Предсказание котировок криптовалютных активов с помощью элементов Машинного обучения

Иван Кузнецов, Георгий Рябов

Апрель 2022

1 Введение

Криптовалюта — это альтернативное платежное средство. Объемы торгов этим типом активов постоянно растут, а значит интерес с точки зрения прогнозирования растет. Для стабильного заработка на криптовалютном рынке нужно уметь прогнозировать изменение котировок рынков в будущем, зная показатели рынка в моменте. Для решения такой задачи удобнее всего пользоваться методами Машинного обучения (ML), потому что они наиболее универсальны. Хотя, например, задачу рыночного арбитража, так же свойственную для этого типа активов, можно решать и стандартным алгоритмическим подходом. [1]

Предсказание котировок криптовалютных активов с помощью ML требует применения на практике базовых методов машинного обучения и сбора данных. Удобнее всего для таких целей подходит язык программирования Python с его многочисленными библиотеками. Сведем задачу прогнозирования к задаче бинарной классификации, другими словами необходимо предсказать к какому из двух классов: "изменение цены положительно"или "изменение цены отрицательно — будет относиться цена актива в следующий промежуток времени. Такие изменения часто называют "свечами". Будем ориентироваться на высокочастотную торговлю, предполагающую совершение большого количества сделок в маленький промежуток времени.

Задача работы доказать или опровергнуть утверждение, что качество ML моделей зависит от количества рассматриваемых свечей, а также отобрать наиболее полезные признаки, по которым можно определить наиболее подходящую ML модель для решения этой задачи.

Можно выделить соответствующие подзадачи:

- Сбор данных с криптобиржи.
- Обработка данных.
- Обучение ML моделей.
- Сравнение качества ML моделей.

2 Сбор данных

Для качественного обучения алгоритма нам нужен большой объем данных. Удобнее всего воспользоваться встроенным API криптобиржи Binance. [2]

API (Application Programming Interface) - это специальный интерфейс программирования приложений, позволяющий сервисам взаимодействовать, получать доступ и обмениваться данными. В данном случае API позволяет взаимодействовать с криптобиржей с помощью языка Python. [3]

Для получения доступа к этому протоколу мы создали Binance кошелёк, после чего нам стали доступны apikey и secretkey - специальные ключи для взаимодействия с биржей и нашими активами на ней. Далее мы подключили библиотеку python-binance для удобного взаимодействия с функционалом API. Выберем пару активов BTCUSDT, так как именно торги этой парой активов составляют наибольшую часть оборота криптобиржи. Выделим минутные изменения рынка с начала этого года. Сырые данные представляют собой таблицу 1 на странице 4, состоящую из 11 столбцов и 1140000 строк. Данные предстваляют собой свечи.

Столбцы таблицы:

Open Time - время открытия торгов
Open - цена актива на открытии торгов
High - максимум цены актива за торговую сессию
Low - минимум цены актива за торговую сессию
Close - цена актива на закрытии торгов
Low - минимум цены актива за торговую сессию
Volume - объем торгов за торговую сессию
Close Time - время закрытия торгов
Quote Asset Volume - Объем торгов второго элемента пары
Number of Trades - Количество сделок
TB Base Volume - Количество ордеров на покупку первого элемента пары
TB Quote Volume - Количество ордеров на покупку второго элемента пары

Этого объема данных достаточно для обучения алгоритмов бинарной классификации и оценки его качества.

3 Обработка данных

Перед тем, как обучать ML модели нужно изучить датасет и определить пространство признаков. Данных у нас очень много, а это неизбежно влечет статистические выбросы и различные ошибки в форматировании. Это все можно сделать с помощью функционала библиотек Pandas и Sckit-Learn. Избавимся от пустых (Not a Number) значений и дубликатов в таблице, изменим тип столбцов. Удалим признак Close Time и сформируем новые признаки, ориентируясь на столбец Open Time выделим день недели (Day of Week), день месяца (Day of Month), номер месяца (Month of Year), номер неделю (Week of Year), час дня (Hour of Day), и минуту (Minute of Hour). Далее вычислим изменение цены в следующий момент времени $Delta = Close - Open$ и сопоставим его текущем. Далее создадим столбец SignDelta применив функцию $f(x) = sign(x)$ к каждому элементу столбца. Исключим из таблицы объекты для которых определить такое целевое значение нельзя. В результате получаем столбец, элементы которого принимают бинарные значения. Их мы будем предсказывать. Обновленное пространство признаков представлено в таблице 2 на странице 5.

