**Институт интеллектуальных кибернетических систем НИЯУ МИФИ**

**Онлайн магистратура: группа М24-525**

**01.04.02 Прикладная математика и информатика,**

**Студент:**

**Медведев Игорь Олегович**

**Классическое машинное обучение**

**Курсовая работа (vo\_PJ)**

**Исследование лекарственной активности**

**2025**

**Введение**

**Цель работы**

На основе предоставленных данных по химическим соединениям построить прогноз эффективности веществ с целью подбора оптимального состава лекарственного препарата. Основной фокус — на предсказании ключевых показателей активности соединений (IC50, CC50, SI) и их классификации на "сильные" / "слабые" ингибиторы.

**Задачи исследования**

1. Провести исследовательский анализ данных (EDA) и оценить информативность признаков.
2. Построить модели машинного обучения:

* Регрессия для IC50
* Регрессия для CC50
* Регрессия для SI
* Классификация: превышает ли значение IC50 медианное значение выборки
* Классификация: превышает ли значение CC50 медианное значение выборки
* Классификация: превышает ли значение SI медианное значение выборки
* Классификация: превышает ли значение SI значение 8

3. Выполнить сравнительный анализ качества моделей по метрикам: 4. Выбрать наиболее эффективные модели и обосновать выбор. Предложить рекомендации по использованию финальной модели в практической работе.

Предобработка данных

Описание датасета:

Датасет представляет собой таблицу, содержащую данные по 1001 химическому соединению. Каждая строка соответствует одному веществу, столбцы — его физико химическим признакам и биологической активности

Признаки:

• Числовые признаки : 107 колонок с типом float64

• Целочисленные признаки : 107 колонок с типом int64

Признаки описывают структурные, физико-химические и молекулярные свойства соединений.

Таблица 1. Целевые переменные датасета

|  |  |
| --- | --- |
| IC₅₀, mM | Концентрация соединения (в миллимолях), требуемая для подавления вирусной активности на 50% |
| CC₅₀, mM | Концентрация соединения (в миллимолях), вызывающая гибель 50% клеток (цитотоксичность) | |
| SI (Selectivity Index) | Индекс селективности, рассчитываемый как отношение CC₅₀ к IC₅₀ (чем выше значение, тем более селективен препарат) |

**Полный перечень признаков**

Электронные и энергетические параметры:

• MaxAbsEStateIndex — максимальный электроотрицательный индекс состояния по абсолютному значению • MaxEStateIndex — максимальный индекс состояния

• MinAbsEStateIndex — минимальный электроотрицательный индекс по абсолютному значению

• MinEStateIndex — минимальный индекс состояния

• MaxPartialCharge — максимальный частичный заряд атома

• MinPartialCharge — минимальный частичный заряд атома

• MaxAbsPartialCharge — максимальный частичный заряд (по модулю)

• MinAbsPartialCharge — минимальный частичный заряд (по модулю)

Молекулярные дескрипторы:

• MolWt — молекулярная масса

• HeavyAtomMolWt — масса без учёта атомов водорода

• ExactMolWt — точная молекулярная масса

• NumValenceElectrons — количество валентных электронов

• NumRadicalElectrons — количество радикальных электронов

• qed — Quantitative Estimate of Drug-likeness (оценка качества молекулы как кандидата в лекарства)

• SPS — сумма поляризационных поверхностей растворителя

Физико-химические свойства:

• MolLogP — коэффициент распределения (оценка липофильности)

• MolMR — молярный рефракционный показатель (мера молекулярного объёма и поляризуемости)

Структурные признаки:

• HeavyAtomCount — число тяжёлых атомов (все, кроме H)

• NHOHCount — число групп OH и NH

• NOCount — число атомов N и O

• NumRotatableBonds — число ротируемых связей (мера гибкости молекулы)

• RingCount — общее число колец • FractionCSP3 — доля sp³-гибридизованных атомов углерода

• NumAliphaticRings — число алифатических колец

• NumAromaticRings — число ароматических колец

• NumHAcceptors — число акцепторов водородных связей • NumHDonors — число доноров водородных связей

• NumHeteroatoms — число гетероатомов

Дескрипторы Morgan Fingerprint Density:

• FpDensityMorgan1, FpDensityMorgan2, FpDensityMorgan3 — плотность фингерпринтов разного радиуса

BCUT-дескрипторы (атомные свойства):

• BCUT2D\_MWHI, BCUT2D\_MWLOW — массовые дескрипторы

• BCUT2D\_CHGHI, BCUT2D\_CHGLO — зарядовые дескрипторы

• BCUT2D\_LOGPHI, BCUT2D\_LOGPLOW — оценка липофильности

• BCUT2D\_MRHI, BCUT2D\_MRLOW — оценка молярного рефракционного индекса

Топологические дескрипторы:

• BalabanJ — балабановский индекс (топологическая характеристика молекулы)

• BertzCT — индекс сложности молекулы (fragment complexity contribution)

• HallKierAlpha, Ipc, Kappa1, Kappa2, Kappa3 — структурные индексы Холла Кьера

Площадь поверхности доступности (ASA):

• LabuteASA — площадь доступной растворителю поверхности

PEOE\_VSA — дескрипторы по зарядам:

(PEOE — Partial Equalization of Orbital Electronegativity)

• PEOE\_VSA1–PEOE\_VSA14 — разделённые по диапазонам значения атомных зарядов и поляризации

SMR\_VSA — молекулярное рефракционное значение по участкам:

• SMR\_VSA1–SMR\_VSA10 — молярная рефракция по различным диапазонам

SlogP\_VSA — логP по областям молекулы:

• SlogP\_VSA1–SlogP\_VSA12 — дескрипторы липофильности по участкам молекулы

TPSA — полярная поверхность:

• TPSA — суммарная полярная поверхность (Topological Polar Surface Area)

EState\_VSA — электроотрицательность по зонам:

• EState\_VSA1–EState\_VSA11 — деление молекулы на участки по электроотрицательности VSA\_EState — вариация EState по размеру:

• VSA\_EState1–VSA\_EState9 — деление по электроотрицательности с участием площади поверхности

Часто используемые фрагменты (fr\_...):

Функциональные группы и их наличие в молекуле:

