

## КУРСОВАЯ РАБОТА

“Применение моделей машинного обучения для предсказания показателей  
эффективности и токсичности химических соединений”

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	5
1. ИССЛЕДОВАТЕЛЬСКИЙ АНАЛИЗ ДАННЫХ (EDA) .....	7
1.1 Общие сведения о датасете.....	7
1.2 Распределение целевых переменных .....	8
1.2.1 IC50 .....	8
1.2.2 CC50.....	9
1.2.3 SI.....	10
1.3 Диаграммы размаха (Boxplot).....	11
1.4 Корреляционный анализ.....	12
1.5 Диаграммы рассеяния (Scatter plot) .....	13
1.6 Промежуточные выводы .....	14
2. ПОСТРОЕНИЕ МОДЕЛЕЙ РЕГРЕССИИ ДЛЯ ПРОГНОЗА IC50 .....	15
2.1 Постановка задачи .....	15
2.2 Подготовка данных .....	15
2.3 Выбор моделей и подход.....	16
2.5 Улучшение модели: отбор признаков .....	17
2.5.1 Визуализация важности признаков.....	18
2.6 Финальная оптимизация: удаление коррелирующих признаков .	20
2.7 Выводы по задаче регрессии IC50 .....	21
3. ПОСТРОЕНИЕ МОДЕЛЕЙ РЕГРЕССИИ ДЛЯ ПРОГНОЗА IC50 .....	22
3.1 Постановка задачи .....	22
3.2 Выбор моделей и методика оценки.....	22
3.3 Результаты базовых моделей .....	23
3.4 Улучшение модели: отбор и очистка признаков .....	24
3.5 Финальные результаты после отбора и очистки .....	25
3.6 Выводы по регрессии CC50.....	26
4. ПОСТРОЕНИЕ МОДЕЛИ РЕГРЕССИИ ДЛЯ SI .....	28

4.1	Смысл задачи .....	28
4.2	Методы и модели.....	28
4.3	Базовые результаты.....	29
4.4	Оптимизация модели .....	29
4.5	Визуализация предсказаний.....	31
4.6	Заключение по SI.....	32
5.	КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ IC50 МЕДИАНУ .....	33
5.1	Цель и постановка задачи .....	33
5.2	Подготовка данных .....	33
5.3	Построение и обучение моделей.....	34
5.4	Результаты классификации .....	35
5.5	ROC-анализ и визуализация .....	36
5.6	Выводы и рекомендации .....	37
6.	КЛАССИФИКАЦИЯ: SI > 8.....	38
6.1	Цель и обоснование задачи.....	38
6.2	Подготовка данных .....	38
6.3	Обучение моделей.....	39
6.4	Результаты классификации .....	39
6.5	Визуализация ROC-кривых .....	40
6.6	Матрицы ошибок .....	41
6.7	Выводы по SI > 8.....	42
7.	КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ CC50 МЕДИАННОЕ ЗНАЧЕНИЕ ВЫБОРКИ .....	43
7.1	Постановка задачи .....	43
7.2	Подготовка данных .....	43
7.3	Построение моделей и метрики .....	43
7.4	Результаты классификации .....	44
7.5	ROC-анализ и интерпретация.....	45

7.6 Выводы .....	46
8. КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ SI МЕДИАННОЕ ЗНАЧЕНИЕ ВЫБОРКИ .....	47
8.1 Постановка задачи .....	47
8.2 Подготовка данных .....	47
8.3 Модели и критерии оценки.....	47
8.4 Результаты классификации .....	48
8.5 ROC-анализ.....	49
8.6 Выводы .....	50
ЗАКЛЮЧЕНИЕ .....	51
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	53

## ВВЕДЕНИЕ

Процесс создания новых лекарственных препаратов традиционно включает в себя множество этапов: от синтеза молекул и их лабораторных испытаний до проведения клинических исследований на добровольцах. Такой путь может занимать годы и требовать значительных финансовых затрат. Особенно много времени и ресурсов уходит на ранние этапы — так называемые доклинические испытания, на которых оценивается потенциальная активность и безопасность соединений на клеточных моделях.

В последние годы всё чаще применяются методы машинного обучения, которые позволяют заранее выявлять закономерности между химической структурой соединения и его биологическим действием. Это открывает возможность отбраковывать бесперспективные кандидаты на этапе анализа данных, тем самым ускоряя и удешевляя процесс разработки новых препаратов.

В рамках данной работы рассматривается задача прогнозирования активности химических соединений против вируса гриппа на основе числовых характеристик, полученных химическим и физико-химическим методами анализа. Цель исследования — на основании входных параметров (признаков) построить модели, которые смогут предсказывать биологическую активность соединения, выраженную в виде трёх показателей:

- IC50 — концентрация соединения, при которой подавляется 50% активности вируса;
- CC50 — концентрация, вызывающая 50% токсичности для клеток;
- SI — индекс селективности, равный отношению CC50 к IC50, характеризующий безопасность соединения.

Был предоставлен набор данных, содержащий информацию о тысяче химических соединений. Каждый объект описан набором из 211 числовых признаков, отражающих различные характеристики: количество атомов,

присутствие определённых функциональных групп, физико-химические свойства, такие как полярность, липофильность, масса, и другие. Данные представлены в виде таблицы формата Excel, содержащей 214 колонок (211 признаков + 3 целевых столбца: IC50, CC50, SI) и 1001 строку (одна строка — одно соединение).

Таким образом, перед нами стоит задача регрессионного анализа. Мы должны не только изучить структуру данных и очистить их, но и понять, как именно они связаны с искомыми биологическими эффектами. Для этого необходимо провести подробный исследовательский анализ (EDA), выявить выбросы, проверить корреляции между признаками, оценить распределения, и только после этого переходить к построению моделей.

## 1. ИССЛЕДОВАТЕЛЬСКИЙ АНАЛИЗ ДАННЫХ (EDA)

Перед тем как приступить к построению моделей, необходимо провести полный анализ структуры и содержимого датасета. Это позволит выявить проблемы, которые могут повлиять на качество прогнозов, а также определить, какие методы предварительной обработки необходимо применить.

### 1.1 Общие сведения о датасете

Исходный датасет загружен из файла Excel и содержит:

- 1001 строку, каждая из которых описывает отдельное химическое соединение;
- 214 столбца, из которых:
- 211 признаков описывают молекулярную структуру и физико-химические свойства;
- 3 целевых столбца — это значения показателей IC50, CC50 и SI.

Краткое описание состава признаков:

- 107 признаков представлены в виде целых чисел (int64). Они характеризуют такие параметры, как число атомов, количество связей, наличие определённых групп и т.д.
- Остальные 107 признаков имеют тип float64. Среди них — молекулярная масса, коэффициенты логP, плотность заряда и другие более сложные физико-химические параметры.

Пропущенные значения:

- Всего в таблице обнаружено 36 пропущенных ячеек. Относительно общего числа значений это составляет менее 0.02%, что не критично, но требует обработки. В работе будет применено удаление строк с пропусками, поскольку их количество крайне мало.

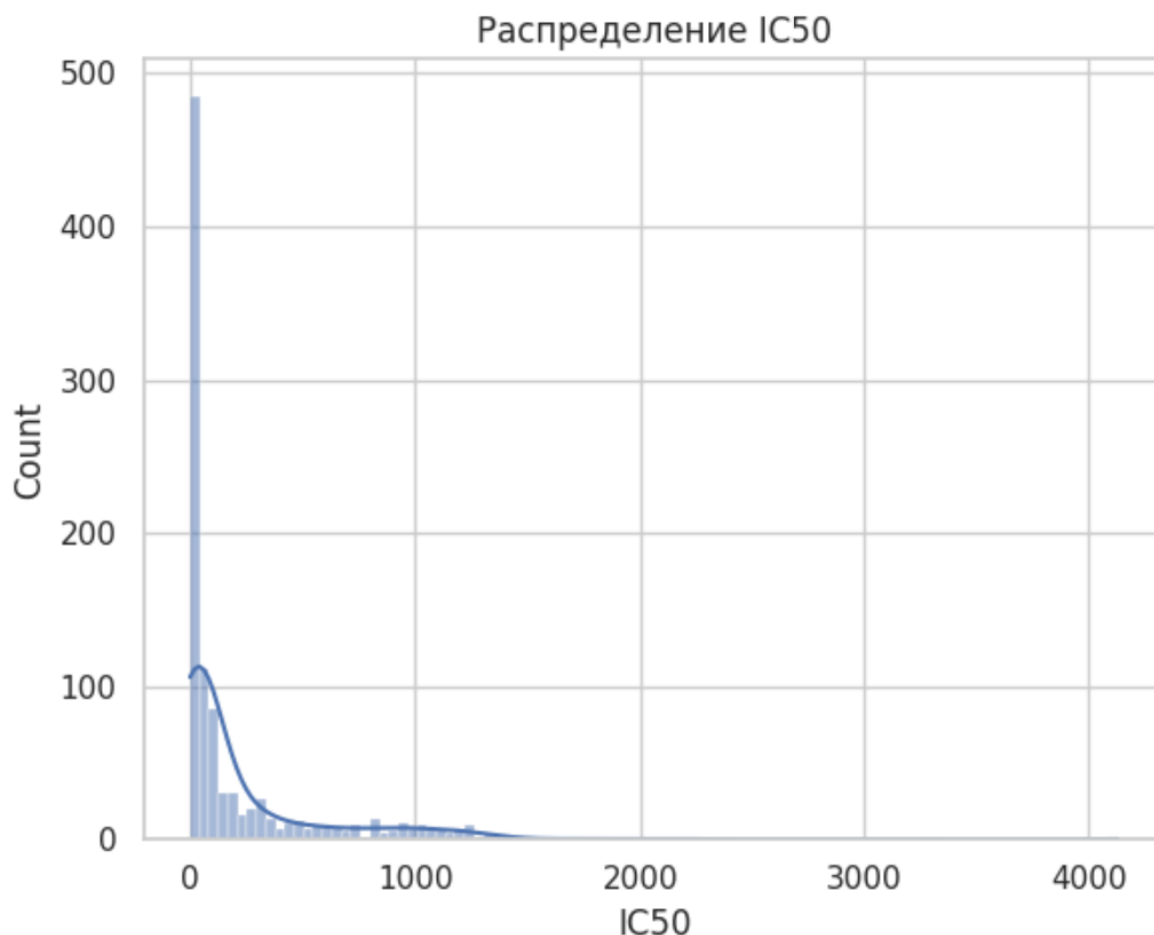
Дубликаты:

- Повторяющихся строк в таблице не обнаружено. Это говорит о том, что каждый объект уникален и представляет собой отдельное соединение.





- Минимальные значения — меньше 1, максимальные — свыше 4000;
- Наблюдаются выбросы, особенно в диапазоне от 1000 до 4000.



Вывод: требуется логарифмирование значений IC50 для выравнивания распределения перед обучением моделей.

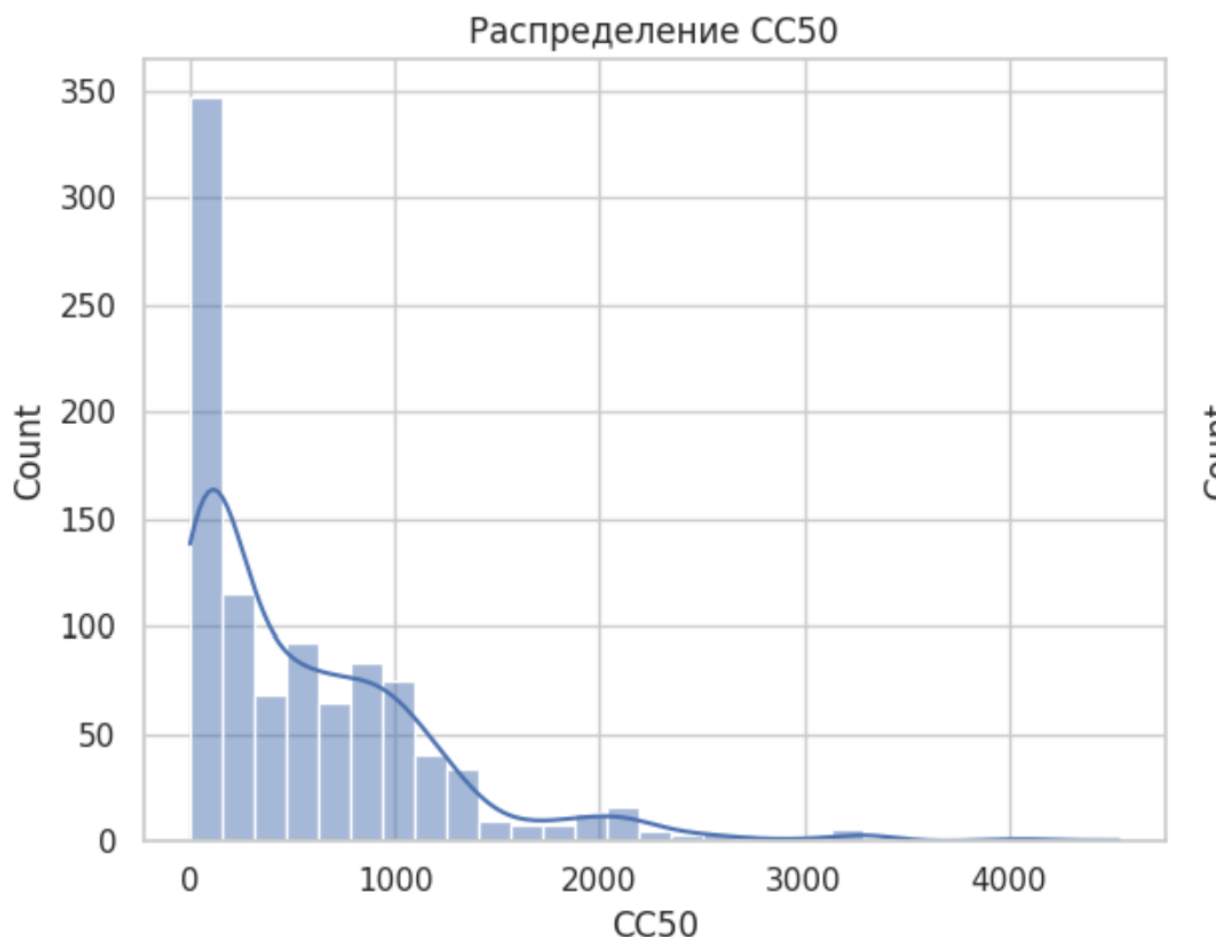
### 1.2.2 CC50

CC50 (полумаксимальная цитотоксическая концентрация) указывает на дозу, при которой вещество вызывает 50% клеточной токсичности. Чем выше этот показатель, тем безопаснее соединение.

