Содержание

Введение

Процесс создания нового лекарственного препарата является сложным и многоэтапным. Необходимо определить его химическую формулу, синтезировать соединение, провести первичные биологические испытания и организовать тестирование. Все эти этапы требуют значительного времени, однако современные методы машинного обучения способны существенно ускорить данный процесс.

Так, например, с помощью различных моделей можно спрогнозировать эффективность соединений и подобрать наиболее подходящие сочетания параметров для разработки лекарственных средств.

Тем не менее, для достижения качественного результата в подобного рода задачах важно наладить эффективное взаимодействие между химиками и специалистами по машинному обучению, что зачастую оказывается непростой задачей.

Постановка задачи

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

Для этого требуется:

1. Проанализировать текущие параметры с использованием различных методов.
2. Научиться предсказывать их эффективность.
3. Создать несколько максимально эффективных моделей для решения следующих задач:
4. Регрессия для IC50
5. Регрессия для CC50
6. Регрессия для SI
7. Классификация: превышает ли значение IC50 медианное значение выборки
8. Классификация: превышает ли значение CC50 медианное значение выборки
9. Классификация: превышает ли значение SI медианное значение выборки
10. Классификация: превышает ли значение SI значение 8

Теоретическая часть

Предоставленный датасет от химиков содержит конфиденциальные данные о 1000 химических соединений с указанием их эффективности против вируса гриппа. Параметры, характеризующие эффективность препарата, относятся к фармакологическим показателям, такие как:

1. IC50 (мМ) – Концентрация вещества, необходимая для подавления биологической активности (например, ферментативной активности или клеточного процесса) на 50%;
2. CC50 (мМ) – Концентрация вещества, вызывающая гибель 50% клеток в тестах на цитотоксичность;
3. SI (Selectivity Index, индекс селективности) – Отношение CC50 к IC50, показывает, насколько вещество избирательно действует на мишень, а не на здоровые клетки (SI = CC50 / IC50).

Дадим определения остальным химическим свойствам.

Дескрипторы электронного состояния (E-State Indices):

1. MaxAbsEStateIndex – Максимальное абсолютное значение электронного состояния атома в молекуле;
2. MaxEStateIndex – Максимальное значение электронного состояния атома в молекуле;
3. MinAbsEStateIndex – Минимальное абсолютное значение электронного состояния атома в молекуле;
4. MinEStateIndex – Минимальное значение электронного состояния атома в молекуле.

Физико-химические и стерические свойства:

1. qed (Quantitative Estimate of Drug-likeness) – мера сходства свойств вещества с характеристиками пероральных лекарств;
2. SPS (Synthetic Accessibility Score) – Оценка сложности синтеза молекулы (обычно от 1 – легко, до 10 – сложно);
3. MolWt (Molecular Weight) – Молекулярная масса соединения (в

г/моль);

4. HeavyAtomMolWt – Молекулярная масса без учета водородов;
5. ExactMolWt – Точная молекулярная масса (с учетом изотопов);
6. NumValenceElectrons – Количество валентных электронов в молекуле;
7. NumRadicalElectrons – Количество неспаренных электронов;

Электростатические параметры:

1. MaxPartialCharge – Максимальный парциальный заряд на атоме;
2. MinPartialCharge – Минимальный парциальный заряд на атоме;
3. MaxAbsPartialCharge – Максимальное абсолютное значение парциального заряда;
4. MinAbsPartialCharge – Минимальное абсолютное значение парциального заряда.

Fingerprint и плотности Morgan:

1. FpDensityMorgan1, FpDensityMorgan2, FpDensityMorgan3 – Плотность Fingerprint Morgan (степень ветвления молекулы).

BCUT-дескрипторы (2D-дескрипторы на основе матрицы связей):

1. BCUT2D_MWHI / BCUT2D_MWLOW – Высокие/низкие значения молекулярного веса в BCUT-дескрипторах;
2. BCUT2D_CHGHI / BCUT2D_CHGLO – Высокие/низкие значения зарядов в BCUT;
3. BCUT2D_LOGPHI / BCUT2D_LOGPLOW – Высокие/низкие значения logP;
4. BCUT2D_MRHI / BCUT2D_MRLOW – Высокие/низкие значения молярной рефракции.

Топологические индексы:

1. AvgIpc (Average Information Content) – Средняя информационная емкость молекулы;
2. BalabanJ – Индекс Балабана (мера компактности молекулы);
3. BertzCT – Индекс сложности молекулы Берца;
4. Chi0, Chi0n, Chi0v, Chi1, Chi1n, Chi1v, Chi2n, Chi2v, Chi3n, Chi3v,

Chi4n, Chi4v – Различные ки-индексы (топологические дескрипторы, учитывающие связи и атомы);

5. HallKierAlpha – Альфа-индекс Холла-Кира (стерический параметр);

6. Ipc (Information Content) – Информационная емкость молекулы;

7. Kappa1, Kappa2, Kappa3 – Каппа-индексы (форма молекулы и разветвленность);

8. LabuteASA (Approximate Surface Area) – Приблизительная площадь поверхности молекулы.

PEOE_VSA (атомные вклады в полярность поверхности):

1. PEOE_VSA1–14 – Вклады атомов в полярность поверхности на основе парциальных зарядов.

SMR_VSA (атомные вклады в молярную рефракцию):

1. SMR_VSA1–10 – Вклады атомов в молярную рефракцию.

SlogP_VSA (атомные вклады в липофильность):

1. SlogP_VSA1–12 – Вклады атомов в logP.

Другими важными дескрипторами являются:

1. TPSA (Topological Polar Surface Area) – Топологическая полярная поверхностная площадь (в Å²);

2. EState_VSA1–11 – Вклады электронных состояний в площадь поверхности;

3. VSA_EState1–10 – Вклады электронных состояний в VSA.

Стереохимические и структурные параметры:

1. FractionCSP3 – Доля sp^3-гибридизированных атомов углерода;

2. HeavyAtomCount – Количество "тяжелых" атомов (не водородов);

3. NHOHCount – Количество гидроксильных и амино-групп;

4. NOCount – Количество азотов и кислородов;

5. NumAliphaticCarbocycles / Heterocycles / Rings – Количество алифатических карбо-/гетероциклов;

6. NumAromaticCarbocycles / Heterocycles / Rings – Количество ароматических циклов;

7. NumHAcceptors / HDonors – Количество акцепторов/доноров водородных связей;

8. NumHeteroatoms – Количество гетероатомов (O, N, S и др.);

9. NumRotatableBonds – Количество вращающихся связей;

10. NumSaturatedRings – Количество насыщенных циклов;

11. RingCount – Общее количество циклов;

Физико-химические константы:

1. MolLogP – Расчетный коэффициент распределения октанол/вода;

2. MolMR – Молярная рефракция.

Функциональные группы (fr_*):

1. fr_Al_COO, fr_Ar_OH, fr_COO, fr_NH2 и др. – Бинарные индикаторы наличия функциональных групп (например, карбоксилаты, фенолы, амины и т. д.).

Все эти параметры используются в компьютерном дизайне лекарств (CADD), QSAR-моделировании и оценке ADME-свойств.

Практическая часть

Анализ и пред обработка данных

Перед построением моделей классического ML необходимо выполнить анализ и пред обработку предоставленных данных. В начале анализа была выполнена загрузка датасета и вывод основной информации, как показано на рисунке 1.

```
#Прочитаем датасет
course_df = pd.read_excel('./data/course_data.xlsx')
```

```
#Выведем основую информацию о датасете
display(course_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Columns: 214 entries, Unnamed: 0 to fr_urea
dtypes: float64(107), int64(107)
memory usage: 1.6 MB
None
```

Датасет содержит 1001 запись и 214 полей

```
#Выведем первые и последние 5 строк
display(course_df.head(5))
display(course_df.tail(5))
```

	Unnamed: 0	IC50, mM	CC50, mM	SI	MaxAbsEStateIndex	MaxEStateIndex	MinAbsEStateIndex	MinEStateIndex	qed	SPS	MolWt	Hea
0	0	6.239374	175.482382	28.125000	5.094096	5.094096	0.387225	0.387225	0.417362	42.928571	384.652	
1	1	0.771831	5.402819	7.000000	3.961417	3.961417	0.533868	0.533868	0.462473	45.214286	388.684	
2	2	223.808778	161.142320	0.720000	2.627117	2.627117	0.543231	0.543231	0.260923	42.187500	446.808	
3	3	1.705624	107.855654	63.235294	5.097360	5.097360	0.390603	0.390603	0.377846	41.862069	398.679	
4	4	107.131532	139.270991	1.300000	5.150510	5.150510	0.270476	0.270476	0.429038	36.514286	466.713	

	Unnamed: 0	IC50, mM	CC50, mM	SI	MaxAbsEStateIndex	MaxEStateIndex	MinAbsEStateIndex	MinEStateIndex	qed	SPS	MolWt	He
996	996	31.000104	34.999650	1.129017	12.934891	12.934891	0.048029	-0.476142	0.382752	49.133333	414.542	
997	997	31.999934	33.999415	1.062484	13.635345	13.635345	0.030329	-0.699355	0.369425	44.542857	485.621	
998	998	30.999883	33.999458	1.096761	13.991690	13.991690	0.026535	-0.650790	0.284923	41.973684	545.742	
999	999	31.998959	32.999644	1.031272	13.830180	13.830180	0.146522	-1.408652	0.381559	39.000000	522.635	
1000	1000	99.999531	99.999531	1.000000	13.380863	13.380863	0.002425	-0.447978	0.452565	48.580645	426.597	

Рисунок 1. Загрузка датасета и вывод основной информации.

