

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б10-мм

Разработка медицинского инструмента для классификации заболеваний щитовидной ЖЕЛЕЗЫ

Кушнир Артем Юрьевич

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Гориховский В.И.

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	5
2. Обзор требований к инструменту	6
2.1. Метрики	7
2.2. Обзор используемых технологий	8
2.3. Выводы	8
3. Описание решения	9
3.1. Feature Engineering Layer	9
3.2. DataAdapter	11
3.3. Models	12
3.4. Tuners	14
3.5. TrainingSystem	15
3.6. InferenceSystem	15
4. Эксперимент	17
4.1. Дизайн эксперимента	17
4.2. Условия эксперимента	17
4.3. Исследовательские вопросы	18
4.4. Метрики	18
4.5. Результаты	18
4.6. Обсуждение результатов	19
Заключение	21
Список литературы	22

Введение

В современном мире онкологические заболевания и различные виды новообразований представляют собой одну из наиболее актуальных проблем здравоохранения. Значительное число людей ежегодно сталкивается с онкологической патологией [3], и своевременное выявление этих заболеваний играет ключевую роль в эффективности лечения. Многие новообразования на ранней стадии не представляют серьёзной угрозы и могут быть успешно устранены, однако поздняя диагностика значительно снижает шансы на выздоровление.

Одним из наиболее распространённых и эффективных методов диагностики в онкологии являются визуализирующие методы обследования, такие как рентгенография, радионуклидная визуализация, ультразвуковое исследование (УЗИ), компьютерная томография (КТ) и магнитно-резонансная томография (МРТ). Несмотря на широкое распространение этих методов, качество диагностики во многом зависит от квалификации и внимательности врача. Человеческий фактор, утомляемость и ограниченный опыт пока что нерешенные проблемы, они могут привести к диагностическим ошибкам, которые могут стоить пациенту здоровья или жизни.

В осеннем семестре была проведена исследовательская работа [12], направленная на изучение различных алгоритмов машинного обучения для классификации патологий щитовидной железы на основе ультразвуковых изображений [12]. Результаты продемонстрировали высокую эффективность применённых подходов: полученные модели показали высокие значения таких метрик, как точность, полнота и F1-мера. Однако данное исследование носит предварительный характер и не предназначено для практического применения в широком масштабе. Его следует рассматривать как подтверждение целесообразности дальнейшего развития данного направления.

В продолжение этой работы в рамках весенней практики была поставлена задача разработки прототипа медицинского инструмента, предназначенного для автоматической обработки ультразвуковых изображений

и классификации заболеваний, в первую очередь патологий щитовидной железы, с использованием методов машинного обучения.

1. Постановка задачи

Целью данной работы является разработка медицинского программного инструмента для классификации заболеваний на основе медицинских изображений. В рамках проекта предусмотрены два основных направления: обработка изображений и выделение информативных признаков, построение и обучение моделей машинного обучения. Первой частью занимается Сазонова Ирина, а данная работа сосредоточена на втором направлении. Для его реализации были сформулированы следующие задачи:

1. Спроектировать архитектуру инструмента.
2. Реализовать модули инструмента, отвечающие за обучение и выполнение предсказаний моделей машинного обучения.
3. Разработать интерфейс для обучения моделей и использования инструмента.
4. Провести экспериментальное сравнение с результатами других статей.
5. Реализовать комплекс unit-тестов и end-to-end тестов.

2. Обзор требований к инструменту

В рамках осенней практики было проведено экспериментальное сравнение различных моделей машинного обучения, применяемых для классификации патологий щитовидной железы на основе ультразвуковых изображений, с использованием одного из открытых датасетов [5]. По итогам эксперимента было принято решение о разработке универсального медицинского инструмента, способного выполнять автоматическую классификацию заболеваний на основе визуальных данных медицинских обследований. Важно подчеркнуть, что планируемая система не ограничивается только патологией щитовидной железы, а ориентирована на расширяемость и возможность работы с различными типами медицинских изображений.