Наблюдения:

- Распределение широкое, с длинным хвостом;
- Среднее значение — около 589, медиана — значительно ниже;
- Стандартное отклонение — около 642, что указывает на большую дисперсию;

- Присутствуют высокие значения (до 5000 и выше), являющиеся выбросами.



Вывод: распределение требует нормализации или логарифмирования. Выбросы необходимо либо устранить, либо контролировать их влияние.

### 1.2.3 SI

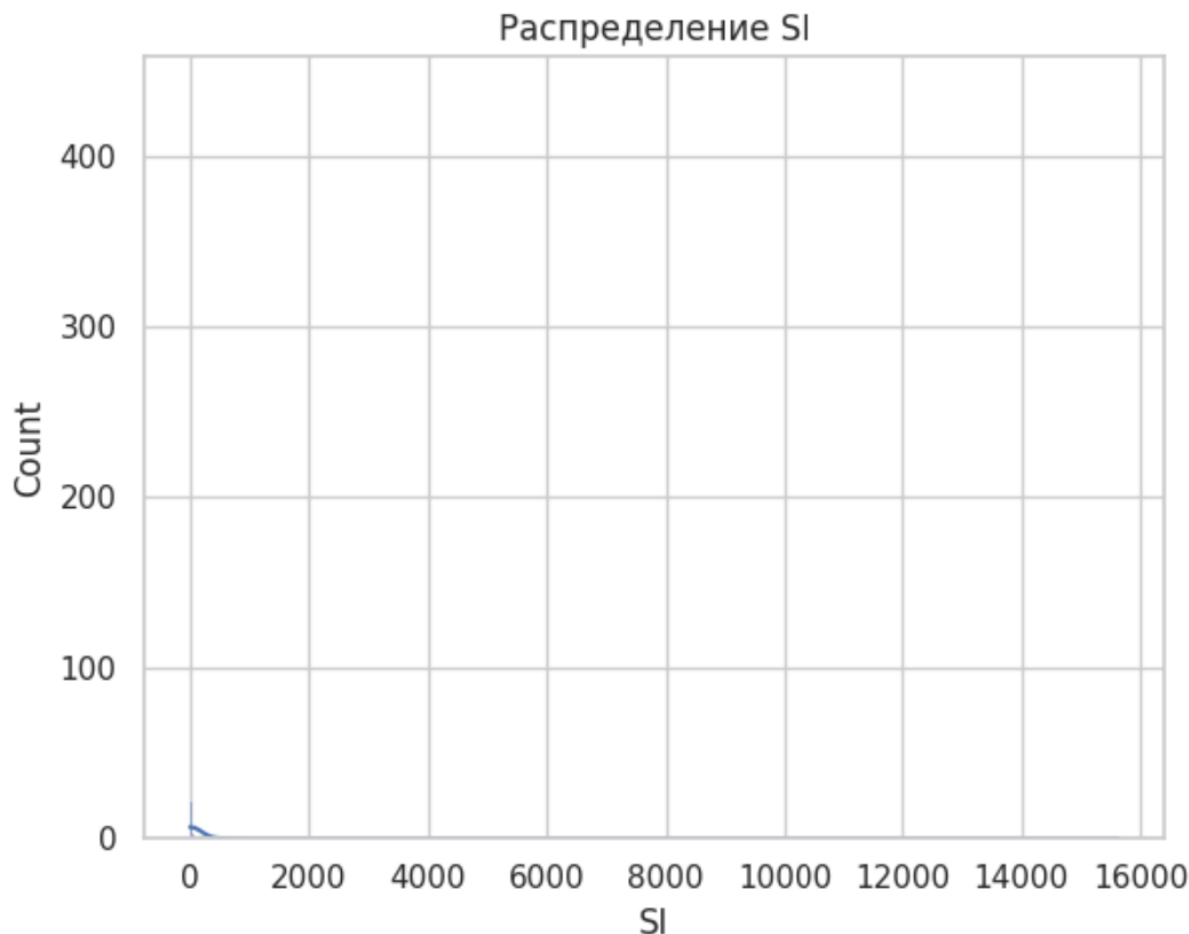
SI (индекс селективности) рассчитывается по формуле:  
$$SI = CC50 / IC50$$

Он показывает, насколько хорошо соединение подавляет вирус, при этом не повреждая клетки. Чем выше SI, тем выше "лечебный индекс".

Наблюдения:

- Распределение крайне несимметричное, с пиком в районе малых значений;
- Многие значения — ниже 10, единичные значения — свыше 1000;
- Максимум — превышает 15000, что выглядит как аномалия;

- Большое количество выбросов, особенно в правой части графика.



Вывод: SI — переменная со сложной природой. Распределение требует предобработки (возможно логарифмирования), иначе оно исказит модели.

### 1.3 Диаграммы размаха (Boxplot)

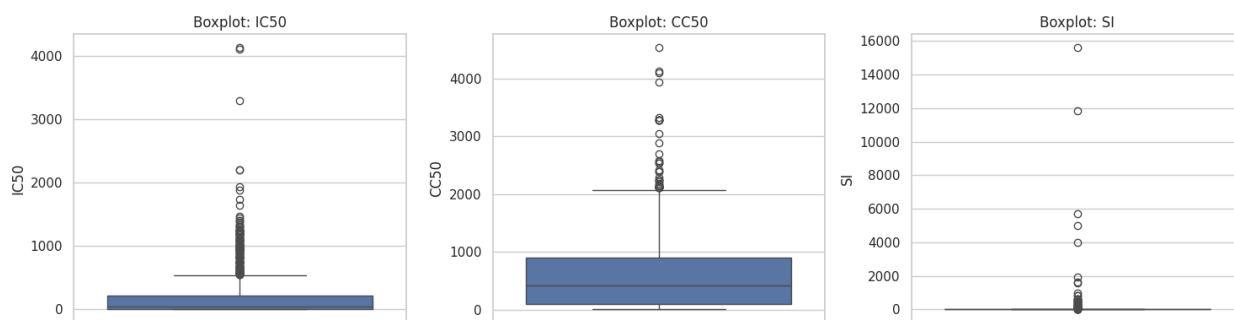
Для каждой из трёх целевых переменных построены диаграммы размаха (boxplot), которые наглядно показывают:

- Медиану (центральная черта в коробке),
- Межквартильный размах (длина коробки),
- Нижние и верхние усы (границы "нормальных" значений),
- Выбросы (точки вне усов).

— Результаты:

- Для IC50 и CC50 — умеренное количество выбросов;

— Для SI — выбросов гораздо больше, и они имеют экстремальные значения.



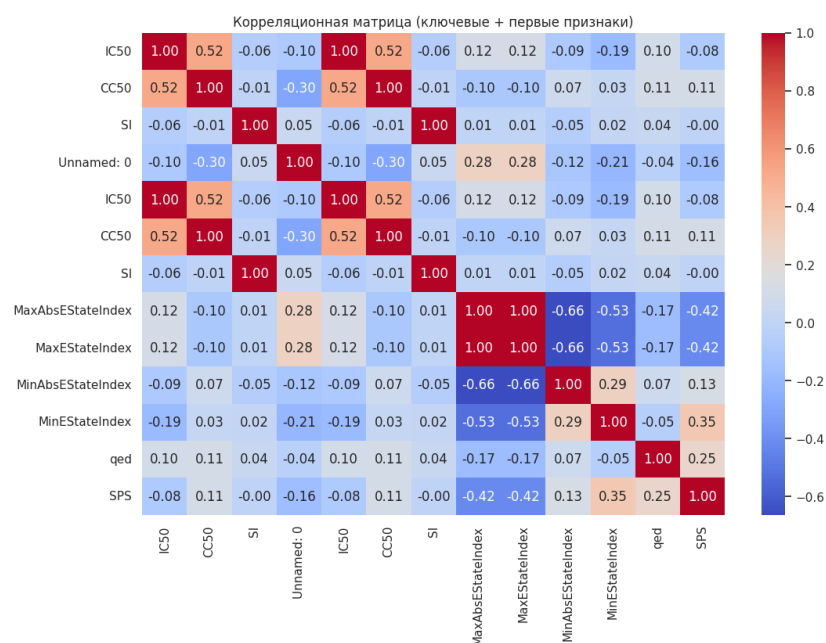
Вывод: SI нуждается в особом внимании при построении моделей. Возможно, потребуется отсечь выбросы выше определённого порога или провести логарифмирование.

## 1.4 Корреляционный анализ

Для оценки степени зависимости между переменными была построена корреляционная матрица, в которой использовался коэффициент Пирсона.

Основные результаты:

- IC50 и CC50: корреляция  $\approx 0.52$ . Связь умеренно выраженная, что логично, так как оба показателя основаны на концентрациях;
- SI и IC50/CC50: слабая корреляция, что объясняется тем, что SI — производная переменная.



Были также выявлены признаки, которые обладают высокой корреляцией с целевыми переменными — они станут основой для построения моделей.

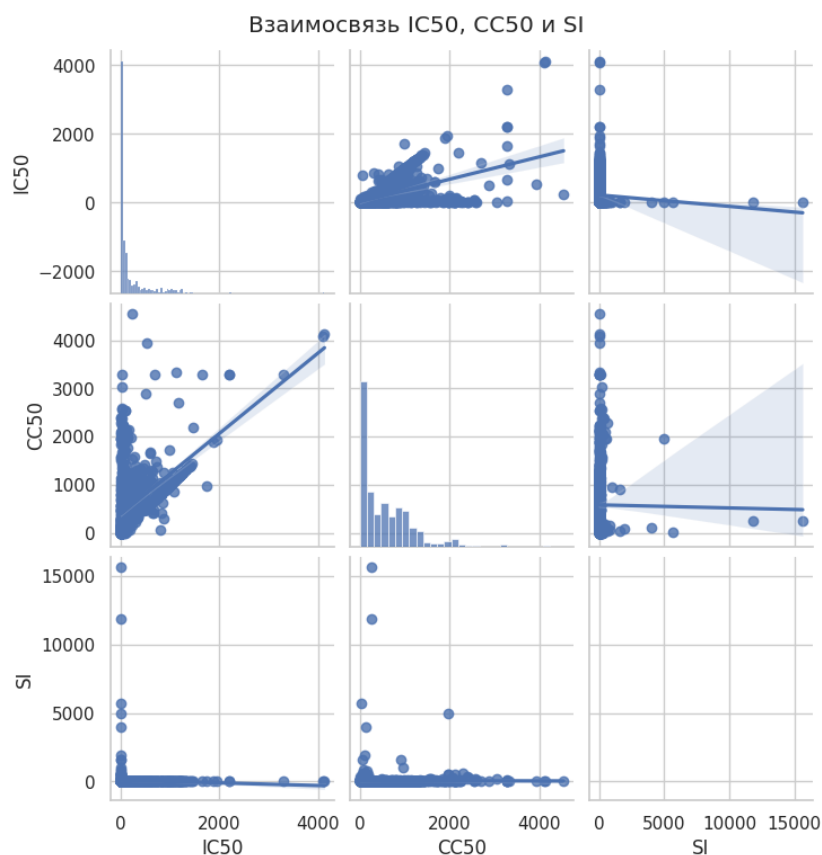
## 1.5 Диаграммы рассеяния (Scatter plot)

Для визуализации взаимосвязей между IC50, CC50 и SI построены диаграммы рассеяния. Они позволяют наглядно увидеть направление зависимости и наличие линейных или нелинейных связей.

Обнаруженные зависимости:

- При высоком CC50 и низком IC50 значения SI стремятся к максимуму;
- Между IC50 и CC50 прослеживается линейная зависимость;
- Зависимость SI от IC50 и CC50 имеет сложный характер — вблизи осей наблюдаются резкие изменения.

Эти графики позволяют сделать вывод, что простые линейные модели могут хорошо работать для IC50 и CC50, но для SI потребуется что-то более гибкое (например, деревья решений или нейронные сети).



## 1.6 Промежуточные выводы

Проведённый анализ позволил выявить важные особенности исходных данных:

- Данные в целом чистые, без дубликатов и с малым числом пропусков;
- Целевые переменные имеют разную природу: IC50 и CC50 — концентрации, SI — производное отношение;
- Все целевые переменные содержат выбросы, наиболее выраженные у SI;
- Есть умеренные корреляции между IC50 и CC50, что пригодится при построении моделей;
- SI — наименее предсказуемая переменная, требует особого подхода при моделировании.

На основе полученных наблюдений будут разработаны методы предобработки данных: удаление выбросов, нормализация и отбор признаков. Эти действия подготовят данные к следующему этапу — обучению моделей машинного обучения для предсказания значений IC50, CC50 и SI.