Из данный информации были получены сведения о том, что датафрейм содержит 1001 запись и 214 полей. Все данные представлены в формате float64, что позволяет избежать кодирования категориальных признаков.

В данных обнаружен признак «Unnamed: 0», который содержит индекс вещества. Так как индексы абсолютны уникальны в каждой строке, то такие признаки будут только ухудшать качество модели при прогнозировании. Поэтому выполняем удаление данного признака.

Следующим шагом является анализ на пропуски данных в признаках. Как

показано на рисунке 2, в датасете было обнаружено 36 пропущенных ячеек в трёх строках двенадцати признаков. Так как количество строк с пропусками не велико, было принято решение удалить строки имеющие пропуски в признаках. После данной операции в датасете осталось 988 записей.

```
#Удаление поля "Unnamed: 0"
course_df.drop('Unnamed: 0', axis=1, inplace=True)
```

```
#Проверим наличие пропусков в данных
print('Количество пропущеных ячеек', course_df.isna().sum().sum())
print('Количество строк с пропусками', course_df.shape[0] - course_df.dropna().shape[0])
print('Колонки с пропусками:')
columns_with_null = course_df.columns[course_df.isnull().any()]
for column_name in columns_with_null:
    print(f'• {column_name}')
```

```
Количество пропущеных ячеек 36
Количество строк с пропусками 3
Колонки с пропусками:
• MaxPartialCharge
• MinPartialCharge
• MaxAbsPartialCharge
• MinAbsPartialCharge
• BCUT2D_MWHI
• BCUT2D_MWLOW
• BCUT2D_CHGHI
• BCUT2D_CHGLO
• BCUT2D_LOGPHI
• BCUT2D_LOGPLOW
• BCUT2D_MRHI
• BCUT2D_MRLOW
```

Так как во всем датасете пропуски есть только в 3х строках, можем ими пренебречь и удалить.

```
#Удаляем строки с пропусками
course_df.dropna(axis=0, inplace=True)
print('Размерность датасета после удаления =', course_df.shape)
```

```
Размерность датасета после удаления = (998, 213)
```

Рисунок 2. Анализ пропусков в данных.

Далее был проведён анализ на уникальность значений в признаках. Так было выявлено 18 признаков в которых присутствовало всего одно уникальное значение. Такие данные бесполезны для модели, поэтому данные признаки удаляем. На рисунке 3 показан процесс отбора и удаления признаков с единственным уникальным значением.

```
not_uniq_features = list()

print('Количество уникальных значений в признаках:')
for colname in course_df.columns:
    uniq_count = course_df[colname].nunique()
    print('{0:25} - {1}'.format(colname, uniq_count))
    if uniq_count <= 1:
        not_uniq_features.append(colname)

print(f'\nКоличество признаков с одним значением: {len(not_uniq_features)}')
```

```
Количество уникальных значений в признаках:
IC50, mM                  - 950
CC50, mM                  - 885
SI                        - 767
MaxAbsEStateIndex         - 791
MaxEStateIndex            - 791
MinAbsEStateIndex         - 788
MinEStateIndex            - 794
qed                       - 756
SPS                       - 595
MolWt                     - 710
HeavyAtomMolWt            - 603
ExactMolWt                - 691
NumValenceElectrons       - 112
NumRadicalElectrons       - 1
MaxPartialCharge          - 722
MinPartialCharge          - 654
```

Проанализировав признаки на уникальные значения, получаем, что в 18 признаках имеется всего одно значение. Следовательно данные признаки неинформативны для вычисления и их можно удалить.

```
#Удаление признаков с единственным уникальным значением
course_df.drop(not_uniq_features, axis=1, inplace=True)
print('Размерность датасета после удаления =', course_df.shape)
```

```
Размерность датасета после удаления = (998, 195)
```

Рисунок 3. Выявление и удаление признаков с единственным уникальным значением.

Следующим важным этапом является анализ и обработка выбросов в данных, так как выбросы сильно искажают результаты предсказания модели. Для анализа выбросов была описана функция «show_outliers_iqr», выводящая распределение значения признаков в виде гистограммы. По верх гистограммы наложены карасиные линии, показывающие доверительный интервал по межквартильному размаху, зелёная пунктирная линия, показывающая среднее значение, и красные бины, показывающие аномальные выбросы на основе метода LocalOutlierFactor из библиотеки sklearn.neighbors.

Автоматическое удаление или замена выбросов на медианное значени приводило к плохим результатам при построении моделей ML. Поэтому все признаки были вручную проанализированы на наличие аномальных значений. После анализа были выделены следующие признаки с диапазоном выбросов:

1. IC50, mM > 2500 (рисунок 4)
2. CC50, mM > 4000 (рисунок 5)
3. SI > 800 (рисунок 6)

4. MinAbsEStateIndex > 1 (рисунок 7)
5. MinPartialCharge > -0.2 и <-0.7 (рисунок 8)
6. MaxAbsPartialCharge > 0.5 (рисунок 9)
7. BCUT2D_MWLOW < 2 (рисунок 10)
8. BCUT2D_MRHI > 13 (рисунок 11)
9. Ipc > 1 * 10**12 (рисунок 12)
10. SMR_VSA2 > 5 (рисунок 13)

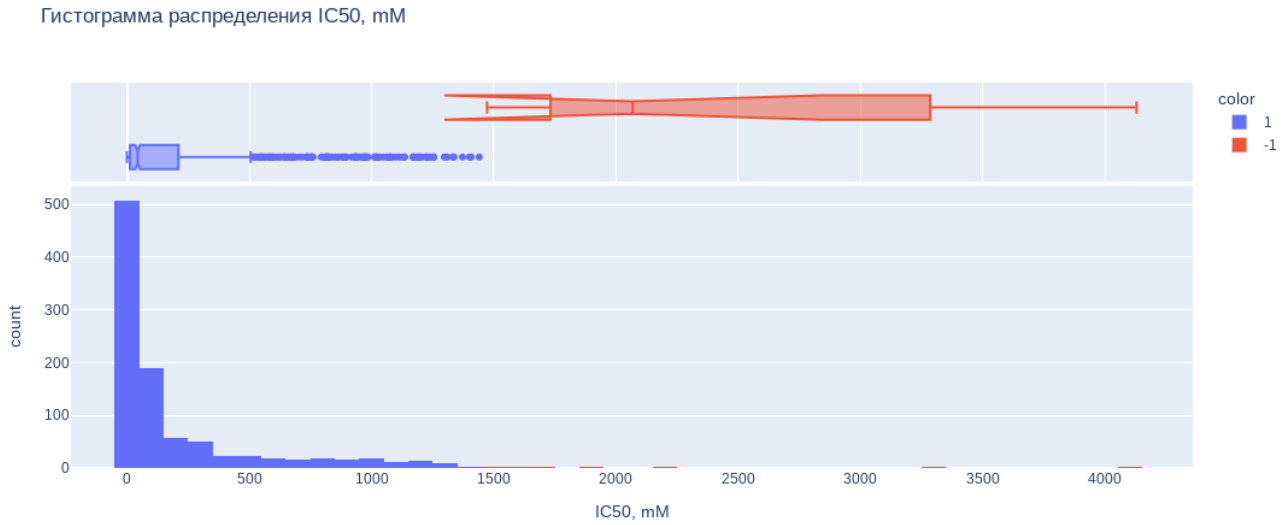

Рисунок 4. Поиск выбросов в признаке «IC50, mM».

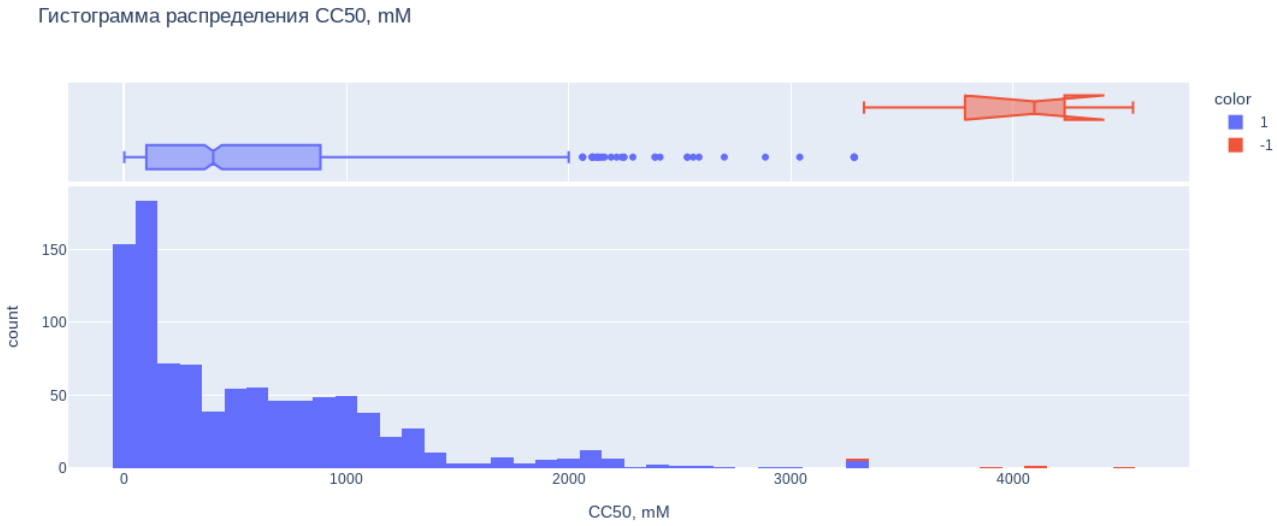

Рисунок 5. Поиск выбросов в признаке «CC50, mM».