На основе анализа поставленных задач и проделанной работы были сформулированы следующие ключевые функциональные и нефункциональные требования к системе:

1. Поддержка различных типов медицинских изображений и широкого спектра заболеваний, что обеспечит универсальность и масштабируемость решения.
2. Модульная архитектура, допускающая гибкую замену и настройку компонентов, включая возможность переключения между различными классификаторами без необходимости переработки всей системы.
3. Обеспечить согласованность архитектуры с этапами обработки изображений и извлечения признаков.
4. Высокая точность и приемлемое время отклика, обеспечивающие практическую применимость инструмента в условиях клинической или исследовательской работы.
5. Полная автоматизация обработки, включающая все этапы: загрузку и предобработку данных, извлечение признаков, классификацию и визуализацию результатов.

6. Выделение подозрительных областей на изображениях, что позволит повысить интерпретируемость и доверие пользователя к результатам предсказания.

Дополнительно можно рассмотреть включение следующих требований:

- Возможность обучения новых моделей на пользовательских данных, включая удобный интерфейс загрузки и аннотации.
- Интерпретируемость предсказаний.
- Интеграция с медицинскими информационными системами.

2.1. Метрики

Были выбраны основные метрики, которые используются в задачах классификации опасных заболеваний.

- *Accuracy* — метрика, показывающая точность классификации, долю верно предсказанных объектов. Определяется формулой:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{где}$$

TP — True Positive, TN — True Negative, FP — False Positive, FN — False Negative.

- *Recall* — метрика, показывающая долю верно классифицированных изображений положительного класса среди всех объектов положительного класса. Определяется следующей формулой:

$$Recall = \frac{TP}{TP + FN}, \quad \text{где}$$

- F_1 -мера — метрика, которая объединяет *Precision* и *Recall*, считается их среднегармоническое. Важно ее также учитывать, так как, если у нас будет дисбаланс классов и модель всегда будет

предсказывать положительный класс, то модель будет не очень хорошей, что и отобразит эта метрика. Определяется формулой:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \quad \text{где}$$

2.2. Обзор используемых технологий

Для разработки медицинского инструмента, предназначенного для классификации заболеваний на основе медицинских изображений, был выбран язык программирования Python. Такой выбор обусловлен рядом весомых причин. Прежде всего, Python является одним из наиболее популярных языков в области анализа данных и машинного обучения благодаря своей простоте, читаемости и широкой поддержке сообщества.

Кроме того, для Python существует обширная экосистема специализированных библиотек, предназначенных для решения задач компьютерного зрения и машинного обучения. В частности, библиотеки `scikit-learn` [9], `PyTorch` [8], `XGBoost` [2], `CatBoost` [1], `LightGBM` [6] и `Optuna` [7] предоставляют эффективные реализации как классических, так и современных алгоритмов машинного обучения, а также инструменты для настройки и оптимизации гиперпараметров моделей.

Дополнительным преимуществом является наличие готовых модулей для загрузки, предобработки и аугментации изображений, что критически важно при работе с медицинскими данными. Все это делает Python оптимальным выбором для создания гибкого, масштабируемого и воспроизводимого инструмента для автоматизированного анализа медицинских изображений.

2.3. Выводы

Таким образом, формализованные требования создают основу для архитектурного проектирования и дальнейшей реализации прототипа инструмента.

3. Описание решения

На UML-диаграмме на рис. 1 представлена разработанная архитектура инструмента, синим цветом выделена моя часть работы. Основными компонентами разрабатываемого инструмента являются: TransformPipeline, TransformPlugin, DataAdapter, Storage, Model, Tuner, TrainSystem, InferenceSystem. Далее будут описаны детали реализации этих компонентов, а также то, за что они отвечают.