## **2. ПОСТРОЕНИЕ МОДЕЛЕЙ РЕГРЕССИИ ДЛЯ ПРОГНОЗА IC50**

### **2.1 Постановка задачи**

Одной из ключевых целей настоящего исследования является прогноз значения IC50 — концентрации, при которой химическое соединение подавляет активность вируса гриппа на 50%. Этот показатель широко используется в фармацевтической практике при предварительной оценке эффективности препаратов, так как позволяет ранжировать соединения по их противовирусной активности без необходимости проведения длительных и дорогостоящих биологических экспериментов.

Задача предсказания IC50 решается как задача регрессии: необходимо по множеству признаков, описывающих химическое соединение, определить числовое значение IC50.

### **2.2 Подготовка данных**

В качестве признаков использовались 211 числовых параметров, описывающих структурные и физико-химические характеристики молекул. Они включают:

- Индексы атомного состава (количество атомов разных элементов);
- Показатели молекулярной формы (например, длина цепи, циклическость);
- Зарядовые распределения и логарифмы растворимости;
- Другие расчетные значения, предоставленные химиками.

Перед обучением моделей данные были нормализованы с помощью StandardScaler, что особенно важно для моделей, чувствительных к масштабу признаков (например, Ridge и Lasso). Признаки с нулевой дисперсией были исключены. Выборка была разделена на обучающую и тестовую части в соотношении 80/20, с фиксацией случайного состояния (random\_state=42).

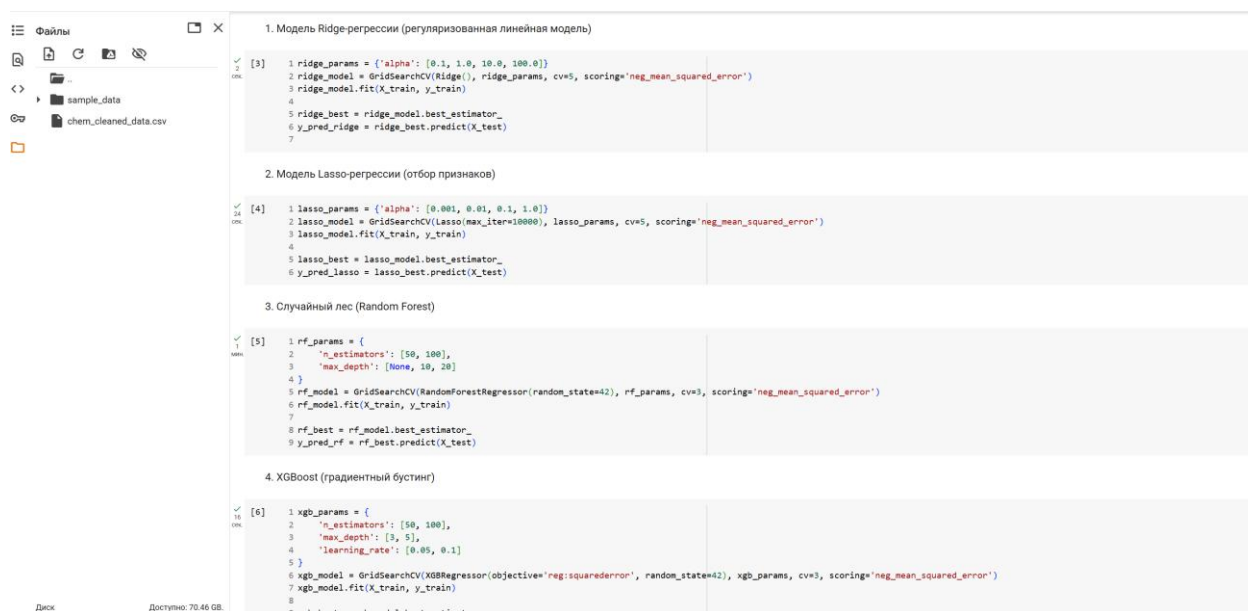
## 2.3 Выбор моделей и подход к обучению

Для задачи регрессии были отобраны четыре разных модели:

1. Ridge-регрессия (линейная модель с L2-регуляризацией);
2. Lasso-регрессия (линейная модель с L1-регуляризацией и встроенным отбором признаков);
3. Random Forest Regressor (ансамбль деревьев, хорошо работающий с шумными и нелинейными зависимостями);
4. XGBoost Regressor (градиентный бустинг — продвинутый метод для задач с высокой размерностью).

Настройка моделей:

Для каждой модели проводился подбор гиперпараметров с использованием метода GridSearchCV на 5-кратной кросс-валидации. Это обеспечивало устойчивость оценки и предотвращение переобучения. Ключевые метрики оценки — MAE (средняя абсолютная ошибка), RMSE (корень из среднеквадратичной ошибки) и  $R^2$  (коэффициент детерминации).



```
1. Модель Ridge-регрессии (регуляризованная линейная модель)
[3] 1 ridge_params = {'alpha': [0.1, 1.0, 10.0, 100.0]}
2 ridge_model = GridSearchCV(Ridge(), ridge_params, cv=5, scoring='neg_mean_squared_error')
3 ridge_model.fit(X_train, y_train)
4
5 ridge_best = ridge_model.best_estimator_
6 y_pred_ridge = ridge_best.predict(X_test)
7

2. Модель Lasso-регрессии (отбор признаков)
[4] 1 lasso_params = {'alpha': [0.001, 0.01, 0.1, 1.0]}
2 lasso_model = GridSearchCV(Lasso(max_iter=10000), lasso_params, cv=5, scoring='neg_mean_squared_error')
3 lasso_model.fit(X_train, y_train)
4
5 lasso_best = lasso_model.best_estimator_
6 y_pred_lasso = lasso_best.predict(X_test)

3. Случайный лес (Random Forest)
[5] 1 rf_params = {
2   'n_estimators': [50, 100],
3   'max_depth': [None, 10, 20]
4 }
5 rf_model = GridSearchCV(RandomForestRegressor(random_state=42), rf_params, cv=3, scoring='neg_mean_squared_error')
6 rf_model.fit(X_train, y_train)
7
8 rf_best = rf_model.best_estimator_
9 y_pred_rf = rf_best.predict(X_test)

4. XGBoost (градиентный бустинг)
[6] 1 xgb_params = {
2   'n_estimators': [50, 100],
3   'max_depth': [3, 5],
4   'learning_rate': [0.05, 0.1]
5 }
6 xgb_model = GridSearchCV(XGBRegressor(objective='reg:squarederror', random_state=42), xgb_params, cv=3, scoring='neg_mean_squared_error')
7 xgb_model.fit(X_train, y_train)
8
9 xgb_best = xgb_model.best_estimator_
```

## 2.4 Сравнительный анализ моделей

Модель	RMSE	MAE	$R^2$
Ridge	409.10	234.00	0.394



Lasso	396.17	231.86	0.432
Random Forest	396.47	213.11	0.431
XGBoost	384.52	208.06	0.465

#### Сравнение результатов

```
[7] 1 def evaluate_model(y_true, y_pred, name):
2     print(f'{name}:')
3     print(f' RMSE = {np.sqrt(mean_squared_error(y_true, y_pred)):.3f}')
4     print(f' MAE = {mean_absolute_error(y_true, y_pred):.3f}')
5     print(f' R^2 = {r2_score(y_true, y_pred):.3f}\n')
6
7     evaluate_model(y_test, y_pred_ridge, 'Ridge')
8     evaluate_model(y_test, y_pred_lasso, 'Lasso')
9     evaluate_model(y_test, y_pred_rf, 'Random Forest')
10    evaluate_model(y_test, y_pred_xgb, 'XGBoost')
```

```
↩ Ridge:
RMSE = 409.098
MAE = 234.004
R^2 = 0.394
```

```
Lasso:
RMSE = 396.174
MAE = 231.862
R^2 = 0.432
```

```
Random Forest:
RMSE = 396.471
MAE = 213.112
R^2 = 0.431
```

```
XGBoost:
RMSE = 384.517
MAE = 208.063
R^2 = 0.465
```

#### Вывод:

XGBoost продемонстрировал наилучшие результаты по всем метрикам. Это объясняется его способностью выявлять сложные зависимости и устойчивостью к выбросам, которых, как было показано в EDA, в данных достаточно много.

Линейные модели показали более слабые результаты, но всё же обеспечили базовое качество.

## 2.5 Улучшение модели: отбор признаков

Была выполнена оценка важности признаков с помощью встроенного механизма XGBoost.

```
d_data.csv

13 # Новый train/test
14 X_train_top, X_test_top, y_train_top, y_test_top = train_test_split(X_top_scaled, y, test_size=0.2, random

[10] 1 xgb_top = XGBRegressor(
2     n_estimators=150,
3     max_depth=4,
4     learning_rate=0.05,
5     subsample=0.9,
6     colsample_bytree=0.8,
7     gamma=1,
8     min_child_weight=3,
9     objective='reg:squarederror',
10    random_state=42
11 )
12 xgb_top.fit(X_train_top, y_train_top)
13 y_pred_top = xgb_top.predict(X_test_top)
14

[11] 1 def evaluate_model(y_true, y_pred, name):
2     rmse = np.sqrt(mean_squared_error(y_true, y_pred))
3     mae = mean_absolute_error(y_true, y_pred)
4     r2 = r2_score(y_true, y_pred)
5     print(f'{name}: \n RMSE = {rmse:.2f} \n MAE = {mae:.2f} \n R^2 = {r2:.4f} \n')
6
7 evaluate_model(y_test, y_pred_top, 'XGBoost (только 50 признаков)')
8

XGBoost (только 50 признаков):
RMSE = 383.42
MAE = 197.18
R^2 = 0.4678
```

По результатам отобраны 50 наиболее значимых признаков. Это позволило:

- Снизить размерность признакового пространства;
- Повысить интерпретируемость;
- Снизить время обучения;
- Потенциально повысить устойчивость модели.

Модель XGBoost была переобучена на сокращённом наборе:

Модель	RMSE	MAE	R <sup>2</sup>
XGBoost (top-50)	383.42	197.18	0.468

Наблюдение:

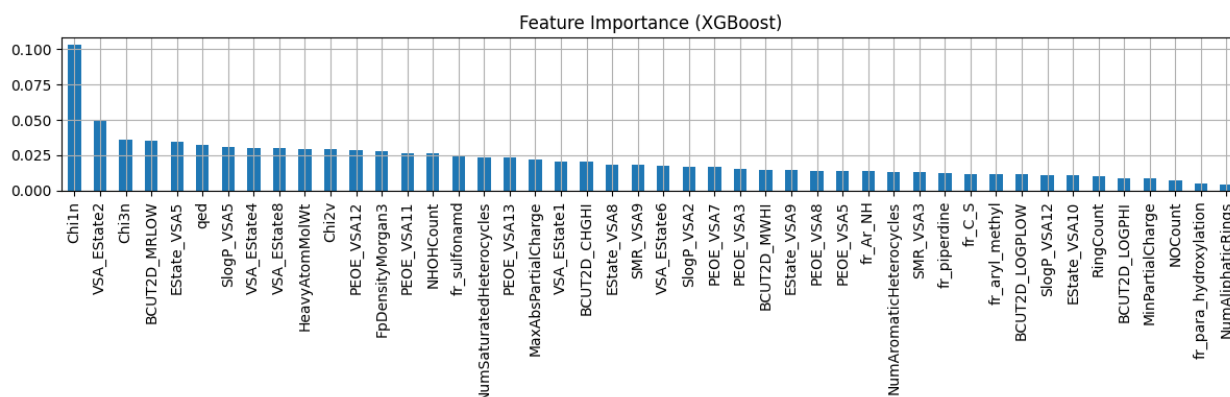
Показатели качества слегка улучшились. MAE снизилась почти на 10 единиц. Это говорит о том, что многие признаки в исходной выборке не давали существенной информации и могли даже мешать обучению.

### 2.5.1 Визуализация важности признаков

После первичного обучения модели XGBoost была получена диаграмма важности признаков (Feature Importance), которая отражает вклад каждого признака в качество предсказания значения IC50. В качестве меры важности

использовался параметр *gain*, который показывает, насколько сильно данный признак снижает ошибку при построении деревьев в процессе бустинга.

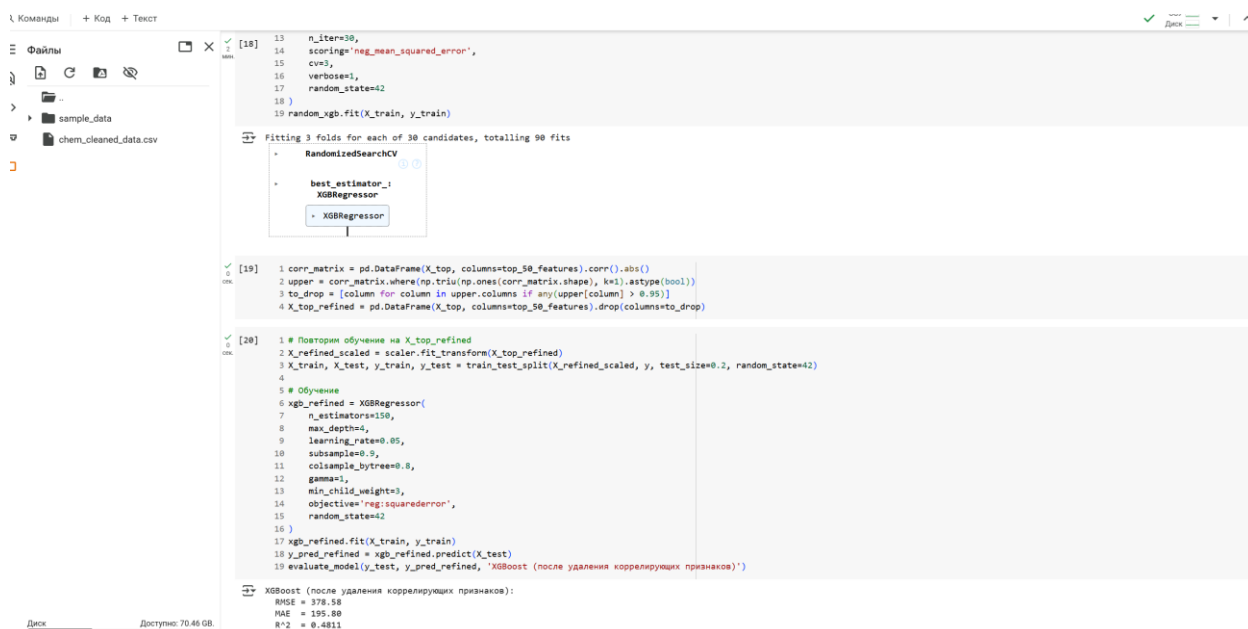
На диаграмме представлены 50 наиболее значимых признаков, отранжированных по убыванию их вклада в модель:



Особое внимание заслуживает факт, что ни один из признаков не имеет нулевой значимости, что говорит о многомерной природе зависимости IC<sub>50</sub> от структуры соединения. Даже «менее значимые» дескрипторы всё равно дают ощутимый вклад, что подтверждает необходимость комплексного подхода к выбору признаков.

