

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**МАШИННОЕ ОБУЧЕНИЕ И ТРЕЙДИНГ**  
**КУРСОВАЯ РАБОТА**

студента 3 курса 331 группы  
направления 10.05.01 — Компьютерная безопасность  
факультета КНиИТ  
Никитина Арсения Владимировича

Научный руководитель

д.ф.-м.н., доцент

\_\_\_\_\_

М. Б. Абросимов

Заведующий кафедрой

д.ф.-м.н., доцент

\_\_\_\_\_

М. Б. Абросимов

Саратов 2022

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	5
1 Теоретическая часть .....	7
1.1 Временной ряд .....	7
1.2 Основы технического анализа .....	7
1.2.1 Японская свеча .....	7
1.2.2 Определение тенденции с помощью макисмумов и мини- мумов .....	8
1.2.3 Линия роста (падения) .....	9
1.3 Перцептрон как один из методов реализации нейронных сетей .....	10
1.4 Использование нейронных сетей для поиска фигур технического анализа .....	11
2 Практическая часть .....	13
2.1 Описание инструментов и библиотек программной реализации .....	13
2.2 Программная реализация алгоритма .....	14
2.2.1 Флаги и вымпелы .....	14
2.2.2 Прямоугольник .....	16
2.2.3 Ромб .....	17
2.2.4 Голова и плечи .....	18
2.3 Поиск экстремальных значений .....	19
2.4 Проверка работы алгоритма .....	20
2.4.1 Energy Focus .....	21
2.4.2 Yandex .....	22
2.4.3 Rectruiter.com Group .....	23
2.4.4 Greenbox POS .....	24
2.4.5 Campbell Soup Company .....	25
2.4.6 Johnson and Johnson .....	26
2.4.7 Altria Group .....	27
2.4.8 Sprouts Farmers Market .....	28
2.4.9 Bitcoin .....	29
2.4.10 EOS .....	30
ЗАКЛЮЧЕНИЕ .....	32

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	33
Приложение А Код drawing.py .....	34
Приложение Б Код figures.py .....	37
Приложение В Код main.py .....	42

## ВВЕДЕНИЕ

В последнее время анализ курсов акций является немаловажной проблемой. Так различают фундаментальный анализ, основанный на финансовой и производственной деятельности компаний, а также на общей экономической обстановке выбранной сферы. Также определен технический анализ, который основан на прогнозировании изменения курсов в зависимости от закономерностей изменений цен в прошлом в аналогичных обстоятельствах.

Таким образом, в данной работе будет рассмотрена концепция технического анализа, его основные методы, инструменты, а также будет разработан детерминированный алгоритм нахождения технических фигур технического анализа.

Более 150 лет торговые залы Чикагской фондовой биржи были местом проведения торговых сделок покупки и продажи.

Вплоть до 1997 года более 10 тысяч людей торговали в общих залах Чикагских бирж (в так называемых «биржевых ямах»). Позднее, в том же году появилась торговля с помощью компьютеров. Сейчас только около 10% трейдеров продолжают торговать в биржевых залах (остальные 90% торгуют с компьютеров).

Несложно догадаться, что большинство трейдеров, использующих в качестве основного инструмента заработка компьютер, пользуются также различными вспомогательными средствами, обеспечивающие им уверенность в своих действиях.

Итак, одним из средств, помогающих выполнять анализ поведения котировок акций, является технический анализ и, в частности, нахождение и анализ технических фигур.

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Перед анализом теоретической составляющей машинного обучения и технического анализа в трейдинге, стоит ввести ряд терминов и определений, являющихся основой для понимания работы алгоритмов поиска технических фигур, а также для определения основных составляющих трейдинга.

*Трейдинг* — покупка и продажа акций компаний с целью заработка на ежедневном демпинге цен. Трейдеры внимательно следят за краткосрочными колебаниями цен на эти акции, а затем пытаются купить подешевле и продать подороже. Этот краткосрочный подход отличает биржевых трейдеров от традиционных инвесторов фондового рынка, которые склонны работать в долгосрочной перспективе.

*Технический анализ* — это торговая дисциплина, используемая для оценки инвестиций и выявления торговых возможностей путем анализа статистических тенденций, полученных в результате торговой деятельности, таких как движение цены и объем. В отличие от фундаментального анализа, который пытается оценить стоимость ценной бумаги на основе результатов бизнеса, таких как продажи и прибыль, технический анализ фокусируется на изучении цены и объема.

*Фигура технического анализа* — определенная закономерность подряд идущих значений курса акции, которая отображаются на графике и позволяют трейдеру получать определенные сигналы о том, как стоимость актива будет строиться в будущем.

*Волатильность* — финансовый показатель, отражающий то, как сильно меняется цена на актив или товар за короткий промежуток времени.[1]

*Искусственный интеллект* (англ. Artificial Intelligence) — технология создания алгоритмов, лежащих в основе проектирования интеллектуальных машин и программ, способных имитировать деятельность человека.

*Нейронная сеть (нейросеть)* (англ. Neural Network) — математическая модель, чаще всего имеющая программную интерпретацию, сутью которой является реализация деятельности, похожей на деятельность биологических нейронных сетей. Нейронная сеть используется при создании какого-либо из алгоритмов искусственного интеллекта и состоит из совокупности нейронов, соединенных между собой связями.

*Признак* (англ. Feature) — каждый отдельный элемент информации, вклю-

чаемый в представление о каком-либо анализируемом объекте.

*Машинное обучение* (англ. Machine Learning) — область науки об искусственном интеллекте, которая изучает способы создания алгоритмов, которые могут обучаться (развиваться).

# 1 Теоретическая часть

## 1.1 Временной ряд

*Временной ряд* — это определенные данные, собранные в разные моменты времени, графиком которого является значение времени по оси абсцисс и значение наблюдаемой величины в текущий момент времени по оси ординат. [1]

Временной ряд может иметь детерминированный компонент в определенный момент времени, тогда говорят, что данные временного ряда имеют временной тренд.

Временные тренды в данных временных рядов также имеют значение для тестирования и моделирования. Надежность модели временных рядов зависит от правильного определения и учета временных тенденций.

Определить, наличие временного тренда на графике можно с помощью проведения возрастающей (убывающей) линии и выявления сосредоточения значений наблюдаемой величины в окрестности этой линии.

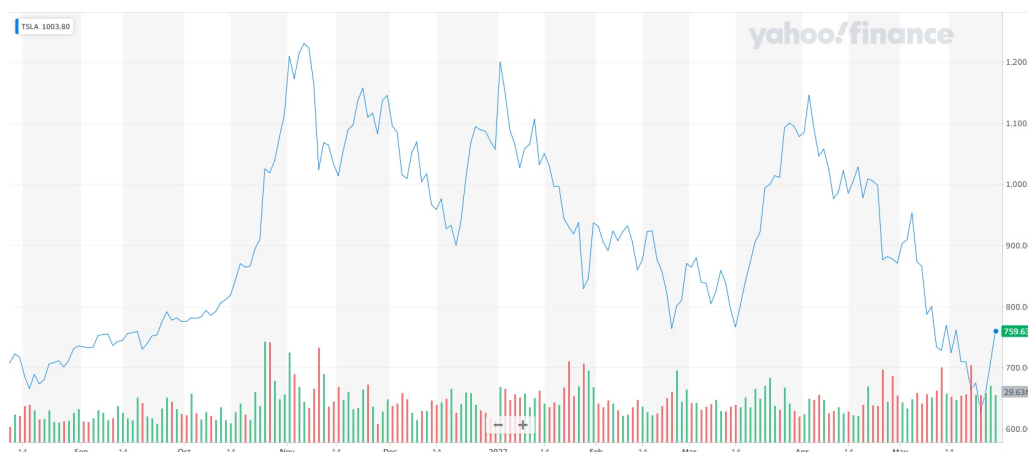


Рисунок 1 – Пример временного ряда

## 1.2 Основы технического анализа

### 1.2.1 Японская свеча

Графики свечей — наиболее популярная в Японии и самая древняя форма технического анализа. «Японские свечи» появились раньше, чем штриховые графики и графики «крестики-нолики». Японцы осознали важность технического анализа давным-давно. Они первыми стали торговать фьючерсами. В середине XVII в. они торговали «пустыми» рисовыми контрактами (рисом, которого еще не было, — иначе говоря, рисовыми фьючерсами). [3]

Японская свеча состоит из следующих компонентов:

1. Open (цена открытия) — цена на начало периода.
2. Close (цена закрытия) — цена на конец периода.
3. High — максимальная цена в течение периода.
4. Close — минимальная цена в течение периода.

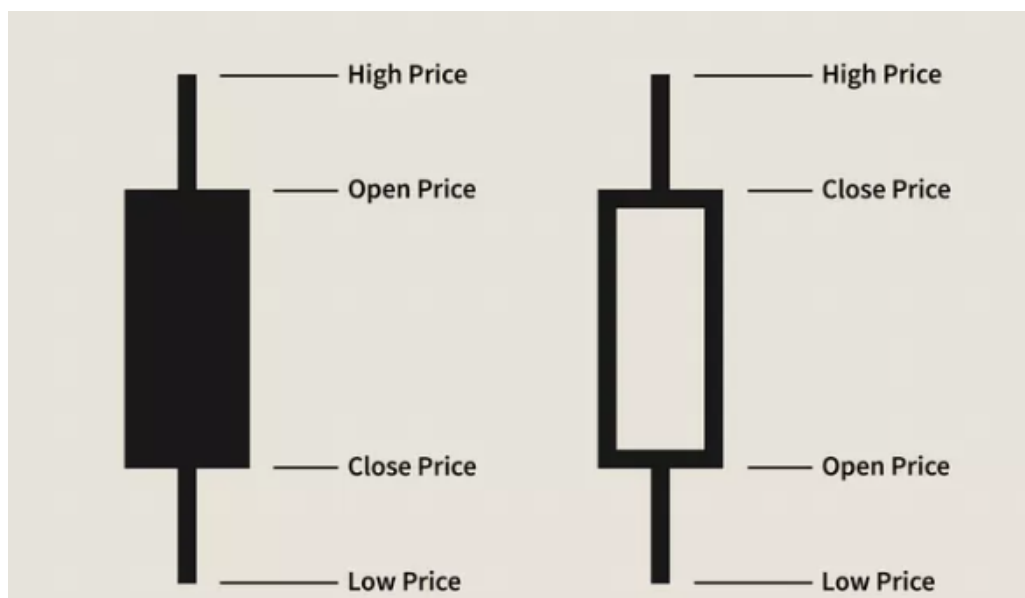


Рисунок 2 – Японская свеча

Для того, чтобы можно было отрисовать график акций, требуется использовать один из показателей японской свечи. В данной работе далее используется показатель *Close*.

### 1.2.2 Определение тенденции с помощью максимумов и минимумов

В качестве одного из стандартных методов определения тенденции повышения можно рассмотреть последовательности более высоких максимумов и более низких минимумов.

