная диаграмма помогает определить, какое значение принимает каждый признак с наибольшей частотой для каждого класса.
- Роевая диаграмма помогает определить, для каких признаков происходит более четкое разделение параметров.

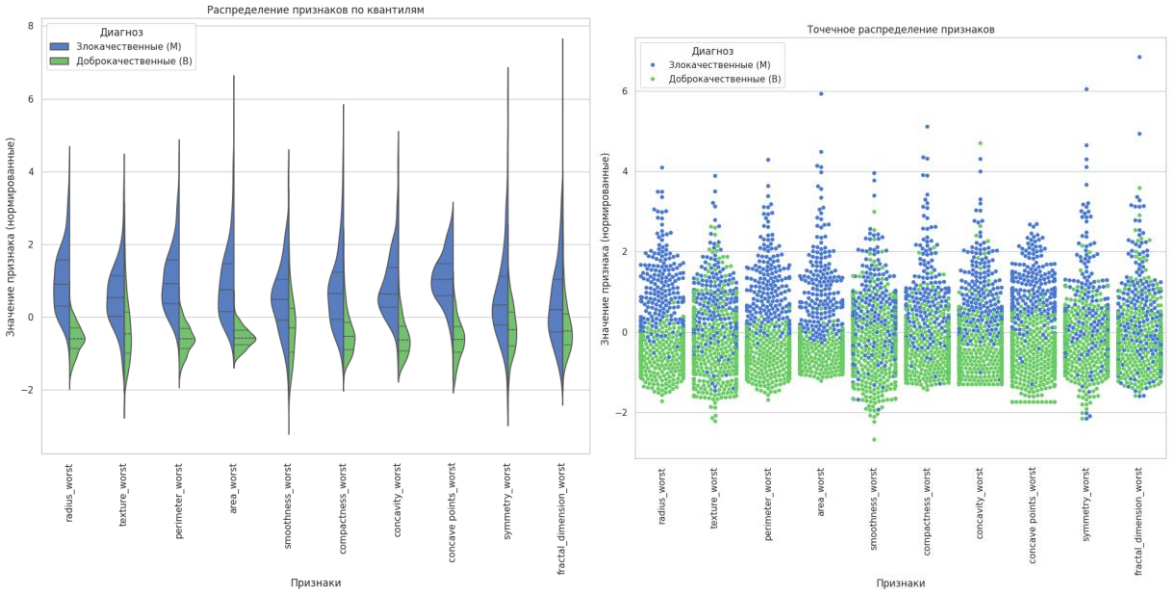

Рис. 4.1.6 - Скрипичная и роевая диаграммы для признаков 21 - 30

Для оптимизации набора признаков и снижения избыточности данных была построена корреляционная матрица (рисунок 4.1.7), которая выявляет линейно зависимые признаки (значение корреляции 1.0 или -1.0). В каждой группе линейно зависимых признаков был оставлен один представитель. Выбор признаков проводился на основе анализа скрипичных и роевых графиков. Приоритет отдавался признакам с наиболее четким разделением между классами.

В результате были отобраны следующие признаки:

1. radius_mean, perimeter_mean, area_mean → **area_mean**
2. compactness_mean, concavity_mean, concave points_mean → **concavity_mean**
3. radius_se, perimeter_se, area_se → **area_se**
4. radius_worst, perimeter_worst, area_worst → **area_worst**
5. compactness_worst, concavity_worst, concave points_worst → **concavity_worst**
6. compactness_se, concavity_se, concave points_se → **concavity_se**
7. texture_mean, texture_worst → **texture_mean**
8. area_worst и area_mean → **area_mean**