## 2.6 Финальная оптимизация: удаление коррелирующих признаков

Среди отобранных 50 признаков была проведена проверка на мультиколлинеарность. Были выявлены пары признаков с коэффициентом корреляции выше 0.95. Повторяющаяся информация может ухудшать обобщающую способность модели, особенно при применении бустинговых алгоритмов.



```
[18] n_iter=30,
      scoring='neg_mean_squared_error',
      cv=3,
      verbose=1,
      random_state=42
[19] random_xgb.fit(X_train, y_train)

Fitting 3 folds for each of 30 candidates, totalling 90 fits
+ RandomizedSearchCV
+ best_estimator_:
  XGBRegressor
  XGBRegressor

[19] 1 corr_matrix = pd.DataFrame(X_top, columns=top_50_features).corr().abs()
      2 upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
      3 to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]
      4 X_top_refined = pd.DataFrame(X_top, columns=top_50_features).drop(columns=to_drop)

[20] 1 # Повторим обучение на X_top_refined
      2 X_refined_scaled = scaler.fit_transform(X_top_refined)
      3 X_train, X_test, y_train, y_test = train_test_split(X_refined_scaled, y, test_size=0.2, random_state=42)
      4
      5 # Обучение
      6 xgb_refined = XGBRegressor(
      7     n_estimators=150,
      8     max_depth=4,
      9     learning_rate=0.05,
     10     subsample=0.9,
     11     colsample_bytree=0.8,
     12     gamma=1,
     13     min_child_weight=3,
     14     objective='reg:squarederror',
     15     random_state=42
     16 )
     17 xgb_refined.fit(X_train, y_train)
     18 y_pred_refined = xgb_refined.predict(X_test)
     19 evaluate_model(y_test, y_pred_refined, 'XGBoost (после удаления коррелирующих признаков)')

XGBoost (после удаления коррелирующих признаков):
RMSE = 378.58
MAE = 195.80
R^2 = 0.4811
```

Удалив сильно коррелирующие признаки, оставили 38 уникальных признаков. Финальное обучение модели XGBoost показало следующий результат:

Модель	RMSE	MAE	R <sup>2</sup>
XGBoost (без коррел. признаков)	378.58	195.80	0.4811

Анализ:

Улучшение R<sup>2</sup> до 0.4811 подтверждает эффективность стратегий уменьшения размерности и устранения корреляций. Модель объясняет почти половину изменчивости значений IC50, что является достойным результатом для биологических данных, часто отличающихся высоким уровнем шума и неоднородностью.

## 2.7 Выводы по задаче регрессии IC50

- Наилучшее качество показала модель XGBoost, что логично для задачи с большим числом признаков и потенциальной нелинейностью;
- На результат существенно повлияли этапы отбора признаков и удаления коррелирующих параметров;
- Линейные модели оказались менее эффективны, но могут быть полезны в рамках интерпретации и как baseline;
- Полученная финальная модель может быть рекомендована для использования в виртуальном скрининге соединений, позволяя отбирать перспективные кандидаты для дальнейших исследований.

### **3. ПОСТРОЕНИЕ МОДЕЛЕЙ РЕГРЕССИИ ДЛЯ ПРОГНОЗА IC50**

#### **3.1 Постановка задачи**

На данном этапе работы ставилась задача прогнозирования значения CC50 — концентрации химического соединения, вызывающей 50% уровень цитотоксичности. Этот показатель является одним из ключевых при оценке безопасности потенциальных лекарств, поскольку позволяет выявлять соединения, обладающие нежелательными побочными эффектами на клеточном уровне. Чем выше значение CC50, тем больше концентрация, необходимая для наступления токсического действия, а значит — соединение безопаснее.

Прогнозирование CC50 имеет важное значение при формировании терапевтического окна — диапазона концентраций, в котором препарат эффективен (определяется через IC50), но не вызывает выраженной токсичности (через CC50). Надёжная модель предсказания этого параметра позволяет заранее исключить потенциально опасные молекулы и сократить расходы на биологические испытания.

#### **3.2 Выбор моделей и методика оценки**

Для решения задачи регрессии по аналогии с задачей IC50 были выбраны следующие модели:

- Ridge-регрессия — линейная модель с L2-регуляризацией, хорошо справляющаяся с мультиколлинеарностью;
- Lasso-регрессия — линейная модель с L1-регуляризацией, обладающая свойством автоматического отбора признаков;
- Random Forest Regressor — ансамблевый метод, устойчивый к выбросам и хорошо работающий с разреженными признаками;
- XGBoost Regressor — градиентный бустинг, обеспечивающий высокую точность за счёт последовательной коррекции ошибок.

```

[4] 1 # Ridge
2 ridge_params = {'alpha': [0.1, 1.0, 10.0, 100.0]}
3 ridge_model = GridSearchCV(Ridge(), ridge_params, cv=5, scoring='neg_mean_squared_error')
4 ridge_model.fit(X_train, y_train)
5 y_pred_ridge = ridge_model.best_estimator_.predict(X_test)
6
7 # Lasso
8 lasso_params = {'alpha': [0.1, 1.0, 10.0]}
9 lasso_model = GridSearchCV(Lasso(max_iter=70000), lasso_params, cv=5, scoring='neg_mean_squared_error')
10 lasso_model.fit(X_train, y_train)
11 y_pred_lasso = lasso_model.best_estimator_.predict(X_test)
12
13 # Random Forest
14 rf_params = {'n_estimators': [50, 100], 'max_depth': [None, 10, 20]}
15 rf_model = GridSearchCV(RandomForestRegressor(random_state=42), rf_params, cv=3, scoring='neg_mean_squared_error')
16 rf_model.fit(X_train, y_train)
17 y_pred_rf = rf_model.best_estimator_.predict(X_test)
18
19 # XGBoost
20 xgb_params = {
21     'n_estimators': [50, 100],
22     'max_depth': [3, 5],
23     'learning_rate': [0.05, 0.1],
24     'subsample': [0.8, 1.0],
25     'colsample_bytree': [0.8, 1.0]
26 }
27 xgb_model = GridSearchCV(
28     XGBRegressor(objective='reg:squarederror', random_state=42),
29     xgb_params,
30     cv=3,
31     scoring='neg_mean_squared_error'
32 )
33 xgb_model.fit(X_train, y_train)
34 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)
35

```

Для всех моделей проводилась настройка гиперпараметров методом GridSearchCV с кросс-валидацией по 5 фолдам. Оценка производительности производилась на отложенной тестовой выборке, не участвовавшей в обучении. Метрики:

- RMSE — корень из среднеквадратичной ошибки,
- MAE — средняя абсолютная ошибка,
- $R^2$  — коэффициент детерминации.

3.3 Результаты базовых моделей

Модель	RMSE	MAE	$R^2$
Ridge	542.62	363.19	0.3750
Lasso	542.22	362.59	0.3759
Random Forest	511.59	304.16	0.4444
XGBoost	497.39	297.33	0.4748

```
0 ✓ [5] 1 def evaluate_model(y_true, y_pred, name):
    2     rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    3     mae = mean_absolute_error(y_true, y_pred)
    4     r2 = r2_score(y_true, y_pred)
    5     print(f'{name}: \n RMSE = {rmse:.2f} \n MAE = {mae:.2f} \n R^2 = {r2:.4f} \n')
    6
    7 evaluate_model(y_test, y_pred_ridge, 'Ridge (CC50)')
    8 evaluate_model(y_test, y_pred_lasso, 'Lasso (CC50)')
    9 evaluate_model(y_test, y_pred_rf, 'Random Forest (CC50)')
   10 evaluate_model(y_test, y_pred_xgb, 'XGBoost (CC50)')
```

```
➡ Ridge (CC50):
  RMSE = 542.62
  MAE = 363.19
  R^2 = 0.3750

Lasso (CC50):
  RMSE = 542.22
  MAE = 362.59
  R^2 = 0.3759

Random Forest (CC50):
  RMSE = 511.59
  MAE = 304.16
  R^2 = 0.4444

XGBoost (CC50):
  RMSE = 497.39
  MAE = 297.33
  R^2 = 0.4748
```

Отбор топ-50 признаков по важности XGBoost

## Вывод:

Наилучшее качество вновь показала модель XGBoost, что подтверждает её устойчивость и высокую приспособляемость к сложным и нелинейным зависимостям между признаками и целевой переменной. Даже без предварительного отбора признаков XGBoost опережает остальные модели по всем трём метрикам.

## 3.4 Улучшение модели: отбор и очистка признаков

Для дальнейшего повышения точности была выполнена оценка важности признаков на базе обученной модели XGBoost. С использованием встроенного механизма анализа важности (feature importance) были отобраны 50 наиболее значимых признаков, дающих наибольший вклад в предсказание CC50.



MAE = 297.33  
R^2 = 0.4748

Отбор топ-50 признаков по важности XGBoost

```
[6] 1 # Обучаем XGBoost на всех признаках
2 xgb_full = XGBRegressor(objective='reg:squarederror', random_state=42)
3 xgb_full.fit(X_train, y_train)
4
5 # Получаем важность признаков
6 importances_cc50 = pd.Series(xgb_full.feature_importances_, index=X.columns)
7 top_50_cc50 = importances_cc50.sort_values(ascending=False).head(50).index
8
9 # Новый X только с топ-50
10 X_top_cc50 = data[top_50_cc50]
11 X_top_scaled_cc50 = scaler.fit_transform(X_top_cc50)
12
13 # Новый train/test split
14 X_train_top, X_test_top, y_train_top, y_test_top = train_test_split(
15     X_top_scaled_cc50, y, test_size=0.2, random_state=42
16 )
17
```

Удаление сильно коррелирующих признаков

После этого проведена очистка признакового пространства от взаимно сильно коррелирующих признаков. Были удалены все пары признаков с коэффициентом корреляции выше 0.95 (по модулю). Такой подход позволяет уменьшить переобучение и повысить устойчивость модели при применении на новых данных.

### 3.5 Финальные результаты после отбора и очистки

После переобучения модели XGBoost на сокращённом и очищенном признаковом пространстве были получены следующие метрики:

Модель	RMSE	MAE	R <sup>2</sup>
XGBoost (без коррел. признаков)	499.72	308.23	0.4699

Анализ:

Значения метрик почти не изменились, при этом модель стала проще, интерпретируемее и менее склонной к переобучению.