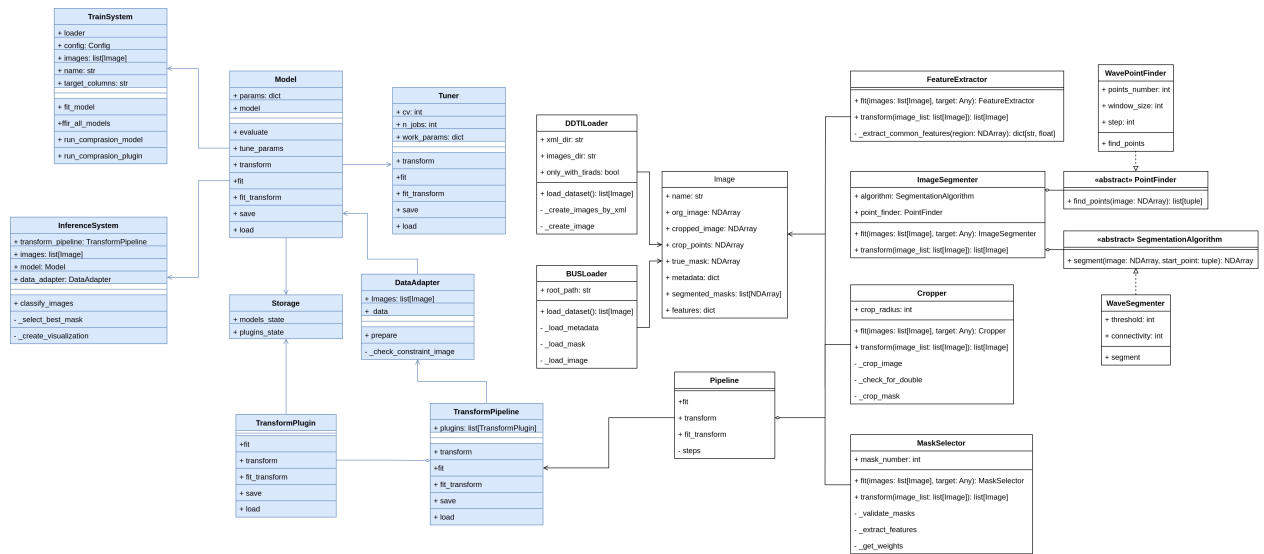


Рис. 1: архитектура реализованного инструмента

3.1. Feature Engineering Layer

Эта часть инструмента отвечает за выделения дополнительных признаков, отбора наиболее значимых признаков, а также за предобработку признаков выделенных из изображений.

Реализована система плагинов, которые можно включать в пайплайн обработки изображений. Всего было написано 10 плагинов:

- StandardScaler — стандартизация: из каждого признака вычитается среднее значение по признаку и делится на стандартное отклонение. Реализовано с помощью функции `sklearn.preprocessing.StandardScaler`.

- `RobustScaler` — масштабирование признаков с устойчивостью к выбросам: используется медиана и интерквартильный размах вместо среднего и стандартного отклонения. Реализовано с помощью `sklearn.preprocessing.RobustScaler`.
- `MinMaxScaler` — масштабирование признаков в заданный диапазон (по умолчанию от 0 до 1). Реализовано с помощью `sklearn.preprocessing.MinMaxScaler`.
- `SelectKBest` — выбор k лучших признаков на основе заданной статистики (например, `f_classif` или `mutual_info`). Реализовано с помощью `sklearn.feature_selection.SelectKBest`.
- `PCA` — понижение размерности с помощью метода главных компонент (Principal Component Analysis), позволяющего сохранить максимальную дисперсию данных в меньшем числе признаков. Реализовано с помощью `sklearn.decomposition.PCA`.
- `FillNan` — заполнение пропущенных значений (NaN) заданным способом, например, медианой, средним или константой.
- `OutlierRemoval` — удаление выбросов из данных с использованием метода Z-оценки.
- `MutualInfo` — отбор признаков по значению взаимной информации (mutual information) с целевой переменной.
- `KMeans` — кластеризация методом k -средних, применяется, например, для генерации новых признаков (кластерных меток, расстояние до центра своего кластера). Реализовано с помощью `sklearn.cluster.KMeans`.
- `FastICA` — независимый компонентный анализ (Independent Component Analysis) для выделения скрытых независимых источников в данных. Реализовано с помощью `sklearn.decomposition.FastICA`.