• fr\_Al\_COO – аллильная карбоновая группа

• fr\_Al\_OH – спиртовые OH-группы

• fr\_Al\_OH\_noTert – OH-группы, за исключением третичных

• fr\_ArN – ароматические N

• fr\_Ar\_COO – ароматические карбоновые кислоты

• fr\_Ar\_N – ароматические амины

• fr\_Ar\_NH – ароматические аминогруппы

• fr\_Ar\_OH – фенольные OH

• fr\_COO – карбоновые кислоты

• fr\_COO2 – вторая форма карбоновой кислоты

• fr\_C\_O – карбонильные группы

• fr\_C\_O\_noCOO – карбонилы, кроме карбоновых

• fr\_C\_S – группы с атомами C=S

• fr\_HOCCN – цианиды с OH-группой

• fr\_Imine – имины

• fr\_NH0 – первичные NH-группы

• fr\_NH1 – вторичные NH-группы

• fr\_NH2 – третичные NH-группы

• fr\_N\_O – связи N–O

• fr\_Ndealkylation1, fr\_Ndealkylation2 – маркеры реакции N-деалкилирования

• fr\_Nhpyrrole – пиррольные NH-группы

• fr\_SH – тиольные группы

• fr\_aldehyde – альдегиды

• fr\_alkyl\_carbamate – карбаматы

• fr\_alkyl\_halide – алкилгалогениды

• fr\_allylic\_oxid – метки для окисления аллильных групп

• fr\_amide – амиды

• fr\_amidine – амидины

• fr\_aniline – анилины

• fr\_aryl\_methyl – арилметильные группы

• fr\_azide – азида

• fr\_azo – азо-соединения

• fr\_barbitur – барбитуровая кислота или её производные

• fr\_benzene – бензольные кольца

• fr\_benzodiazepine – бензодиазепиновые структуры

• fr\_bicyclic – двухкольцевые структуры

• fr\_diazo – диазосоединения

• fr\_dihydropyridine – дигидропиридины

• fr\_epoxide – эпоксиды

• fr\_ester – эфиры

• fr\_ether – простые эфиры

• fr\_furan – фурановые кольца

• fr\_guanido – гуанидиновые группы

• fr\_halogen – галогены

• fr\_hdrzine – гидразиновые группы

• fr\_hdrzone – гидразоны

• fr\_imidazole – имидазолы

• fr\_imide – имиды

• fr\_isocyan – изоцианиды

• fr\_isothiocyan – изотиоцианиды

• fr\_ketone – кетоны

• fr\_ketone\_Topliss – кетоны (по Topliss)

• fr\_lactam – лактамы

• fr\_lactone – лактоны

• fr\_methoxy – метокси-группы

• fr\_morpholine – морфолиновые структуры

• fr\_nitrile – нитрилы

• fr\_nitro – нитрогруппы

• fr\_nitro\_arom – нитроароматические соединения

• fr\_nitro\_arom\_nonortho – нитроароматические, не орто-замещённые

• fr\_nitroso – нитрозо-соединения

• fr\_oxazole – оксазолы

• fr\_oxime – оксимы

• fr\_para\_hydroxylation – метки для пара-гидроксилирования

• fr\_phenol – фенольные OH-группы

• fr\_phenol\_noOrthoHbond – фенолы без орто-водородных связей

• fr\_phos\_acid – фосфорные кислоты

• fr\_phos\_ester – фосфорные эфиры

• fr\_piperdine – пиперидиновые структуры

• fr\_piperzine – пиперазиновые структуры

• fr\_priamide – первичные амиды

• fr\_prisulfonamd – сульфонамиды

• fr\_pyridine – пиридиновые кольца

• fr\_quatN – четвертичные атомы азота

• fr\_sulfide – сульфиды

• fr\_sulfonamd – сульфонамиды

• fr\_sulfone – сульфоны

• fr\_term\_acetylene – терминальные ацетилены

• fr\_tetrazole – тетразолы

• fr\_thiazole – тиазолы

• fr\_thiocyan – тиоцианаты

• fr\_thiophene – тиофеновые кольца

• fr\_unbrch\_alkane – неразветвлённые алканы

• fr\_urea – мочевина и её производные

На основании предоставленных данных от химиков необходимо построить прогноз, позволяющий подобрать наиболее эффективное сочетание параметров для создания лекарственных препаратов.

Ссылка на датасет:

url='https://lms.skillfactory.ru/asset-v1:SkillFactory+MIFIML-2sem+2025+type@asset+block@%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5\_%D0%B4%D0%BB%D1%8F\_%D0%BA%D1%83%D1%80%D1%81%D0%BE%D0%B2%D0%BE%D0%B8\_\_%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5\_%D0%9C%D0%9E.xlsx'

Файл: 'Данные\_для\_курсовои\_Классическое\_МО.xlsx'

Основная информация о DataFrame

* Количество записей: 1001
* Индекс: от 0 до 1000
* Количество колонок: 213

Статистика по колонкам

* Название первой колонки: IC50
* Тип данных первой колонки: float64
* Последняя колонка: fr\_urea
* Тип данных последней колонки: int64

Динамика типов данных

* float64: 107 колонок
* int64: 106 колонок

Затраты памяти

* Использование памяти: 1.6 MB

**1. Качество проведённого EDA**

Файл: Kurs\_EDA.ipynb

# Проверка типов данных

# Поиск нечисловых столбцов

# Получение описания, статистики, включая все типы данных

# Замена имен столбцов 'IC50, mM': 'IC50', 'CC50, mM': 'CC50'

# Вывод колонок с категориальным типом

# Вывод колонок с числовым типом

# Cтатистики для числовых столбцов

# информацию о столбцах, состоящих только из нулей

# Удаление нулевых столбцов из исходного датафрейма

# Построение гистограмм с KDE и линиями медианы и среднего

# Построения боксплотов для визуализации выбросов

* **Вывод:**
* IC50, CC50, SI характеризуются правосторонней асимметрией распределений с экстремальными значениями (особенно выражено для SI).
* Медиана < Среднего во всех случаях, что указывает на влияние длинных хвостов распределения. В случае с другими признаками мы сочлимы двнные особенности за выбросы,но поскольк мы имеем дело с целевыми переменными то будем относится к ним как важным особенностям.

Выводы по анализу свойств молекул

1. MaxAbsEStateIndex (электронные свойства)

* Распределение:
  + Близко к нормальному с небольшим смещением вправо.
  + Большинство значений сосредоточены в диапазоне от 8 до 15.
  + Наблюдаются редкие выбросы, достигающие 16.