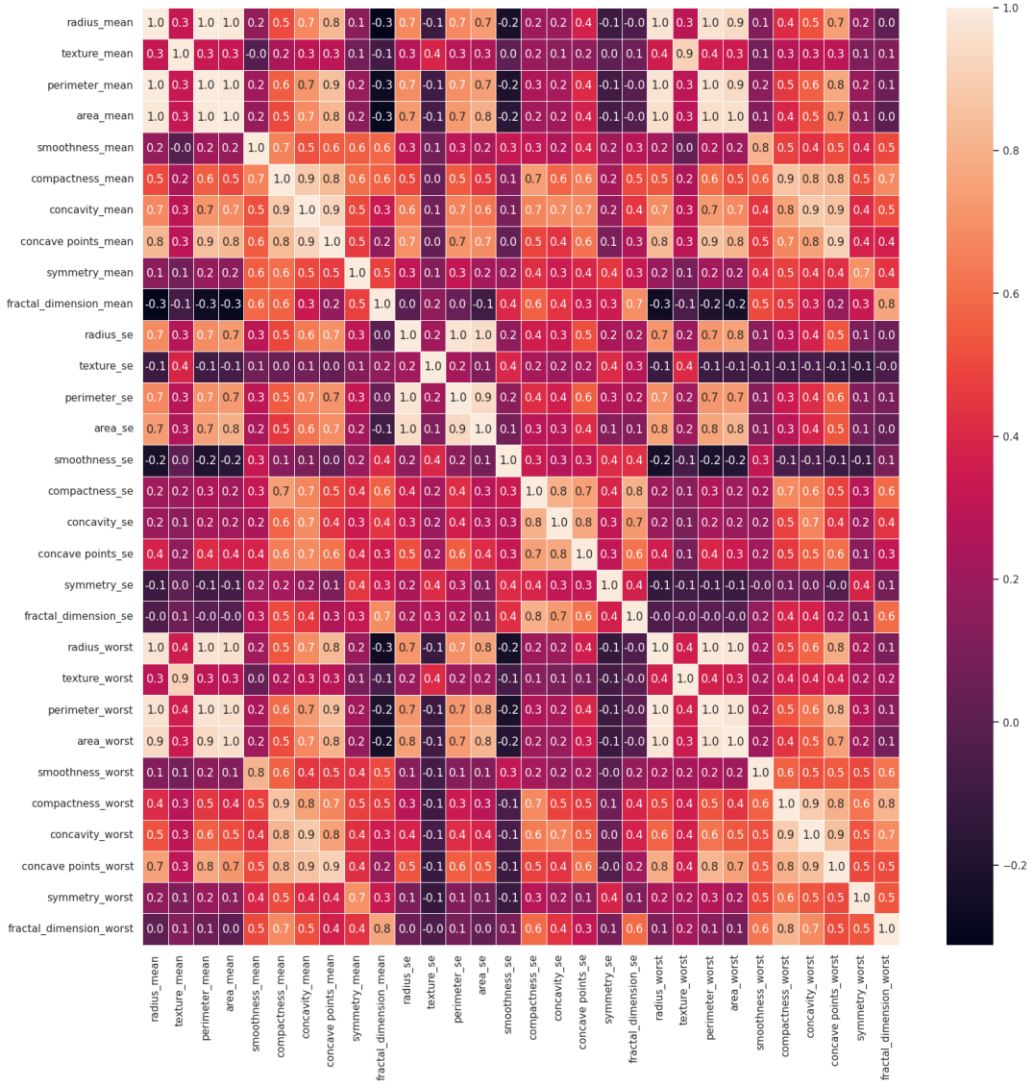

Рис. 4.1.7 - Корреляционная матрица для полного набора признаков

Корреляционная матрица для сокращенного набора признаков изображена на рисунке 4.1.8. Корреляция признаков значительно уменьшилась (максимально значение корреляции вне главной диагонали - 0.9, т.е. зависимость признаков не полная). Далее этот набор считается полным набором признаков.

Некоторые из реализованных нами методы классификации лучше работают с нормированными данными, поэтому в дальнейшем мы использовали их. Данные были разделены на тестовую и обучающую выборку в соотношении 30% / 70%. Для воспроизводимости во всех методах со случайными и при разделении использовался общий random state, равный 9.