Дополнительно реализован модуль `PluginRegistry`, представляющий собой реестр всех доступных плагинов. Для регистрации нового плагина достаточно обернуть его в специальный декоратор, после чего он автоматически сохраняется в реестре и становится доступным по имени с использованием метода `get_plugin`. Такой подход обеспечивает гибкость расширения системы и упрощает интеграцию новых компонентов.

`TransformPipeline` — класс, предназначенный для последовательного применения различных плагинов к изображениям и их признаковому описанию. Предусмотрена возможность сохранения состояния всех плагинов, полученного на этапе обучения, что позволяет впоследствии использовать зафиксированные статистики при предсказании на тестовых данных или в процессе эксплуатации инструмента. Ключевыми методами класса являются: `fit(images)` — для вычисления и сохранения необходимой статистики каждого подключённого плагина, и `predict(images)` — для последовательного преобразования входных изображений на основе обученных плагинов.

3.2. DataAdapter

Данный модуль отвечает за преобразование изображений в формат, пригодный для последующей подачи в модели. Он реализует обёртку над исходными данными в виде абстракции `DataAdapter`. В основе реализации лежит базовый класс `BaseDataAdapter`, от которого наследуются конкретные адаптеры. На вход каждому адаптеру подаётся список изображений. Ключевой метод `prepare` осуществляет преобразование данных в требуемый формат, после чего результат становится доступен через атрибут `data`.

В настоящий момент реализованы следующие адаптеры:

- `SklearnDataAdapter` — адаптер для моделей, совместимых с библиотекой `scikit-learn`. Преобразует изображения в формат `pandas.DataFrame` [11].

- **ImageDataAdapter** — адаптер для моделей на основе PyTorch, работающих исключительно с изображениями. Формирует пользовательский датасет, унаследованный от `torch.utils.data.Dataset`.
- **PytorchHybridDataAdapter** — адаптер для моделей PyTorch, принимающих на вход как изображения, так и статистические признаки, извлечённые из них. Формирует специальный пользовательский датасет, содержащий как изображения, так и соответствующие векторные признаки.

3.3. Models

BaseModel представляет собой абстрактный базовый класс, от которого наследуются все остальные модели. Он определяет единый интерфейс, обязательный для реализации во всех производных классах: методы обучения, предсказания, сохранения и загрузки модели. Также реализован механизм автоматического подбора гиперпараметров на основе абстрактного класса **BaseTuner**. По умолчанию для обучения нейросетевых моделей используется функция потерь Focal Loss, зарекомендовавшая себя в задачах медицинской классификации, однако поддерживаются и альтернативные функции потерь, а также различные оптимизаторы.

Поддерживаются два основных типа моделей:

- **SklearnModel** — обёртка над моделями, реализованными в библиотеке `scikit-learn`.
- **PyTorchModel** — обёртка над нейросетевыми моделями, реализованными с использованием библиотеки PyTorch.

3.3.1. SklearnModel

Класс **SklearnModel** поддерживает любые модели, реализующие стандартный интерфейс `scikit-learn`. В текущей реализации доступны следующие алгоритмы:

- Support Vector Machine (SVM)
- Random Forest
- CatBoostClassifier
- XGBoostClassifier
- LGBMClassifier

3.3.2. PyTorchModel

Класс `PyTorchModel` включает в себя нейросетевые архитектуры, предназначенные как для обработки изображений (в частности, ультразвуковых), так и для работы с изображениями в сочетании со статистическими признаками, извлечёнными из них.