4 Описание ML моделей

Рассмотрим наиболее простые модели, которые применяются в решении задач бинарной классификации: логистическая регрессия, метод ближайших соседей, дерево решений и случайный лес.

4.1 Логистическая регрессия

Логистическая регрессия — это модель, часто используемая для прогнозирования вероятности возникновения некоторого события.

Пусть некоторый объект описывается n числовыми признакамиf_j: X$\to R$, $j=1,\ldots,n$. Тогда пространство признаков объекта есть X = R^n. Обозначим Y — конечное множество классов. Пусть задана выборка, на которой можно обучить модель $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$.

	0	1	2
Open Time	1651363200000	1651363260000	1651363320000
Open	37630.80000000	37645.92000000	37642.28000000
High	37665.29000000	37645.93000000	37684.71000000
Low	37612.43000000	37622.48000000	37642.28000000
Close	37645.92000000	37642.29000000	37672.10000000
Volume	85.80484000	28.27385000	70.97044000
Close Time	1651363259999	1651363319999	1651363379999
Quote Asset Volume	3229135.93967060	1064073.13040180	2673472.22944150
Number of Trades	1271	677	904
TB Base Volume	37.66777000	18.31689000	29.13056000
TB Quote Volume	1417547.36651720	689329.63879220	1097093.92147740

Рис. 1: Транспонированная таблица сырых данных

	0	1	2	3	4
Open	3.764592e+04	3.764228e+04	3.767211e+04	3.773709e+04	37717.790000
High	3.764593e+04	3.768471e+04	3.773897e+04	3.773709e+04	37722.050000
Low	3.762248e+04	3.764228e+04	3.766189e+04	3.770074e+04	37684.020000
Close	3.764229e+04	3.767210e+04	3.773708e+04	3.771779e+04	37711.910000
Volume	2.827385e+01	7.097044e+01	8.117584e+01	3.843930e+01	23.574000
Quote Asset Volume	1.064073e+06	2.673472e+06	3.059013e+06	1.449841e+06	888831.694443
Number of Trades	6.770000e+02	9.040000e+02	1.168000e+03	1.022000e+03	742.000000
TB Base Volume	1.831689e+01	2.913056e+01	5.489804e+01	1.517617e+01	9.336230
TB Quote Volume	6.893296e+05	1.097094e+06	2.068797e+06	5.723768e+05	352003.848227
Day of Week	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	6.000000
Day of Month	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000
Month of Year	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000
Week of Year	1.700000e+01	1.700000e+01	1.700000e+01	1.700000e+01	17.000000
Hour of Day	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
Minute of Hour	1.000000e+00	2.000000e+00	3.000000e+00	4.000000e+00	5.000000

Рис. 2: Транспонированная таблица обработанных данных

Случай двух классов
Положим Y=$\{-1, +1\}$. В логистической регрессии алгоритм классификации: X→Y будет иметь вид:

a(x,w) = $\text{sign}\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right) = \text{sign}\langle x, w\rangle$, где w_j — вес j-го признака, w_0 — порог принятия решения, w=(w_0,w_1,...,w_n) — вектор весов. Введем, что искусственный признак для удобства записи: f_0(x) = - 1.

Обучение линейного классификатора состоит в том, чтобы по ограниченной выборке объектов для обучения X^m найти вектор w. В логистической регрессии это задача сводится к задаче минимизации, где функция потерь выглядит так:

Q(w) = $\sum_{i=1}^{m} \ln\left(1 + \exp(-y_i\langle x_i, w\rangle)\right) \to \min_w$

Найденный вектор-w решение, позволяет не только решить задачу классификацию a(x) = $\text{sign}\langle x, w\rangle$ для произвольного объекта x, но и оценивать вероятности его принадлежности к классам:

$P\{y|x\} = \sigma\left(y\langle x, w\rangle\right), \quad y \in Y,$

где $\sigma(z) = \dfrac{1}{1 + e^{-z}}$ — сигмоидная функция. [4]

4.2 Метод ближайших соседей

Метод k-ближайших соседей это алгоритм, который применяется для решения задач автоматической классификации объектов или регрессии.