2. MolWt (молекулярная масса)

* Распределение:
  + Близко к нормальному распределению с явным пиком в диапазоне 300–400.
  + Длинный правый хвост указывает на наличие молекул с высокой массой (более 700).

# Построение тепловой карты корреляционной матрицы

# Построение корреляционной матрицы для заданных ключевых признаков

['IC50', 'CC50', 'SI', 'MolWt', 'MolLogP', 'MaxAbsEStateIndex']

* Выводы:
* Анализ мультиколлинеарности показывает высокие корреляции между несколькими признаками в данных.
* Сильная корреляция: Наиболее заметные пары признаков, такие как ExactMolWt и MolWt (0.999999), а также Chi1 и HeavyAtomCount (0.998742), указывают на то, что эти переменные практически линейно зависимы. Это может привести к проблемам в моделировании, так как мультиколлинеарность затрудняет оценку значимости отдельных признаков.
* Сильная корреляция:
* Наиболее заметные пары признаков, такие как ExactMolWt и MolWt (0.999999), а также Chi1 и HeavyAtomCount (0.998742), указывают на то, что эти переменные практически линейно зависимы. Это может привести к проблемам в моделировании, так как мультиколлинеарность затрудняет оценку значимости отдельных признаков.

# Список признаков для удаления, по результату анализа матрицы корреляции

[

'HeavyAtomMolWt',

'Chi0',

'LabuteASA',

'Chi0n'

]

# Вычисление основных статистик

# Удаляем выбросы в IC50

# Вычисляем медиану для IC50 без выбросов

# Заменяем выбросы в IC50

# Пересчитываем SI

# Удаляем выбросы в SI

# Вычисляем медиану для SI без выбросов

# Заменяем выбросы в SI

# Проверка результатов

Анализ структуры данных и их предобработка:

# Проверка на наличие пропусков

# Поиск строк с пропусками

# Заполнение пустых значений медианой по каждому столбцу

# Построение гистограмм

# тесты на нормальность распределения данных в заданных столбцах DataFrame

тест Шапиро-Уилка, Андерсона-Дарлина

# статистические тесты, на нормальность распределения для ['CC50', 'IC50', 'SI']

# графическая визуализация только для ['CC50', 'IC50', 'SI']

* *На основании результатов тестов на нормальность — Шапиро-Уилка и Андерсона-Дарлина — можно сделать следующие выводы:*

*Тест Шапиро-Уилка используется для проверки нормально ли распределены данные. В ваших результатах указаны статистики тестов и соответствующие p-значения для трёх наборов данных: CC50, IC50 и SI.*

* Вот основные выводы:
* *CC50:*
* Статистика = 0.809
* p-значение = 0.000
* Интерпретация: p-значение значительно меньше 0.05, что указывает на отклонение от нормальности распределения данных CC50.
* *IC50:*
* Статистика = 0.642
* p-значение = 0.000
* Интерпретация: p-значение также значительно меньше 0.05, что говорит о том, что данные IC50 не распределены нормально.
* *SI:*
* Статистика = 0.330
* p-значение = 0.000
* Интерпретация: Как и в предыдущих случаях, p-значение несколько меньше 0.05 указывает на отклонение от нормальности для данных SI.
* Общий вывод:
* Для всех трёх наборов данных (CC50, IC50 и SI) тест Шапиро-Уилка показывает, что данные не являются нормально распределёнными.

*Тест Андерсона-Дарлина (Anderson-Darling test) служит для проверки гипотезы о том, что данные следуют определённому распределению, и предоставляет статистику теста, которую можно сравнить с критическими значениями для различных уровней значимости.*

* *Вот интерпретация ваших результатов:*
* *Результаты теста:*
* *CC50:*
* *Статистика = 42.665*
* *Критические значения:*
* *15.0%: 0.574*
* *10.0%: 0.653*
* *5.0%: 0.784*
* *2.5%: 0.914*
* *1.0%: 1.088*
* *Вывод: Статистика теста (42.665) значительно превышает максимальное критическое значение (1.088 при уровне значимости 1.0%). Это указывает на наличие отклонения от нормального распределения.*
* *IC50:*
* *Статистика = 134.582*
* *Критические значения:*
* *15.0%: 0.574*
* *10.0%: 0.653*
* *5.0%: 0.784*
* *2.5%: 0.914*
* *1.0%: 1.088*
* *Вывод: Статистика теста (134.582) также значительно превышает все критические значения, что свидетельствует о том, что данные IC50 не распределены нормально.*
* *SI:*
* *Статистика = 206.889*
* *Критические значения:*
* *15.0%: 0.574*
* *10.0%: 0.653*
* *5.0%: 0.784*
* *2.5%: 0.914*
* *1.0%: 1.088*
* *Вывод: Статистика теста (206.889) вновь значительно превышает все критические значения, что указывает на применение ненормального распределения данных SI.*
* *Общий вывод: Для всех трёх наборов данных (CC50, IC50 и SI) результаты теста Андерсона-Дарлина подтверждают выводы теста Шапиро-Уилка и показывают, что данные не распределены нормально.*

# вывод признаков с низкой вариативностью для удаления

# Удаление признаков из датафрейма

# Проверка на количество NaN значений

# Заполнение NaN значений медианным значением по каждому столбцу

# Проверка на наличие NaN значений после обработки

# Применяем StandardScaler к оставшимся данным

# Сохраняю df\_scaled в CSV файл

**2. Качество решения задачи регрессии IC50**

Kurs\_regression\_on\_IC50.ipynbНачало формы

* В коде использовалось четыре разных регрессионных модели для предсказания целевой переменной IC50. Каждая модель прошла этап оптимизации гиперпараметров с помощью библиотеки Optuna. Вот краткое описание каждой из моделей:

**1. XGBRegressor**

* + **Описание**: Это реализация алгоритма градиентного бустинга, разработанная для повышения производительности и скорости. Лучше всего работает с большими наборами данных.
  + **Гиперпараметры, оптимизированные в коде**:
    - learning\_rate: Скорость обучения.
    - max\_depth: Максимальная глубина дерева.
    - n\_estimators: Количество деревьев в ансамбле.
    - subsample: Доля выборки, используемая для каждого дерева.
    - colsample\_bytree: Доля признаков, используемых для построения каждого дерева.