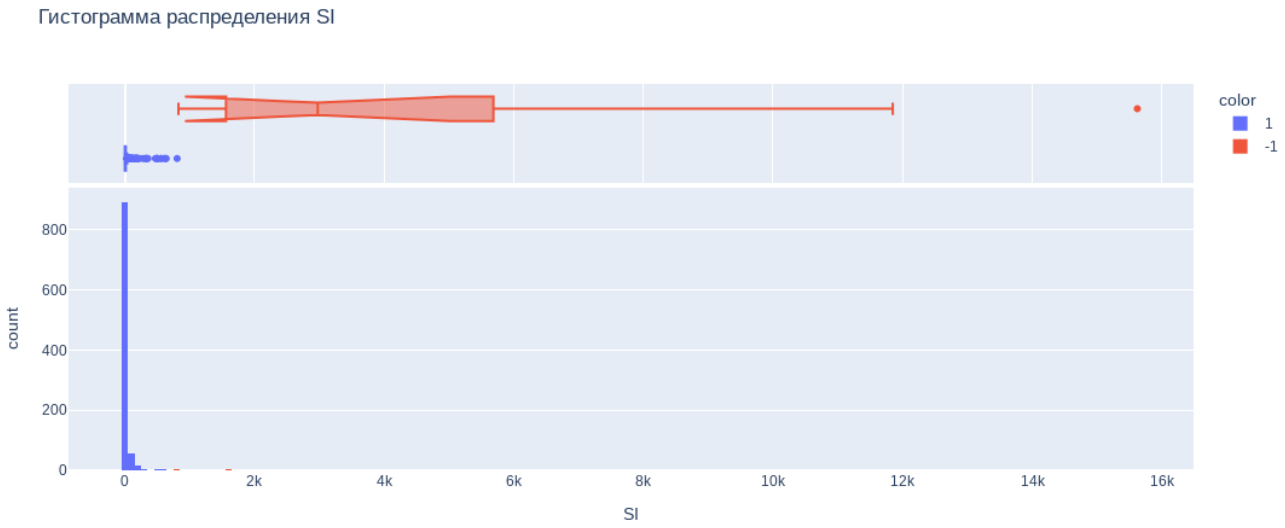

Рисунок 6. Поиск выбросов в признаке «SI».

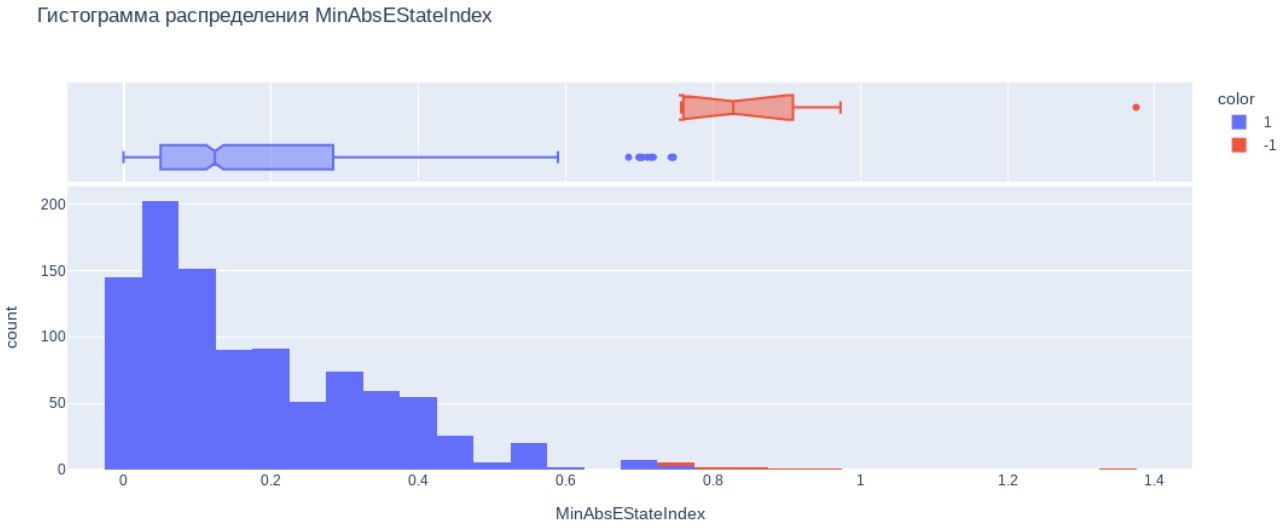

Рисунок 7. Поиск выбросов в признаке «MinAbsEStateIndex».

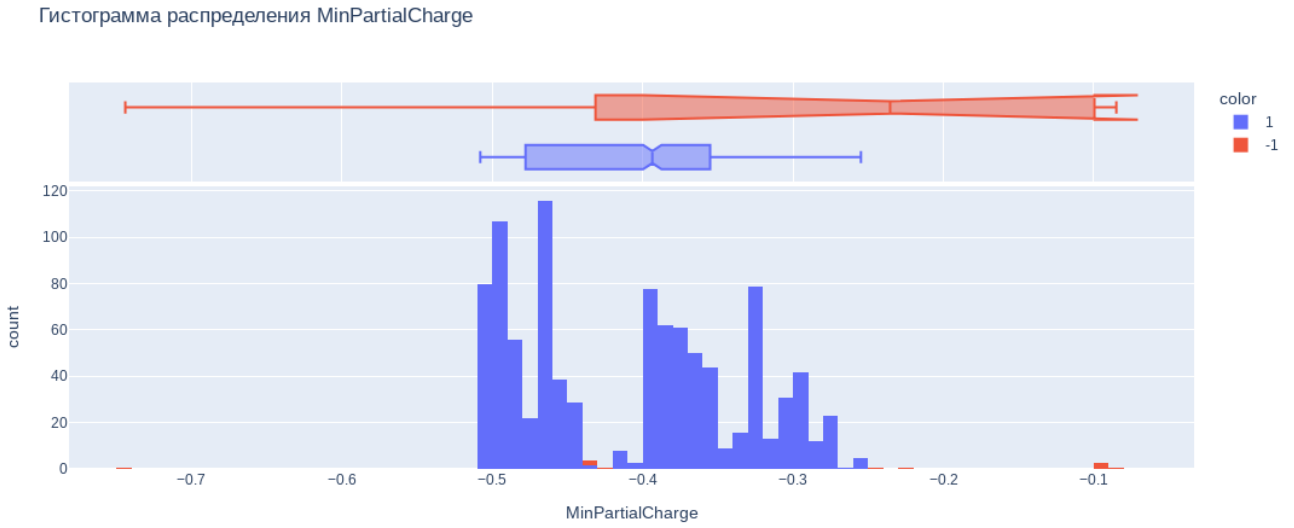

Рисунок 8. Поиск выбросов в признаке «MinPartialCharge».

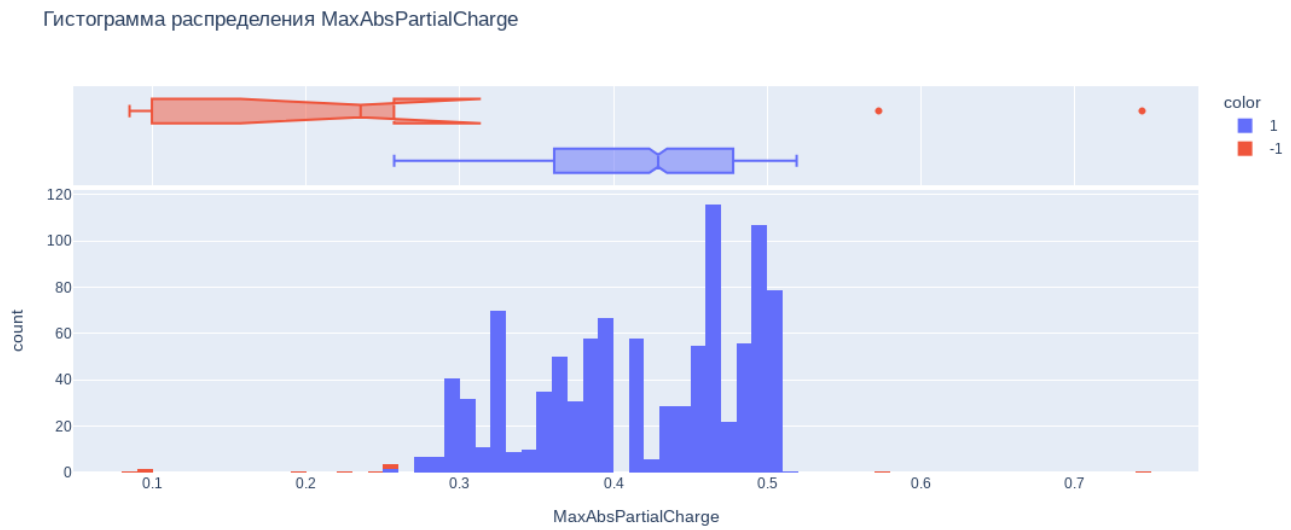

Рисунок 9. Поиск выбросов в признаке «MaxAbsPartialCharge».