Итак, тенденция повышения является ненарушенной ровно до тех пор, пока зафиксированный предыдущий относительный минимум не будет пробит, то есть не образуется новый минимум. Если такое происходит, то делается вывод о том, что тенденция закончилась или же в скором времени это произойдет.

Аналогично можно определить и понижающую тенденцию как последовательность более низких минимумов и более низких максимумов. Понижающая тенденция является ненарушенной до тех пор, пока зафиксированный предыдущий максимум не пробит. Если же наблюдается пробитие максимума, то возможно понижающая тенденция подходит к своему завершению.





Рисунок 3 – Пример нисходящего тренда

### 1.2.3 Линия роста (падения)

Кумулятивная дневная линия роста / падения является одним из наиболее известных индикаторов разброса рынка и часто используется для обнаружения дивергенции (расхождения) с одним из свободных индексов рынка. Как правило, кумулятивная линия роста (падения) вычисляется как скользящая сумма чистого количества растущих акций.

Вычисление значений линии роста (падения) проводится в два этапа:

1. Вычисление разности между количеством растущих и количеством падающих акций с сохранением знака результата. Полученное значение также называется *чистым количеством растущих акций*.
2. Прибавить вычисленную сумму за данный день к значению накопленной суммы ежедневных количеств *растущих акций*.

Также предлагается приступать к вычислению кумулятивной скользящей только после проведения нормализации дневных данных по формуле:

$$N = \frac{A-D}{A+D},$$

где  $N$  — отношение чистого количества растущих акций к общему числу акций, показавших изменение цены,  $A$  — количество растущих акций,  $D$  — количество падающих акций.

### 1.3 Перцептрон как один из методов реализации нейронных сетей

Типичным примером модели глубокого обучения является глубокая сеть прямого распространения, или многослойный перцептрон (МСП). Многослойный перцептрон — это математическая функция, отображающая множество входных значений на множество выходных. Эта функция является композицией нескольких более простых функций. Каждое применение одной математической функции можно рассматривать как новое представление входных данных.

Глубина позволяет обучать многошаговую компьютерную программу. Каждый слой представления можно мыслить себе как состояние памяти компьютера после параллельного выполнения очередного набора инструкций. Чем больше глубина сети, тем больше инструкций она может выполнить последовательно. Последовательное выполнение инструкций расширяет возможности, поскольку более поздние инструкции могут обращаться к результатам выполнения предыдущих.

Есть два основных способа измерить глубину модели. Первый оценивает архитектуру на основе числа последовательных инструкций, которые необходимо выполнить. Можно считать, что это длина самого длинного пути в графе, описывающем вычисление каждого выхода модели по ее входам. Как у двух эквивалентных компьютерных программ могут быть разные длины пути в зависимости от языка, на котором они написаны, так и одна и та же функциональность может быть изображена графами с разной длиной пути в зависимости от того, какие функции допускаются в качестве шагов.

При другом подходе, используемом в глубоких вероятностных моделях, глубиной модели считается не глубина графа вычислений, а глубина графа, описывающего связи концепций. В этом случае граф вычислений, выполняемых для вычисления представления каждой концепции, может быть гораздо глубже, чем граф самих концепций. Связано это с тем, что понятие системы о простых концепциях можно уточнять, располагая информацией о более сложных. Например, система ИИ, наблюдающая изображение лица, на котором один глаз находится в тени, первоначально может распознать только один глаз. Но, обнаружив присутствие лица, система может заключить, что должен быть и второй глаз. В таком случае граф концепций содержит только два слоя — для глаз и для лиц, тогда как граф вычислений содержит  $2n$  слоев, если мы  $n$  раз уточняем оценку каждой концепции при известной информации о второй.

Поскольку не всегда ясно, какой из двух подходов – глубина графа вычислений или глубина графа вероятностной модели – более релевантен, и поскольку разные люди по-разному выбирают наборы примитивных элементов, из которых строятся графы, не существует единственно правильного значения глубины архитектуры, как не существует единственно правильной длины компьютерной программы. И нет общего мнения о том, какой должна быть глубина, чтобы модель можно было считать «глубокой». Однако можно все-таки сказать, что глубокое обучение — это наука о моделях, в которых уровень композиции обученных функций или обученных концепций выше, чем в традиционном машинном обучении.

#### **1.4 Использование нейронных сетей для поиска фигур технического анализа**

Существует целая теория распознавания изображений образов, являющаяся разделом кибернетики.

Нейронные сети в рассматриваемой задаче могут дать преимущество в связи с тем, что они имеют аффинную инвариантность к представляемым данным, в частности, к масштабу и углам смещения фигур технического анализа. Тогда данные отклонения можно считать шумом, с которым нейронная сеть, как правило умеет работать.

Одним из первых вопросов при использовании нейронной сети для нашей задачи является количество входов и подаваемая на них информация. Также важным моментом является выбор типа нейронной сети, ее параметров и методов обучения. Рассматриваемая проблема относится к классу задач классификации, а, значит, необходимо понять, к какой фигуре относится текущее поведение временного ряда.

В качестве входной информации для нейронной сети можно использовать координаты начала отрезка, его окончания и угол наклона прямой.

Итак, для задачи о предсказании поведения текущего курса акций разумно в качестве нейросетевого ядра использовать многослойный перцептрон с обучением по методу обратного распространения ошибки. Данный тип нейронной сети хорошо зарекомендовал себя в задачах прогнозирования.

На входы нейронной сети будут подаваться параметры нескольких подряд идущих отрезков. На выходе будут параметры следующего отрезка.

Нейронные сети, безусловно, являются одним из методов поиска фигур

технического анализа и предсказания поведения акций в зависимости от полученных результатов работы. В данной работе же рассматриваются детерминированные подходы которые пытаются выделить технические фигуры на заданном временном ряде и затем провести прогнозирование поведения ряда в зависимости от найденных фигур.

## 2 Практическая часть

### 2.1 Описание инструментов и библиотек программной реализации

В виду своей удобности использования, простоты написания кода, а также различных дополнительных библиотек для анализа акций, в данной работе используется язык программирования Python. Простота применения Python для написания кода и возможность его отладки через такие пакетные менеджеры, как `conda`, упрощают обмен библиотеками и результатами исследований. Его экосистема библиотек для работы в сфере трейдинга, таких как `NumPy`, `yfinance` и `Matplotlib`, упрощают анализ результатов работы алгоритма, получение котировок акций с различной временной градацией.

`Yfinance` — популярная библиотека с открытым исходным кодом, разработанная Раном Арусси как средство доступа к финансовым данным, расположенным на ресурсе `Yahoo Finance`.

`Yahoo Finance` предлагает обширный набор рыночных данных по акциям, облигациям, валютам и криптовалютам. Для удобства использования в данной библиотеке присутствует возможность получения отчетов и анализов данных, а также дополнительные параметры получаемых данных, что отличает его от ряда конкурентов.

Также стоит отметить, что отличительной особенностью `yfinance` является то, что библиотека позволяет получить градуирование данных вплоть одной минуты. Однако, что данные за текущий день получить нельзя. Если требуется получить данные за день (то есть градуирование составляет по часам и меньше), то минимально возможная разница в днях с текущей датой составляет 60 дней.

Получаемые данные представляют из себя `csv`-файл, в котором присутствуют вышеописанные элементы свечей: максимальная (минимальная) цена, открытие (закрытие) акции.

`Matplotlib` — это кроссплатформенная библиотека для визуализации данных и графического построения графиков для Python и `NumPy`. В данной работе использование библиотеки является необходимым для отображения котировок акций и найденных фигур технического анализа с предполагаемым ростом (падением) курса. По оси абсцисс будем брать градуирование даты, а по оси ординат будем использовать полученные с помощью `yfinance` котировки, используя поле `close`.

## 2.2 Программная реализация алгоритма

Детерминированная реализация поиска технических фигур на акциях будет состоять из нескольких этапов:

1. Выбор рассматриваемой акции.
2. Получение котировок за определенный период.
3. Нахождение экстремумов временного ряда.
4. Итеративный проход по полученным значениям с проверкой условия попадания  $k$  подряд идущих значений в некоторые допуски.
5. Принятие решения о нахождении фигуры и прогноз дальнейшего роста или падения курса.

Итак, для начала определимся, каким образом можно найти фигуры технического анализа путем сравнения подряд идущих значений с допусками. Для этого рассмотрим основные фигуры технического анализа и получим закономерности, по которым и будут проверяться условия попадания значений в результат.

### 2.2.1 Флаги и вымпелы

Флагами и вымпелами называются узкие и краткосрочные (например, от одной до трех недель) фазы консолидации внутри трендов. Фигура называется флагом, когда она ограничена параллельными линиями, и вымпелом, когда линии сходятся.

Флаги и вымпелы обычно отражают паузы в сильном тренде, а также зачастую направлены в сторону, противоположную основной тенденции. Иначе говоря, за этими моделями, как правило, следует движение цен в том же направлении, которое предшествовало их формированию.

Пробой границы флага или вымпела может рассматриваться как подтверждение того, что тренд продолжается, и как сигнал к торговле в направлении тренда.

Значительный пробой за пределы флага или вымпела в направлении, противоположном ожидаемому, то есть против основной тенденции, может рассматриваться как сигнал потенциального разворота тенденции.

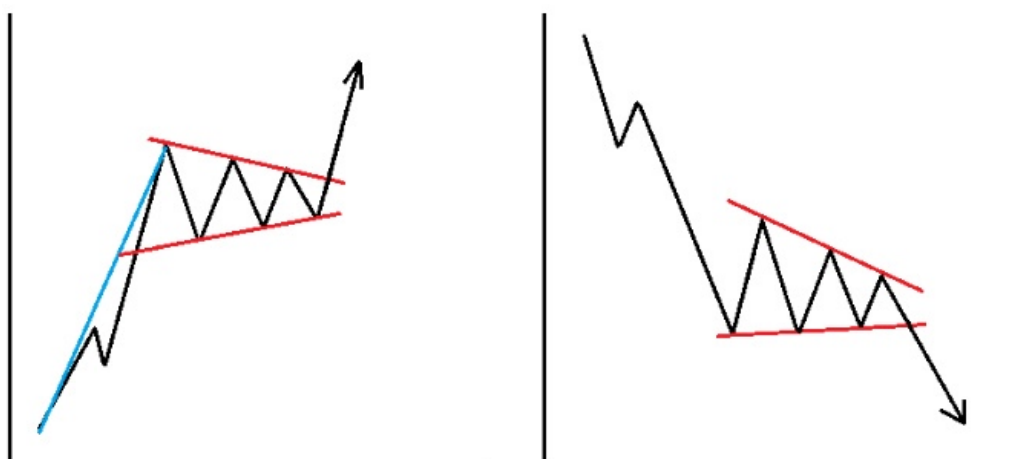


Рисунок 4 – Восходящий и нисходящий вымпелы

Итак, как видно из левой части рисунка, формирование фигуры «восходящий вымпел» будет зависеть от 6-ти подряд идущих значений, причем нечетные значения должны убывать относительно друг друга, а четные значения — возрастать. Также последнее нечетное значение должно быть гарантированно меньше последнего четного. Если до начала формирования восходящего вымпела временной ряд показывал рост, то фигура найдена и предполагаемое значение должно расти.