Если метод используется для классификации, то объекту присваивается тот класс, который является наиболее часто встречается среди k соседей этого объекта. А если метод используется для решения задачи регрессии, то объекту присваивается среднее значение по k ближайших к нему объектам.

Пусть обучающая выборка выглядит также:
$X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Определим функцию расстояния $\rho(x, x')$. Чем больше становится значение этой функции, тем менее схожи два выбранных объекта x,x'.

Произвольно выберем некоторый объект u, расположим объекты обучающей выборки x_i в порядке не убывания их расстояний до u:

$$\rho(u, x_{1;u}) \leq \rho(u, x_{2;u}) \leq \cdots \leq \rho(u, x_{m;u})$$

где через $x_{i;u}$ обозначается тот объект из выбранной обучающей выборки, который является i-м соседом объекта u. Таким же образом обозначим ответ на i-м соседе: $y_{i;u}$. Получаем

$$\text{a(u)} = \arg\max_{y \in Y} \sum_{i=1}^{n} [y(x_{i;u}) = y](w(i;u))$$

В наиболее общем виде алгоритм ближайших выглядит так. [5]

4.3 Дерево решений

Дерево принятия решений (также называют деревом классификации или регрессионным деревом) — это широко распространенное средство поддержки принятия решений. Дерево представляет собой набор «листьев» и «веток». На ветках дерева решения расположены признаки, от которых зависит целевая функция, а в «листьях» определены значения целевой функции. В оставшихся узлах — признаки, по которым различаются случаи. Для классификации нового объекта, необхадимо спуститься по соответствующим ребрам ("веткам") и найти соответствующее значение листа.

Деревья решений могут быть использованы в качестве математических и вычислительных методов, чтобы помочь описать, которые могут быть записаны следующим образом:

$$(x, Y) = (x_1, x_2 \ldots, x_k, Y)$$

Зависимая переменная Y является целевой, которую необходимо классифицировать. Вектор x состоит из входных переменных x_1, x_2, x_3 и т. д. Они используются для выполнения этой задачи. [6]

4.4 Случайный лес

Случайный лес — это распространенный алгоритм машинного обучения, суть которого заключается в создании ансамбля решающих деревьев. Описанный алгоритм не редко применяется в задачах классификации и регрессии. Каждое из деревьев даёт маленькое качество классификации, но за счёт их большого количества результат получается хорошим.

Кратко опишем алгоритм построения случайного леса, состоящего из N деревьев:

1. Сгенерировать выборку X_n
2. Построить решающее дерево b_n по выборке X_n:
   (a) по заданному критерию мы пытаемся определить наилучший признак и продолжать разбиение в дереве уже по нему и так до исчерпания выборки
   (b) дерево строится, пока в каждом листе не более $n_{\min}$ объектов или пока не достигнем определенной высоты дерева
   (c) при каждом разбиении сначала выбирается m случайных признаков из n исходных, и оптимальное разделение выборки ищется только среди них.

В итоге классификатор выглядит так $a(x) = \frac{1}{N} \sum_{i=1}^{N} b_i(x)$ [7]

5 Обучение ML моделей

Изучим зависимость качества модели от глубины данных. Другими словами, будем группировать объекты в таблице в кластеры по 1, 5, 10, 50, 100, 200 штук, увеличивая количество признаков.

Для обучения и сравнения ML моделей будем использовать метод кроссвалидаци (Cross-Validation) из библиотеки sklearn. Другими словами, разобьем датасет на 10 частей. Обучим модель на 9 из них, а на оставшейся протестируем. Повторим процедуру 10 раз, каждый раз по очереди меняя часть данных для теста модели. [8]

Воспользуемся объектом MinMaxScaler из библиотеки Sckit-Learn для нормировки пространства признаков. Значение каждой ячейки соответствующего столбца изменится по формуле:

$$x := x \cdot (max - min) + min$$

x - значение элемента столбца
min - минимальное значение элементов столбца
max - максимальное значение элементов столбца

Обучим описанные выше модели.

6 Сравнение ML моделей

Определим параметры, по которым будет проводиться сравнение моделей. Для удобства записи построим таблицу 3 на стр. 9

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{Precision \cdot Recall}{Precision + Recall}$$

Метрика Accuracy показывает долю верно угаданных ответов. Но этого мало, чтобы делать выводы о качестве модели. Имеют место ошибки 1 рода (FN) и ошибки 2 рода (FP). Для их контроля определим Recall и Precision соответственно и возьмем их среднее-гармоническое. Это F1 метрика. [9] Для сравнения качества моделей будем использовать именно F1 метрику, так как она позволяет контролировать ошибки 1 и 2 рода.