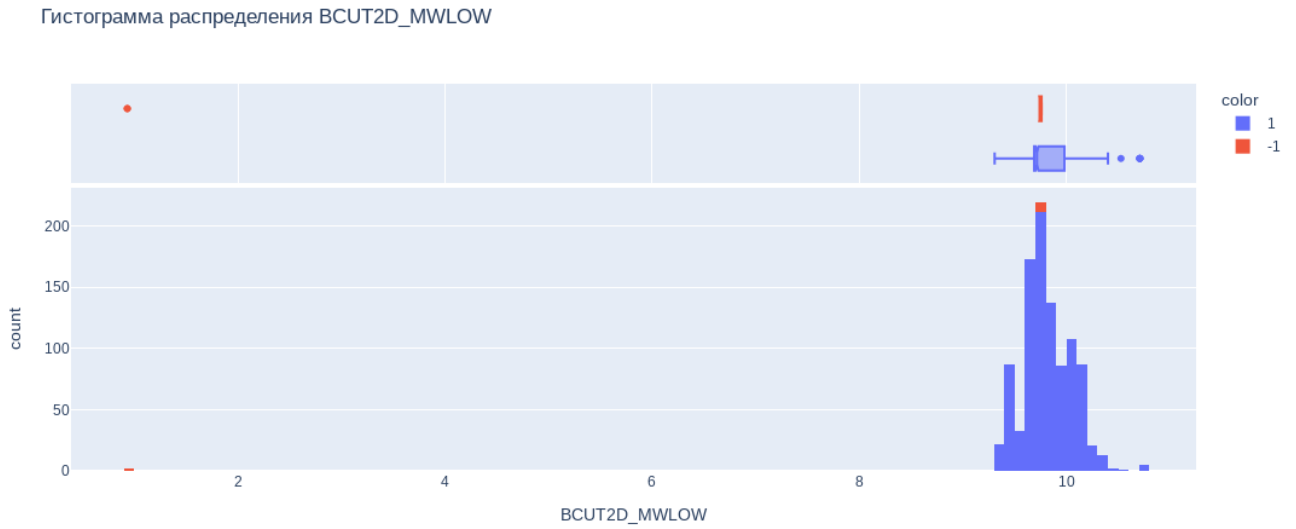

Рисунок 10. Поиск выбросов в признаке «BCUT2D_MWLOW».

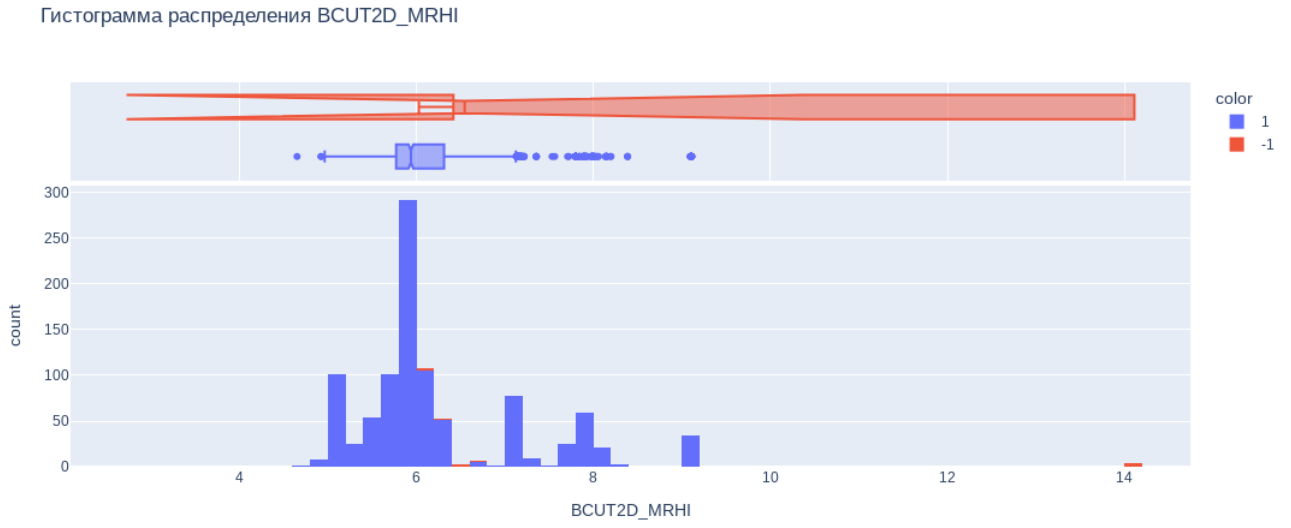

Рисунок 11. Поиск выбросов в признаке «BCUT2D_MRHI».

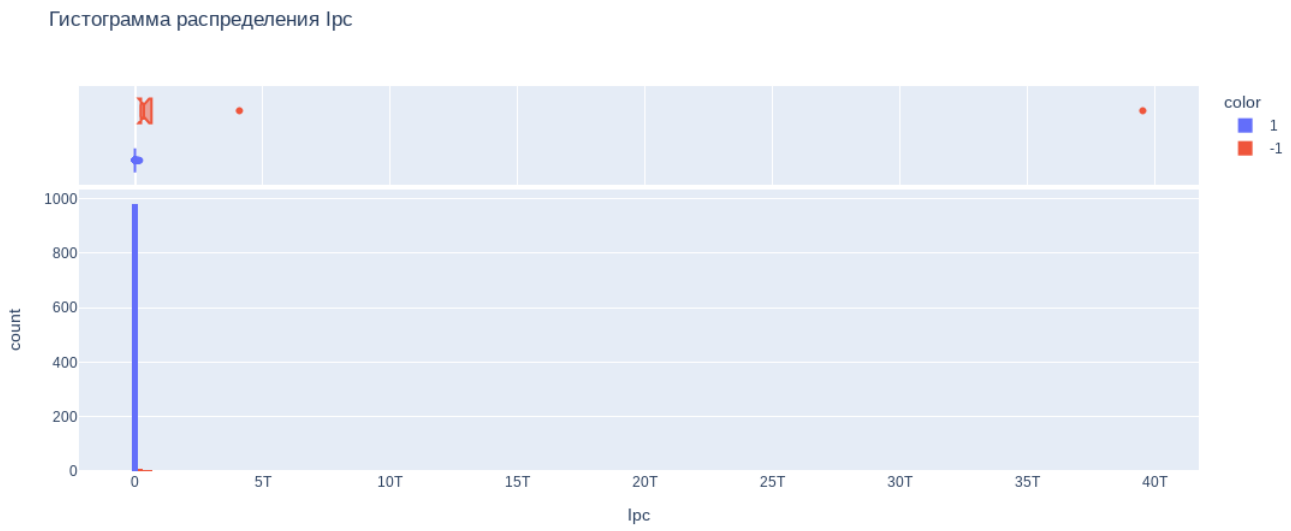

Рисунок 12. Поиск выбросов в признаке «Ipc».

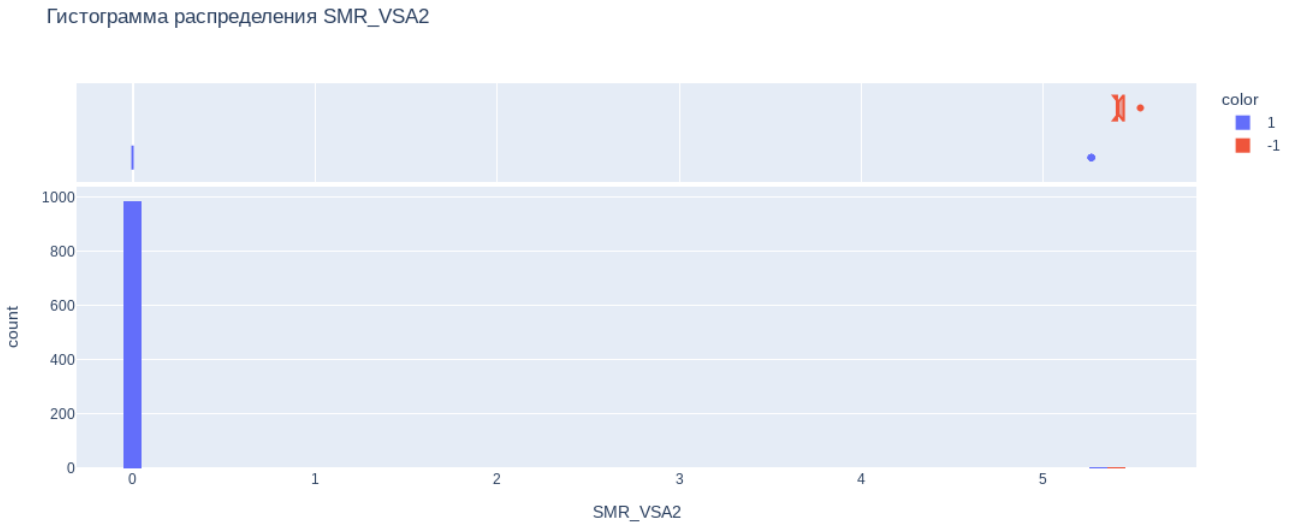

Рисунок 13. Поиск выбросов в признаке «SMR_VSA2».

Все представленные графики объединяет то, что у них основное распределение представлено малым количеством бинов в одной части гистограммы. Это говорит о том, что в признаке присутствуют данные, которые имеют аномально большое или малое значение по отношению к распределению в основной выборке.

Количество строк с выбросами по данным признакам составило 108 записей. Было принято решение удалить строки имеющие выбросы в признаках, так как попытка заменить выбросы на медианное значение ухудшала показатели предсказания моделей.

Далее был выполнен анализ корреляции в данных. Для этого была выведена тепловая карта корреляции, показанная на рисунке 14.