Для фигуры «нисходящий вымпел» также будем проверять 6 подряд идущих значений, только нечетные значения должны убывать относительно друг друга, а четные значения должны быть равны между собой с учетом некоторого допуска. Последнее четное значение также должно быть меньше последнего нечетного. Если до начала формирования фигуры значения акций убывали, то после окончания формирования, тренд останется таким же.

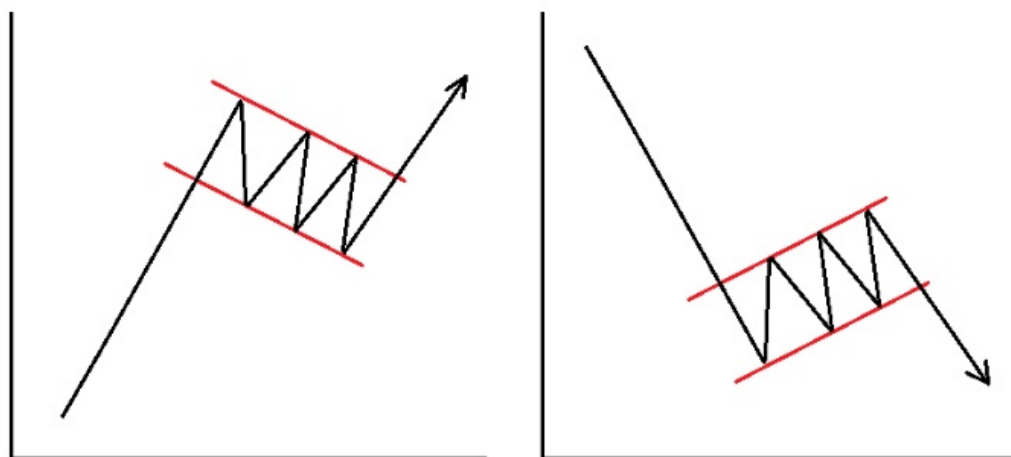


Рисунок 5 – Восходящий и нисходящий флаги

Теперь перейдем к рассмотрению восходящих и нисходящих флагов. Также, как и в случае с вымпелами потребуется сравнить между собой 6 подряд идущих значений временного ряда. Для того, чтобы сделать вывод о формировании флага, требуется, чтобы при соединении четных (нечетных) значений образовывалась прямая, причем полученные две прямые не должны пересекаться. Для этого требуется ввести определенный допуск и посчитать коэффициент прямой по формуле:

$$\frac{y_2 - y_1}{x_2 - x_1},$$

где в качестве значения знаменателя будем брать 1, так как две соседние точки будут иметь разницу ровно один день. А в качестве числителя будем брать разницу между соседними значениями.

Таким образом получим два коэффициента:  $k_1, k_2$ . Теперь вычтем один коэффициент из другого по модулю и сравним с некоторым допуском  $\varepsilon$ . Если  $|k_1 - k_2| < \varepsilon$ , то фигура найдена. Если полученные коэффициенты отрицательны и до этого тренд был возрастающим, то он сохранится. Если же коэффициенты положительны и тренд был нисходящим, то он также сохранится.

### 2.2.2 Прямоугольник

Фигура «прямоугольник» представляет своего рода канал, в пределах которого колеблется цена. Возникает, если котировки не в состоянии продолжить движение по тренду. Если движение по тренду останавливается на какое-то время, но при этом цена не в состоянии развернуться против тренда, то с большей вероятностью актуальная тенденция будет продолжена.

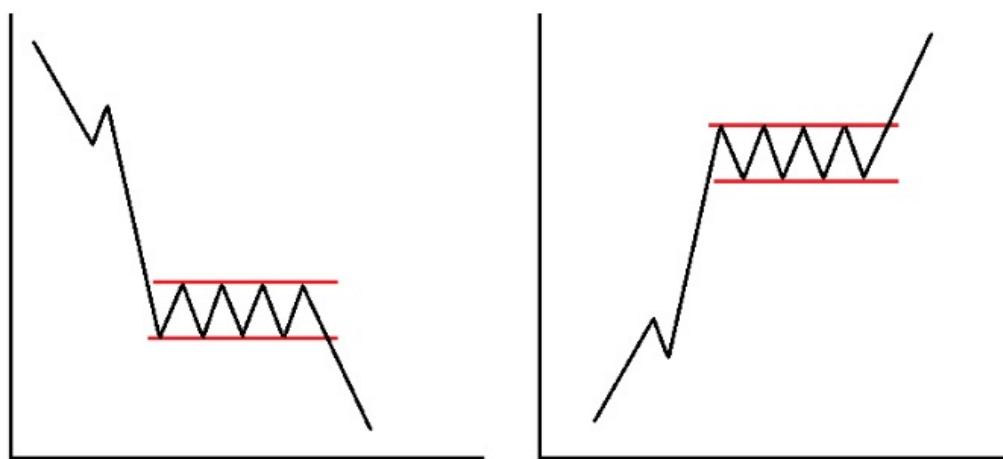


Рисунок 6 – Прямоугольник



Рассмотрим алгоритм нахождения прямоугольников на графике. Будем рассматривать 8 подряд идущих значений и сравнивать их с определенным допуском. Четные и нечетные значения должны быть примерно равны между собой, то есть  $|y_i - y_{i+2}| < \varepsilon$ .

Если до начала формирования фигуры тренд шел на понижение, то после окончания ее формирования он останется. Для возрастающего тренда ситуация аналогичная.

### 2.2.3 Ромб

Начало ромба возникает в конце импульсного движения по тренду. Цена затормаживается, после чего ее колебания постепенно возрастают, но происходят в пределах расходящегося треугольника (левая половина ромба). Затем происходит обратное — волатильность плавно снижается, и мы наблюдаем окончательное формирование ромба.

Как и в случае с другими фигурами, объемы торгов являются отличным вспомогательным инструментом при определении границ и момента завершения ромба. Например, импульсный рост объемов возле границы ромба увеличивает вероятность завершения фигуры.

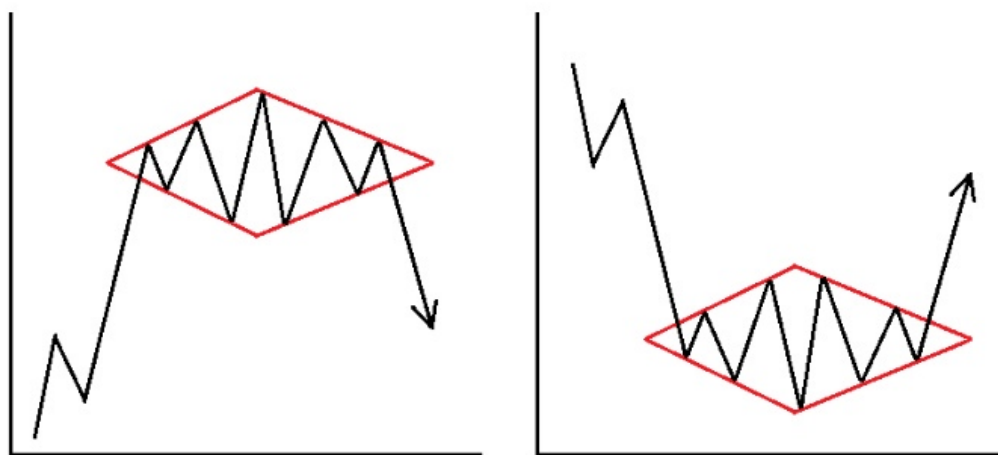


Рисунок 7 – Ромб

Будем рассматривать 9 подряд идущих значений: первые 3 нечетных значения должны возрасть (в случае нахождения средней вершины ромба как точки перегиба в убывающий тренд, если же средняя вершина будет точной перегиба в возрастающий тренд, то будем учитывать первые 2 четных вершины как возрастающие), последние 2 нечетных значения должны убывать. Четные же

вершины наоборот сначала должны убывать по значениям, а затем возрасти. Также стоит учитывать то, что значение средней вершины должно быть локальным максимумом (минимумом) среди всех подряд идущих. Таким образом, будет сделан вывод о формировании фигуры «ромб».

Если до начала формирования фигуры тренд шел на повышение и среднее значение ромба является локальным максимумом, то тренд после окончания формирования фигуры будет идти на понижение. Если же указанное значение является локальным минимумом, а тренд шел на понижение, то он развернется и пойдет на повышение.

#### 2.2.4 Голова и плечи

«Голова и плечи» — одна из самых известных графических моделей. Формация «голова и плечи» представляет собой конфигурацию из трех вершин, причем средняя из них выше двух — предшествующей и последующей. Аналогичным образом, перевернутая «голова и плечи» представляет собой конфигурацию из трех впадин, причем средняя впадина ниже соседних. Данная фигура не считается завершенной, пока не пробита линия «шеи». Более того, подлинная «голова и плечи» формируется только после того, как после пробоя «линии шеи» произошло значительное движение цен. Модели, которые похожи на «голову и плечи», но не удовлетворяют последнему требованию, могут оказаться ложными.



Рисунок 8 – Голова и плечи

Итак, для того, чтобы найти данную фигуру, будем рассматривать 5 подряд идущих значений временного ряда. Первое значение должно быть строго большим второго значения, чтобы получилось «левое плечо», причем тренд

до начала фигуры должен быть возрастающим. Третье значение в случае не перевернутой фигуры должно быть максимальным из рассматриваемых. Аналогично, четвертое значение должно быть строго меньше пятого. Если все данные условия выполнены, то делается вывод о том, что сформировалась очередная фигура «голова и плечи» и, значит, тренд пойдет вниз [1].

Перевернутая голова и плечи строится аналогично: второе значение строго больше первого, четвертое значение строго больше пятого и третье значение является минимальным из рассматриваемых. Тренд до образования фигуры убывающий, а, значит, после окончания формирования, должен пойти вверх.

### 2.3 Поиск экстремальных значений

Воспользуемся одним из методов технического анализа — поиск экстремальных значений котировок и составление нового графика на их основе. Для этого будем сравнивать подряд идущие значения  $x_1, x_2, x_3$ .

Если  $x_2 < x_3$  и  $x_2 < x_1$ , то точка  $x_2$  будет являться экстремальным значением с дальнейшим восходящим трендом, начиная с точки  $x_3$ .

Если же  $x_2 > x_3$  и  $x_2 > x_1$ , то точка  $x_2$  будет являться экстремальным значением с дальнейшим нисходящим трендом, начиная с точки  $x_3$ .

Возьмем, например, акции Netflix за последние 5 лет и проверим работу рассматриваемого алгоритма.