**2. DecisionTreeRegressor**

* + **Описание**: Модель, которая строит дерево решений для предсказания целевой переменной, основываясь на значениях признаков.
  + **Гиперпараметры, оптимизированные в коде**:
    - max\_depth: Максимальная глубина дерева.
    - min\_samples\_split: Минимальное количество образцов для разделения узла.
    - min\_samples\_leaf: Минимальное количество образцов, необходимых для листа.

**3. RandomForestRegressor**

* + **Описание**: Метод ансамблевого обучения, который строит множество деревьев решений и объединяет их результаты для повышения точности предсказаний и уменьшения переобучения.
  + **Гиперпараметры, оптимизированные в коде**:
    - n\_estimators: Количество деревьев в случайном лесу.
    - max\_depth: Максимальная глубина дерева.
    - min\_samples\_split: Минимальное количество образцов для разделения узла.
    - min\_samples\_leaf: Минимальное количество образцов, необходимых для листа.

**4. CatBoostRegressor**

* + **Описание**: Это градиентный бустинг с поддержкой категориальных признаков, который хорошо работает с различными типами данных и обычно требует минимальной настройки.
  + **Гиперпараметры, оптимизированные в коде**:
    - learning\_rate: Скорость обучения.
    - depth: Глубина дерева.
    - iterations: Количество итераций (деревьев) в модели.
    - l2\_leaf\_reg: Регуляризация L2 для укрепления устойчивости модели.
    - verbose: Уровень вывода информации во время обучения, установлен в 0 для предотвращения вывода.

**Выводы**

Код выполняет оптимизацию гиперпараметров для каждой из этих моделей с использованием библиотеки Optuna и затем оценивает производительность каждой модели на тестовых данных, используя такие метрики, как MSE (среднеквадратичная ошибка), MAE (средняя абсолютная ошибка) и R² (коэффициент детерминации).

На основе предоставленных результатов моделей регрессии, мы можем сделать несколько выводов:

* 1. Сравнение по MSE (Mean Squared Error)
* CatBoostRegressor имеет наименьшее значение MSE (1.037), что указывает на лучшее обобщение модели на тестовых данных.
* XGBRegressor и RandomForestRegressor демонстрируют схожие значения MSE, что говорит о сопоставимой производительности.
* DecisionTreeRegressor показывает наивысшее значение MSE (1.109), что свидетельствует о худшей производительности среди рассмотренных моделей.
* *2. Сравнение по MAE (Mean Absolute Error)*
* CatBoostRegressor также демонстрирует наименьший MAE (0.658), что указывает на меньшую среднюю ошибку в предсказаниях.
* RandomForestRegressor и XGBRegressor имеют сопоставимые MAE (0.661 и 0.669 соответственно).
* DecisionTreeRegressor имеет наибольший MAE (0.696), подтверждая снижение точности модели.
* 3. Сравнение по R^2 (Коэффициент детерминации)
* CatBoostRegressor имеет наилучший R² (0.147), что означает, что он лучше всего объясняет вариацию зависимой переменной.
* RandomForestRegressor и XGBRegressor показывают схожие значения R² (0.126 и 0.123 соответственно), что указывает на их аналогичную способность объяснять вариацию.
* DecisionTreeRegressor имеет наименьше значение R² (0.088), что говорит о его худшей способности к объяснению данных.
* Общий вывод
* На основании всех трех метрик (MSE, MAE, R²) CatBoostRegressor представляется наиболее эффективной моделью для данной задачи регрессии. В то время как DecisionTreeRegressor показывает наихудшие результаты во всех сравнении, другие модели (XGBRegressor и RandomForestRegressor) являются конкурентоспособными, но немного уступают CatBoost.

Выполнена визуализация результатов.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**3. Качество решения задачи регрессии CC50**

Kurs\_regression\_on\_CC50.ipynb

В коде используется несколько популярных моделей для регрессии. Давайте рассмотрим каждую из них более подробно:

**1. XGBRegressor**

* **Описание**: Это реализация градиентного бустинга, которая считается одной из лучших для задач регрессии и классификации благодаря своей высокой скорости и точности.
* **Гиперпараметры**:
  + learning\_rate: Скорость обучения, влияющая на обновление модели на каждом шаге.
  + max\_depth: Максимальная глубина дерева, контролирующая сложность модели.
  + n\_estimators: Количество деревьев в модели (итерации).
  + subsample: Степень выборки данных для каждого дерева.
  + colsample\_bytree: Доля признаков, используемых для построения каждого дерева.

**2. DecisionTreeRegressor**

* **Описание**: Модель, использующая дерево решений для предсказания значений. Она проста в интерпретации, но может переобучаться на большом количестве данных.
* **Гиперпараметры**:
  + max\_depth: Максимальная глубина дерева.
  + min\_samples\_split: Минимальное количество образцов для разделения узла.
  + min\_samples\_leaf: Минимальное количество образцов в листовом узле.

**3. RandomForestRegressor**

* **Описание**: Метод ансамблевого обучения, который использует множество деревьев решений для улучшения общей точности и устойчивости модели к переобучению.
* **Гиперпараметры**:
  + n\_estimators: Количество деревьев в случайном лесу.
  + max\_depth: Максимальная глубина каждого дерева.
  + min\_samples\_split: Минимальное количество образцов для разделения узла.
  + min\_samples\_leaf: Минимальное количество образцов в листовом узле.

**4. CatBoostRegressor**

* **Описание**: Это современная библиотека для градиентного бустинга, которая специально разработана для работы с категориальными данными и минимизации проблем с переобучением.
* **Гиперпараметры**:
  + learning\_rate: Скорость обучения.
  + depth: Максимальная глубина дерева.
  + iterations: Количество итераций (деревьев) в модели.
  + l2\_leaf\_reg: Регуляризация для уменьшения переобучения.
  + verbose: Уровень отображения информации (установлен в 0, чтобы подавить вывод).

**Общие моменты**

* Все модели оптимизируются с использованием библиотеки **Optuna**, которая позволяет настраивать гиперпараметры путем минимизации ошибки (в данном случае MSE – средняя квадратичная ошибка).
* Модели обучаются на обучающей выборке, а затем их производительность оценивается на тестовой выборке с использованием нескольких метрик: MSE, MAE (средняя абсолютная ошибка) и R² (коэффициент детерминации), что помогает понять, насколько хорошо модели предсказывают целевые значения.