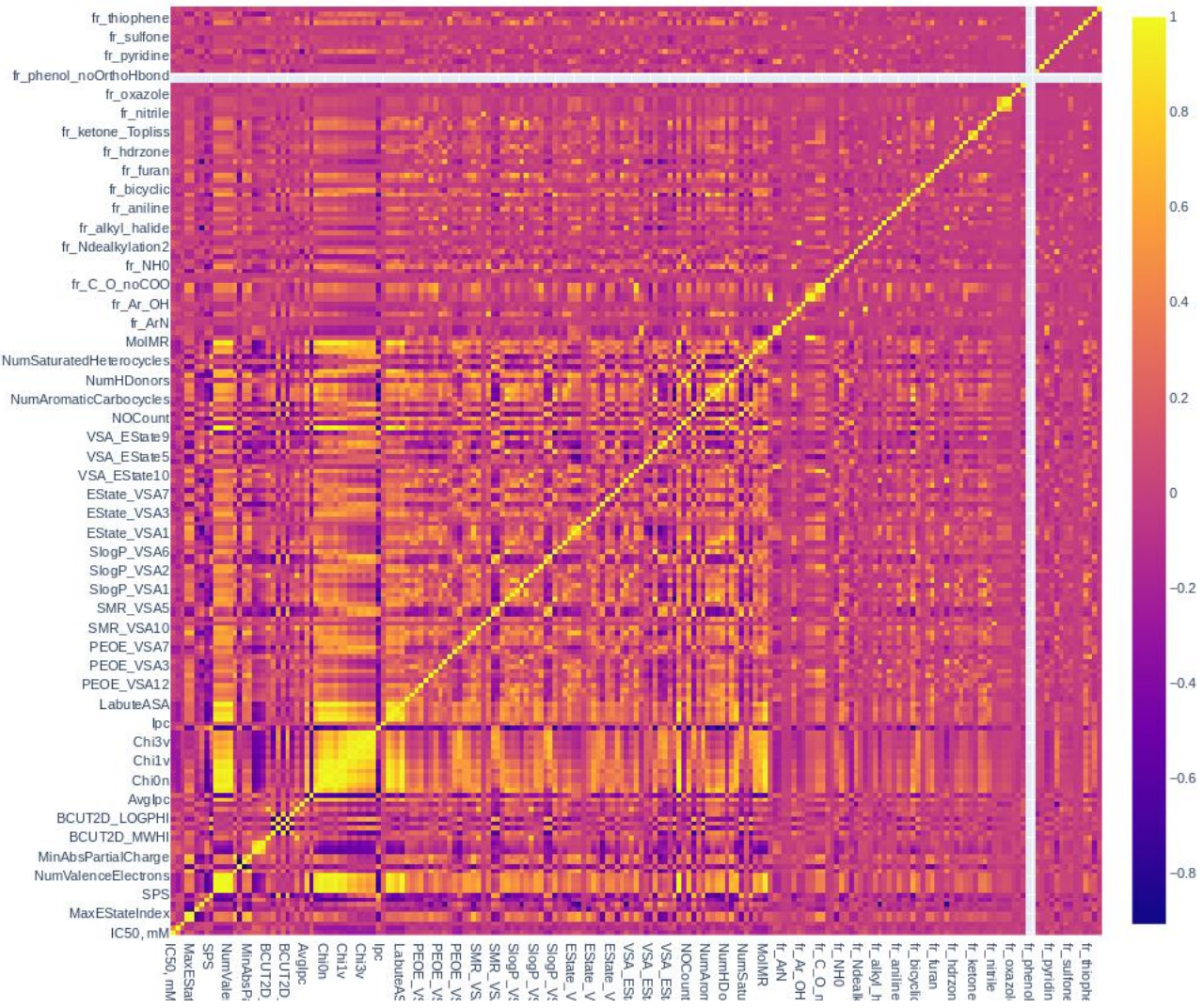

Рисунок 14. Тепловая карта корреляции признаков.

На тепловой карте отчётливо видно, что присутствуют признаки, которые сильно коррелируют между собой. Данные признаки необходимо удалят, так как они линейно зависимы и содержат туже информацию. Также был проведён анализ корреляции между каждым признаком. Как показано на рисунке 15, выяснилось, что некоторые признаки имеют 100% корреляцию между собой. Было принято решение для каждого из таргетов «IC50, mM», «CC50, mM» и «SI» удалять признаки, имеющие корреляцию между собой более 0.95, при этом из двух признаков оставлять тот, что больше коррелирует с таргетом . Так как каждый признак имеет разную корреляцию к каждому из таргетов, с этого момента датасет разделяется на три «IC50_df», «CC50_df» и «SI_df», где данные будут подбираться наилучшим образом для каждого из таргетов.

```
#Определим признаки которые сильно коррелируют между собой
top_corr = get_top_abs_correlations(course_df).reset_index()
display(top_corr)
```

	level_0	level_1	0
0	fr_COO	fr_COO2	1.000000
1	fr_Ar_NH	fr_Nhpyrrole	1.000000
2	MaxAbsEStateIndex	MaxEStateIndex	1.000000
3	NumAromaticCarbocycles	fr_benzene	1.000000
4	MolWt	ExactMolWt	0.999998
5	Chi1	HeavyAtomCount	0.998627
6	MolWt	HeavyAtomMolWt	0.996562
7	HeavyAtomMolWt	ExactMolWt	0.996524
8	Chi0	HeavyAtomCount	0.995670
9	NumValenceElectrons	Chi0	0.994938
10	LabuteASA	HeavyAtomCount	0.994452
11	Chi0n	Chi0v	0.992763
12	Chi1	LabuteASA	0.992388
13	NumValenceElectrons	LabuteASA	0.991088
14	Chi0	LabuteASA	0.990303
15	Chi0	Chi1	0.990118
16	ExactMolWt	HeavyAtomCount	0.989867
17	fr_Al_COO	fr_COO2	0.989748
18	fr_Al_COO	fr_COO	0.989748
19	NumValenceElectrons	HeavyAtomCount	0.989706
20	MolWt	HeavyAtomCount	0.989674

Рисунок 15. Отображение значения корреляции между каждым признаком в порядке убывания.

Последним этапом стал отбор признаков, которые наибольшим образом влияют на прогнозирование целевых показателей. Для этого использовалась модель линейной регрессии, на предсказаниях которой, вычислялась метрика R^2 которую и стремились оптимизировать. Данные для линейной регрессии формировались в цикле, каждый раз добавляя новый признак, имеющий наибольшую корреляцию к таргету. Тем самым удалось определить количество признаков с наибольшей корреляцией к таргету, которые улучшали метрику R^2.

На рисунках 16, 17 и 18 отображены линейные графики изменения метрики R^2 по мере добавления новых признаков в каждом из датасетов «IC50_df», «CC50_df» и «SI_df».

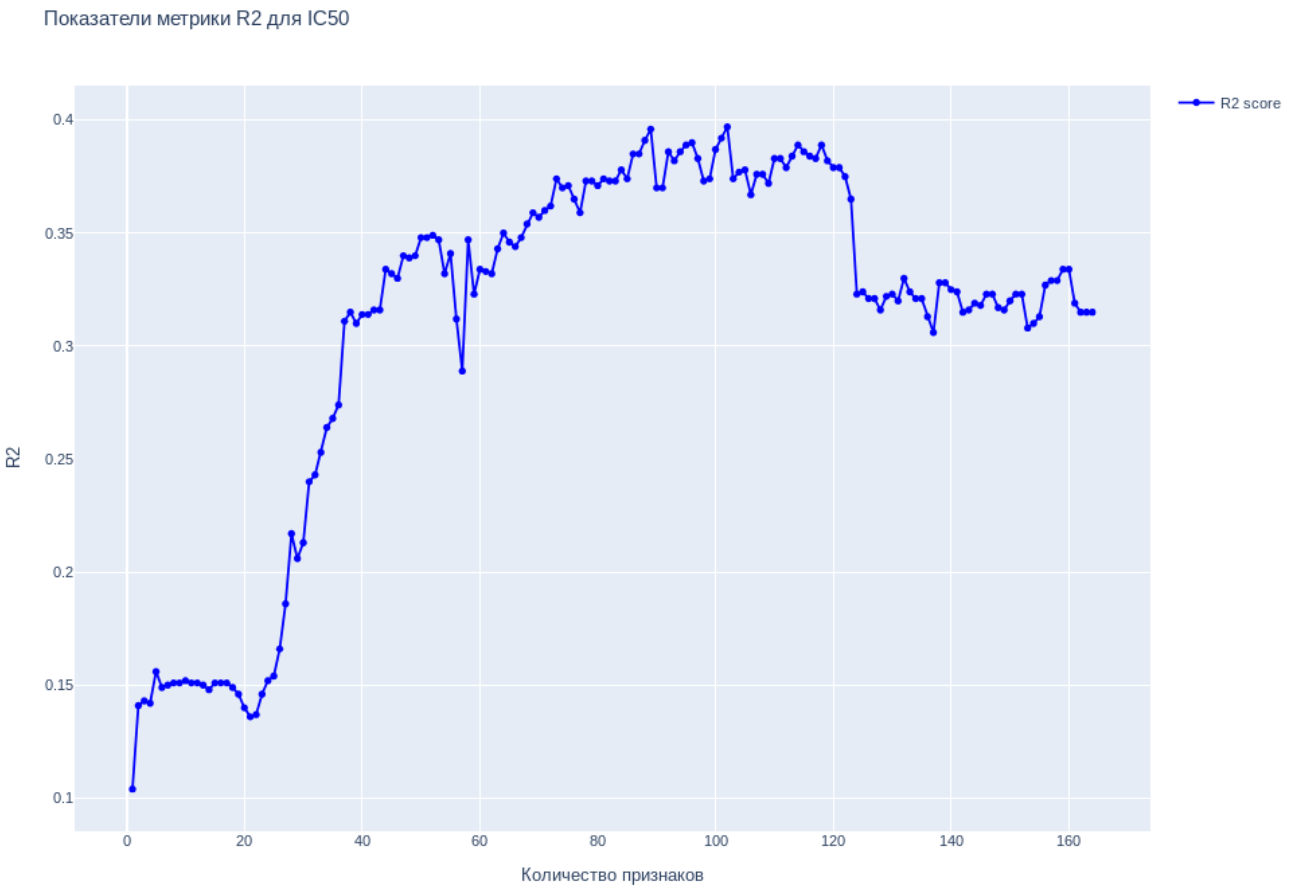

Рисунок 16. Изменение метрики R^2 при добавлении признаков в датасете IC50_df.