Реализованные сверточные нейронные сети:

- DenseNet
- EfficientNet
- ResNet
- SEResNet
- Swin Transformer

Кроме того, разработаны три гибридные модели, обрабатывающие одновременно визуальные данные и табличные признаки:

- **CNN-MLP** — гибридная архитектура, сочетающая сверточную нейронную сеть для анализа ультразвуковых изображений с многослойным перцептроном (MLP), обрабатывающим статистические признаки.
- **DenseNetHybrid** — модель на основе предобученной DenseNet, объединяющая визуальные и табличные признаки с применением механизма внимания.

- **EfficientNetHybrid** — модель, построенная на базе предобученной EfficientNet-B0, объединяет визуальные и статистические признаки с помощью механизма внимания.

3.4. Tuners

Механизм подбора гиперпараметров реализован через абстрактный базовый класс `BaseTuner`, от которого наследуются все конкретные реализации тюнеров. Этот класс задаёт единый интерфейс для процессов оптимизации как в классических моделях, так и в нейросетевых архитектурах.

Для моделей, основанных на `SklearnModel`, реализованы два метода подбора гиперпараметров:

- **Grid Search** — метод полного перебора гиперпараметров в заданном диапазоне, реализованный с использованием `sklearn.model_selection.GridSearchCV`.
- **Байесовская оптимизация** — реализована с использованием библиотеки `Optuna`; обеспечивает более эффективный поиск в пространстве гиперпараметров за счёт построения аппроксимации функции отклика.

Для нейросетевых моделей, представленных классом `PyTorchModel`, доступен только метод байесовской оптимизации на базе `Optuna`, поскольку он лучше подходит для высокоразмерных и дорогостоящих в вычислительном отношении моделей.

Дополнительно разработана подсистема `HyperparameterSelectionSystem`, представляющая собой набор вспомогательных классов-обёрток над конфигурациями гиперпараметров. Она обеспечивает удобный интерфейс для задания, сохранения и загрузки оптимальных конфигураций гиперпараметров. Конфигурация может быть загружена из внешнего хранилища на основе имени модели или передана непосредственно при инициализации тюнера.

3.5. TrainingSystem

Разработана система для обучения заданных моделей с возможностью сохранения результатов, а также автоматизированного проведения экспериментов. Поддерживаются два типа экспериментов:

- **Сравнение моделей** — проводится обучение заданного набора моделей. Для каждой модели можно указать, следует ли сохранять обученную модель и выполнять ли подбор гиперпараметров. В процессе эксперимента фиксируется время обучения, а также вычисляются ключевые метрики качества на тестовой выборке.
- **Сравнение плагинов** — выполняется обучение одной выбранной модели с различными наборами предобработки (плагинов). Для каждого набора вычисляются целевые метрики, что позволяет оценить влияние различных этапов предварительной обработки данных на итоговое качество модели.

Вся информация, связанная с обучением и экспериментами, логируется как в консоль, так и в лог-файл. По завершении эксперимента автоматически формируется подробный отчёт с метриками и визуализациями, включая графики, отражающие сравнительный анализ моделей или конфигураций предобработки.

3.6. InferenceSystem

Это модуль взаимодействия с пользователем, предоставляющий удобный интерфейс для классификации медицинских изображений. Пользователь может загрузить одно или несколько изображений, а также указать модель, которая должна быть использована для анализа.

После получения входных данных система автоматически загружает соответствующую предварительно обученную модель из хранилища и выполняет классификацию. По завершении анализа формируется подробный отчёт, включающий: исходное изображение; маску, выделяющую область, на которую ориентировалась модель; предсказанную метку класса и вероятность.

Данный модуль ориентирован на конечного пользователя и предназначен для упрощения процесса интерпретации результатов модели.