Выполнена визуализация результатов.

На основании представленных данных с тестовыми показателями разных регрессоров, можно сделать следующие выводы:

* Сравнение моделей
* XGBRegressor
* MSE: 0.4473
* MAE: 0.4886
* R²: 0.5944
* DecisionTreeRegressor
* MSE: 0.7112
* MAE: 0.5282
* R²: 0.3551
* RandomForestRegressor
* MSE: 0.4420
* MAE: 0.4601
* R²: 0.5993
* CatBoostRegressor
* MSE: 0.4470
* MAE: 0.4931
* R²: 0.5947
* Выводы
* Лучшие показатели из всех моделей демонстрирует RandomForestRegressor, который имеет наименьший MSE (0.4420) и MAE (0.4601), а также самый высокий R² (0.5993), что указывает на хорошую предсказательную способность модели.
* XGBRegressor и CatBoostRegressor показывают схожие результаты, однако CatBoost имеет немного меньший MSE, но больший MAE по сравнению с XGBRegressor.
* DecisionTreeRegressor имеет наихудшие результаты среди рассмотренных моделей, с самым высоким MSE и MAE, а также самым низким R². Это говорит о том, что модель сильно переобучается и менее эффективна для данной задачи.
* *Рекомендации* Для задач регрессии, судя по данной выборке, стоит отдавать предпочтение RandomForestRegressor из-за его лучших показателей. В зависимости от конкретных требований и вычислительных ресурсов, могут также рассматриваться XGBRegressor и CatBoostRegressor.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**4. Качество решения задачи регрессии SI**

Kurs\_regression\_on\_SI.ipynb

В коде используются четыре регрессионных алгоритма для решения задачи регрессии. Каждый из них имеет свои особенности и параметры, которые могут быть настроены для оптимизации производительности. Распишем каждую модель подробнее:

**1. XGBoost Regressor**

**Описание:** XGBoost (Extreme Gradient Boosting) - это мощный алгоритм градиентного бустинга, который известен своей эффективностью и высокой производительностью. Он поддерживает регуляризацию, что помогает избежать переобучения.

**Ключевые параметры:**

* learning\_rate: скорость обучения, которая контролирует, как сильно каждое дерево влияет на финальную модель.
* max\_depth: максимальная глубина дерева, которая определяет, сколько уровней у дерева может быть.
* n\_estimators: количество деревьев в ансамбле.
* subsample: доля образцов, которые будут использоваться для обучения каждого дерева.
* colsample\_bytree: доля признаков, используемых для обучения каждого дерева.

**2. Decision Tree Regressor**

**Описание:** Решающее дерево - это простая и интерпретируемая модель, которая делит данные на основе значений признаков. Однако она может склоняться к переобучению, особенно если не задать ограничений.

**Ключевые параметры:**

* max\_depth: максимальная глубина дерева.
* min\_samples\_split: минимальное количество образцов, необходимых для разделения узла.
* min\_samples\_leaf: минимальное количество образцов, которое должно быть в листе.

**3. Random Forest Regressor**

**Описание:** Случайный лес - это ансамблевый метод, который строит множество решающих деревьев и делает предсказания на основе среднего их предсказаний. Он помогает снизить переобучение и увеличить стабильность модели.

**Ключевые параметры:**

* n\_estimators: количество деревьев в лесу.
* max\_depth: максимальная глубина каждого дерева.
* min\_samples\_split: минимальное количество образцов, необходимых для разделения узла.
* min\_samples\_leaf: минимальное количество образцов, которое должно быть в листе.

**4. CatBoost Regressor**

**Описание:** CatBoost - это библиотека создания градиентного бустинга, разработанная для работы с категориальными признаками и минимизации необходимости их предварительной обработки. CatBoost часто показывает высокую производительность даже без серьезной настройки параметров.

**Ключевые параметры:**

* learning\_rate: скорость обучения.
* depth: максимальная глубина деревьев.
* iterations: количество деревьев в ансамбле.
* l2\_leaf\_reg: коэффициент L2-регуляризации, который помогает уменьшить переобучение.
* verbose: вывод подробной информации во время обучения (0 - отключить).

**Заключение**

Каждая из перечисленных моделей имеет свои преимущества и недостатки.

Выполнена визуализация результатов.

*На основе представленных результатов тестирования различных моделей регрессии можно сделать следующие выводы:*

* *Сравнительный анализ моделей*
* *XGBRegressor*
* MSE: 1.0531
* MAE: 0.4228
* R²: 0.1538
* Вывод: Модель показывает разумные результаты, но не является лучшей среди представленных.
* *DecisionTreeRegressor*
* MSE: 1.1373
* MAE: 0.4600
* R²: 0.0862
* Вывод: Производительность этой модели значительно ниже других. Высокие показатели MSE и MAE указывают на то, что дерево решений не смогло адекватно уловить зависимости в данных.
* *RandomForestRegressor*
* MSE: 1.0510
* MAE: 0.4291
* R²: 0.1555
* Вывод: Сравнимые результаты с XGBRegressor, но все же незначительно хуже. Показатели близки к лучшим результатам, но не являются самыми оптимальными.
* *CatBoostRegressor*
* MSE: 1.0225 (лучшее значение)
* MAE: 0.4134 (лучшее значение)
* R²: 0.1784 (лучшее значение)
* Вывод: Эта модель показала наилучшие результаты по всем метрикам, что свидетельствует о ее способности лучше всего подстраиваться под данные. *Общий вывод*
* На основании тестовых показателей CatBoostRegressor продемонстрировал наилучшие результаты среди указанных моделей, особенно по метрике MSE, MAE и R². Данная модель, вероятно, лучше всего справилась с задачей регрессии в данном контексте. DecisionTreeRegressor оказался наихудшим выбором, что свидетельствует о слабой способности модели захватывать сложные зависимости в данных.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**5. Предсказание: IC50 > медианы**

Kurs\_IC50\_exceed\_median\_value\_classification.ipynb

**Общее описание**

В представленном коде используются несколько классификаторов для предсказания бинарной целевой переменной (y\_binary), основанной на значении IC50 (среднего значения). Основная цель — оптимизировать гиперпараметры различных моделей с помощью библиотеки Optuna и выбрать лучшие модели для предсказания на основе тестовой выборки.