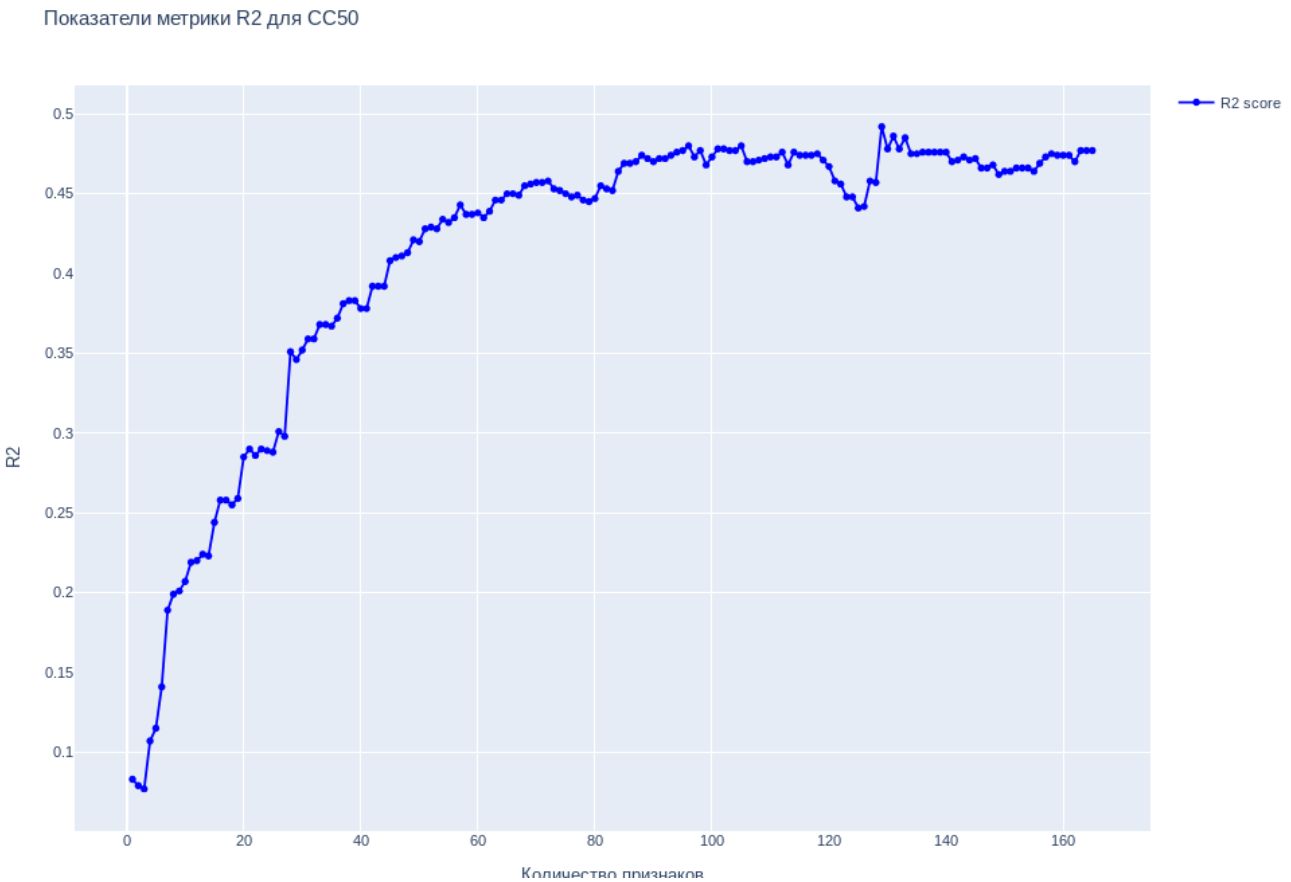

Рисунок 17. Изменение метрики R^2 при добавлении признаков в датасете CC50_df.

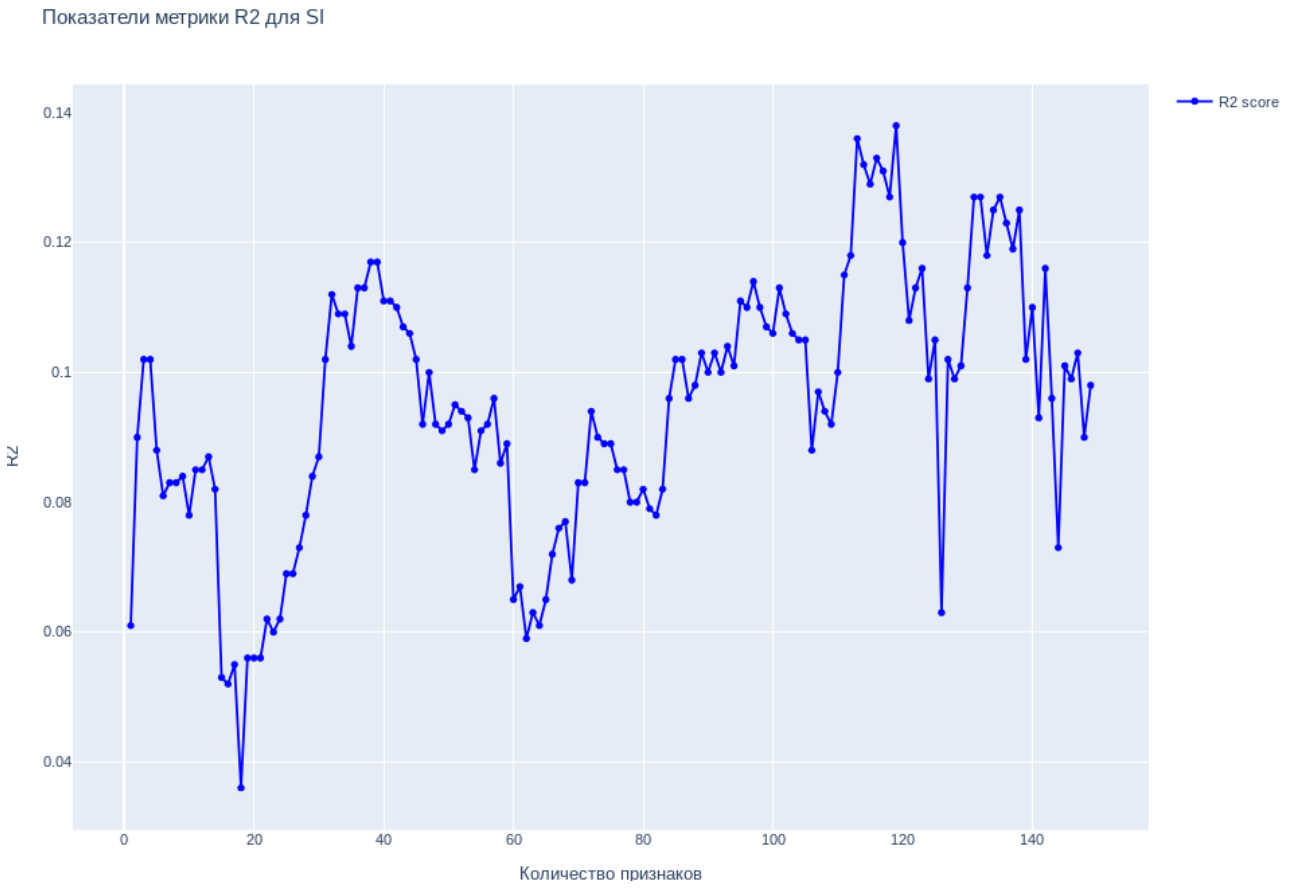

Рисунок 18. Изменение метрики R^2 при добавлении признаков в датасете SI_df.

Из приведённых графиков можно сделать следующие выводы:

1. Для таргета IC50 наилучший показатель R^2 при 102 признаках с наибольшей корреляцией;
2. Для таргета CC50 наилучший показатель R^2 при 129 признаках с наибольшей корреляцией;
3. Для таргета IS наилучший показатель R^2 при 119 признаке с наибольшей корреляцией.

Далее из датасетов удаляются признаки, имеющие негативное влияние не метрику предсказания модели.

На данном этапе готово три основных датьасета «IC50_df», «CC50_df» и «SI_df» для дальнейшего построения моделей.

Реализация моделей регрессии

Для решения задачи регрессии были использованы методы из библиотеки sklearn.

Так как данные уже пред обработаны и собранны под конкретные таргеты, можно приступать к обучению моделей и подбору гипер параметров.

На первом шаге, для каждого из таргетов загружаются свой датасет после чего разделяется на таргет и признаки. Для приведения признаков к единому масштабу, с помощью метода StandardScaler выполняется стандартизация признаков, после чего с помощью метода train_test_split датасет делиться на тренировочную и тестовую выборки в соотношении 80% и 20% соответственно. Пример данного процесса показан на рисунке 19.