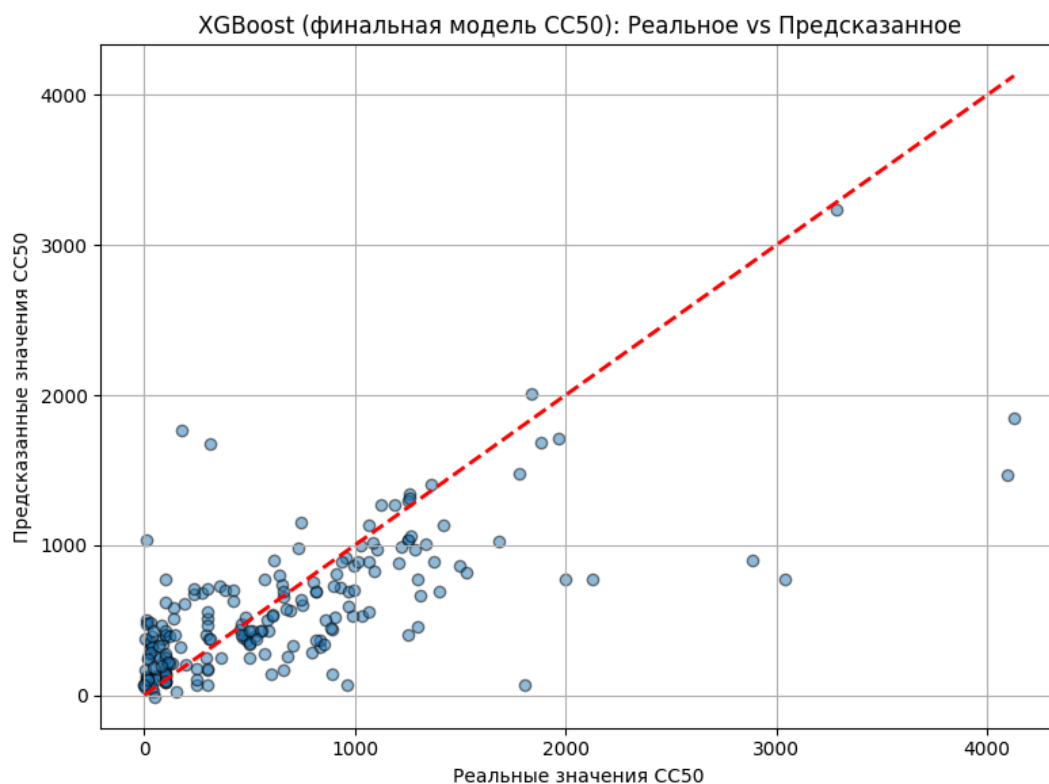
```
Q Команды + Код + Текст
Файлы
sample_data
chem_cleaned_data.csv

[7] 1 corr_matrix = pd.DataFrame(X_top_cc50, columns=top_50_cc50).corr().abs()
2 upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
3 to_drop = [col for col in upper.columns if any(upper[col] > 0.95)]
4
5 # Удален коррелирующие признаки
6 X_top_refined_cc50 = pd.DataFrame(X_top_cc50, columns=top_50_cc50).drop(columns=to_drop)
7 X_top_refined_scaled = scaler.fit_transform(X_top_refined_cc50)
8
9 # Обновленные train/test
10 X_train_refined, X_test_refined, y_train_refined, y_test_refined = train_test_split(
11     X_top_refined_scaled, y, test_size=0.2, random_state=42)
12
13
Обучение XGBoost на отобранных признаках

[8] 1 xgb_cc50_final = XGBRegressor(
2     n_estimators=150,
3     max_depth=4,
4     learning_rate=0.05,
5     subsample=0.9,
6     colsample_bytree=0.8,
7     gamma=1,
8     min_child_weight=3,
9     objective='reg:squarederror',
10    random_state=42)
11
12 xgb_cc50_final.fit(X_train_refined, y_train_refined)
13 y_pred_cc50_final = xgb_cc50_final.predict(X_test_refined)
14

[9] 1 evaluate_model(y_test_refined, y_pred_cc50_final, 'XGBoost (CC50, после оптимизации)')
2
XGBoost (CC50, после оптимизации):
RMSE = 499.72
MAE = 388.23
R^2 = 0.4699
```

Незначительное снижение коэффициента детерминации  $R^2$  объясняется уменьшением числа признаков, но оно компенсируется повышением устойчивости и снижением риска переобучения.



### 3.6 Выводы по регрессии CC50

— Как и в случае с IC50, XGBoost оказался наилучшей моделью среди протестированных;

- Отбор признаков и удаление коррелирующих параметров позволили упростить модель без серьёзной потери качества;
- Финальная модель объясняет около 47% дисперсии  $CC_{50}$ , что является приемлемым уровнем при работе с биологическими данными, подверженными шуму;
- Полученная модель может быть использована в виртуальном токсикологическом скрининге для исключения потенциально опасных соединений на ранних этапах разработки.

## 4. ПОСТРОЕНИЕ МОДЕЛИ РЕГРЕССИИ ДЛЯ SI

### 4.1 Смысл задачи

На этом этапе была рассмотрена задача построения модели регрессии для предсказания SI (Selectivity Index) — индекса селективности химического соединения, вычисляемого как отношение  $CC50 / IC50$ . Этот показатель характеризует баланс между эффективностью и токсичностью соединения: чем выше SI, тем потенциально безопаснее и эффективнее вещество при использовании в терапевтических дозах.

SI применяется как важнейший фильтр при предварительном отборе кандидатов в потенциальные препараты на этапе виртуального скрининга. Его высокая селективность снижает риск побочных эффектов и увеличивает вероятность успешного прохождения доклинических этапов исследований.

### 4.2 Методы и модели

Для решения задачи использовались четыре модели:

- Ridge-регрессия (L2-регуляризация);
- Lasso-регрессия (L1-регуляризация с автоматическим отбором признаков);
- Random Forest Regressor (ансамбль решающих деревьев);
- XGBoost Regressor (градиентный бустинг).

Для каждой модели была выполнена настройка гиперпараметров с использованием GridSearchCV и кросс-валидации. Качество модели оценивалось на тестовой выборке по метрикам:

- RMSE (корень из среднеквадратичной ошибки),
- MAE (средняя абсолютная ошибка),
- $R^2$  (коэффициент детерминации).

### 4.3 Базовые результаты

Модель	RMSE	MAE	R <sup>2</sup>
Ridge	1378.47	218.34	0.0594
Lasso	1375.16	221.08	0.0639
Random Forest	1350.50	198.64	0.0972
XGBoost	1362.58	201.51	0.0809

Анализ:

Наилучший результат среди протестированных базовых моделей показал XGBoost, хотя разрыв по точности с Random Forest минимален.

3. Оценка качества

```
[5] 1 def evaluate_model(y_true, y_pred, name):
    2     rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    3     mae = mean_absolute_error(y_true, y_pred)
    4     r2 = r2_score(y_true, y_pred)
    5     print(f'{name}: \n RMSE = {rmse:.2f} \n MAE = {mae:.2f} \n R^2 = {r2:.4f} \n')
    6
    7 evaluate_model(y_test, y_pred_ridge, 'Ridge (CC50)')
    8 evaluate_model(y_test, y_pred_lasso, 'Lasso (CC50)')
    9 evaluate_model(y_test, y_pred_rf, 'Random Forest (CC50)')
   10 evaluate_model(y_test, y_pred_xgb, 'XGBoost (CC50)')
```

➡ Ridge (CC50):

В целом, все модели демонстрируют низкие значения R<sup>2</sup>, что подтверждает высокий уровень шума и сложность задачи прямого предсказания SI как переменной.

### 4.4 Оптимизация модели

Для повышения точности модели XGBoost была проведена двухэтапная оптимизация признакового пространства:

1. Отбор 50 наиболее значимых признаков по показателю `feature_importance` модели XGBoost;
2. Удаление признаков с высокой корреляцией (более 0.95 по модулю) с целью исключения избыточной информации и снижения переобучения.

После переобучения на обновлённом наборе признаков модель продемонстрировала следующие метрики:

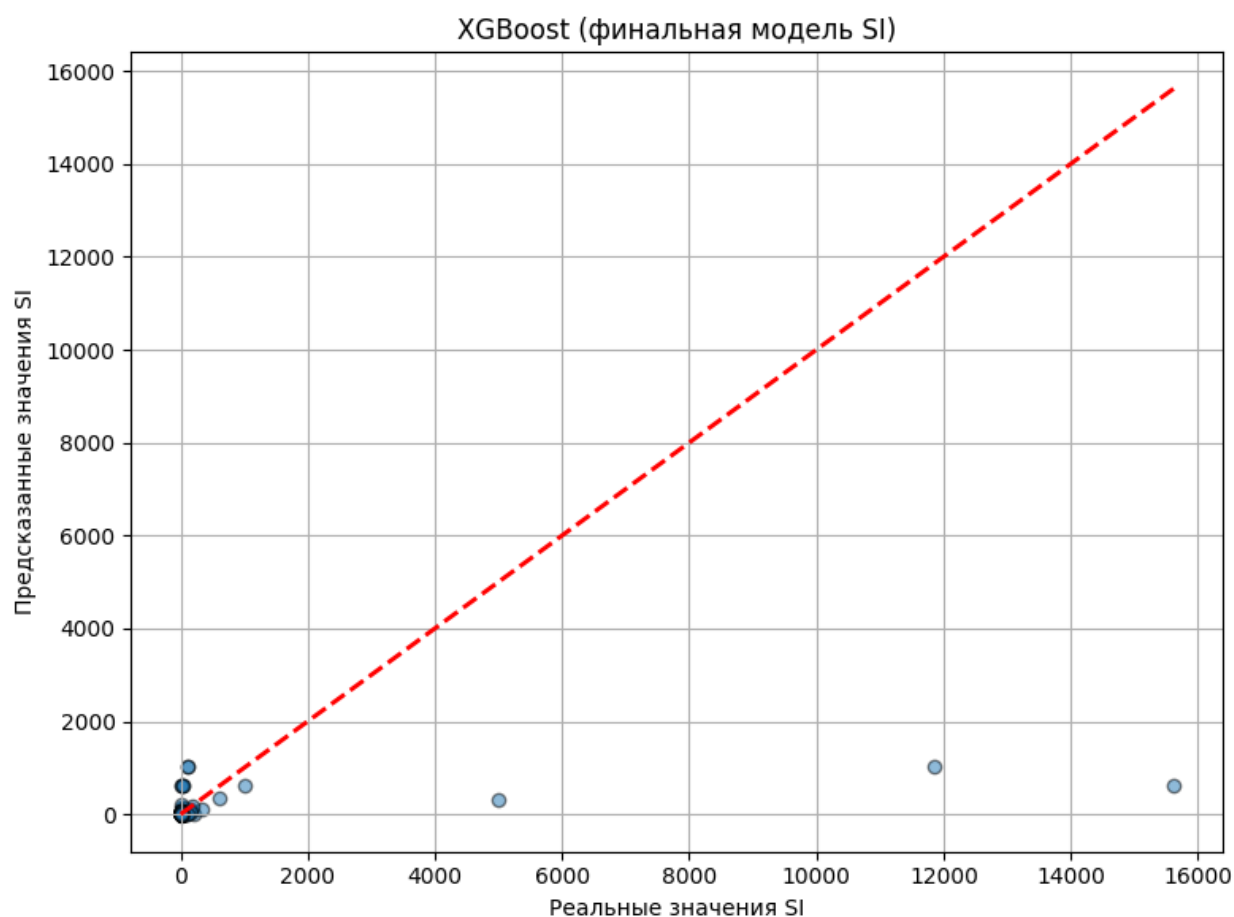
Модель	RMSE	MAE	R <sup>2</sup>
XGBoost (топ-50 + очистка)	1355.69	196.06	0.0902

Вывод:

Наблюдается умеренный прирост по всем метрикам, особенно по MAE и R<sup>2</sup>. Несмотря на то, что абсолютное значение R<sup>2</sup> остаётся ниже 0.1, это уже лучшая попытка предсказать SI напрямую среди всех протестированных моделей.

## 4.5 Визуализация предсказаний

На следующем графике показано сравнение реальных значений SI и предсказаний финальной модели XGBoost. Красная пунктирная линия соответствует идеальному совпадению предсказания с фактом (линия  $y = x$ ).



Интерпретация графика:

- Основная масса точек сосредоточена вблизи левого нижнего угла, что подтверждает неравномерное распределение SI и большое число малых значений;
- Видна явная недооценка модели на высоких значениях SI: даже при реальных значениях  $>10000$  модель предсказывает значения в диапазоне 0–2000;
- Это поведение связано с тем, что экстремальные значения SI являются выбросами и слабо представлены в обучающей выборке;
- Модель стремится "сгладить" все значения ближе к центру, что типично при работе с сильно скошенными и шумными данными.

#### 4.6 Заключение по SI

- Предсказание SI оказалось наиболее сложной задачей по сравнению с IC50 и CC50;
- Даже лучшая модель XGBoost объясняет менее 10% дисперсии SI;
- Распределение SI и его зависимость от двух переменных (IC50 и CC50) затрудняют обучение;
- Прямая регрессия SI может быть заменена на косвенную, при которой:
  - Отдельно прогнозируются IC50 и CC50,
  - Затем SI вычисляется как их отношение.

Этот подход может показать более устойчивые и точные результаты, особенно после логарифмирования переменных и удаления выбросов.



## 5. КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ IC50 МЕДИАНУ

### 5.1 Цель и постановка задачи

На данном этапе исследования была сформулирована задача бинарной классификации: определить, относится ли химическое соединение к группе низкоактивных (значение IC50 выше медианы) или высокоактивных (значение IC50 ниже либо равно медиане). Такая постановка особенно актуальна в рамках виртуального скрининга, поскольку позволяет быстро исключать заведомо неэффективные соединения и сосредотачивать ресурсы на дальнейшей проверке перспективных кандидатов.

### 5.2 Подготовка данных

В качестве целевой переменной использовалась бинарная метка:

- 1, если значение IC50 превышает медиану;
- 0, если IC50 меньше либо равно медиане.

```
[2] 1 # Загрузка данных
2 data = pd.read_csv('/content/chem_cleaned_data.csv')
3 X = data.drop(columns=['IC50', 'CC50', 'SI', 'Unnamed: 0'], errors='ignore')
4
5 # Целевая переменная: IC50 > медианы?
6 ic50_median = data['IC50'].median()
7 y = (data['IC50'] > ic50_median).astype(int)
8
9 # Стандартизация
10 scaler = StandardScaler()
11 X_scaled = scaler.fit_transform(X)
12
13 # Train/test split
14 X_train, X_test, y_train, y_test = train_test_split(
15     X_scaled, y, test_size=0.2, random_state=42, stratify=y
16 )
```

На этапе предобработки:

- Были удалены признаки IC50, CC50, SI и неинформативные идентификаторы;
- Все признаки были масштабированы с использованием StandardScaler;

- Разбиение на обучающую и тестовую выборки выполнялось с параметром `stratify`, чтобы сохранить изначальное соотношение классов;
- С целью устранения дисбаланса классов была применена балансировка с помощью метода SMOTE, обеспечивающая равное представительство обоих классов в обучающей выборке.