**Классификаторы в коде**

1. **XGBClassifier (Extreme Gradient Boosting)**
   * **Описание**: Это мощный алгоритм градиентного бустинга, который подходит для решения задач классификации и регрессии. Этот алгоритм работает, создавая ансамбль слабых моделей (обычно деревьев решений) и улучшая их предсказания по мере добавления новых деревьев.
   * **Оптимизируемые параметры**:
     + learning\_rate: Скорость обучения для контроля сложности обновлений модели.
     + max\_depth: Максимальная глубина деревьев, это предотвращает переобучение.
     + n\_estimators: Количество деревьев в ансамбле.
     + subsample: Доля экземпляров, которые используются для обучения каждого дерева.
     + colsample\_bytree: Доля признаков, которые учитываются для каждого дерева.
2. **DecisionTreeClassifier**
   * **Описание**: Это простой и интуитивно понятный алгоритм, который делит данные на основе значений признаков, создавая дерево решений. Он хорош для быстрых предсказаний, но может страдать от переобучения, если параметры не оптимизированы.
   * **Оптимизируемые параметры**:
     + max\_depth: Как и в XGBoost, в этом параметре задается максимальная глубина дерева.
     + min\_samples\_split: Минимальное количество образцов, необходимых для разбиения узла.
     + min\_samples\_leaf: Минимальное количество образцов, которые должны быть в листовом узле.
3. **RandomForestClassifier**
   * **Описание**: Это ансамблевый метод, который использует множество деревьев решений с целью уменьшения переобучения, путем усреднения предсказаний нескольких моделей. Разбивая данные на случайные подвыборки, он создает разнообразные деревья, которые затем комбинируются для получения окончательного результата.
   * **Оптимизируемые параметры**:
     + n\_estimators: Число деревьев в лесу.
     + max\_depth: Максимальная глубина каждого дерева.
     + min\_samples\_split и min\_samples\_leaf: Эти параметры аналогичны тем, что используются в DecisionTreeClassifier, контролируя минимальное количество образцов для разбиения и минимальные образцы в листовом узле соответственно.
4. **CatBoostClassifier**
   * **Описание**: Этот алгоритм также основан на градиентном бустинге и оптимизирован для работы с категориальными переменными. Он автоматически учитывает категориальные признаки и может работать с несбалансированными данными.
   * **Оптимизируемые параметры**:
     + learning\_rate: Управляет скоростью обучения, аналогично параметру в других моделях.
     + depth: Максимальная глубина деревьев в ансамбле.
     + iterations: Общее количество деревьев в модели.
     + l2\_leaf\_reg: Регуляризация для предотвращения переобучения.

**Процесс работы кода**

1. **Загрузка и подготовка данных**:
   * Данные загружаются, и целевая переменная y\_binary создается на основе медианного значения IC50. Признаки X формируются путем исключения определенных колонок.
2. **Разделение на обучающую и тестовую выборки**:
   * Данные разделяются на обучающую (70%) и тестовую (30%) выборки.
3. **Оптимизация гиперпараметров**:
   * Для каждого классификатора используется библиотека Optuna, которая тестирует различные комбинации гиперпараметров, выбирая те, которые максимизируют точность модели (accuracy).
4. **Оценка моделей**:
   * После обучения каждой из моделей на обучающей выборке производится предсказание на тестовых данных. Рассчитываются и выводятся точность (accuracy) и F1-метрика. Это позволяет увидеть, насколько хорошо каждая из моделей справляется с задачей.
5. **Вывод результатов**:
   * В конце код выводит оптимальные гиперпараметры для каждой модели и их производительность на тестовой выборке.

Сделана визуализация результатов.

*На основе представленных результатов тестовых метрик для различных классификаторов можно сделать следующие выводы:*

* *CatBoostClassifier:*
* Тестовая точность: 0.711
* Тестовый F1-score: 0.672
* Показал наилучшие результаты среди всех классификаторов как по тестовой точности, так и по F1-score. Это свидетельствует о его хорошей способности к классификации.
* *RandomForestClassifier:*
* Тестовая точность: 0.701
* Тестовый F1-score: 0.656
* Второй по эффективности. Несмотря на то, что точность немного ниже, F1-score близок к XGBClassifier.
* *XGBClassifier:*
* Тестовая точность: 0.698
* Тестовый F1-score: 0.656
* Результаты сопоставимы с RandomForestClassifier, однако он немного уступает по точности.
* *DecisionTreeClassifier:*
* Тестовая точность: 0.619
* Тестовый F1-score: 0.584
* Наименее эффективный классификатор из представленных. Значительно ниже точность и F1-score по сравнению с другими методами. *Общие выводы:*
* CatBoostClassifier явно выделяется среди остальных алгоритмов, демонстрируя наилучшие результаты.
* RandomForestClassifier и XGBClassifier показывают сравнимые результаты, хоть и уступают CatBoost.
* DecisionTreeClassifier следует рассматривать с осторожностью, поскольку его показатели значительно ниже, что может указывать на переобучение или недостаточную сложность модели.
* Рекомендуется использовать CatBoostClassifier в дальнейших исследованиях и выборах моделей, учитывая его лучшие аналитические показатели.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**6. Предсказание: CC50 > медианы**

Kurs\_CC50\_exceed\_median\_\_value\_classification.ipynb

* В коде используются четыре различных классификатора, каждый из которых имеет свои уникальные особенности и параметры. Вот краткое описание каждой модели:

**1. XGBoost Classifier (XGBClassifier)**

* + **Описание:** Это реализация градиентного бустинга, которая специально разработана для масштабируемости и производительности. Она широко используется для задач классификации благодаря высокой эффективности и точности.
  + **Параметры:**
    - learning\_rate: Уровень обучения (шаг), который контролирует размер обновлений на каждом шаге.
    - max\_depth: Максимальная глубина деревьев, использующихся в модели.
    - n\_estimators: Количество деревьев (учеников), используемых для обучения.
    - subsample: Доля обучающей выборки, используемой для обучения каждого дерева.
    - colsample\_bytree: Доля признаков, используемых для обучения каждого дерева.