[2]:
```
#Загрузи данные
IC50_df = pd.read_csv('./data/IC50.csv')
display(IC50_df)
```

	Unnamed: 0	IC50, mM	MaxEStateIndex	MinAbsEStateIndex	MinEStateIndex	qed	SPS	MaxPartialCharge	MaxAbsPartialCharge	FpDensityMorg
0	0	6.239374	5.094096	0.387225	0.387225	0.417362	42.928571	0.038844	0.293526	0.642
1	1	0.771831	3.961417	0.533868	0.533868	0.462473	45.214286	0.012887	0.313407	0.607
2	2	223.808778	2.627117	0.543231	0.543231	0.260923	42.187500	0.094802	0.325573	0.562
3	3	1.705624	5.097360	0.390603	0.390603	0.377846	41.862069	0.038844	0.293526	0.620
4	4	107.131532	5.150510	0.270476	0.270476	0.429038	36.514286	0.062897	0.257239	0.600
...	...	...	...	...	...	...	...	...	...	
885	996	31.000104	12.934891	0.048029	-0.476142	0.382752	49.133333	0.317890	0.468587	1.133
886	997	31.999934	13.635345	0.030329	-0.699355	0.369425	44.542857	0.327562	0.467493	1.085
887	998	30.999883	13.991690	0.026535	-0.650790	0.284923	41.973684	0.327887	0.467485	1.157
888	999	31.998959	13.830180	0.146522	-1.408652	0.381559	39.000000	0.312509	0.468755	0.756
889	1000	99.999531	13.380863	0.002425	-0.447978	0.452565	48.580645	0.311311	0.468587	1.064

890 rows × 105 columns

[3]:
```
#Выведем целевой признак
target = IC50_df['IC50, mM']
features = IC50_df.drop(['IC50, mM', 'Unnamed: 0'], axis=1)
```

[4]:
```
#Выполним стандартизацию данных
scaler = StandardScaler()
features = scaler.fit_transform(features)
```

[5]:
```
#Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

Рисунок 19. Загрузка датасета и подготовка к обучению модели регрессии.

Для прогнозирования числового значения были использованны такие модели как:

1. Linear regression;
2. DecisionTree;
3. Random Forest;

4. SVR;
5. KNN;
6. CatBoostRegressor.

После получения числовых прогнозов, с помощью библиотеки metrics выводятся основные метрики, такие как:

1. MAE - средняя абсолютная величина ошибки;
2. MSE - средняя квадратичная ошибка;
3. RMSE - квадратный корень из MSE;
4. R^2 - статистическая метра соответствия регрессионной модели реальным данным.

В каждой модели вручную подбирались гипер параметры для получения наилучшего результата метрики R^2. Результаты метрик для каждого из таргетов представлены в таблицах 1, 2 и 3.

Таблица 1. Прогнозирование таргета IC50.

Название модели	**MAE**	**MSE**	**RMSE**	**R^2**
Linear regression	210.00	80 689.42	284.06	0.38
DecisionTree	178.15	79 025.26	281.11	0.40
Random Fores	176.24	68 954.82	262.59	0.47
SVR	167.58	85 078.64	291.68	0.35
KNN	174.26	74 818.18	273.53	0.43
CatBoostRegressor	167.74	73 046.39	270.27	0.44

Таблица 2. Прогнозирование таргета CC50.

Название модели	**MAE**	**MSE**	**RMSE**	**R^2**
Linear regression	351.29	204 901.02	452.66	0.48
DecisionTree	337.27	227 427.29	476.89	0.42
Random Fores	299.98	175 769.39	419.25	0.55
SVR	296.58	203 537.91	451.15	0.48
KNN	303.01	207 263.84	45.26	0.47
CatBoostRegressor	281.86	175 741.74	419.22	0.55

Таблица 3. Прогнозирование таргета SI

Название модели	**MAE**	**MSE**	**RMSE**	**R^2**
Linear regression	35.10	5 230.09	72.32	0.12
DecisionTree	27.63	4 374.26	66.14	0.26
Random Fores	27.64	4 148.01	64.41	0.30
SVR	27.10	5 214.58	72.21	0.12

Название модели	MAE	MSE	RMSE	R^2
KNN	28.52	4 563.80	67.56	0.23
CatBoostRegressor	27.24	4251.78	65.21	0.28

Обучение модели на примере Random Forest показано на рисунке 20.

Модель Random Forest

```
#Обучение модели
model = RandomForestRegressor(
    n_estimators=35,
    max_depth=9,
    random_state=42
)
model.fit(X_train, y_train)

# Предсказание
y_pred = model.predict(X_test)
print('Ансамбль деревьев:')
print_model_metrics(y_test, y_pred)
```

```
Ансамбль деревьев:
MAE  = 176.23588470435337
MSE  = 68954.82002436121
RMSE = 262.5924980351899
R2   = 0.47301096016541677
```

Рисунок 20. Пример обучения модели.

После определения лучшей модели с ручным подбором гипер параметров, для этой же модели опробован автоматический подбор гипер параметров, как показано на рисунке 21, для более точной настройки и возможного улучшения метрик.

На основе проведенных экспериментов с подбором моделей и гипер параметров, можно сделать вывод, что во всех трёх задачах регрессии наилучший показатель по метрике R^2 у модели Random Forest. Попытка подобрать наилучшие гипер рапаметры в автоматическом режиме не дало результатов. При автоматическом подборе метрика R^2 хуже, чем при ручном подборе. Возможно, это связанно с малым количество итераций.

Для лучшей модели попробуем автоматический подбор гипер параметров

```
param_dist = {
    'n_estimators': randint(3, 100),
    'max_depth': randint(2, 20),
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 10),
    'max_features': ['sqrt', 'log2', None],
    'criterion': ['friedman_mse', 'absolute_error', 'poisson'],
    'max_samples': uniform(0.1, 0.9),
}
random_search = RandomizedSearchCV(
    estimator= RandomForestRegressor(random_state=42),
    param_distributions=param_dist,
    n_iter=100,
    scoring='r2',
    cv=5,
    random_state=42,
    n_jobs=-10
)

random_search.fit(X_train, y_train)

best_params = random_search.best_params_
print("Лучшие параметры:", best_params)

best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)

print('Лучшие метрики Random Forest при подборе:')
print_model_metrics(y_test, y_pred)
```

```
Лучшие параметры: {'criterion': 'poisson', 'max_depth': 12, 'max_features': 'sqrt', 'max_samples': 0.663345
7030914155, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 42}
Лучшие метрики Random Forest при подборе:
MAE  = 178.85579905286704
MSE  = 71324.17295300814
RMSE = 267.0658588307538
R2   = 0.4549031176033481
```

Рисунок 21. автоматический подбор гипер параметров модели для задач регрессии.

Реализация моделей классификации

Для решения задачи бинарной классификации были использованы методы из библиотеки sklearn.

Так как данные уже пред обработаны и собранны под конкретные таргеты, можно приступать к обучению моделей и подбору гипер параметров.

На первом шаге, для каждого из таргетов загружаются свой датасет после чего разделяется на таргет и признаки. Для приведения признаков к единому масштабу, с помощью метода StandardScaler выполняется стандартизация признаков, после чего с помощью метода train_test_split датасет делиться на тренировочную и тестовую выборки в соотношении 80% и 20% соответственно. Пример данного процесса показан на рисунке 22.

Для обучения модели классификации важен баланс классов в выборке, что бы модель не заучила определять все записи, как доминирующий класс. Тренировочная выборка сбалансирована, так как разделение велось по медианному значению.

```
#Загрузи данные
IC50_df = pd.read_csv('./data/IC50.csv')
display(IC50_df)
```

	Unnamed: 0	IC50, mM	MaxEStateIndex	MinAbsEStateIndex	MinEStateIndex	qed	SPS	MaxPartialCharge	MaxAbsPartialCha
0	0	6.239374	5.094096	0.387225	0.387225	0.417362	42.928571	0.038844	0.293
1	1	0.771831	3.961417	0.533868	0.533868	0.462473	45.214286	0.012887	0.313
2	2	223.808778	2.627117	0.543231	0.543231	0.260923	42.187500	0.094802	0.325
3	3	1.705624	5.097360	0.390603	0.390603	0.377846	41.862069	0.038844	0.293
4	4	107.131532	5.150510	0.270476	0.270476	0.429038	36.514286	0.062897	0.257
...	...	...	...	...	...	...	...	...	
885	996	31.000104	12.934891	0.048029	-0.476142	0.382752	49.133333	0.317890	0.468
886	997	31.999934	13.635345	0.030329	-0.699355	0.369425	44.542857	0.327562	0.467
887	998	30.999883	13.991690	0.026535	-0.650790	0.284923	41.973684	0.327887	0.467
888	999	31.998959	13.830180	0.146522	-1.408652	0.381559	39.000000	0.312509	0.468
889	1000	99.999531	13.380863	0.002425	-0.447978	0.452565	48.580645	0.311311	0.468

890 rows × 105 columns

Преобразим целеавой признак в True и False для выполнения классификации.

Значение True установим если значение IC50 > медианного значения иначе установим False.

Так как медианное значение это значение центрального элемента выборки, то имеем сбалансированный датасет для обучения.

```
#Преобразим целевой признак
target = IC50_df['IC50, mM'] >= IC50_df['IC50, mM'].median()
```

```
#Выведем признаки
features = IC50_df.drop(['IC50, mM', 'Unnamed: 0'], axis=1)
```

```
#Выполним стандартизацию данных
scaler = StandardScaler()
features = scaler.fit_transform(features)
```

```
#Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

Рисунок 22. Загрузка датасета и подготовка к обучению модели классификации.

Для классификации были опробованы такие модели как:

1. Logistic regression;
2. DecisionTree;
3. Random Forest;
4. SVC;
5. KNN;
6. CatBoostRegressor.

После получения принадлежности к классам, с помощью библиотеки metrics выводятся основные метрики, такие как:

1. Accuracy - доля правильных предсказаний модели среди всех сделанных предсказаний;

2. Precision - доля объектов, выделенных как положительные, действительно являются положительными;

3. Recall - доля положительных объектов правильно идентифицирована моделью;

4. f1-score - гармоническое среднее между precision и recall;

В каждой модели вручную подбирались гипер параметры для получения наилучшего результата метрики Recall. Результаты метрик классификации для каждого из таргетов представлены в таблицах 4, 5 и 6.