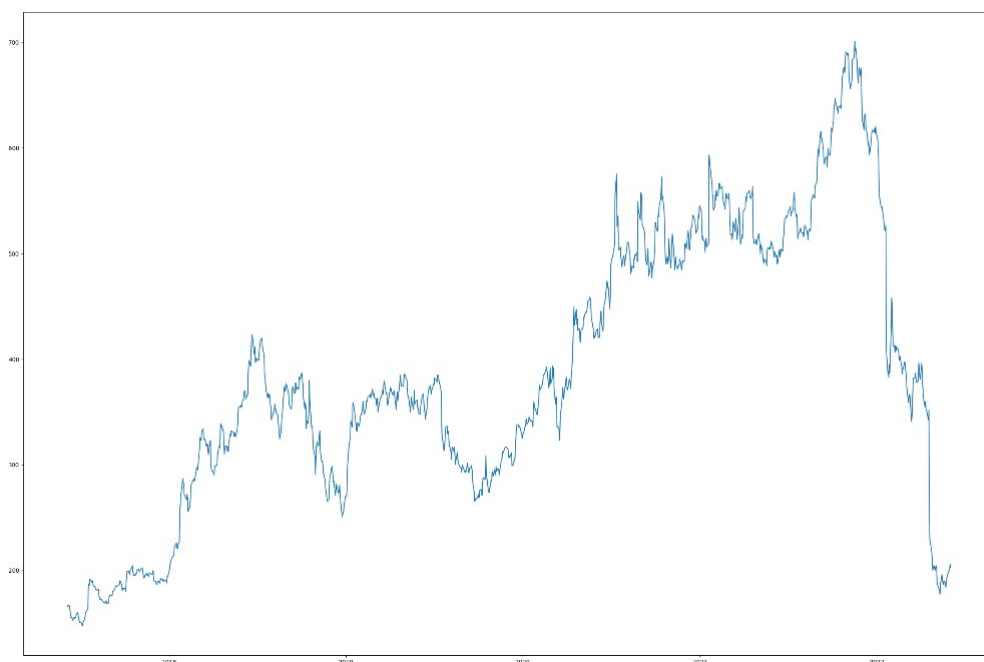


Рисунок 9 – График акций Netflix

Всего было получено 1260 значений цен. Теперь воспользуемся алгоритмом получения экстремальных значений, а также аналогично изобразим временной ряд.

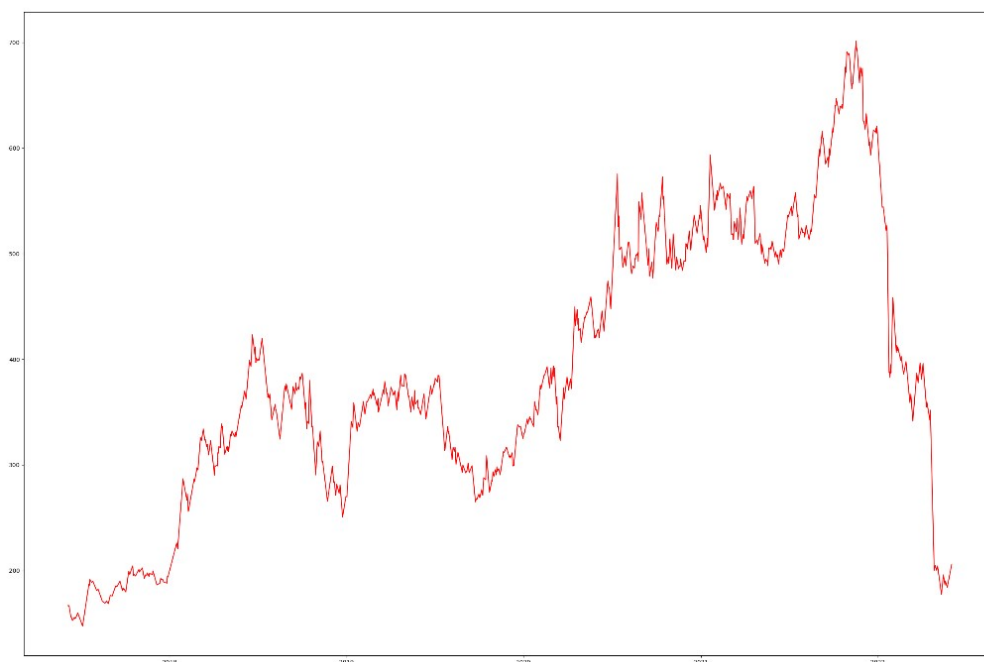


Рисунок 10 – График акций Netflix по экстремумам

После работы алгоритма, было получено 560 экстремальных значений.

## 2.4 Проверка работы алгоритма

Итак, перейдем к проверке работы алгоритма, а также к анализу получаемого результата нахождения технических фигур. Для этого сделаем анализ котировок четырех акций с высокой волатильностью, четырех с низкой волатильностью, а также возьмем в рассмотрение две котировки акций криптовалют. Затем сделаем вывод о том, какие из рассматриваемых акций поддаются техническому анализу, а также получим общий вывод по работоспособности алгоритма. Все акции будем рассматривать за последние пять лет, следовательно цены на акции будем брать с промежутком в один день.

На графике будем изображать работу алгоритма поиска экстремумов, а также будем наносить поверх графика экстремумов найденные фигуры с предполагаемым дальнейшим ростом или падением курса.

Для начала рассмотрим акции с высокой волатильностью.

## 2.4.1 Energy Focus

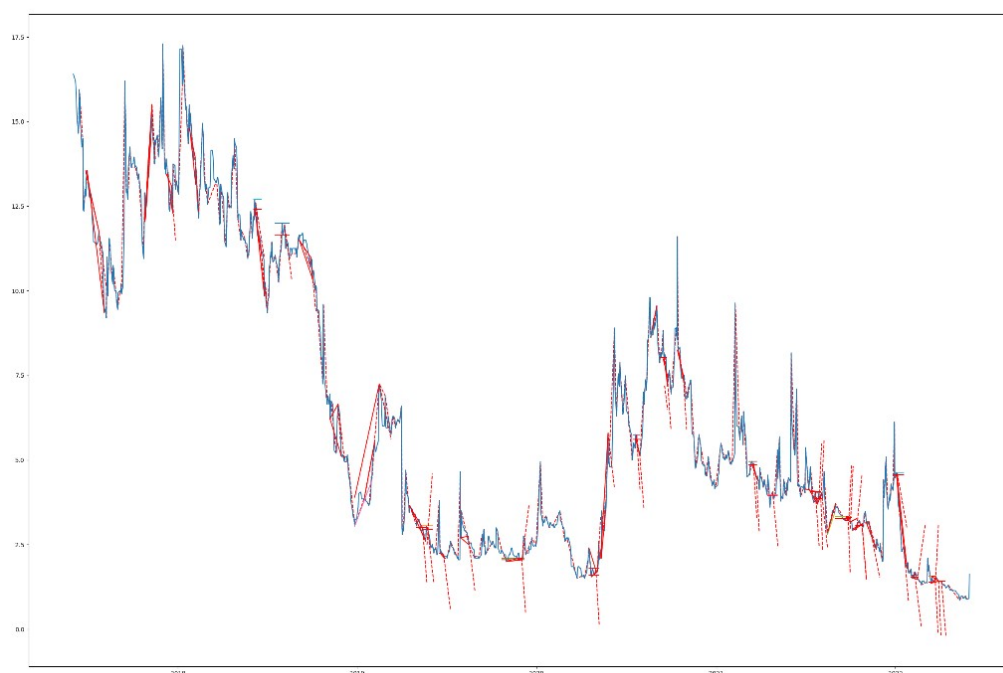


Рисунок 11 – График акций и технические фигуры Energy Focus

Фигура	Количество найденных	Совпадение	Доля
Вымпел	2	2	0.023
Нисходящий флаг	2	2	0.023
Восходящий флаг	4	3	0.047
Прямоугольник	3	3	0.035
Нисходящий ромб	1	1	0.012
Восходящий ромб	0	0	0
Голова и плечи	2	2	0.023
Перевернутые голова и плечи	3	3	0.035

Исходя из полученных данных, можно сделать вывод, что для акций Energy Focus не было найдено достаточное количество технических фигур для проведения анализа, но для большинства найденных фигур предполагаемый курс акций совпадал с действительным.

## 2.4.2 Yandex

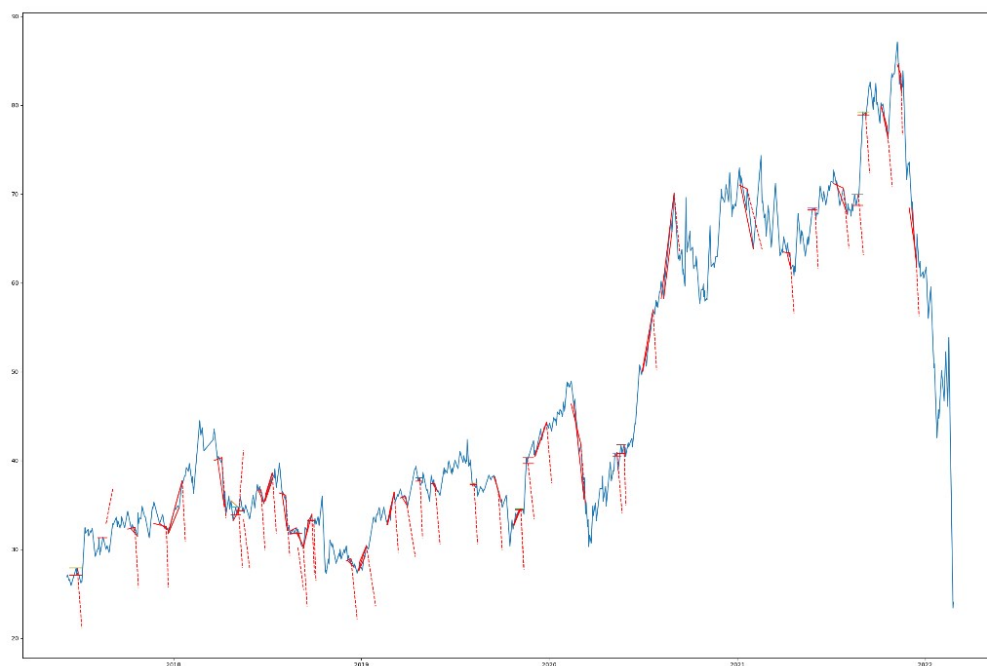


Рисунок 12 – График акций и технические фигуры Yandex

Фигура	Количество найденных	Совпадение	Доля
Вымпел	1	1	0.013
Нисходящий флаг	0	0	0
Восходящий флаг	6	6	0.078
Прямоугольник	0	0	0
Нисходящий ромб	0	0	0
Восходящий ромб	0	0	0
Голова и плечи	3	3	0.039
Перевернутые голова и плечи	1	1	0.013

В данный момент Yandex относится к компаниям, имеющим акции с высокой волатильностью. Исходя из полученных данных, можно сделать вывод что данные котировки можно отнести к неподдающимся техническому анализу, так как фигур «прямоугольник» и «ромб» вообще не было найдено, а остальные фигуры были найдены в недостаточном количестве.