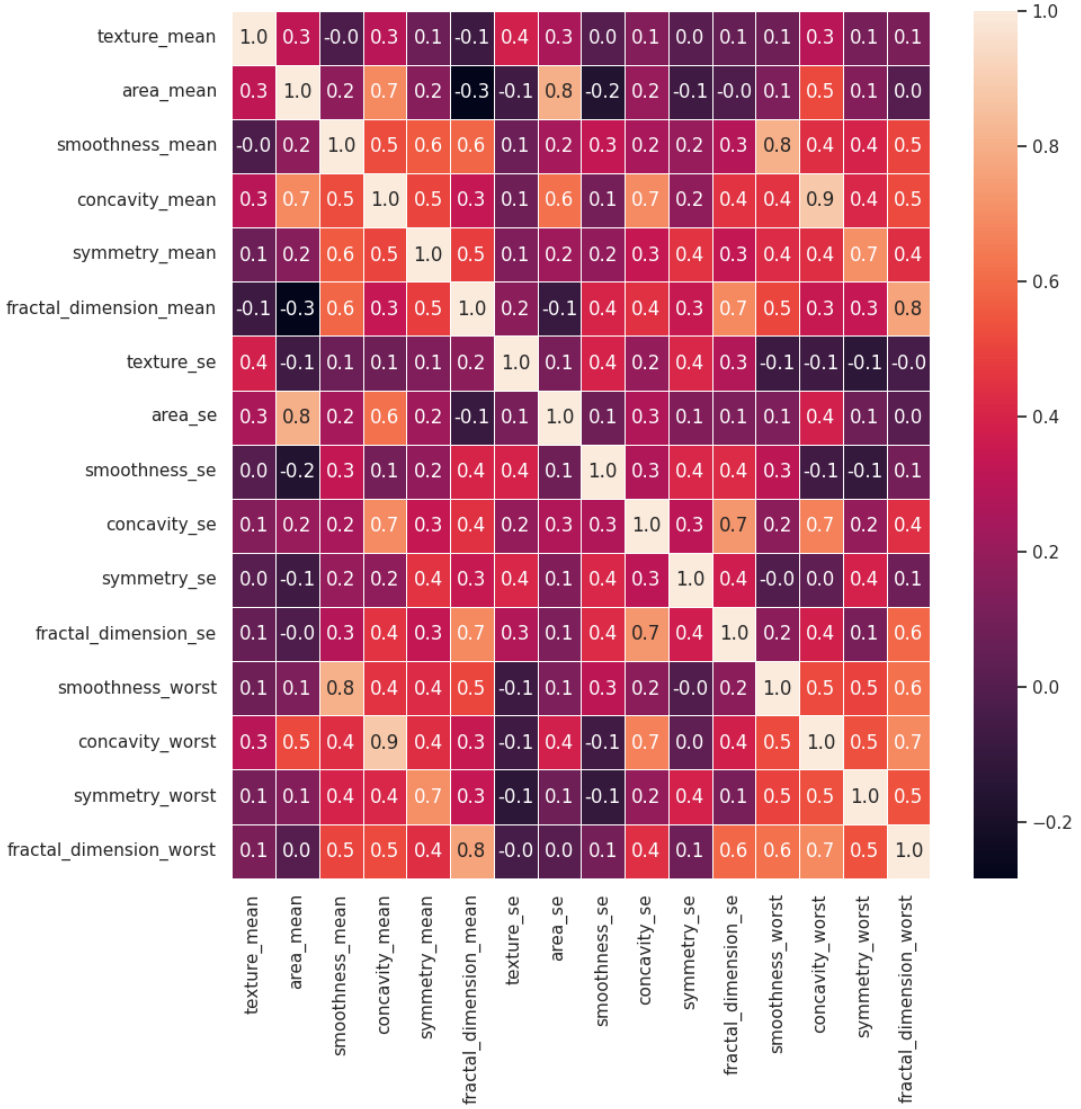

Рис. 4.1.8 - Корреляционная матрица для сокращенного набора признаков

4.2 Статистические методы

4.2.1 Логистическая регрессия

Для реализации метода логистической регрессии был использован алгоритм, описанный в теоретической части, и методы для работы с линейной алгебры из библиотеки NumPy. На рисунке 4.2.1.1 представшена матрица ошибок классификации модели. Так модель классифицировала:

- Количество объектов класса 'M', правильно классифицированных как 'M': 59
- Количество объектов класса 'B', правильно классифицированных как 'B': 109
- Количество объектов класса 'B', ошибочно классифицированных как 'M': 0
- Количество объектов класса 'M', ошибочно классифицированных как 'B': 3

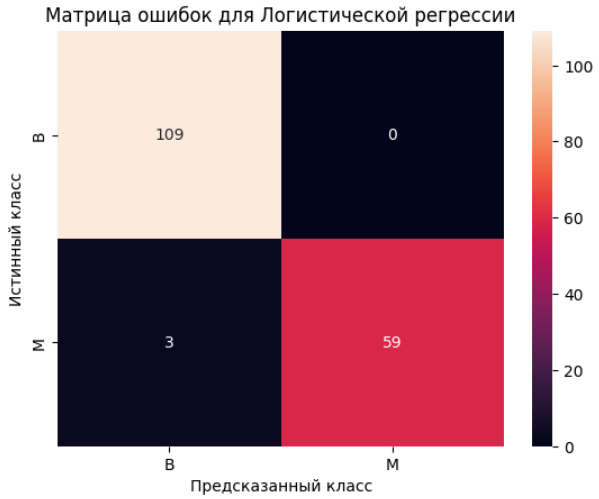

Рис. 4.2.1.1 - Матрица ошибок для логистической регрессии

Точность работы метода составляет 98.2%. Таким образом логистическая регрессия хорошо справляется с задачей бинарной классификации на основе имеющихся признаков. Параметры скорости обучения и количество итераций также были подобраны оптимально.