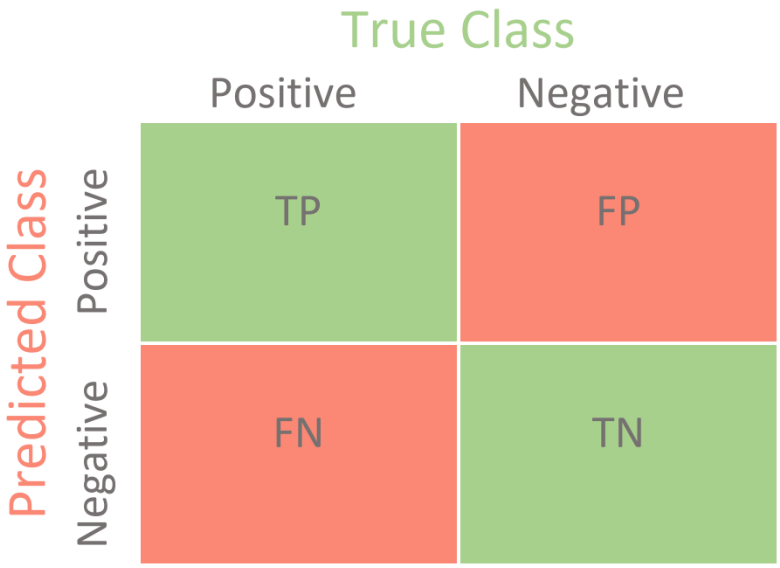

Рис. 3: Confusion matrix

Во время процесса Cross-Validation получаем 10 значений Accuracy, Recall, Precision и F1. Посчитаем их среднее и определим стандартное отклонение. Графики приведены далее.

По графикам 4 - 10 кажется на стр 10, что качество модели выше при обучении на кластере из 100 свечей. Определим, значимо ли это отличие. Проведем проверку статистической гипотезы о равенстве двух средних. Воспользуемся Т-тестом Стьюдента. [10]

Импортируем библиотеку scipy для python. Нулевая гипотеза теста — это предположение, что разница качество моделей не значима статистически, и находится на уровне значимости 5% а альтернативная гипотеза — это предположение, что разница значима статистически. Воспользуемся функцией ttest_ind и сравним массивы F1 метрик для 1 и 100 свечей для каждой из моделей, полученных в результате кросс-валидации.

Результаты теста для всех моделей показывают достигаемый уровень значимости выше 5%. Таким образом, мы не отвергаем нулевую гипотезу в пользу альтернативы, что свидетельствует, что качество не зависит от глубины данных.

F1 score	Кол-во свечей
0.473623	1.0
0.503301	5.0
0.494786	10.0
0.521666	50.0
0.516934	100.0
0.448425	150.0

F1 score	Кол-во свечей
0.482753	1.0
0.478543	5.0
0.469562	10.0
0.471014	50.0
0.481811	100.0
0.402592	150.0

Рис. 4: Логистическая регрессия и Метод ближайших соседей

F1 score	Кол-во свечей
0.461972	1.0
0.438819	5.0
0.421750	10.0
0.451765	50.0
0.507939	100.0
0.427495	150.0

	F1 score	Кол-во свечей
0	0.421700	1.0
1	0.410791	5.0
2	0.427052	10.0
3	0.467691	50.0
4	0.498399	100.0
5	0.411402	150.0

Рис. 5: Дерево решений и Случайный лес

Дальнейшее сравнение моделей между собой, в попытках определить какая лучше справляется с поставленной задачей, будем проводить при 100 свечах. Проведем такие же тесты для каждой пары моделей. Нулевую и альтернативную гипотезы сформулируем аналогично. Результаты каждого теста также показывают достигаемый уровень значимости выше 5%. Отвергать нулевую гипотезу в каждом из этих случае не стоит.