### 2.4.3 Rectruiter.com Group

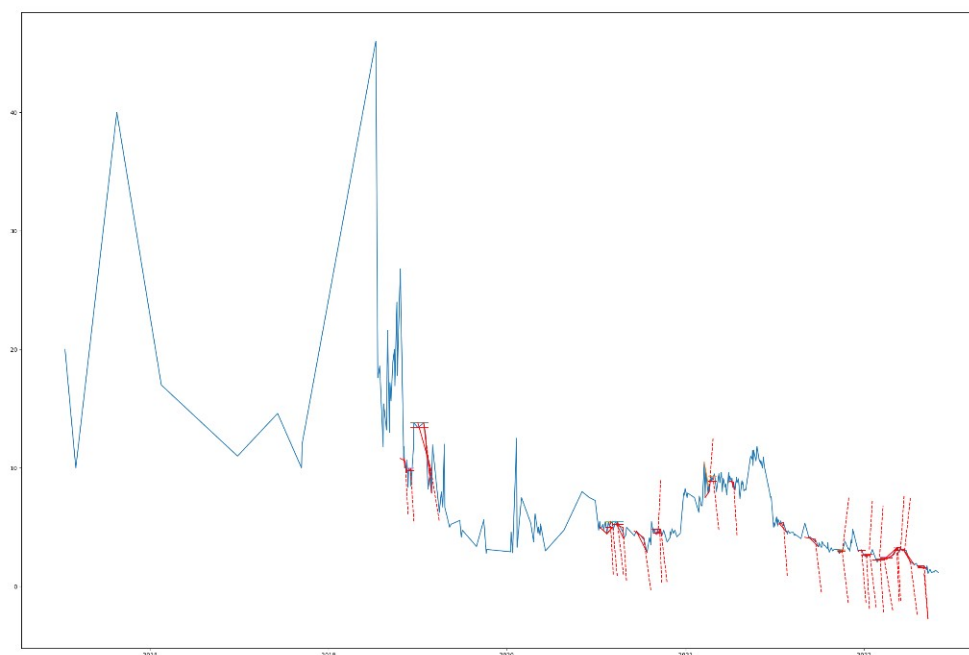


Рисунок 13 – График акций и технические фигуры Recruiter.com

Фигура	Количество найденных	Совпадение	Доля
Вымпел	2	1	0.038
Нисходящий флаг	1	0	0.019
Восходящий флаг	2	2	0.038
Прямоугольник	2	2	0.038
Нисходящий ромб	1	0	0.019
Восходящий ромб	0	0	0
Голова и плечи	1	1	0.019
Перевернутые голова и плечи	4	4	0.075

В данный момент, особенно в связи с пандемией, онлайн–рекрутмент стал одним из самых популярных направлений бизнеса, поэтому рассмотрим в качестве представителя акций высокой волатильности американскую компанию Recruiter.

Для данной акции, начиная с середины 2021 года, технический анализ представляется возможным, так как наибольшее число фигур было найдено в это время. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

#### 2.4.4 Greenbox POS

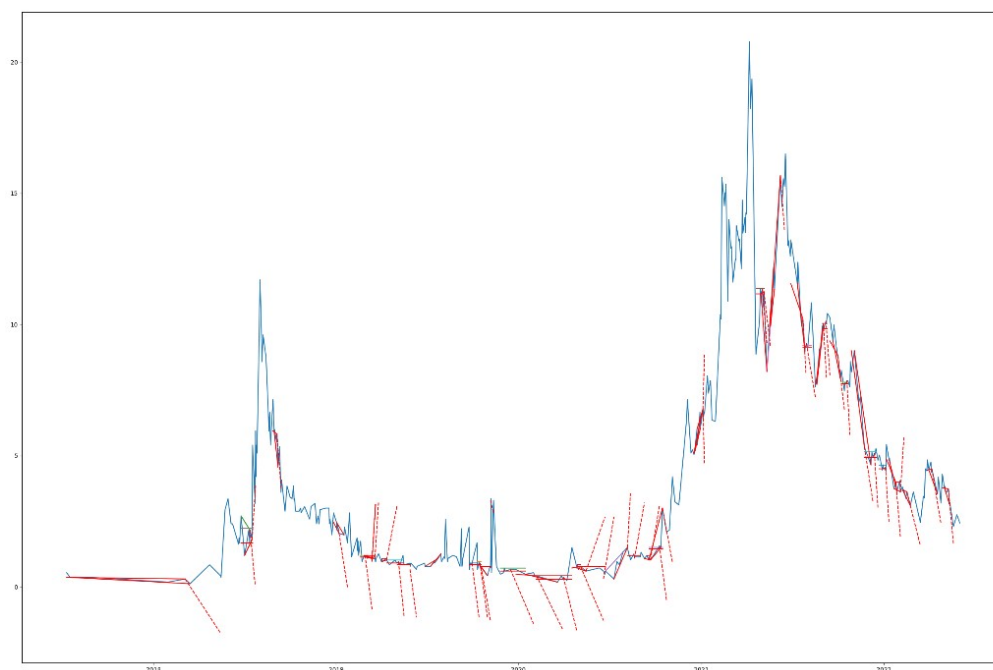


Рисунок 14 – График акций и технические фигуры Greenbox POS

Фигура	Количество найденных	Совпадение	Доля
Вымпел	1	1	0.014
Нисходящий флаг	1	1	0.014
Восходящий флаг	4	2	0.058
Прямоугольник	1	0	0.014
Нисходящий ромб	0	0	0
Восходящий ромб	0	0	0
Голова и плечи	1	1	0.014
Перевернутые голова и плечи	4	1	0.058

Ввиду развития технологий электронных платежей с использованием блокчейна, рассмотрим котировки компании Greenbox POS — одной из самых волатильных акций в данной сфере.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

Теперь перейдем к анализу акций с низкой волатильностью.



## 2.4.5 Campbell Soup Company

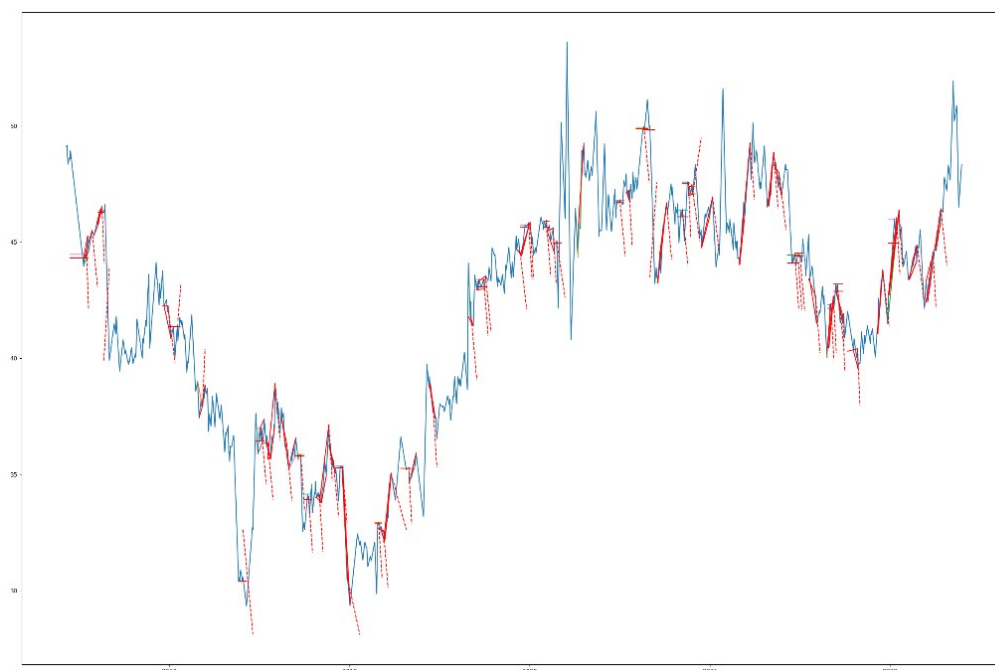


Рисунок 15 – График акций и технические фигуры Campbell Soup Company

Фигура	Количество найденных	Совпадение	Доля
Вымпел	1	1	0.01
Нисходящий флаг	0	0	0
Восходящий флаг	14	14	0.136
Прямоугольник	1	1	0.01
Нисходящий ромб	0	0	0
Восходящий ромб	0	0	0
Голова и плечи	4	4	0.039
Перевернутые голова и плечи	1	1	0.01

Campbell Soup Company (также известная как Campbell's) — американская компания, являющаяся крупнейшим в мире производителем консервированных супов.

Стоит сказать, что для данной акции всего было найдено 103 технические фигуры за рассматриваемый период, что свидетельствует о возможности технического анализа. В сводной таблице же приведены рассматриваемые в данной работе только те технические фигуры, смысл рассматривать которые есть, так как другие фигуры технического анализа не являются эталонными для рассмотрения.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

#### 2.4.6 Johnson and Johnson

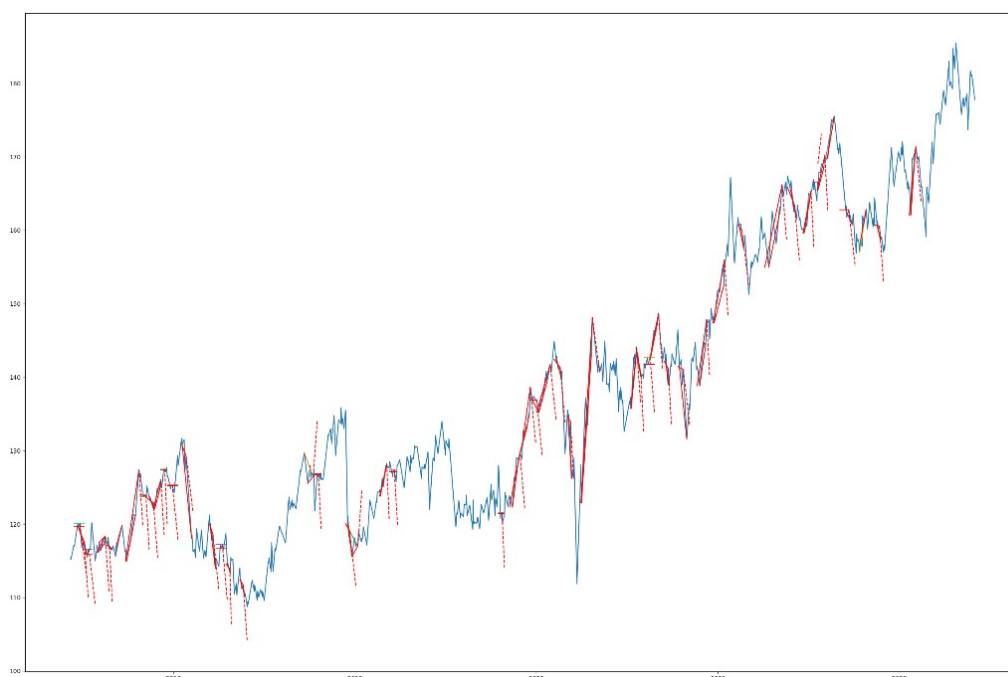


Рисунок 16 – График акций и технические фигуры Johnson and Johnson

Фигура	Количество найденных	Совпадение	Доля
Вымпел	2	2	0.025
Нисходящий флаг	0	0	0
Восходящий флаг	6	6	0.076
Прямоугольник	0	0	0
Нисходящий ромб	0	0	0
Восходящий ромб	0	0	0
Голова и плечи	0	0	0
Перевернутые голова и плечи	1	1	0.013

Johnson and Johnson — американская холдинговая компания, возглавляющая группу из более чем 250 дочерних компаний по всему миру, производящих лекарственные препараты, санитарно-гигиенические товары и медицинское оборудование.