Ложноположительные и ложноотрицательные результаты могут быть следствием частичного перекрытия классов, затрудняющих линейное разделение.

4.2.2 Линейный дискриминантный анализ (LDA)

Прежде чем реализовывать линейный дискриминантный анализ необходимо было оценить ковариационные матрицы для каждого класса. При сравнении матриц, представленных на рисунке 4.2.2.1, кажется, что они похожи, однако статистический тест Бокса, показывает, что матрицы различны. Это может повлиять на точность классификации.

Одно из условий применения методов дискриминантного анализа — нормальное распределение признаков в каждом классе. Проведя тест Шапиро-Уилка, установлено, что большинство признаков в выборке для каждого класса не соответствуют нормальному распределению. Это также может повлиять на точность классификации.

Для реализации LDA аналогично был использован алгоритм, описанный в теоретической части, и методы для работы с линейной алгебры из библиотеки NumPy. На рисунке 4.2.2.1 представшена матрица ошибок классификации модели.

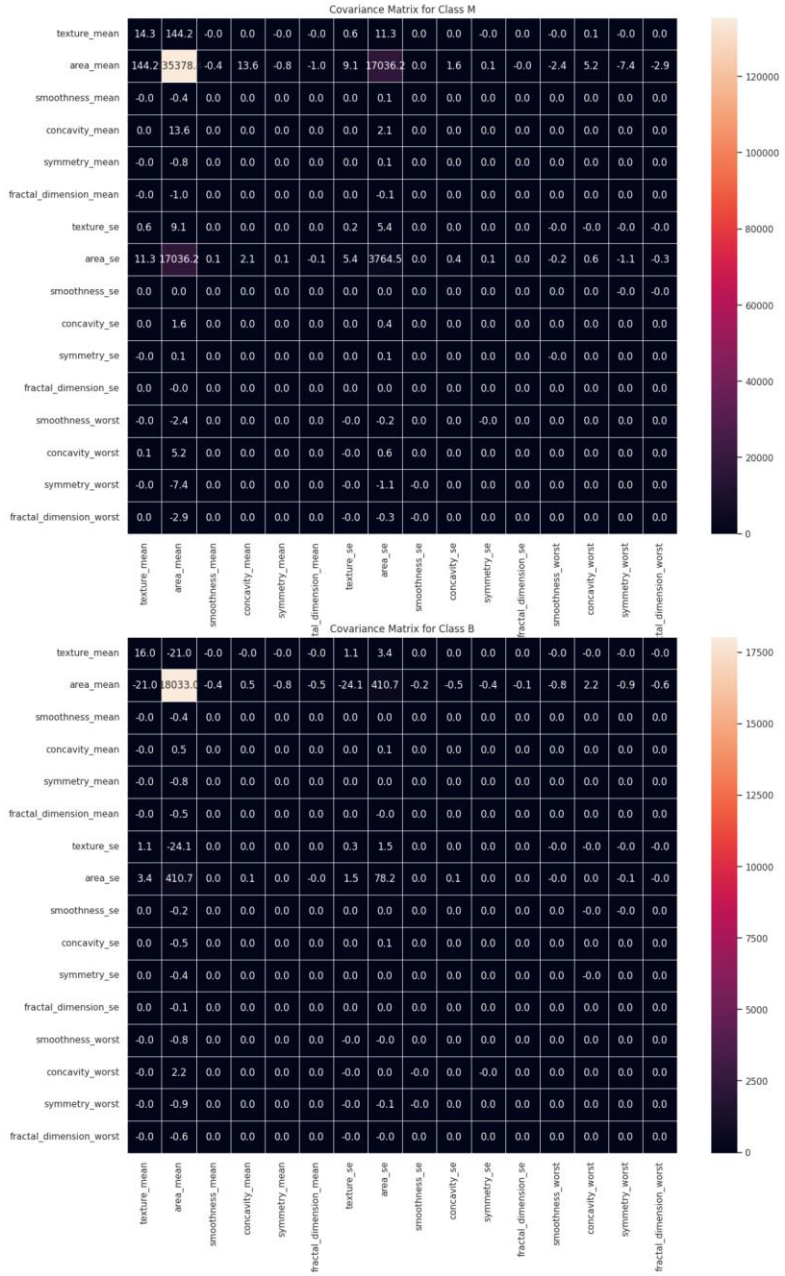

Рис. 4.2.2.1 - Ковариационные матрицы классов

Точность модели составляет 94.7%. Это меньше, чем у логистической регрессии и связано с различием ковариационных матриц, описанным в начале раздела, а также с несоответствием признаков классов нормальному распределению. Однако, метод выигрывает при работе с большим числом классов и не требует дополнительной настройки параметров (скорость обучения, количество итераций и т.д.).