7 Выводы

В нашей работе мы изучили основы процесса сбора и обработки данных, изучили основные модели машинного обучения, применяемые в задачах бинарной классификации и обучили модель предсказывать знак изменения

цены актива на следующем промежутке времени, зная показатели рынка в моменте. Мы пришли к выводу, что различия в качестве моделей не статистически значимо, при рассмотрении разного количества свечей. Различия в качестве модели логистической регрессии, метода ближайших соседей, дерева решений и модели случайного леса в нашей задаче, также оказались незначительны. Скорее всего, это связано с тем, что модели построенные нами слишком просты и не способны уловить сложные зависимости рынка криптовалюты. В перспективе, модели можно улучшать, добавляя дополнительные признаки, основанные на техническом анализе.

8 Обязанности

Иван Кузнецов выполнил:

1. Выполнил отчистку данных собранных с использованием api Binance.
2. Применил и обучил модель машинного обучения "Дерево решений".
3. Применил и обучил модель машинного обучения "Случайный лес".
4. Сравнил и проанализировал различные модели машинного обучения.

Георгий Рябов выполнил:

1. Выполнил сбор данных с использованием api Binance.
2. Выполнил первичный анализ моделей машинного обучения для дальнейшего использования.
3. Применил и обучил модель машинного обучения "Логистическая регрессия".
4. Применил и обучил модель машинного обучения "Метод ближайших соседей".

9 Список литературы

[1] - https://www.netinbag.com
[2] - https://www.binance.com/ru
[3] - https://www.programmableweb.com
[4] - http://www.machinelearning.ru/wiki/LogisticRegression
[5] - http://www.machinelearning.ru/wiki/KNearestNeighbors
[6] - https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[7] - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[8] - https://scikit-learn.org/stable/modules/cross_validation.html
[9] - https://towardsdatascience.com
[10] - https://r-analytics.blogspot.com