Исходя из полученных данных, можно сделать вывод, что акции не относятся к техничным. Из рассматриваемых фигур не были найдены: «Нисходящий флаг», «Прямоугольник», «Ромб», «Голова и плечи».

Котировка не считается технически анализируемой.

#### 2.4.7 Altria Group

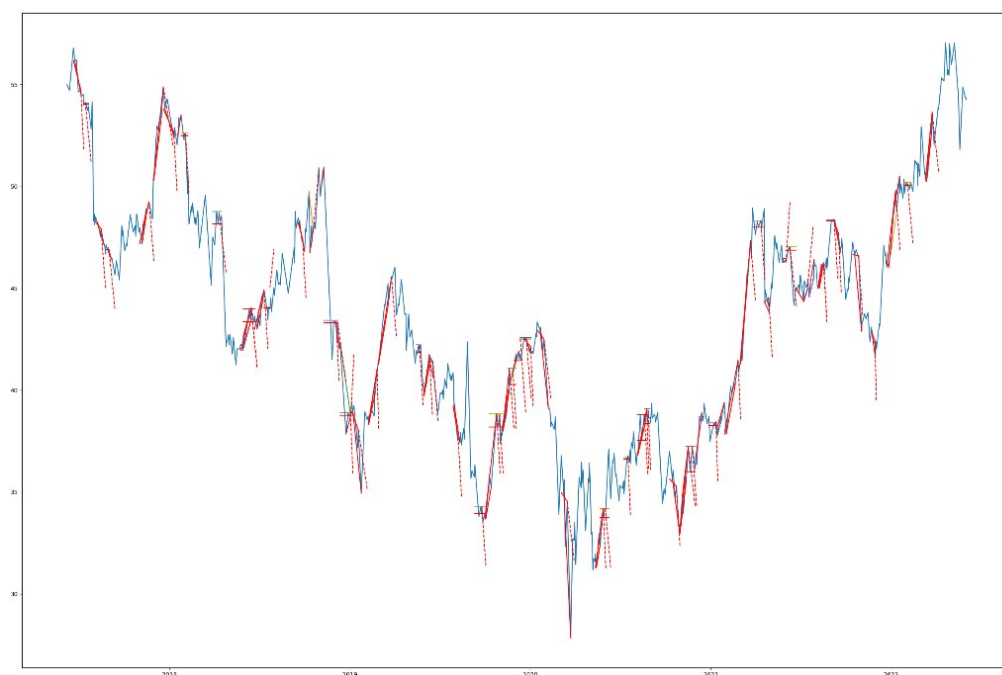


Рисунок 17 – График акций и технические фигуры Altria Group

Фигура	Количество найденных	Совпадение	Доля
Вымпел	2	2	0.018
Нисходящий флаг	0	0	0
Восходящий флаг	17	17	0.153
Прямоугольник	0	0	0
Нисходящий ромб	1	1	0.009
Восходящий ромб	0	0	0
Голова и плечи	1	1	0.009
Перевернутые голова и плечи	1	1	0.009

Altria Group — один из лидеров мирового рынка табачных изделий, который является материнской компанией таких производителей сигарет, как Philip Morris USA, John Middleton и U.S. Smokeless Tobacco Company.

Для данной акции всего было найдено 111 технических фигур за рассматриваемый период, что свидетельствует о возможности технического анализа.

Информация о фигурах, представляющих интерес представлена в таблице.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

#### 2.4.8 Sprouts Farmers Market

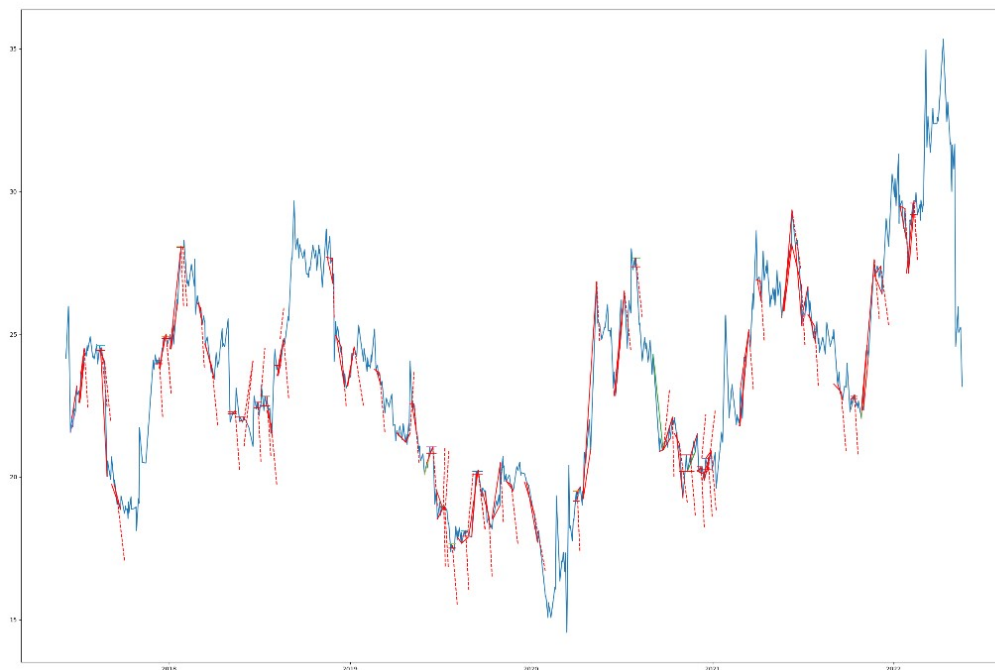


Рисунок 18 – График акций и технические фигуры Sprouts Farmers Market

Фигура	Количество найденных	Совпадение	Доля
Вымпел	2	2	0.017
Нисходящий флаг	1	1	0.009
Восходящий флаг	15	15	0.128
Прямоугольник	0	0	0
Нисходящий ромб	0	0	0
Восходящий ромб	3	3	0.026
Голова и плечи	0	0	0
Перевернутые голова и плечи	4	4	0.034

Sprouts Farmers Market — сеть супермаркетов, специализирующаяся на свежих, натуральных и органических продуктах.

Для данной акции всего было найдено 117 технических фигур за рассматриваемый период, что свидетельствует о возможности технического анализа. Информация о фигурах, представляющих интерес представлена в таблице.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

Теперь перейдем к анализу котировок криптовалют. Для начала возьмем в качестве анализируемой акции одну из самых популярных валют — Bitcoin. Затем наоборот возьмем для анализа не самую популярную валюту — EOS.

#### 2.4.9 Bitcoin

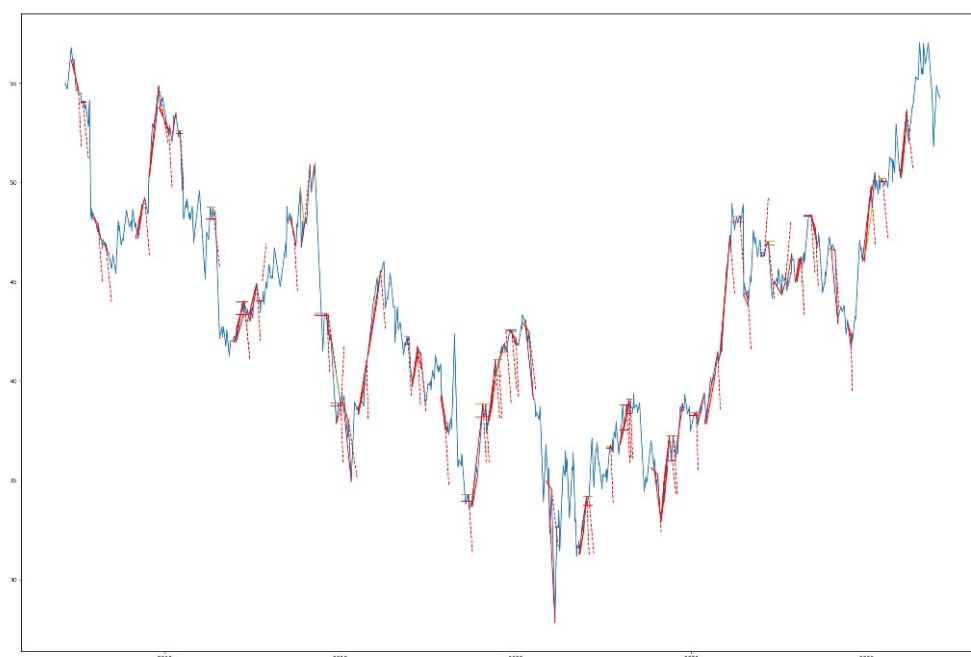


Рисунок 19 – График акций и технические фигуры Bitcoin

Фигура	Количество найденных	Совпадение	Доля
Вымпел	1	1	0.013
Нисходящий флаг	1	1	0.013
Восходящий флаг	2	2	0.026
Прямоугольник	3	3	0.039
Нисходящий ромб	1	1	0.013
Восходящий ромб	1	1	0.013
Голова и плечи	5	5	0.065
Перевернутые голова и плечи	5	5	0.065

Bitcoin является самой популярной криптовалютой в мире.