4. Эксперимент

4.1. Дизайн эксперимента

Необходимо убедиться, что разработанный медицинский инструмент пригоден как для практического применения, так и для объективного сравнения с существующими аналогами, включая проведение детального сравнительного эксперимента. Кроме того, важно оценить качество классификации, достигаемое разработанными гибридными моделями, которые используют при обучении и изображения, и выделенные признаки.

В рамках оценки будет проведён анализ результатов обучения всех доступных моделей машинного обучения с последующим сравнением их метрик качества.

4.2. Условия эксперимента

Сравнение моделей будет проводиться на двух датасетах:

- **DDTI: Thyroid Ultrasound Images** [5] — датасет с ультразвуковыми изображениями заболеваний щитовидной железы, использовавшийся в эксперименте в рамках предыдущей учебной практики.
- **BUS-BRA: A Breast Ultrasound Dataset** [4] — датасет с изображениями молочной железы, размеченными по типам патологий.

Включение второго датасета позволяет убедиться в том, что разработанный инструмент не подстроен под данные щитовидной железы, а также продемонстрировать его универсальность и способность к обобщению на медицинские изображения другой природы.

Одним из критериев выбора датасетов была возможность сравнения с результатами, представленными в научных публикациях. Так, в работе [5], посвящённой датасету DDTI, предложен подход с использованием обучения с учебным планом. Авторы достигли следующих метрик: accuracy — 73.28%, precision — 63.02%, recall — 73.64%, F1-мера — 67.87%.

Для датасета BUS-BRA ориентиром является статья [10], в которой представлен метод UltraSam — модификация SAM, дообученная на ультразвуковых данных. На BUS-BRA модель показала precision — 87.6%, recall и accuracy — 87.5%.

4.3. Исследовательские вопросы

RQ1 : какие модели лучше всего справились с классификацией?

RQ2 : сопоставимы ли результаты инструмента с опубликованными методами диагностики по УЗИ

RQ3 : превосходят ли разработанные гибридные модели базовые решения по качеству классификации?

4.4. Метрики

Для сравнения моделей классификации были выбраны метрики:

- Accuracy.
- Recall.
- F_1 -мера.

4.5. Результаты

Результаты эксперимента, представленные в таблице 1, демонстрируют сопоставимый уровень точности с результатами, опубликованными в открытых исследованиях. Алгоритм CatBoostClassifier показал стабильную эффективность на обоих использованных датасетах.

4.5.1. RQ1

На датасете, содержащем изображения щитовидной железы, наилучшие результаты показали модели CatBoostClassifier, Swin и SVM. На втором датасете (молочная железа) лучшими оказались CatBoostClassifier,

Таблица 1: Сравнение результатов классификации разработанного инструмента с результатами, представленными в других опубликованных работах на выбранных датасетах

Решение	Модель	BUS-BRA				DDTI			
		acc	prec	recall	f1	acc	prec	recall	f1
<i>ThyroidScan</i>	CatBoost	83.3	73.6	88.8	80.5	65.9	66.0	86.5	74.9
	Swin	73.0	70.5	76.1	73.2	63.8	67.2	82.4	74.0
	SVM	61.6	55.2	88.9	68.1	65.1	63.2	97.3	76.6
	EfficientNetH	87.6	83.8	90.2	86.9	61.4	60.4	71.3	65.4
	SEResNet	75.5	92.1	81.2	86.3	57.9	59.1	66.8	62.7
<i>UltraSam</i>	UltraSam	-	87.6	87.5	87.5	-	-	-	-
<i>Less is More</i>	DenseNet	-	-	-	-	72.9	60.74	77.8	67.99

EfficientNetHybrid и SEResNet. Эти результаты позволяют сделать вывод о высокой обобщающей способности модели CatBoostClassifier.

4.5.2. RQ2

Да, полученные результаты сопоставимы с опубликованными в открытых исследованиях: модели продемонстрировали либо сходные значения, либо превосходящие по ключевым метрикам качества.