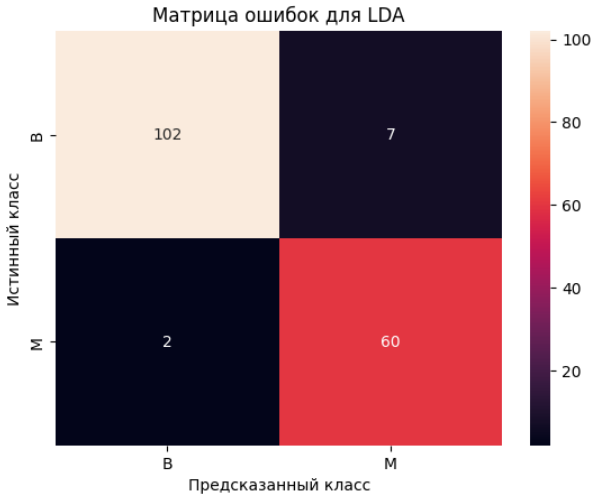

Рис. 4.2.2.2 - Матрица ошибок для линейного дискриминантного анализа

4.2.3 Квадратичный дискриминантный анализ (QDA)

Для реализации QDA аналогично был использован алгоритм, описанный в теоретической части, и методы из библиотеки NumPy. На рисунке 4.2.3.1 представшена матрица ошибок классификации модели. Точность метода составляет 96.5%.

Точность модели оказалась выше, чем у LDA, следовательно метод смог эффективно учесть различия ковариационных матриц для построения более точной границы классов. Однако несоответствие признаков классов нормальному распределению не позволило добится большей точности.

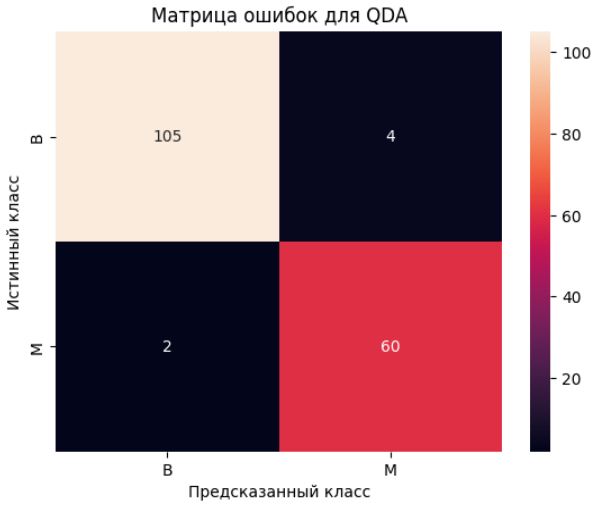

Рис. 4.2.3.1 - Матрица ошибок для квадратичного дискриминантного анализа

4.3 Машинное обучение

4.3.1 Случайный лес (Random Forest)

Для первоначальной оценки эффективности случайного леса был использован стандартный класс RandomForestClassifier из библиотеки scikit-learn (импортируется как sklearn). Модель была обучена с параметрами по умолчанию: количество деревьев - 100, глубина деревьев не ограничена.

Точность обученной модели на тестовой выборке составила 97%. Матрица ошибок представлена на рисунке 4.3.1.1.

Модель хорошо справляется с классификацией как доброкачественных, так и злокачественных опухолей. Таким образом, реализация "из коробки" обеспечивает отличные результаты при минимальных усилиях, что делает ее удобным решением для быстрого старта.