Для данной акции всего было найдено 77 технических фигур за рассматриваемый период, что свидетельствует о возможности технического анализа. Информация о фигурах, представляющих интерес представлена в таблице.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

#### 2.4.10 EOS

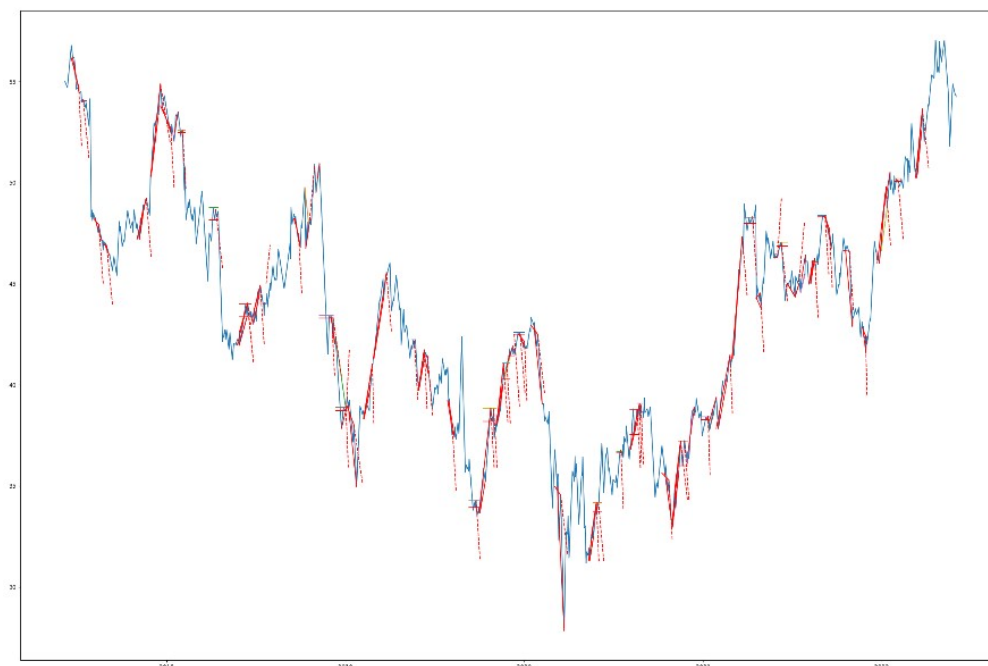


Рисунок 20 – График акций и технические фигуры EOS

Фигура	Количество найденных	Совпадение	Доля
Вымпел	1	1	0.004
Нисходящий флаг	1	1	0.004
Восходящий флаг	24	23	0.09
Прямоугольник	8	8	0.03
Нисходящий ромб	3	3	0.011
Восходящий ромб	1	1	0.004
Голова и плечи	13	13	0.049
Перевернутые голова и плечи	11	11	0.041

Для данной акции всего было найдено 266 технических фигур за рассматриваемый период и является максимальным количеством из рассматриваемых.

Для данной акции, за весь рассматриваемый период технический анализ представляется возможным. В большинстве случаев предполагаемый курс совпал с действительным.

Котировка считается технически анализируемой.

## ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены детерминированные методы поиска технических фигур на курсах акций. Проведен анализ работы алгоритма на акциях с высокой волатильностью и низкой, а также на акциях криптовалют. Алгоритм в некоторых случаях может работать некорректно в следствие неправильной подобранных значений допуска  $\varepsilon$  в фигурах, где это значение используется. Также были рассмотрены теоретические основы нейросетей, описаны основные принципы работы и их роль в решении, основы технического анализа и основные определения трейдинга.

В качестве практической части работы была описана программная реализация алгоритма поиска технических фигур на акциях, получение статистики по фигурам, а также отрисовка данных фигур на графике акции с предполагаемым ростом или падением курса.



## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

- 1 Создание приложения Windows Forms с помощью .NET Framework (C++) [Электронный ресурс]. — URL: [http://msdn.microsoft.com/ru-ru/library/vstudio/ms235634\(v=vs.100\).aspx](http://msdn.microsoft.com/ru-ru/library/vstudio/ms235634(v=vs.100).aspx) (Дата обращения 12.07.2013). Загл. с экр. Яз. рус.

## ПРИЛОЖЕНИЕ А

### Код drawing.py

```
def draw_pennants_rising_flags(plt, indexes, prices, date, delta):
    for index in indexes:
        x_upper_line = [date[index], date[index + 4]]
        y_upper_line = [prices[index], prices[index + 4]]
        plt.plot(x_upper_line, y_upper_line)
        x_lower_line = [date[index + 1], date[index + 5]]
        y_lower_line = [prices[index + 1], prices[index + 5]]
        plt.plot(x_lower_line, y_lower_line, color='red')
        if prices[index - 1] < prices[index]:
            x_rising_line = [date[index + 5], date[index + 8]]
            y_rising_line = [prices[index + 5], prices[index + 5] + delta]
            plt.plot(x_rising_line, y_rising_line, color='red')

def draw_descending_flags(plt, indexes, prices, date, delta):
    for index in indexes:
        x_upper_line = [date[index], date[index + 4]]
        y_upper_line = [prices[index], prices[index + 4]]
        plt.plot(x_upper_line, y_upper_line)
        x_lower_line = [date[index + 1], date[index + 5]]
        y_lower_line = [prices[index + 1], prices[index + 5]]
        plt.plot(x_lower_line, y_lower_line, color='red')
        if prices[index - 1] > prices[index]:
            x_rising_line = [date[index + 5], date[index + 8]]
            y_rising_line = [prices[index + 5], prices[index + 5] + delta]
            plt.plot(x_rising_line, y_rising_line, color='red')

def draw_rectangles(plt, indexes, prices, date, delta):
    for index in indexes:
        x_upper_line = [date[index + 1], date[index + 7]]
        y_upper_line = [prices[index + 1], prices[index + 7]]
        plt.plot(x_upper_line, y_upper_line, color='red')
        x_lower_line = [date[index], date[index + 6]]
        y_lower_line = [prices[index], prices[index + 6]]
        plt.plot(x_lower_line, y_lower_line, color='red')
        x_line = [date[index + 7], date[index + 10]]
        y_line_2nd_arg = prices[index + 7] - delta if \
            prices[index] < prices[index - 1] else \
                prices[index + 7] + delta
        y_line = [prices[index + 7], y_line_2nd_arg]
        plt.plot(x_line, y_line, color='red')

def draw_rhombuses(plt, indexes, prices, date, delta):
    for index in indexes:
```

```

y_middle_value = (prices[index + 4] + prices[index + 3]) / 2

x_1st_line = [date[index - 1], date[index + 4]]
y_1st_line = [y_middle_value, prices[index + 4]]
plt.plot(x_1st_line, y_1st_line, color='red')

x_2nd_line = [date[index + 4], date[index + 9]]
y_2nd_line = [prices[index + 4], y_middle_value]
plt.plot(x_2nd_line, y_2nd_line, color='red')

x_3rd_line = [date[index + 9], date[index + 4]]
y_3rd_line = [y_middle_value, prices[index + 4]]
plt.plot(x_3rd_line, y_3rd_line, color='red')

x_4th_line = [date[index + 4], date[index - 1]]
y_4th_line = [prices[index + 4], y_middle_value]
plt.plot(x_4th_line, y_4th_line, color='red')

y_value_line = prices[index + 8] + delta if \
    prices[index] < prices[index - 1] else \
    prices[index + 8] - delta
y_values_line = [prices[index + 8], y_value_line]
x_values_line = [date[index + 9], date[index + 13]]
plt.plot(x_values_line, y_values_line, color='red')

def draw_double_tops(plt, indexes, prices, date, delta):
    for index in indexes:
        x_upper_line = [date[index - 2], date[index + 4]]
        y_upper_line = [prices[index], prices[index]]
        plt.plot(x_upper_line, y_upper_line)
        x_lower_line = [date[index - 2], date[index + 4]]
        y_lower_line = [prices[index + 1], prices[index + 1]]
        plt.plot(x_lower_line, y_lower_line, color='red')
        if prices[index] > prices[index - 1]:
            y_line = [prices[index + 2], prices[index + 2] - delta]
            x_line = [date[index + 2], date[index + 5]]
            plt.plot(x_line, y_line)

def draw_triple_bottoms(plt, indexes, prices, date, delta):
    for index in indexes:
        x_upper_line = [date[index + 1], date[index + 3]]
        y_upper_line = [prices[index + 1], prices[index + 1]]
        plt.plot(x_upper_line, y_upper_line)
        x_lower_line = [date[index], date[index + 4]]
        y_lower_line = [prices[index], prices[index + 4]]
        plt.plot(x_lower_line, y_lower_line, color='red')

```

```

    if prices[index] < prices[index - 1]:
        x_line = [date[index + 4], date[index + 7]]
        y_line = [prices[index + 4], prices[index + 4] + delta]
        plt.plot(x_line, y_line)

def draw_head_and_shoulders(plt, indexes, prices, date, delta):
    for index in indexes:
        x_line = [date[index - 1], date[index + 5]]
        y_line = [prices[index + 1], prices[index + 1]]
        plt.plot(x_line, y_line, color='red')
        x_line = [date[index + 4], date[index + 7]]
        y_value = prices[index + 4] - delta if \
            prices[index] > prices[index - 1] else prices[index + 4] + delta
        y_line = [prices[index + 7], y_value]
        plt.plot(x_line, y_line)

def draw_wedges(plt, indexes, prices, date, flag, delta_y):
    # Флаг нужен для отрисовки нисходящих / восходящих
    delta = 4 if flag else 6
    for index in indexes:
        x_upper_line = [date[index], date[index + delta]]
        y_upper_line = [prices[index + 1], prices[index + delta]]
        plt.plot(x_upper_line, y_upper_line)
        x_lower_line = [date[index + 1], date[index + 5]]
        y_lower_line = [prices[index + 1], prices[index + 5]]
        plt.plot(x_lower_line, y_lower_line, color='red')
        x_line = [date[index + delta], date[index + delta + 3]]
        y_line_1st = prices[index + delta] + delta_y if delta == 5 else \
            prices[index + delta] - delta_y
        y_line_2nd = prices[index + delta]
        y_line = [y_line_2nd, y_line_1st]
        plt.plot(x_line, y_line)

```

## ПРИЛОЖЕНИЕ Б

### Код figures.py

```
def make_bolting(indexes, delta):
    res = []
    if indexes:
        res.append(indexes[0])
    if len(indexes) > 1:
        for i in range(1, len(indexes) - 1):
            if indexes[i] - indexes[i + 1] <= delta and \
                indexes[i] - res[-1] <= delta:
                continue
            else:
                res.append(indexes[i])
    return res

def get_pennants(prices):
    res = []
    col = 0
    for i in range(0, len(prices) - 6):
        if prices[i] > prices[i + 2] > prices[i + 4] > prices[i + 5] > \
            prices[i + 3] > prices[i + 1]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 6)

def get_descending_flags(prices):
    res = []
    col = 0
    eps = 1.25
    for i in range(0, len(prices) - 6):
        if prices[i] > prices[i + 2] > prices[i + 4] > prices[i + 5] > \
            prices[i + 3] > prices[i + 1] and \
            abs(prices[i] - prices[i + 4] - prices[i + 1] \
                + prices[i + 5]) < eps:
            res.append(i)
            col += 1
    return col, make_bolting(res, 6)

def get_rising_flags(prices):
    res = []
    col = 0
    eps = 0.5
    for i in range(0, len(prices) - 6):
        if prices[i] < prices[i + 2] < prices[i + 4] < prices[i + 5] and \
            prices[i + 1] < prices[i + 3] < prices[i + 5] \
```

```

        and abs(prices[i] - prices[i + 4] - prices[i + 1] + \
                prices[i + 5]) < eps:
            res.append(i)
            col += 1
    return col, make_bolting(res, 6)

def get_rectangles(prices):
    res = []
    col = 0
    eps = 0.15
    for i in range(len(prices) - 8):
        if abs(prices[i] - prices[i + 2]) < eps and \
            abs(prices[i + 2] - prices[i + 4]) < eps and \
            abs(prices[i + 4] - prices[i + 6]) < eps and \
            abs(prices[i + 1] - prices[i + 3]) < eps and \
            abs(prices[i + 3] - prices[i + 5]) < eps and \
            abs(prices[i + 5] - prices[i + 7]) < eps and \
            prices[i + 7] > prices[i + 6]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 8)

def get_descending_rhombuses(prices):
    res = []
    col = 0
    eps = 0.40
    for i in range(len(prices) - 9):
        if prices[i + 2] < prices[i] < prices[i + 1] and \
            prices[i + 6] < prices[i + 8] < prices[i + 7] and \
            abs(prices[i + 3] - prices[i + 5]) < eps and \
            abs(prices[i + 2] - prices[i + 6]) < eps and \
            prices[i + 4] < prices[i + 2] and \
            abs(prices[i + 1] - prices[i + 7]) < eps and \
            abs(prices[i] - prices[i + 8]) < eps and \
            prices[i + 3] > prices[i + 8]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 9)

def get_rising_rhombuses(prices):
    res = []
    col = 0
    eps = 0.15
    for i in range(len(prices) - 9):
        if prices[i + 2] > prices[i] > prices[i + 1] and \