**2. Decision Tree Classifier (DecisionTreeClassifier)**

* + **Описание:** Это базовая модель машинного обучения, которая использует древовидную структуру для принятия решений на основе значений признаков. Легко интерпретируется и обучается, но может быть подвержена переобучению.
  + **Параметры:**
    - max\_depth: Максимальная глубина дерева.
    - min\_samples\_split: Минимальное количество образцов, необходимое для разделения узла.
    - min\_samples\_leaf: Минимальное количество образцов, необходимое для узла в листе.

**3. Random Forest Classifier (RandomForestClassifier)**

* + **Описание:** Это ансамблевая модель, которая строит множество решающих деревьев и объединяет их результаты для улучшения точности и контроля переобучения. Работает по принципу голосования.
  + **Параметры:**
    - n\_estimators: Количество деревьев в лесу.
    - max\_depth: Максимальная глубина деревьев.
    - min\_samples\_split: Минимальное количество образцов, необходимое для разделения узла.
    - min\_samples\_leaf: Минимальное количество образцов, необходимое для узла в листе.

**4. CatBoost Classifier (CatBoostClassifier)**

* + **Описание:** Это градиентный бустинг, который хорошо справляется с категориальными признаками без необходимости их предварительного кодирования. Высокоэффективен и устойчив к переобучению.
  + **Параметры:**
    - learning\_rate: Уровень обучения для контроля прироста весов каждого дерева.
    - depth: Максимальная глубина деревьев, используемых в модели.
    - iterations: Количество итераций (деревьев), которые нужно построить.
    - l2\_leaf\_reg: Регуляризация для уменьшения сложностей нелинейных моделей.
    - verbose: Управляет выводом информации о процессе обучения (установлено в 0 для отключения вывода).

**Использование**

В коде проводится оптимизация гиперпараметров для каждого из указанных классификаторов с использованием библиотеки Optuna. Затем каждая модель обучается и оценивается на тестовых данных. Результаты (точность и F1-score) каждой модели выводятся на экран.

Сделана визуализация результатов.

*На основе предоставленных результатов тестирования для различных классификаторов, можно сделать следующие выводы:*

* *1. Сравнение моделей*
* *XGBClassifier:*
* Точность: 0.8007
* F1-score: 0.8105
* *DecisionTreeClassifier:*
* Точность: 0.7526
* F1-score: 0.7647
* *RandomForestClassifier:*
* Точность: 0.7904
* F1-score: 0.8026
* *CatBoostClassifier:*
* Точность: 0.8076
* F1-score: 0.8133
* *2. Анализ моделей*
* Лучшие показатели:
* CatBoostClassifier достигает наилучших результатов по обоим метрикам: точности (0.8076) и F1-score (0.8133). Это говорит о его высокой способности к обобщению и балансировке между точностью и полнотой.
* Наименьшие показатели:
* DecisionTreeClassifier показывает наименьшие результаты по обеим метрикам. Это может указывать на переобучение или недостаточную сложность модели для решения данной задачи.
* *3. Выводы*
* CatBoostClassifier и XGBClassifier являются наиболее эффективными моделями для этой задачи, с CatBoost, который слегка опережает XGBoost по обоим метрикам.
* RandomForestClassifier также демонстрирует приемлемые результаты, но не достигает уровня CatBoost и XGBoost.
* Результаты DecisionTreeClassifier указывают на необходимость его оптимизации или рассмотрения более сложных моделей для улучшения производительности.
* *4. Рекомендации*
* Рассмотреть возможность использования CatBoostClassifier для дальнейшего применения, поскольку он продемонстрировал наивысшую производительность.
* Можно провести дополнительный анализ и оптимизацию гиперпараметров для всех моделей, чтобы улучшить их производительность.
* Рассмотреть использование методов ансамблевого обучения для дальнейшего повышения точности.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**7. Предсказание: SI > медианы**

Kurs\_SI\_exceed\_median\_value\_classification.ipynb

* В приведенном коде используются несколько моделей классификации для решения задачи бинарной классификации на основе набора данных, содержащего целевую переменную SI. Давайте более подробно опишем каждую из моделей, которая используется в коде.

**1. XGBoost Classifier**

**XGBoostClassifier** - это реализация метода градиентного бустинга, которая известна своей высокой производительностью и эффективностью. Она часто используется в соревнованиях по машинному обучению.

* + **Параметры:**
    - objective: устанавливает цель обучения (в данном случае, бинарная логистическая регрессия).
    - learning\_rate: скорость обучения, которая влияет на то, насколько сильно обновляются модели на каждом шаге.
    - max\_depth: максимальная глубина каждого дерева, что контролирует переобучение.
    - n\_estimators: количество деревьев в модели.
    - subsample: часть данных, используемых для обучения каждого дерева, помогает уменьшить переобучение.
    - colsample\_bytree: доля признаков, используемых для создания каждого дерева.

**2. Decision Tree Classifier**

**DecisionTreeClassifier** - это простой и интерпретируемый метод классификации, который строит дерево принятия решений на основе значений признаков.

* + **Параметры:**
    - max\_depth: максимальная глубина дерева.
    - min\_samples\_split: минимальное количество образцов, необходимых для разделения узла.
    - min\_samples\_leaf: минимальное количество образцов в листьях дерева.

**3. Random Forest Classifier**

**RandomForestClassifier** - это ансамблевый метод, который строит множество деревьев решений и объединяет их результаты, чтобы улучшить точность и предотвратить переобучение.

* + **Параметры:**
    - n\_estimators: количество деревьев в лесу.
    - max\_depth: максимальная глубина каждого дерева.
    - min\_samples\_split: минимальное количество образцов, необходимых для разделения узла.
    - min\_samples\_leaf: минимальное количество образцов в листьях.

**4. CatBoost Classifier**

**CatBoostClassifier** - это разновидность градиентного бустинга, специально разработанная для обработки категориальных признаков.

* + **Параметры:**
    - learning\_rate: скорость обучения.
    - depth: максимальная глубина дерева.
    - iterations: общее количество итераций (или деревьев), которые будут построены.
    - l2\_leaf\_reg: регуляризация, предотвращающая переобучение.
    - verbose: уровень вывода информации во время обучения (0 означает отсутствие вывода).

**Оптимизация гиперпараметров**

Optuna - это инструмент для автоматической оптимизации гиперпараметров, используемый в коде для поиска наилучших значений параметров для каждой модели. Каждая модель обучается на обучающей выборке, и их производительность оценивается с помощью метрики F1-score на тестовой выборке.