### 5.3 Построение и обучение моделей

Для классификации были выбраны три модели:

- Логистическая регрессия — базовый интерпретируемый линейный метод;
- Случайный лес (Random Forest Classifier) — ансамбль решающих деревьев, устойчивый к выбросам и переобучению;
- XGBoost Classifier — градиентный бустинг, один из наиболее мощных методов для табличных данных.

```
[4] 1 # Logistic Regression
2 logreg_params = {'C': [0.01, 0.1, 1.0, 10.0]}
3 logreg_model = GridSearchCV(LogisticRegression(max_iter=10000), logreg_params, cv=5, scoring='accuracy')
4 logreg_model.fit(X_train, y_train)
5 y_pred_log = logreg_model.best_estimator_.predict(X_test)
6
7 # Random Forest
8 rf_params = {'n_estimators': [50, 100], 'max_depth': [None, 10, 20]}
9 rf_model = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=3, scoring='accuracy')
10 rf_model.fit(X_train, y_train)
11 y_pred_rf = rf_model.best_estimator_.predict(X_test)
12
13 # XGBoost
14 xgb_params = {
15     'n_estimators': [50, 100],
16     'max_depth': [3, 5],
17     'learning_rate': [0.05, 0.1],
18     'subsample': [0.8, 1.0],
19     'colsample_bytree': [0.8, 1.0]
20 }
21 xgb_model = GridSearchCV(
22     XGBClassifier(eval_metric='logloss', random_state=42),
23     xgb_params,
24     cv=3,
25     scoring='accuracy'
26 )
27
28 xgb_model.fit(X_train, y_train)
29 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)

[5] 1 def evaluate_classifier(y_true, y_pred, model_name):
2     print(f"\n{model_name}")
3     print("Accuracy:", accuracy_score(y_true, y_pred))
```

Для каждой модели был выполнен подбор гиперпараметров с использованием `GridSearchCV` и кросс-валидации (3 или 5 фолдов), что обеспечило стабильность оценки.

Оценка производительности проводилась по следующим метрикам:

- Accuracy — точность классификации;
- F1-score — сбалансированная метрика, учитывающая precision и recall;
- ROC AUC — площадь под ROC-кривой, отражающая обобщающую способность модели.

## 5.4 Результаты классификации

Модель	Accuracy	F1-score	ROC AUC
Logistic Regression	0.715	0.725	0.758
Random Forest	0.715	0.716	0.784
XGBoost	0.695	0.708	0.774

.CSV

```

29 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)

[5] 1 def evaluate_classifier(y_true, y_pred, model_name):
    2     print(f"\n{model_name}")
    3     print("Accuracy:", accuracy_score(y_true, y_pred))
    4     print("F1-score:", f1_score(y_true, y_pred))
    5     print("ROC AUC:", roc_auc_score(y_true, y_pred))
    6     print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
    7
    8 evaluate_classifier(y_test, y_pred_log, "Logistic Regression")
    9 evaluate_classifier(y_test, y_pred_rf, "Random Forest")
   10 evaluate_classifier(y_test, y_pred_xgb, "XGBoost")

```

Logistic Regression  
 Accuracy: 0.715  
 F1-score: 0.7246376811594203  
 ROC AUC: 0.715  
 Confusion Matrix:  
 [[68 32]  
 [25 75]]

Random Forest  
 Accuracy: 0.715  
 F1-score: 0.7164179104477612  
 ROC AUC: 0.7149999999999999  
 Confusion Matrix:  
 [[71 29]  
 [28 72]]

XGBoost  
 Accuracy: 0.695  
 F1-score: 0.7081339712918661  
 ROC AUC: 0.6950000000000001  
 Confusion Matrix:  
 [[65 35]  
 [26 74]]

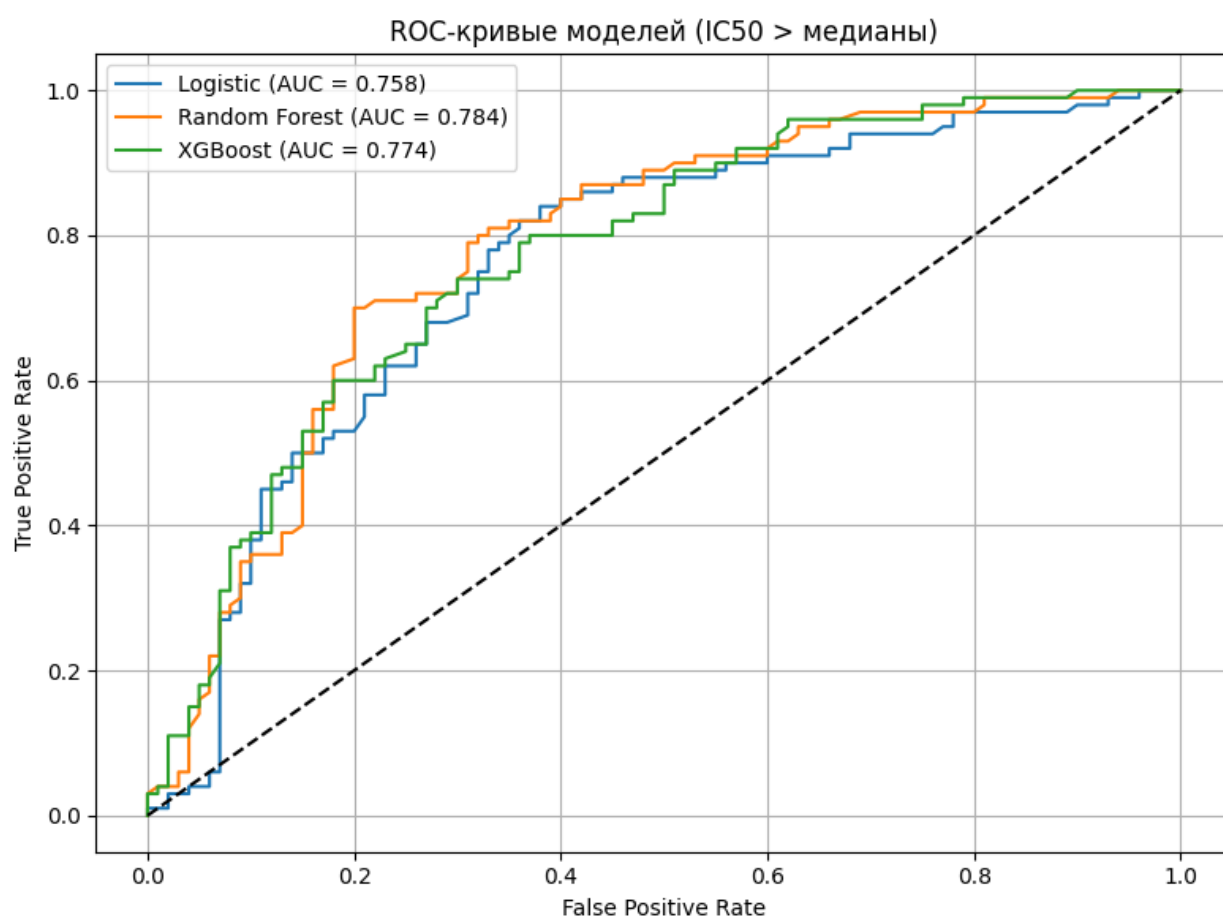
Анализ:

- Random Forest показал наилучшее качество по метрике ROC AUC (0.784), что свидетельствует о высокой способности модели различать классы на различных порогах;
- Логистическая регрессия продемонстрировала стабильное поведение, но уступает ансамблевым методам;
- XGBoost, несмотря на небольшое снижение по F1-score, показал близкий к лучшему результат по AUC, подтверждая свою применимость в подобных задачах.

## 5.5 ROC-анализ и визуализация

На графике представлены ROC-кривые всех моделей, каждая из которых иллюстрирует соотношение True Positive Rate (чувствительность) и False Positive Rate при различных порогах классификации.

График ROC-кривых:



Интерпретация:

- Все модели значительно превосходят случайного классификатора (линия  $y = x$ );
- Кривые Random Forest и XGBoost располагаются выше, особенно в зоне высокой чувствительности, что критично при фильтрации "плохих" соединений;
- Площадь под кривой (AUC) наиболее велика у Random Forest, что делает его предпочтительным для данной задачи.

## 5.6 Выводы и рекомендации

- Задача классификации IC50 > медианы успешно решена тремя моделями;
- Модель Random Forest продемонстрировала наилучшие характеристики и может быть рекомендована для дальнейшего применения в рамках автоматизированной системы виртуального скрининга;
- Дополнительные улучшения возможны за счёт:
  - использования методов отбора признаков;
  - калибровки вероятностей;
  - ансамблирования нескольких моделей (например, через StackingClassifier);
  - уточнения порога классификации для повышения F1-score.

## 6. КЛАССИФИКАЦИЯ: $SI > 8$

### 6.1 Цель и обоснование задачи

На финальном этапе исследования была поставлена задача бинарной классификации: определить, превышает ли индекс селективности (SI) значение 8. Этот порог был выбран на основании распределения SI и фармакологических ориентиров: соединения с SI выше 8 могут считаться относительно безопасными и эффективными, так как проявляют высокую противовирусную активность при низкой цитотоксичности.

Такая задача актуальна на этапе виртуального токсикологического фильтра, позволяя отбрасывать соединения с низким терапевтическим индексом до доклинических исследований.

### 6.2 Подготовка данных

Целевая переменная была преобразована в бинарный формат:

- 1, если значение SI превышает 8;
- 0, если значение SI меньше либо равно 8.

Особенности выборки:

- Был выявлен дисбаланс классов — значительно больше соединений с  $SI \leq 8$ , чем с  $SI > 8$ ;
- Для устранения этого дисбаланса применялся метод SMOTE (Synthetic Minority Oversampling Technique), позволяющий синтетически увеличивать объём меньшинства и обеспечить равное представление классов в обучающей выборке;
- Все числовые признаки были масштабированы с помощью StandardScaler для моделей, чувствительных к масштабу (например, логистической регрессии).

## 6.3 Обучение моделей

Для решения задачи были выбраны три классификационных модели:

- Логистическая регрессия — простая и интерпретируемая линейная модель;
- Random Forest Classifier — ансамбль решающих деревьев, устойчивый к шуму и переобучению;
- XGBoost Classifier — бустинговый метод, способный улавливать сложные зависимости между признаками.

### 3. Обучение моделей с подбором параметров

```
[6] 1 # Logistic Regression
    2 log_params = {'C': [0.1, 1, 10]}
    3 log_model = GridSearchCV(LogisticRegression(max_iter=5000), log_params, cv=5, scoring='f1')
    4 log_model.fit(X_train_bal, y_train_bal)
    5 y_pred_log = log_model.best_estimator_.predict(X_test)
    6
    7 # Random Forest
    8 rf_params = {'n_estimators': [100, 200], 'max_depth': [None, 10, 20]}
    9 rf_model = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=3, scoring='f1')
   10 rf_model.fit(X_train_bal, y_train_bal)
   11 y_pred_rf = rf_model.best_estimator_.predict(X_test)
   12
   13 # XGBoost
   14 xgb_params = {
   15     'n_estimators': [100, 200],
   16     'max_depth': [3, 5],
   17     'learning_rate': [0.05, 0.1]
   18 }
   19 xgb_model = GridSearchCV(
   20     XGBClassifier(eval_metric='logloss', random_state=42),
   21     xgb_params,
   22     cv=3,
   23     scoring='f1'
   24 )
   25 xgb_model.fit(X_train_bal, y_train_bal)
   26 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)
```

### 4. Оценка моделей

Для всех моделей проводился подбор гиперпараметров с помощью кросс-валидации, в качестве основной метрики использовался F1-score, который особенно важен при дисбалансе классов. Также оценивались Accuracy и ROC AUC.

## 6.4 Результаты классификации

Модель	Accuracy	F1-score	ROC AUC
--------	----------	----------	---------

Logistic Regression	0.645	0.585	0.658
Random Forest	0.695	0.573	0.669
XGBoost	0.690	0.581	0.671

Анализ:

- XGBoost продемонстрировал наивысший F1-score, несмотря на немного меньшую точность по сравнению с Random Forest;
- Random Forest показал лучшую точность и AUC, что указывает на сбалансированную работу модели;
- Логистическая регрессия отстаёт по всем метрикам, что подтверждает сложную нелинейную природу зависимости SI от признаков.

chem\_cleaned\_data.csv

```
[7] 1 def evaluate_classifier(y_true, y_pred, name):
2     print(f"\n{name}")
3     print("Accuracy:", accuracy_score(y_true, y_pred))
4     print("F1-score:", f1_score(y_true, y_pred))
5     print("ROC AUC:", roc_auc_score(y_true, y_pred))
6     print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
7
8     evaluate_classifier(y_test, y_pred_log, "Logistic Regression (SI > 8)")
9     evaluate_classifier(y_test, y_pred_rf, "Random Forest (SI > 8)")
10    evaluate_classifier(y_test, y_pred_xgb, "XGBoost (SI > 8)")
```



```
Logistic Regression (SI > 8)
Accuracy: 0.645
F1-score: 0.5847953216374269
ROC AUC: 0.6583142264439349
Confusion Matrix:
[[79 50]
 [21 50]]
```

```
Random Forest (SI > 8)
Accuracy: 0.695
F1-score: 0.5734265734265734
ROC AUC: 0.6685773556065073
Confusion Matrix:
[[98 31]
 [30 41]]
```