Рис. 4.3.1.1 - Матрица ошибок для Random Forest Classifaer

С целью уменьшения размерности пространства признаков был проведен анализ значимости признаков с функцией оценки хи-квадрат. Было выбрано 7 наиболее информативных: area_mean, area_se, texture_mean, concavity_worst, concavity_mean, symmetry_worst, concavity_se.

После повторного обучения модели на сокращенном наборе признаков точность не изменилась. Следовательно, удаление не привело к существенному снижению качества классификации. Уменьшение числа признаков также способствует ускорению работы модели и снижению риска переобучения.

Матрица ошибок, представленная на рисунке 4.3.1.2, совпадает с матрицей ошибок для полного набора признаков. Однако это можно считать особенностью конкретного random state и матрица ошибок может отличаться.

Матрица ошибок после фильтрации признаков сохранила устойчивость в предсказаниях, что подтверждает эффективность подхода к отбору признаков на основе их значимости.

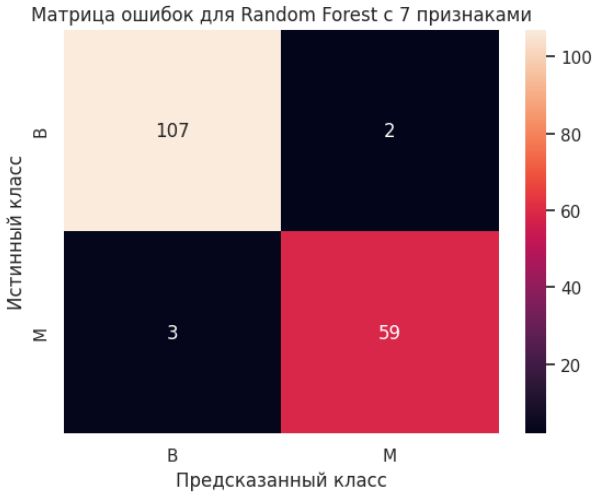

Рис. 4.3.1.2 - Матрица ошибок для Random Forest Classifaer с 7 признаками

Также для метода случайного леса была написана собственная реализация алгоритма из пункта 2.4 на чистом Python. Для упрощения написания кода:

- Не использовалась параллелизация или оптимизация через сторонние библиотеки (например, Cython, применяемый в scikit-learn).
- Все деревья имели ограничение на максимальное количество уровней разбиения, равное 10.
- Финальный прогноз строился с помощью голосования большинством. (в то время как scikit-learn использует вариацию взвешенного голосования)

Точность модели составила 97%, что соответствует точности, достигнутой с помощью scikit-learn. Следовательно, даже базовая, неоптимизированная версия алгоритма показывает высокое качество классификации. Матрица ошибок собственной реализации представлена на рисунке 4.3.1.3.

Однако, стоит отметить, что упрощения привели к существенному увеличению времени обучения даже на небольшом наборе данных.

Рис. 4.3.1.3 - Матрица ошибок для собственной реализации Random Forest

4.3.2 Метод опорных векторов (SVM)

Для первоначальной оценки эффективности метода опорных векторов была использована стандартная реализация SVC из библиотеки scikit-learn (импортируется как sklearn). Модель обучалась на нормализованных данных с заданными параметрами: C = 10, gamma = 0.01, ядро - RBF, взвешивание классов - balanced.

Рис. 4.3.2.1 - Матрица ошибок для SVM

Точность модели на тестовой выборке составила 98.2% . Матрица ошибок, представленная на рисунке 4.3.2.1. Модель показывает высокую точность, однако обучение производилось несколько раз с разными параметрами для поиска оптимальных.

Уменьшение количества используемых признаков для SVM возможно, но в данном случае используется нелинейное ядро RBF, что не позволяет свести задачу к рассмотрению весов вектора w. Следовательно, для оценки значимости каждого признака требуется градиентный анализ. Это является отдельной вычислительно задачей и не рассматривается далее.

Также для метода опорных векторов была написана собственная реализация алгоритма из пункта 2.5 на чистом Python. Для упрощения и ускорения разработки были приняты следующие допущения:

- Реализация поддерживает только RBF-ядро, без возможности использования других типов.
- Модель требует предварительной нормализации данных, так как не содержит встроенной обработки масштаба признаков.
- Для решения задачи оптимизации используется библиотека cvxpy, вместо LIBSVM.
- Отсутствует поддержка параллелизма

Точность собственной реализации составила 94.7%, что ниже значения, достигнутого с помощью scikit-learn. К тому же модель обучалась дольше, в сравнении с остальными реализациями.