4.5.3. RQ3

На текущем этапе дать окончательный ответ затруднительно. Кастомная модель EfficientNet продемонстрировала наилучший результат на датасете с изображениями молочной железы, однако этого недостаточно для формулирования обоснованных выводов.

4.6. Обсуждение результатов

Можно предположить, что нейросетевые модели показали лучшие результаты на втором датасете по следующим причинам: изображения щитовидной железы обладают высоким уровнем шума, низким разрешением и представлены в ограниченном объёме, что затрудняет

эффективное обучение. В то же время датасет, содержащий изображения молочной железы, отличается более высоким качеством визуальных данных, меньшим уровнем шума и большим числом примеров, что способствует лучшей обучаемости нейронных сетей и их способности извлекать значимые признаки.

Заключение

В ходе данной работы были выполнены следующие задачи:

- Были выдвинуты требования разрабатываемого инструмента.
- Спроектирована архитектура
- Реализована часть инструмента с моделями машинного обучения
- Проведено экспериментальное сравнение результатов классификации

В следующем семестре планируется ряд улучшений инструмента:

- Проведение глубокого сравнительного эксперимента со схожими инструментами
- Расширение функциональности
- Разработка алгоритма для выбора лучшей маски

Исходный код инструмента доступен на Github¹

¹<https://github.com/ArtemKushnir/ThyroidScan.git>

Список литературы

- [1] CatBoost: unbiased boosting with categorical features / Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev et al. // Advances in Neural Information Processing Systems. — 2018. — Vol. 31. — P. 6638–6648.
- [2] Chen Tianqi, Guestrin Carlos. XGBoost: A scalable tree boosting system // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining / ACM. — 2016. — P. 785–794.
- [3] Global cancer statistics 2022: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries / Freddie Bray, Mathieu Laversanne, Hyuna Sung et al. // [CA: A Cancer Journal for Clinicians](#). — 2024. — 04. — Vol. 74.
- [4] Gómez-Flores Wilfrido, Gregorio-Calas Maria, Pereira Wagner. BUS-BRA: A breast ultrasound dataset for assessing computer-aided diagnosis systems // [Medical Physics](#). — 2023. — 11. — Vol. 51.
- [5] [Less is More: Adaptive Curriculum Learning for Thyroid Nodule Diagnosis](#) / Haifan Gong, Hui Cheng, Yifan Xie et al. — 2022. — 09. — P. 248–257. — ISBN: [978-3-031-16439-2](#).
- [6] LightGBM: A highly efficient gradient boosting decision tree / Guolin Ke, Qi Meng, Thomas Finley et al. // Advances in Neural Information Processing Systems. — Vol. 30. — 2017. — P. 3146–3154.
- [7] [Optuna: A Next-generation Hyperparameter Optimization Framework](#) / Takuya Akiba, Shotaro Sano, Takeru Yanase et al. // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). — ACM, 2019. — P. 2623–2631.
- [8] PyTorch: An Imperative Style, High-Performance Deep Learning Library / Adam Paszke, Sam Gross, Francisco Massa et al. // Ad-

vances in Neural Information Processing Systems. — Vol. 32. — 2019. — P. 8024–8035.

- [9] Scikit-learn: Machine Learning in Python / Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort et al. // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.
- [10] Meyer Adrien, Murali Aditya, Mutter Didier, Padoy Nicolas. Ultra-Sam: A Foundation Model for Ultrasound using Large Open-Access Segmentation Datasets. — 2024. — [2411.16222](#).
- [11] pandas development team The. pandas-dev/pandas: Pandas // [Zenodo](#). — 2020. — URL: <https://doi.org/10.5281/zenodo.3509134>.
- [12] Кушнир Артем. сайт кафедры системного программирования. — URL: https://se.math.spbu.ru/thesis/texts/Kushnir_Artem_Jur'evich_Autumn_practice_2nd_year_2024_text.pdf (дата обращения: 28 декабря 2024 г.).