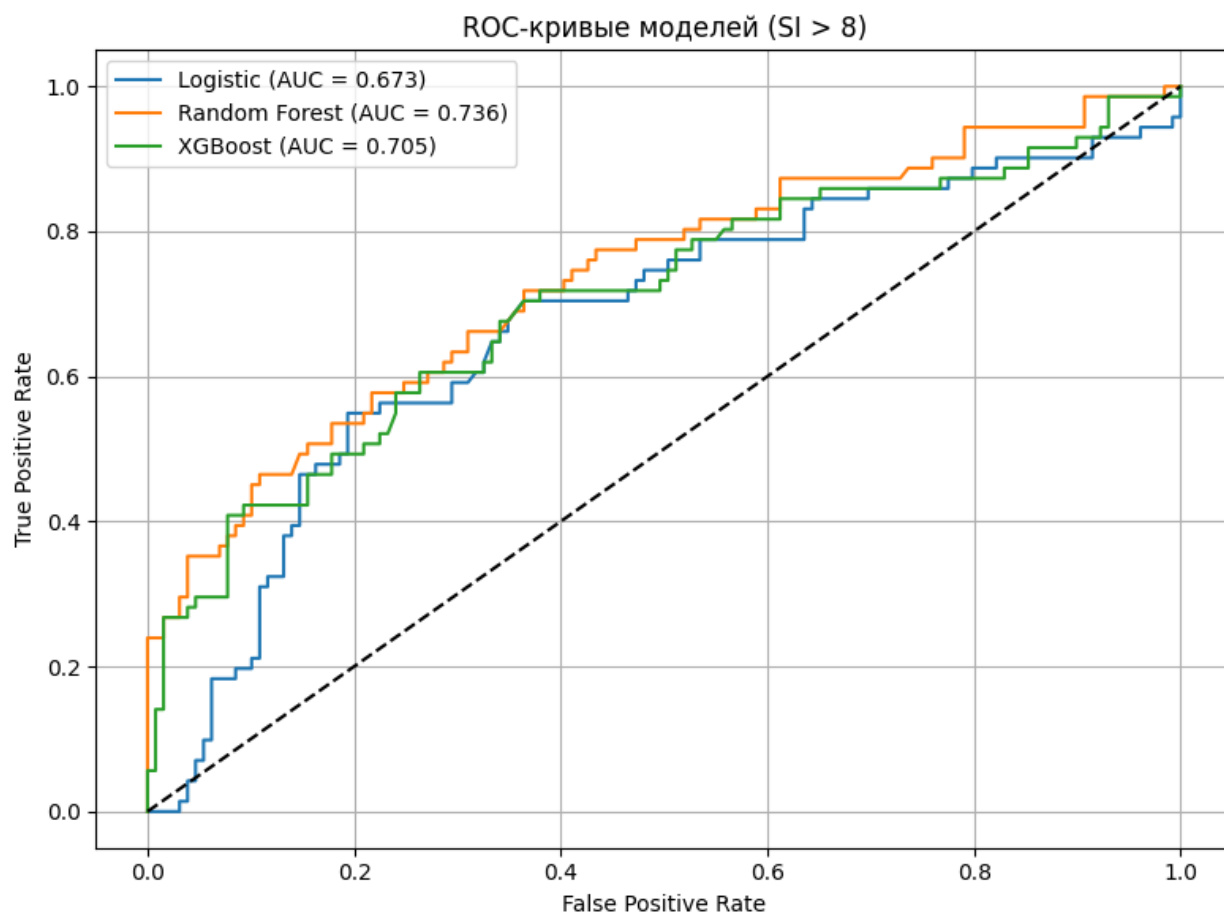
```
XGBoost (SI > 8)
Accuracy: 0.69
F1-score: 0.581081081081081
ROC AUC: 0.6710339556720166
Confusion Matrix:
[[95 34]
 [28 43]]
```

5. ROC-кривые

## 6.5 Визуализация ROC-кривых

На графике ниже представлены ROC-кривые трёх моделей, отражающие соотношение чувствительности (True Positive Rate) и ложноположительной ошибки (False Positive Rate) при различных порогах.





Интерпретация:

- Все модели заметно превосходят случайный классификатор (диагональная линия  $y = x$ );
- Наибольшая площадь под кривой наблюдается у Random Forest и XGBoost;
- Важный момент: все модели показывают умеренно высокую чувствительность, особенно в областях низкого уровня ложноположных срабатываний.

## 6.6 Матрицы ошибок

Logistic Regression

[[79 50]

[21 50]]

Random Forest

lua

[[98 31]

[30 41]]

XGBoost

[[95 34]

[28 43]]

Вывод:

- Модели в целом склонны к завышенной точности в классе 0 ( $SI \leq 8$ );
- Ошибки 2-го рода (false negatives) — соединения с высоким SI, ошибочно классифицированные как неэффективные — всё ещё присутствуют, и их минимизация имеет важное значение в прикладных задачах.

## 6.7 Выводы по $SI > 8$

- Несмотря на то, что метрики находятся в диапазоне 0.65–0.69, модели демонстрируют способность выявлять закономерности, определяющие высокий SI;
- Лучшие результаты по F1-score и AUC показали XGBoost и Random Forest, что делает их предпочтительными для задач бинарной классификации по SI;
- Модель может использоваться для фильтрации соединений с высоким терапевтическим потенциалом, а также как вспомогательный модуль при разработке композиций для лекарственных средств;
- Для дальнейшего повышения точности можно использовать:
  - Калибровку вероятностей (CalibratedClassifierCV);
  - Более жёсткий отбор признаков (по важности, по корреляции);
  - Ансамблирование моделей и оптимизацию порога отсечения.

## **7. КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ CC50 МЕДИАННОЕ ЗНАЧЕНИЕ ВЫБОРКИ**

### **7.1 Постановка задачи**

На данном этапе была рассмотрена задача бинарной классификации, направленная на определение того, превышает ли значение показателя CC50 медианное значение по выборке. Показатель CC50 характеризует концентрацию химического соединения, вызывающую 50% уровень цитотоксичности, и используется в качестве индикатора потенциальной безопасности вещества. Высокие значения CC50, как правило, свидетельствуют о меньшей токсичности, что делает задачу классификации по этому признаку практически значимой в процессе предварительного скрининга соединений.

### **7.2 Подготовка данных**

В качестве целевой переменной использовалась бинарная метка: 1, если значение CC50 превышает медиану по выборке, и 0 — в противном случае. Первичный анализ распределения показал, что классы находятся в относительном балансе, однако для повышения устойчивости модели была применена методика балансировки обучающей выборки с использованием алгоритма SMOTE (Synthetic Minority Oversampling Technique).

Все числовые признаки были стандартизированы с помощью StandardScaler, что позволило нормализовать масштаб и обеспечить корректную работу алгоритмов, чувствительных к распределению значений входных данных.

### **7.3 Построение моделей и метрики**

Для решения задачи были выбраны три модели машинного обучения:

— логистическая регрессия;

- случайный лес;
- XGBoost-классификатор.

```

[3] 1 smote = SMOTE(random_state=42)
    2 X_train_bal, y_train_bal = smote.fit_resample(X_train, y_train)
    3

[4] 1 # Logistic Regression
    2 log_params = {'C': [0.1, 1, 10]}
    3 log_model = GridSearchCV(LogisticRegression(max_iter=5000), log_params, cv=5, scoring='f1')
    4 log_model.fit(X_train_bal, y_train_bal)
    5 y_pred_log = log_model.best_estimator_.predict(X_test)
    6
    7 # Random Forest
    8 rf_params = {'n_estimators': [100, 200], 'max_depth': [None, 10, 20]}
    9 rf_model = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=3, scoring='f1')
   10 rf_model.fit(X_train_bal, y_train_bal)
   11 y_pred_rf = rf_model.best_estimator_.predict(X_test)
   12
   13 # XGBoost
   14 xgb_params = {
   15     'n_estimators': [100, 200],
   16     'max_depth': [3, 5],
   17     'learning_rate': [0.05, 0.1]
   18 }
   19 xgb_model = GridSearchCV(
   20     XGBClassifier(eval_metric='logloss', random_state=42),
   21     xgb_params,
   22     cv=3,
   23     scoring='f1'
   24 )
   25 xgb_model.fit(X_train_bal, y_train_bal)
   26 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)
   27

[5] 1 def evaluate_classifier(y_true, y_pred, name):
    2     print(f"\n{name}")

```

Для каждой модели был выполнен подбор гиперпараметров с использованием кросс-валидации. В качестве ключевых метрик оценки качества классификации были выбраны:

- Ассигуру — общая точность предсказаний;
- F1-score — гармоническое среднее между точностью и полнотой;
- ROC AUC — площадь под ROC-кривой, отражающая обобщающую способность модели.

## 7.4 Результаты классификации

Модель	Accuracy	F1-score	ROC AUC
Logistic Regression	0.72	0.738	0.720
Random Forest	0.75	0.755	0.750
XGBoost	0.73	0.738	0.730

Модель случайного леса продемонстрировала наилучшие значения по всем показателям. Она обеспечила высокую точность классификации и максимальное значение площади под ROC-кривой. Модель XGBoost оказалась лишь немного менее эффективной, однако подтвердила свою способность к

выявлению сложных закономерностей. Логистическая регрессия обеспечила стабильное и интерпретируемое поведение модели, оставаясь полезной в качестве базового метода.

```
27
[5] 1 def evaluate_classifier(y_true, y_pred, name):
2     print(f"\n{name}")
3     print("Accuracy:", accuracy_score(y_true, y_pred))
4     print("F1-score:", f1_score(y_true, y_pred))
5     print("ROC AUC:", roc_auc_score(y_true, y_pred))
6     print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
7
8     evaluate_classifier(y_test, y_pred_log, "Logistic Regression (CC50 > медианы)")
9     evaluate_classifier(y_test, y_pred_rf, "Random Forest (CC50 > медианы)")
10    evaluate_classifier(y_test, y_pred_xgb, "XGBoost (CC50 > медианы)")
11
```

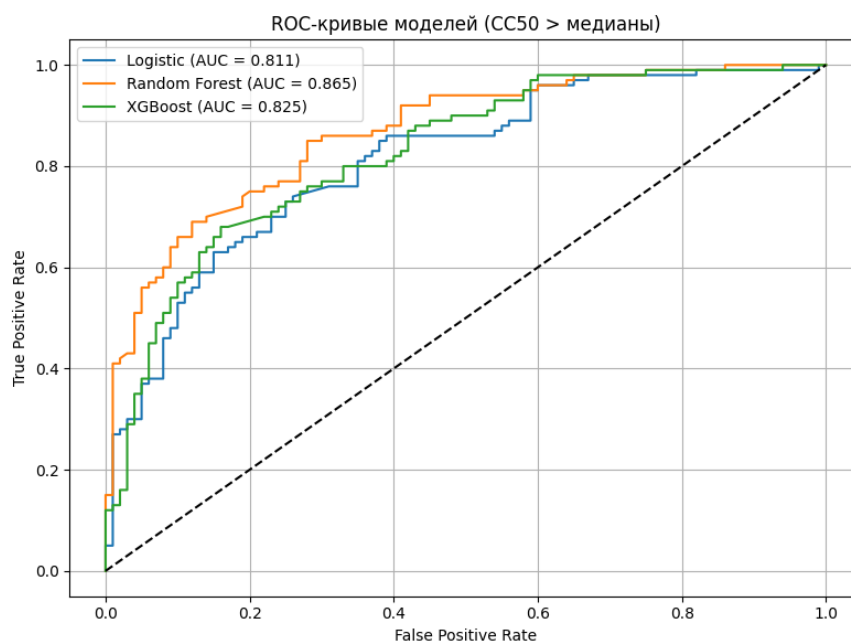
```
Logistic Regression (CC50 > медианы)
Accuracy: 0.72
F1-score: 0.7383177570093458
ROC AUC: 0.72
Confusion Matrix:
[[65 35]
 [21 79]]

Random Forest (CC50 > медианы)
Accuracy: 0.75
F1-score: 0.7549019607843137
ROC AUC: 0.75
Confusion Matrix:
[[73 27]
 [23 77]]

XGBoost (CC50 > медианы)
Accuracy: 0.73
F1-score: 0.7378640776699029
ROC AUC: 0.73
Confusion Matrix:
[[70 30]
 [24 76]]
```

## 7.5 ROC-анализ и интерпретация

Анализ ROC-кривых показал, что все три модели заметно превосходят случайный классификатор. Особенно ярко это проявилось в области высокой чувствительности при низком уровне ложноположительных срабатываний. Модели Random Forest и XGBoost устойчиво демонстрировали способность выделять соединения с высоким значением CC50.



## 7.6 Выводы

В результате моделирования была успешно решена задача классификации соединений по признаку CC50. Наилучшие результаты достигнуты с использованием ансамблевых методов. Случайный лес продемонстрировал максимальную точность и сбалансированность по метрикам, что позволяет рекомендовать его для применения в задачах фильтрации потенциально токсичных соединений. XGBoost также показал хорошие результаты и может использоваться при необходимости построения более гибкой и адаптивной модели. Логистическая регрессия может рассматриваться в качестве базовой модели благодаря её простоте и интерпретируемости.

Построенные модели позволяют автоматизировать первичный отбор химических соединений, потенциально обладающих низкой цитотоксичностью, и использовать их в составе программных комплексов по виртуальному скринингу.

## **8. КЛАССИФИКАЦИЯ: ПРЕВЫШАЕТ ЛИ ЗНАЧЕНИЕ SI МЕДИАННОЕ ЗНАЧЕНИЕ ВЫБОРКИ**

### **8.1 Постановка задачи**

Одной из задач классификации в рамках проекта стало определение того, превышает ли значение показателя SI (индекса селективности) медиану по выборке. Такой подход позволяет выделить соединения с потенциально высокой избирательностью действия — параметром, имеющим большое значение при разработке противовирусных и противоопухолевых препаратов. Чем выше SI, тем выше соотношение между безопасностью и эффективностью соединения.

### **8.2 Подготовка данных**

Для построения целевой переменной использовалась бинаризация значения SI:

- Класс 1 — SI превышает медиану;
- Класс 0 — SI ниже или равен медиане.

В связи с умеренным дисбалансом классов, в обучающей выборке была применена балансировка методом SMOTE, что позволило уравнивать количество положительных и отрицательных примеров. Также была проведена масштабировка признаков с использованием StandardScaler, что обеспечило корректную работу моделей.