Матрица ошибок метода представлена на рисунке 4.3.2.2.

Рис. 4.3.2.2 - Матрица ошибок для собственной реализации SVM

5. Обсуждение и выводы

5.1 Сравнение методов статистического анализа

Логистическая регрессия не делает предположений о распределении данных, что делает ее более точной для датасета, где данные не соответствуют строгому нормальному распределению.

Логистическая регрессия и LDA формируют линейные разделяющие поверхности между классами, однако логистическая регрессия оптимизирует вероятности напрямую, что обеспечивает ей большую устойчивость на нашем датасете. QDA создает более сложные границы, что может быть выгодно при различиях в корреляциях, но в данном случае такой подход оказался избыточным и менее эффективным.

Логистическая регрессия требует меньше всего параметров и вычислительных ресурсов, что позволяет использовать ее для задач с большим числом признаков. Полное сравнение методов по ресурсам и параметром представлено в таблице 5.1.1.

Таблица 5.1.1 - Сравнение числа параметров и вычислительных ресурсов для статистического анализа

Метод	**Число параметров**	**Вычислительные ресурсы**
Логистическая регрессия	$p + 1$ (число признаков p и один свободный член)	Низкие: градиентный спуск
LDA	$C * p + \frac{p*(p+1)}{2} + C$ (средние для каждого класса, ковариационная матрица и априорные вероятностей)	Средние: требуется инверсия ковариационной матрицы
QDA	$C * p + C * \frac{p*(p+1)}{2} + C$ (средние для каждого класса, ковариационные матрицы классов и априорные вероятностей)	Высокие: требуются инверсии ковариационных матриц каждого класса

Таким образом логистическая регрессия превосходит LDA и QDA на выбранном датасете. Хотя QDA учитывает разные ковариации классов, модель неоправданно сложна для данной задачи, а LDA уступает в гибкости, так как требует определенного распределения данных. Таким образом логистическая регрессия является

предпочтительным вариантом благодаря ее устойчивости, интерпретируемости и высокой производительности.

5.2 Сравнение методов машинного обучения

Случайный лес из scikit-learn продемонстрировал хорошую точность классификации, дополнительное изменение параметров не потребовалась. Для SVM понадобилось перебрать несколько наборов параметров, но это значительно повысило точность. Дополнительно RBF ядро сильно усложняет модель и отбор признаков (это необходимо, так как SVM дольше работает с большим количеством признаков).

Собственная реализация для Random Forest в несколько раз проще и не теряет точность в отличии от аналогичной реализация SVM, которая уступает по времени обучения и используемым ресурсам.

Метод случайного леса позволяет легко оценить важность признаков и проследить, как отдельные деревья принимают решения. Метод опорных векторов является "чёрным ящиком" с точки зрения внутренней логики, решающее правило не поддается простой интерпретации из-за нелинейности модели.

Если важна точность, число признаков небольшое или граница между классами четко выражена - стоит выбрать SVM, обучив модель с разными ядрами и параметрами C и γ, и отобрав наилучший набор для конкретных данных.

Если в данных большое число признаков, и может потребоваться отбор некоторых из них или оптимизация времени обучения - оптимальным решением будет Random forest, балансирующий между точностью и простотой реализации и поддерживающий параллельное обучение деревьев (что повышает масштабируемость).

Таблица 5.2.1 - Сравнение различных признаков машинного обучения

	Random Forest	**Cast. Random Forest**	**SVM**	**Cast. SVM**
Точность	97%	97%	98.2%	94.7%
Время обучения	Низкое	Высокое	Среднее	Высокое
Сложность реализации	Низкая	Средняя	Низкая	Высокая

Масштабируемость	Высокая	Низкая
Устойчивость к выбросам	Высокая	Средняя
Интерпретируемость	Прозрачный процесс	Черный ящик, реализующий разделение гиперплоскостью

5.3 Общее сравнение

В предыдущих пунктах мы уже сравнили модели внутри их типов и отобрали те, которые лучше всего подходят для классификации выбранного датасета. Из статистических методов была выбрана логистическая регрессия, а из методов машинного обучения подходят оба, так как их точность сопоставима, и каждый обладает уникальными преимуществами (точность, простота реализации, гибкость). Предполагается, что все методы ML максимально оптимизированы (т.е. сравнение идет с реализацией из scikit-learn).