```

```

        prices[i + 6] > prices[i + 8] > prices[i + 7] and \
            abs(prices[i + 3] - prices[i + 5]) < eps and \
            abs(prices[i + 2] - prices[i + 6]) < eps and \
            prices[i + 4] > prices[i + 2] and \
            abs(prices[i + 1] - prices[i + 7]) < eps and \
            abs(prices[i] - prices[i + 8]) < eps and \
            prices[i + 3] < prices[i + 8]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 9)

def get_double_tops(prices):
    res = []
    col = 0
    eps = 0.1
    for i in range(1, len(prices) - 3):
        if prices[i - 1] < prices[i + 1] < prices[i] and \
            abs(prices[i] - prices[i + 2]) < eps:
            res.append(i)
            col += 1
    return col, make_bolting(res, 3)

def get_triple_bottoms(prices):
    res = []
    col = 0
    eps = 0.1
    for i in range(1, len(prices) - 5):
        if abs(prices[i] - prices[i + 2]) < eps and \
            abs(prices[i + 2] - prices[i + 4]) < eps and \
            abs(prices[i + 1] - prices[i + 3]) < eps and \
            prices[i] < prices[i + 1] and prices[i] < prices[i - 1]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 5)

def get_reversed_head_and_shoulders(prices):
    res = []
    col = 0
    eps = 0.1
    for i in range(1, len(prices) - 5):
        if prices[i] < prices[i - 1] and \
            abs(prices[i + 1] - prices[i + 3]) < eps and \
            abs(prices[i] - prices[i + 4]) < eps and \
            prices[i + 1] > prices[i] > prices[i + 2]:
            res.append(i)

```

```

        col += 1
    return col, make_bolting(res, 5)

def get_head_and_shoulders(prices):
    res = []
    col = 0
    eps = 0.1
    for i in range(1, len(prices) - 5):
        if prices[i] > prices[i - 1] and \
            abs(prices[i + 1] - prices[i + 3]) < eps and \
            abs(prices[i] - prices[i + 4]) < eps and \
            prices[i + 1] < prices[i] < prices[i + 2]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 5)

def get_falling_wedges(prices):
    res = []
    col = 0
    for i in range(len(prices) - 6):
        if prices[i] > prices[i + 2] > prices[i + 4] > prices[i + 5] and \
            prices[i + 1] > prices[i + 3] > prices[i + 5]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 6)

def get_rising_wedges(prices):
    res = []
    col = 0
    for i in range(len(prices) - 7):
        if prices[i] < prices[i + 2] < prices[i + 4] < prices[i + 6] and \
            prices[i + 1] < prices[i + 3] < prices[i + 5] < prices[i + 6]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 7)

def get_rising_triangles(prices):
    res = []
    col = 0
    eps = 0.25
    for i in range(len(prices) - 7):
        if abs(prices[i + 1] - prices[i + 3]) < eps and \
            abs(prices[i + 3] - prices[i + 5]) < eps and \
            prices[i] < prices[i + 2] < prices[i + 4] < \

```



```

        prices[i + 6] < prices[i + 5]:
    res.append(i)
    col += 1
    return col, make_bolting(res, 7)

def get_falling_triangles(prices):
    res = []
    col = 0
    eps = 0.25
    for i in range(len(prices) - 8):
        if abs(prices[i] - prices[i + 2]) < eps and \
            abs(prices[i + 2] - prices[i + 4]) < eps and \
            abs(prices[i + 4] - prices[i + 6]) < eps and \
            prices[i + 6] < prices[i + 7] < prices[i + 5] < \
            prices[i + 3] < prices[i + 1]:
            res.append(i)
            col += 1
    return col, make_bolting(res, 8)

```

## ПРИЛОЖЕНИЕ В

### Код main.py

```
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import figures
import drawing as d
plt.figure(figsize=(30, 20))

def print_statistics(figure_name, indexes, col, prices, flag, c):
    print(f'Статистика по фигуре {figure_name}:')
    print(f'Всего найдено: {col}, что составляет {round(col / c, 3)} \
от общего числа фигур')
    res = []
    # pennants
    if flag == 1:
        for index in indexes:
            res.append([1 if prices[index] > prices[index - 1] and \
prices[index + value] > prices[index + 5] else 0
for value in range(6, 11)])
    #flags
    if flag == 2:
        for index in indexes:
            res.append([1 if prices[index] < prices[index - 1] and \
prices[index + value] > prices[index + 5] or
prices[index] > prices[index - 1] and \
prices[index + 6] < prices[index + value] else 0
for value in range(6, 11)])
    #rectangles
    if flag == 3:
        for index in indexes:
            res.append([1 if prices[index] > prices[index - 1] and \
prices[index + 7] < prices[index + value] or
prices[index] < prices[index - 1] and \
prices[index + 7] > prices[index + value] else 0
for value in range(8, 13)])
    #rhombuses
    if flag == 4:
        for index in indexes:
            res.append([1 if prices[index] > prices[index - 1] and \
prices[index + 8] > prices[index + value] or
prices[index] < prices[index - 1] and \
prices[index + 8] < prices[index + value] else 0
for value in range(9, 14)])
    #head_and_shoulders
    else:
        for index in indexes:
```

```

        res.append([1 if prices[index] > prices[index - 1] and \
                    prices[index + 4] < prices[index + value] or
                    prices[index] < prices[index - 1] and \
                    prices[index + 4] > prices[index + value] \
                    else 0
                    for value in range(5, 10)])

    for subres in res:
        print(subres)

def get_extrema(date, prices):
    indexes = []
    for i in range(1, len(prices) - 1):
        if prices[i] > prices[i - 1] and prices[i] > prices[i + 1] or \
            prices[i] < prices[i - 1] and prices[i] < prices[i + 1]:
            indexes.append(i)

    return [date[index] for index in indexes], \
           [prices[index] for index in indexes]

data = yf.Ticker("AAPL").history(period='max')
prices = np.array(data['High'])
data.reset_index(inplace=True)
date = np.array(data['Date'], dtype='datetime64[D]')

delta_y = (max(prices) - min(prices)) / 10
date_extrema, prices_extrema = get_extrema(date, prices)
# plt.plot(date_extrema, prices_extrema, color='red')
plt.plot(date, prices)
count_pennants, res_pennants = figures.get_pennants(prices_extrema)
d.draw_pennants_rising_flags(plt, res_pennants, prices_extrema, \
    date_extrema, delta_y)

count_descending_flags, res_descending_flags = figures.get_descending_flags(\
    prices_extrema)
d.draw_descending_flags(plt, res_descending_flags, prices_extrema, \
    date_extrema, delta_y)

count_rising_flags, res_rising_flags = figures.get_rising_flags(prices_extrema)
d.draw_pennants_rising_flags(plt, res_rising_flags, prices_extrema, \
    date_extrema, delta_y)

count_rectangles, res_rectangles = figures.get_rectangles(prices_extrema)
d.draw_rectangles(plt, res_rectangles, prices_extrema, date_extrema, delta_y)

count_descending_rhombuses, res_descending_rhombuses = figures.get_descending_rhombuses(\

```

```

    prices_extrema)
d.draw_rhombuses(plt, res_descending_rhombuses, prices_extrema, \
    date_extrema, delta_y)

count_rising_rhombuses, res_rising_rhombuses = figures.get_rising_rhombuses(\
    prices_extrema)
d.draw_rhombuses(plt, res_rising_rhombuses, prices_extrema, \
    date_extrema, delta_y)

count_double_tops, res_double_tops = figures.get_double_tops(prices_extrema)
d.draw_double_tops(plt, res_double_tops, prices_extrema, date_extrema, delta_y)

count_triple_bottoms, res_triple_bottoms = figures.get_triple_bottoms(prices_extrema)
d.draw_triple_bottoms(plt, res_triple_bottoms, prices_extrema, \
    date_extrema, delta_y)

count_head_and_shoulders, res_head_and_shoulders = figures.get_head_and_shoulders(\
    prices_extrema)
d.draw_head_and_shoulders(plt, res_head_and_shoulders, prices_extrema, \
    date_extrema, delta_y)

count_reversed_head_and_shoulders, res_reversed_head_and_shoulders = \
    figures.get_reversed_head_and_shoulders(prices_extrema)
d.draw_head_and_shoulders(plt, res_reversed_head_and_shoulders, \
    prices_extrema, date_extrema, delta_y)

count_falling_wedges, res_falling_wedges = figures.get_falling_wedges(prices_extrema)
d.draw_wedges(plt, res_falling_wedges, prices_extrema, \
    date_extrema, True, delta_y)

count_rising_wedges, res_rising_wedges = figures.get_rising_wedges(prices_extrema)
d.draw_wedges(plt, res_rising_wedges, prices_extrema, \
    date_extrema, False, delta_y)

plt.show()
c = count_pennants + count_descending_flags + count_rectangles + \
    count_descending_rhombuses + count_rising_rhombuses \
    + count_double_tops + count_head_and_shoulders + \
    count_reversed_head_and_shoulders + count_falling_wedges \
    + count_rising_wedges

print(f'Общее количество найденных фигур: {c}')

# print_statistics('вымпел', res_pennants, count_pennants, prices_extrema, 1, c)
# pennants, \
# descending_flags, \
# rising_flags, \
# rectangles, \

```

```
# descending_rhombuses, \
# rising_rhombuses, \
# head_and_shoulders, \
# reversed_head_and_shoulders, \
```

```
# double_tops, \
# triple_bottoms, \
# falling_wedges, \
# rising_wedges, \
```

```
# rising_triangles, \
# falling_triangles
```