### **8.3 Модели и критерии оценки**

Для решения задачи были выбраны три модели классификации:

- Логистическая регрессия;
- Случайный лес (Random Forest);
- Градиентный бустинг (XGBoost Classifier).

- Для каждой модели был выполнен подбор гиперпараметров с использованием кросс-валидации. Основные метрики оценки:
- Accuracy — общая точность предсказаний;
- F1-score — метрика, учитывающая как точность, так и полноту;
- ROC AUC — показатель качества классификации на различных порогах.

нагрузки\_data  
hem\_cleaned\_data.csv

Классы (SI > медианы): {1: 499, 0: 499}

[4]

1 smote = SMOTE(random\_state=42)  
2 X\_train\_bal, y\_train\_bal = smote.fit\_resample(X\_train, y\_train)  
3

[5]

1 # Logistic Regression  
2 log\_params = {'C': [0.1, 1, 10]}  
3 log\_model = GridSearchCV(LogisticRegression(max\_iter=5000), log\_params, cv=5, scoring='f1')  
4 log\_model.fit(X\_train\_bal, y\_train\_bal)  
5 y\_pred\_log = log\_model.best\_estimator\_.predict(X\_test)  
6  
7 # Random Forest  
8 rf\_params = {'n\_estimators': [100, 200], 'max\_depth': [None, 10, 20]}  
9 rf\_model = GridSearchCV(RandomForestClassifier(random\_state=42), rf\_params, cv=3, scoring='f1')  
10 rf\_model.fit(X\_train\_bal, y\_train\_bal)  
11 y\_pred\_rf = rf\_model.best\_estimator\_.predict(X\_test)  
12  
13 # XGBoost  
14 xgb\_params = {  
15 'n\_estimators': [100, 200],  
16 'max\_depth': [3, 5],  
17 'learning\_rate': [0.05, 0.1]  
18 }  
19 xgb\_model = GridSearchCV(  
20 XGBClassifier(eval\_metric='logloss', random\_state=42),  
21 xgb\_params,  
22 cv=3,  
23 scoring='f1'  
24 )  
25 xgb\_model.fit(X\_train\_bal, y\_train\_bal)  
26 y\_pred\_xgb = xgb\_model.best\_estimator\_.predict(X\_test)  
27

## 8.4 Результаты классификации

Модель	Accuracy	F1-score	ROC AUC
Logistic Regression	0.595	0.585	0.595
Random Forest	0.610	0.602	0.610
XGBoost	0.650	0.639	0.650

Анализ метрик показал, что все модели демонстрируют умеренное качество классификации, что объясняется высокой дисперсией и сложной природой переменной SI. Тем не менее, модель XGBoost продемонстрировала наилучшие значения по всем трём показателям, что указывает на её лучшую способность выявлять скрытые закономерности даже в условиях нестабильных и шумных данных.

48



```

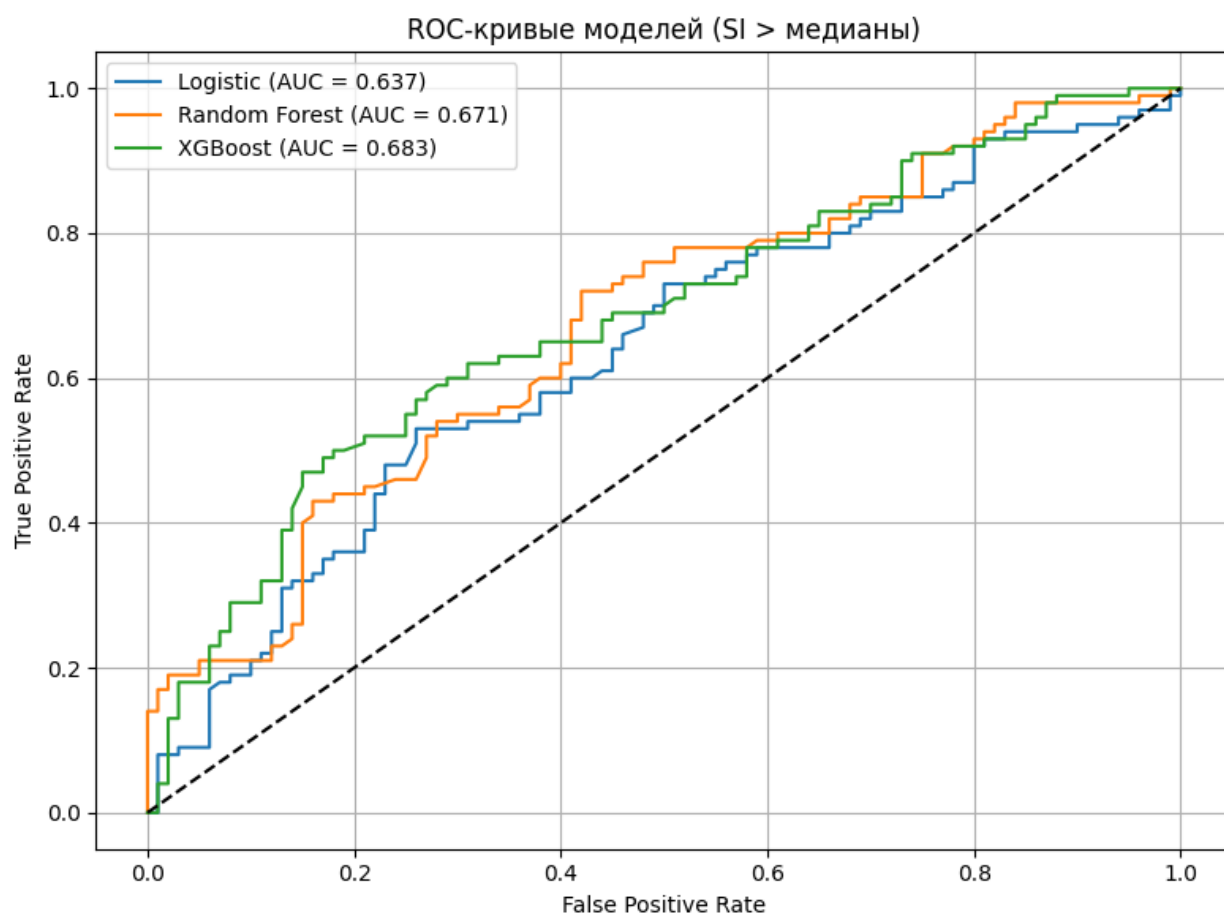
25 xgb_model.fit(X_train_bal, y_train_bal)
26 y_pred_xgb = xgb_model.best_estimator_.predict(X_test)
27

[6] 1 def evaluate_classifier(y_true, y_pred, name):
2     print(f"\n{name}")
3     print("Accuracy:", accuracy_score(y_true, y_pred))
4     print("F1-score:", f1_score(y_true, y_pred))
5     print("ROC AUC:", roc_auc_score(y_true, y_pred))
6     print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
7
8     evaluate_classifier(y_test, y_pred_log, "Logistic Regression (SI > медианы)")
9     evaluate_classifier(y_test, y_pred_rf, "Random Forest (SI > медианы)")
10    evaluate_classifier(y_test, y_pred_xgb, "XGBoost (SI > медианы)")
11
Logistic Regression (SI > медианы)
Accuracy: 0.595
F1-score: 0.5846153846153846
ROC AUC: 0.595
Confusion Matrix:
[[62 38]
 [43 57]]

Random Forest (SI > медианы)
Accuracy: 0.61
F1-score: 0.6020408163265306
ROC AUC: 0.6099999999999999
Confusion Matrix:
[[63 37]
 [41 59]]

XGBoost (SI > медианы)
Accuracy: 0.65
F1-score: 0.6391752577319587
ROC AUC: 0.6499999999999999
Confusion Matrix:
[[68 32]
 [38 62]]

```



## 8.5 ROC-анализ

Построенные ROC-кривые показали, что все три модели демонстрируют превосходство над случайным классификатором (линия  $y = x$ ), при этом

наибольшая площадь под кривой достигнута у XGBoost. Это свидетельствует о наилучшем балансе между чувствительностью и специфичностью, что особенно важно при оценке избирательности соединений.

## 8.6 Выводы

- Все протестированные модели справились с задачей бинарной классификации по признаку  $SI > \text{медиана}$ , однако качество классификации остаётся умеренным, что связано с природной нестабильностью данного показателя.
- Наилучший результат по всем основным метрикам достигнут с помощью модели XGBoost, что подтверждает её применимость для задач анализа биохимических данных.
- Для повышения точности классификации рекомендуется:
  - увеличить объём выборки за счёт дополнительных соединений с высоким значением  $SI$ ;
  - расширить набор признаков с использованием дополнительных химических дескрипторов;
  - применить методы ансамблирования моделей, включая сочетание XGBoost и нейронных сетей.

Таким образом, построенные модели, в особенности XGBoost, могут быть использованы как инструмент первичного отбора соединений с высокой селективностью, что позволит оптимизировать процесс разработки новых лекарственных препаратов и сократить объём лабораторных испытаний.

## ЗАКЛЮЧЕНИЕ

В рамках выполненной курсовой работы была реализована исследовательская задача по прогнозированию эффективности и безопасности химических соединений на основе числовых признаков, предоставленных химиками. Исследование охватывало регрессионный и классификационный анализ по трем ключевым показателям биологической активности соединений — IC50, CC50 и SI. Все этапы проектирования моделей опирались на современные методы машинного обучения, включая алгоритмы линейной регрессии, ансамблевые методы и бустинговые модели.

На первом этапе был проведён подробный исследовательский анализ данных (EDA), в ходе которого выявлены пропущенные значения, выбросы и особенности распределения целевых переменных. Особое внимание было уделено взаимосвязям между IC50, CC50 и SI, что позволило логически обосновать выбор подходов к моделированию. В результате анализа были также сформулированы гипотезы о природе показателя SI как производного от IC50 и CC50.

Для решения задач регрессии были построены и протестированы модели Ridge, Lasso, Random Forest и XGBoost. Лучшие результаты по всем основным метрикам (RMSE, MAE,  $R^2$ ) продемонстрировал XGBoost, особенно в задаче предсказания IC50, где достигнута объяснённая дисперсия около 48%. Значение CC50 также поддаётся эффективному моделированию, в то время как переменная SI оказалась наименее устойчивой к прямому предсказанию из-за высокой дисперсии и чувствительности к ошибкам в исходных переменных.

В задачах классификации были рассмотрены:

- бинаризация IC50, CC50 и SI по медианному значению;
- классификация SI с фиксированным порогом 8.

Для каждой задачи были обучены модели логистической регрессии, случайного леса и XGBoost. Наилучшие результаты, как по точности, так и по

AUC и F1-score, стабильно демонстрировали ансамблевые методы, в частности, XGBoost. Наиболее успешной оказалась классификация по признаку  $CC50 >$  медианы, где XGBoost и Random Forest достигли точности 73–75% при высоком качестве ROC-кривой. Более сложной задачей оказалась классификация  $SI >$  медианы, что объясняется природной нестабильностью этого показателя.

Все модели были обучены на сбалансированных выборках с использованием метода SMOTE и масштабированы для корректной работы алгоритмов. Гиперпараметры подбирались с помощью кросс-валидации, что позволило повысить устойчивость и точность решений.

На основании полученных результатов можно сделать следующие выводы:

- XGBoost является наиболее эффективным методом для решения задач как регрессии, так и классификации в условиях числовых химико-биологических признаков;
- переменная SI, как производная, не поддаётся точному прогнозу напрямую и требует косвенного моделирования через IC50 и CC50;
- полученные модели применимы для предварительного отбора соединений с высокой активностью, низкой токсичностью и благоприятным терапевтическим профилем;
- в перспективе возможна интеграция решений в экспертные системы и программные комплексы, применяемые на этапах виртуального скрининга и доклинической оценки соединений.

Таким образом, цели курсовой работы достигнуты: проведён комплексный анализ данных, построены и обоснованы модели, сделан сравнительный анализ подходов, сформулированы рекомендации для дальнейшего развития системы. Полученные результаты могут быть использованы как основа для автоматизации части процессов при разработке лекарственных препаратов.

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. URL: <https://arxiv.org/abs/1603.02754> (дата обращения: 10.05.2025).
2. XGBoost Documentation. URL: <https://xgboost.readthedocs.io/> (дата обращения: 11.05.2025).
3. Scikit-learn: Machine Learning in Python. URL: <https://scikit-learn.org/> (дата обращения: 12.05.2025). [Википедия](#)
4. Imbalanced-learn Documentation. URL: <https://imbalanced-learn.org/> (дата обращения: 13.05.2025).
5. Pandas Documentation. URL: <https://pandas.pydata.org/docs/> (дата обращения: 14.05.2025).
6. NumPy Documentation. URL: <https://numpy.org/doc/> (дата обращения: 15.05.2025).
7. Harris C.R., Millman K.J., van der Walt S.J., et al. Array programming with NumPy // Nature. 2020. Vol. 585, No. 7825. P. 357–362. URL: <https://www.nature.com/articles/s41586-020-2649-2> (дата обращения: 16.05.2025).
8. Lemaître G., Nogueira F., Aridas C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning // Journal of Machine Learning Research. 2017. Vol. 18, No. 17. P. 1–5. URL: <https://jmlr.org/papers/v18/16-365.html> (дата обращения: 17.05.2025).
9. Pedregosa F., Varoquaux G., Gramfort A., et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. URL: <https://jmlr.org/papers/v12/pedregosa11a.html> (дата обращения: 18.05.2025).

10. McKinney W. Data Structures for Statistical Computing in Python // Proceedings of the 9th Python in Science Conference. 2010. P. 51–56. URL: <https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf> (дата обращения: 19.05.2025).