SVM и логистическая регрессия используют линейную гиперплоскость для разделения классов, но их подходы различны:

- В SVM гиперплоскость оптимизируется для максимального зазора между классами, что делает ее чувствительной к выбросам, находящимся близко к границе классов, однако параметр жесткости C позволяет контролировать их влияние.
- В логистической регрессии гиперплоскость формируется через минимизации логарифмических потерь, учитывая все точки, из-за чего особенно чувствительна к выбросам, находящимся далеко от разделяющей гиперплоскости.

Фокусом логистической регрессии является вероятностное моделирование, в то время как SVM, напротив, смотрит на геометрическое расположение классов в пространстве, что позволяет ей создавать не только линейные границы классов, но и нелинейные. Random Forest - нелинейная ансамблевой модель, которая создает множество деревьев решений, агрегируя их предсказания, что позволяет выявлять сложные нелинейные зависимости в данных.

В плане интерпретируемости логистическая регрессия отличается высокой прозрачностью, так как ее веса напрямую показывают вклад признаков в результат. SVM интерпретируемая при использовании линейного ядра, где веса также отражают влияние признаков, но нелинейные ядра снижают эту прозрачность из-за сложного преобразования данных. Random Forest обладает высокой интерпретируемостью, так как важность признаков можно оценить с помощью функции хи-квадрат.

Что касается требования к предварительной обработке данных, логистическая регрессия не требует нормированности, но нормализация признаков улучшает сходимость градиентного спуска. SVM чувствительна к масштабу данных, особенно при использовании нелинейных ядер, таких как RBF, что делает нормализацию необходимой. Random Forest, напротив, не зависит от масштаба или распределения признаков, так как опирается на пороговые правила разделения.

Таблица 5.3.1 - Сравнение различных признаков Логистической регрессии, SVM и Ramdon forest

	Логистическая регрессия	**SVM**	**Random Forest**
Тип модели	Линейная, вероятностная	Линейная или нелинейная, дискриминативная	Нелинейная, ансамблевая
Принцип работы	Моделирует вероятность класса через сигмоиду, оптимизируя логаримф правдоподобия	Максимизирует зазор между классами, минимизируя ошибки на границе	Строит ансамбль деревьев решений с агрегацией предсказаний
Фокус модели	Вероятности принадлежности к классу	Геометрическое разделение классов	Нелинейные зависимости через ансамбль деревьев
Интерпретируемость	Высокая (веса показывают влияния признаков)	Высокая -для линейных ядер Низкая - для нелинейных	Высокая (легко получить веса признаков и оценить деревья)
Требование предобработки данных	Лучше работает с нормализованными данными	Лучше работает с нормализованными данным	Не чувствителен к масштабу или распределению признаков

Обработка выбросов	Чувствительна	Чувствительна	Устойчива благодаря ансамблю
Скорость обучения	Быстрая (градиентный спуск)	Медленная для больших выборок (квадратичная сложность)	Зависит от максимальной глубины деревьев
Точность	98.2%	98.2%	97%

Все три метода показали высокую точность классификации, что может быть связано с хорошей линейной разделимостью классов в датасете. При этом можно отметить следующее:

- Логистическая регрессия, как статистический метод, работает предсказуемо, дает понятные веса признаков и результаты обучения, не зависящие от случайных состояний.
- SVM и Random Forest, относящиеся к методам машинного обучения, демонстрируют чуть более высокую точность на границах классов, но их результаты могут отличаться при разных запусках, например, из-за случайного выбора признаков при обучении.

После анализа характеристик и особенностей логистической регрессии, SVM и Random Forest в контексте диагностики рака молочной железы, сложно однозначно выделить лучшую модель без конкретной задачи и понимания, где в дальнейшем будет использоваться метод, так как каждый метод имеет свои сильные и слабые стороны.

5.4 Дальнейшие исследования

Далее можно изучить способы объединения преимуществ статистических методов и методов машинного обучения. Такое объединение может быть получено с помощью методов построения ансамблей (бэггинг, бустинг и стекинг).

Можно также провести дополнительный анализ и трансформацию данных к нормальному распределению (логарифмирование, преобразование Бокса-Кокса или преобразование Yeo-Johnson) для улучшения работы дискриминантных методов.

Возможна реализация градиентного анализа для нелинейного ядра SVM и анализ изменений скорости обучения модели и точности классификации после отбрасывания признаков.