# Сохранение наилучших параметров и моделей

best\_models[classifier\_name] = {

'best\_params': study.best\_params,

'best\_model': CLASSIFIERS[classifier\_name](\*\*study.best\_params).fit(X\_train, y\_train)

}

**Оценка моделей**

После оптимизации гиперпараметров код вычисляет точность и F1-score для каждой из моделей на тестовой выборке, позволяя сравнить их производительность:

for classifier\_name, model\_data in best\_models.items():

best\_model = model\_data['best\_model']

y\_test\_pred = best\_model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_test\_pred)

f1 = f1\_score(y\_test, y\_test\_pred)

print(f"{classifier\_name} - Тестовая точность: {accuracy}, Тестовый F1-score: {f1}")

Таким образом, код реализует полное решение задачи классификации с использованием различных алгоритмов, автоматической оптимизации их гиперпараметров и последующей оценки их производительности.

Начало формы

Сделана визуализация результатов.

*На основе представленных метрик тестовой точности и F1-score для различных классификаторов, можно сделать следующие выводы:*

*Сравнение моделей*

* *XGBClassifier:*
* Тестовая точность: 0.6701
* Тестовый F1-score: 0.6547
* Данная модель показывает хорошие результаты, однако не является лучшей из представленных.
* *DecisionTreeClassifier:*
* Тестовая точность: 0.6186
* Тестовый F1-score: 0.6237
* Показатели этой модели являются наименьшими среди всех классификаторов, что свидетельствует о её менее эффективной работе по сравнению с другими.
* *RandomForestClassifier:*
* Тестовая точность: 0.6357
* Тестовый F1-score: 0.5985
* Хотя Random Forest использует ансамблевый подход, его результаты также не впечатляют и находятся между Decision Tree и XGBClassifier.
* *CatBoostClassifier:*
* Тестовая точность: 0.6907
* Тестовый F1-score: 0.6739
* Эта модель показывает наилучшие результаты в обеих метриках, что делает её наиболее предпочтительной среди представленных.
* *Общие выводы*
* Лучшие результаты демонстрирует CatBoostClassifier, с высокой точностью и F1-score.
* XGBClassifier занимает второе место, также демонстрируя приличные показатели, но не дотягивает до CatBoost.
* DecisionTreeClassifier и RandomForestClassifier показывают худшие результаты, с заметным отставанием от первой пары моделей. *Рекомендации*
* Рекомендуется использовать CatBoostClassifier для задач, где важны высокая точность и сбалансированность классификации. Возможно, стоит провести дополнительную настройку гиперпараметров для XGBClassifier, чтобы улучшить его показатели. Следует учитывать, что различия в метриках могут зависеть от особенностей данных, поэтому желательно провести дополнительные испытания и валидацию.

Сделано сохранения и загрузки обученных моделей машинного обучения с использованием библиотеки pickle в Python.

**8. Предсказание: SI > 8**

Kurs\_Classification\_SI\_value\_exceeds\_8.ipynb

* В коде представлены четыре различных алгоритма классификации для задачи бинарной классификации, а именно для оценки целевой переменной из данных (это SI). Вот краткое описание моделей, которые используются в коде:

**1. Random Forest**

* + **Описание**: Метод ансамблевого обучения, который строит множество деревьев принятия решений и объединяет их для улучшения точности и уменьшения переобучения.
  + **Преимущества**:
    - Способен обрабатывать большие наборы данных с высоким количеством признаков.
    - Устойчив к переобучению.
    - Обеспечивает хороший уровень интерпретируемости.

**2. XGBoost (Extreme Gradient Boosting)**

* + **Описание**: Быстрая и эффективная реализация алгоритма градиентного бустинга, которая учитывает скрытые зависимости и взаимодействия между признаками.
  + **Преимущества**:
    - Высокая производительность и скорость.
    - Поддерживает различные функции потерь и регуляризацию, что помогает избежать переобучения.
    - Эффективен на больших датасетах.

**3. LightGBM**

* + **Описание**: Алгоритм градиентного бустинга, который используется для решения задач классификации и регрессии. LightGBM используется для обработки больших наборов данных с высокой скоростью.
  + **Преимущества**:
    - Обрабатывает большие объёмы данных быстрее, чем другие реализации градиентного бустинга.
    - Использует метод, называемый "градиентный бустинг с потоковой выборкой", который уменьшает время обучения.
    - Поддерживает использование категориальных признаков без необходимости их явного кодирования.

**4. CatBoost**

* + **Описание**: Алгоритм градиентного бустинга от Yandex, который автоматически обрабатывает категориальные признаки и основан на решении задач машинного обучения.
  + **Преимущества**:
    - Простота в использовании: не требует предварительного кодирования категориальных переменных.
    - Высокая точность прямо на наборе данных с различными типами данных.
    - Имеет механизмы для обработки многоклассовых задач, что делает его гибким и универсальным.

**Заключение**

Эти модели обеспечивают разнообразные подходы к решению задач классификации. В вашем коде они используются для сравнения их производительности на одной и той же задаче, что позволяет выбрать наиболее подходящую для ваших данных и целей.

Начало формы

Сделана визуализация результатов.

*Общие выводы*

* Высокая точность (Accuracy):
* Все модели демонстрируют одинаковое значение точности — 98.97%. Это указывает на то, что каждая из моделей предоставляет хорошие результаты в классификации.
* Различия в AUC (ROC AUC): Несмотря на одинаковую точность, модели показывают значительные различия в значениях ROC AUC:
* *XGBoost:* 0.626943 — наилучший показатель среди всех моделей, что указывает на хорошее качество классификации, особенно для не сбалансированных классов.
* *CatBoost:* 0.616580 — близок к XGBoost, также показывает неплохие результаты.
* *Random Forest:* 0.419689 — значительно ниже, что может указывать на более низкую способность модели к разделению классов.
* *LightGBM:* 0.295337 — наименьший показатель, что может свидетельствовать о слабой производительности на данном наборе данных. Контрарное сочетание: Все модели имеют высокую точность, но различия в ROC AUC подчеркивают, что высокая точность не всегда подразумевает хорошую способность к различению классов. Следует использовать ROC AUC для более осмысленной оценки.

Начало формы