Таблица 4. Классификация таргета IC50 по медианному значению.

Название модели	**Accuracy**	**precision**		**recall**		**f1-score**	
		< Median	>= Median	< Median	>= Median	< Median	>= Median
Linear regression	0.70	0.76	0.65	0.63	0.77	0.69	0.71
DecisionTree	0.72	0.74	0.71	0.73	0.71	0.74	0.71
Random Fores	0.76	0.79	0.74	0.76	0.77	0.77	0.76
SVC	0.70	0.71	0.68	0.72	0.67	0.72	0.67
KNN	0.74	0.77	0.71	0.72	0.76	0.75	0.74
CatBoostRegressor	0.77	0.78	0.76	0.79	0.75	0.78	0.75

Таблица 5. Классификация таргета CC50 по медианному значению.

Название модели	**Accuracy**	**precision**		**recall**		**f1-score**	
		< Median	>= Median	< Median	>= Median	< Median	>= Median
Linear regression	0.78	0.83	0.73	0.70	0.85	0.76	0.79
DecisionTree	0.70	0.79	0.65	0.57	0.84	0.66	0.73
Random Fores	0.79	0.84	0.76	0.74	0.85	0.78	0.80
SVC	0.76	0.78	0.74	0.74	0.78	0.76	0.76
KNN	0.76	0.84	0.71	0.67	0.86	0.74	0.78
CatBoostRegressor	0.79	0.84	0.74	0.71	0.86	0.77	0.80

Таблица 6. Классификация таргета SI по медианному значению.

Название модели	**Accuracy**	**precision**		**recall**		**f1-score**	
		< Median	>= Median	< Median	>= Median	< Median	>= Median
Linear regression	0.67	0.69	0.66	0.64	0.71	0.66	0.68
DecisionTree	0.67	0.64	0.74	0.81	0.54	0.71	0.62
Random Fores	0.70	0.70	0.71	0.72	0.69	0.71	0.70

Название модели	Accuracy	precision		recall		f1-score	
		< Median	>= Median	< Median	>= Median	< Median	>= Median
SVC	0.67	0.69	0.65	0.62	0.72	0.65	0.68
KNN	0.71	0.73	0.70	0.67	0.75	0.70	0.72
CatBoostRegressor	0.67	0.69	0.66	0.64	0.71	0.66	0.68

Как и в задачах регрессии, после определения лучшей модели с ручным подбором гипер параметров, для этой же модели опробован автоматический подбор гипер параметров для более точной настройки и улучшения метрик, показанный на рисунке 23.

Для лучшей модели попробуем автоматический подбор гипер параметров

```
param_dist = {
    'n_neighbors': randint(1, 15),
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size': randint(1, 100),
}
random_search = RandomizedSearchCV(
    estimator= KNeighborsClassifier(),
    param_distributions=param_dist,
    n_iter=500,
    scoring='roc_auc',
    cv=5,
    random_state=42,
    n_jobs=-10
)

random_search.fit(X_train, y_train)

best_params = random_search.best_params_
print("Лучшие параметры:", best_params)

best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)

print('Лучшие метрики Random Forest при подборе:')
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(classification_report(y_test, y_pred))
```

```
Лучшие параметры: {'algorithm': 'kd_tree', 'leaf_size': 15, 'n_neighbors': 11, 'weights': 'uniform'}
Лучшие метрики Random Forest при подборе:
Accuracy: 0.6741573033707865
              precision    recall  f1-score   support

       False       0.69      0.64      0.66        89
        True       0.66      0.71      0.68        89

    accuracy                           0.67       178
   macro avg       0.67      0.67      0.67       178
weighted avg       0.67      0.67      0.67       178
```

Рисунок 23. Автоматический подбор гипер параметров модели для задач классификации.

Отдельно рассмотрим классификацию таргета SI >= 8. После загрузки датасета, стандартизации и разделения на тестовую и тренировочную выборки,

был произведён анализ баланса классов, как показано на рисунке 24.

```
# Выполним анализ баланса целевых классов
chart_data = SI_df['SI'].value_counts().rename_axis('unique_class').reset_index(name='counts')

fig = px.bar(
    data_frame=chart_data,
    x="unique_class",
    y="counts",
    color='unique_class',
    orientation='v',
    height=500,
    width=1000,
)
fig.show()
```

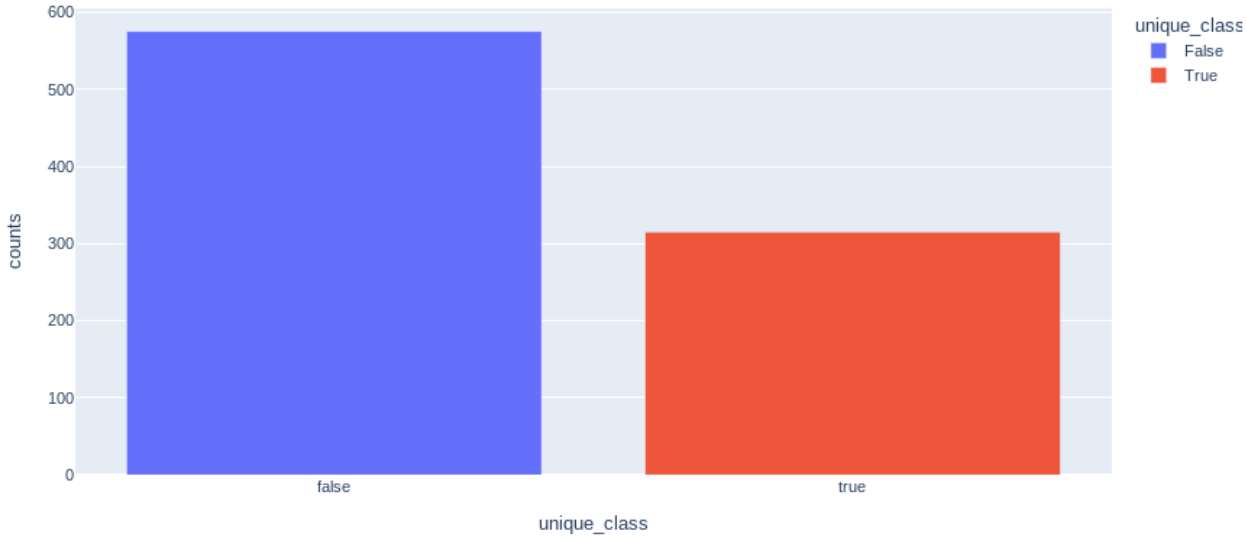

Рисунок 24. Анализ балланса классов в выборке.

Выяснилось, что объектов, у которых значение SI >= 8 на 242 объекта больше, чем объектов, у которых значение SI < 8. Для балансировки классов, как показано на рисунке 25, использовалась технология Oversampling, реализованная в методе SMOTE из библиотеки imblearn.over_sampling. Значение параметра «k_neighbor» равное 5, подобрано вручную основываясь на метриках моделей классификации.

```
#Пробую балансировку Oversampling
smote = SMOTE(sampling_strategy='auto', k_neighbors=5, random_state=42)
features, target = smote.fit_resample(features, target)
```

```
print(features.shape)
```

```
(1150, 120)
```

Рисунок 25. Применение Oversampling для балансировки классов.

После чего было выполнено обучение моделей. Метрики классификации отображены в таблице 7.

Таблица 7. Классификация таргета SI < и >= 8.

Название модели	**Accuracy**	**precision**		**recall**		**f1-score**	
		< 8	>= 8	< 8	>= 8	< 8	>= 8
Linear regression	0.76	0.73	0.79	0.78	0.74	0.75	0.76
DecisionTree	0.74	0.74	0.75	0.71	0.78	0.72	0.76
Random Fores	0.78	0.76	0.81	0.80	0.77	0.78	0.79
SVC	0.80	0.76	0.83	0.83	0.77	0.79	0.80
KNN	0.77	0.79	0.76	0.71	0.83	0.74	0.79
CatBoostRegressor	0.80	0.77	0.82	0.81	0.79	0.79	0.80

На основе проведенных экспериментов с подбором моделей и гипер параметров, можно сделать выводы:

1. Для задачи классификации таргета «IC50» по медианному значению, наилучший показатель accuracy и recall показала модель Random Forest с ручным подбором гипер параметров;
2. Для задачи классификации таргета «CC50» по медианному значению, наилучший показатель accuracy и recall показала модель Random Forest с ручным подбором гипер параметров;
3. Для задачи классификации таргета «SI» по медианному значению, наилучший показатель accuracy и recall показала модель KNN с ручным подбором гипер параметров;
4. Для задачи классификации таргета «SI» по значению «8», наилучший показатель accuracy и recall показала модель SVC с ручным подбором гипер параметров;

Заключение

В ходе выполнение курсовой работы был выполнен анализ предоставленного датасета, а именно:

1. анализ и обработка пропусков в данных;
2. анализ и удаление аномальных значений (выбросов) в данных;
3. анализ и удаление неинформативных признаков;
4. анализ и удаление признаков с высокой корреляцией друг к другу;
5. анализ и подбор оптимального количества признаков с наилучшей корреляцией к таргетам;
6. Пподготовлены датасеты по таргетам для дальнейшего использования в обучении моделей.

Также для задач регрессии и классификации было выполнено обучение нескольких моделей ML, такие как:

1. Linear regression / Logistic regression;
2. DecisionTree;
3. Random Forest;
4. SVR / SVC;
5. KNN;
6. CatBoostRegressor.

На основе метрик были выбрана лучшие модели и сформулированы выводы.