10 Дополнительные графики

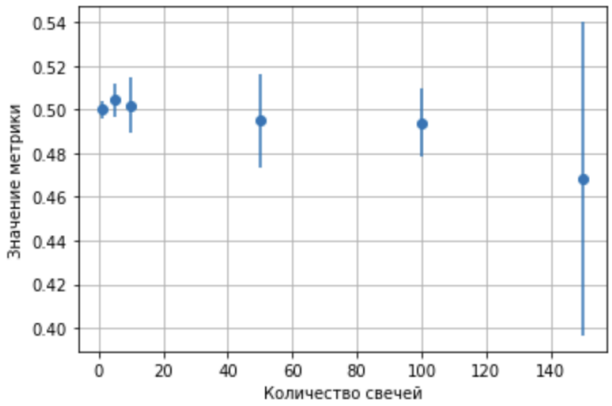

Рис. 6: Логистическая регрессия Accuracy

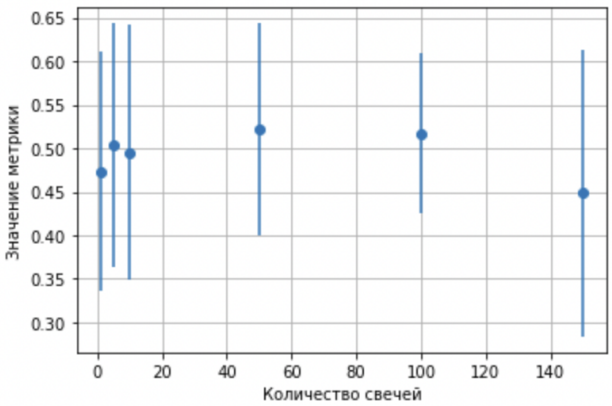

Рис. 7: Логистическая регрессия F1

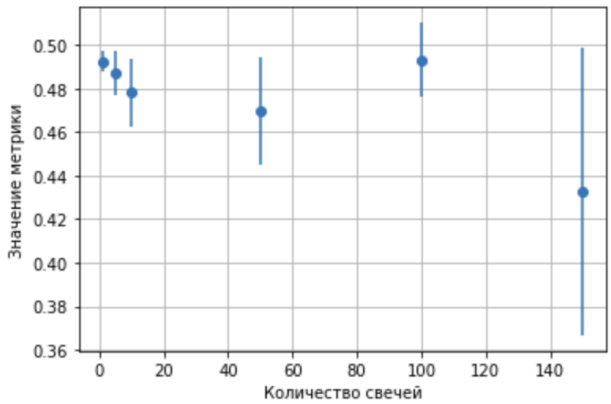

Рис. 8: Метод ближайших соседей Accuracy

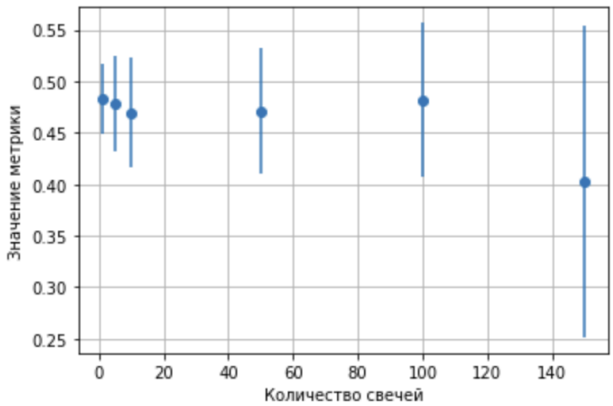

Рис. 9: Метод ближайших соседей F1

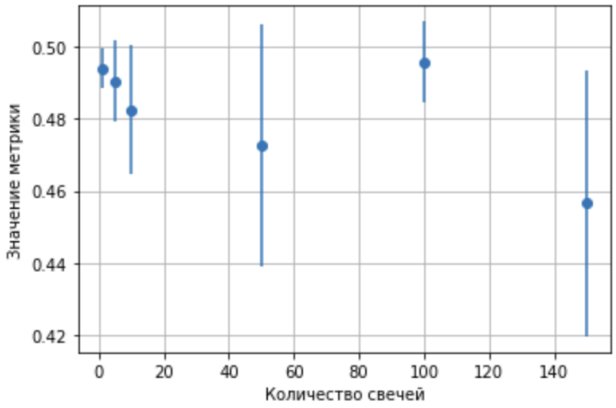

Рис. 10: Дерево решений Accuracy

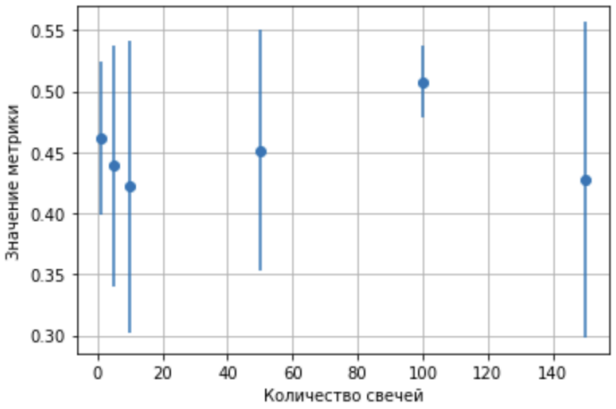

Рис. 11: Дерево решений F1

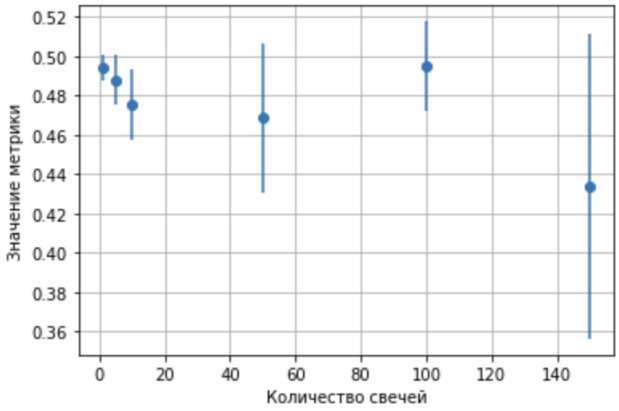

Рис. 12: Случайный лес Accuracy

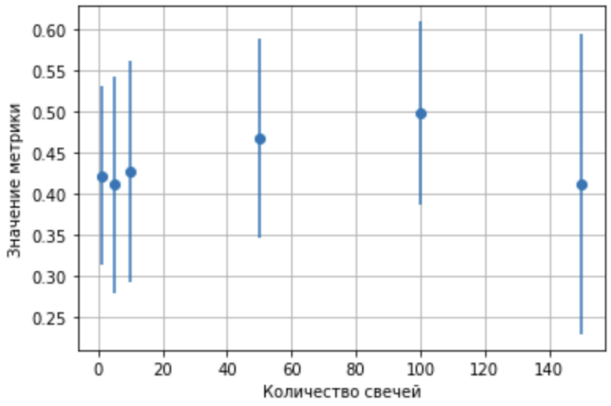

Рис. 13: Случайный лес F1