Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

РАБОТА ПРОВЕРЕНА
Рецензент,_____
_____
_____/_____
“ ____ ” _____ 2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ПМиП,
д.ф.-м.н.
_____/А.А.Замышляева
“ ____ ” _____ 2018 г.

Анализ временных рядов Forex

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ
КВАЛИФИКАЦИОННОЙ РАБОТЕ МАГИСТРА
ЮУрГУ–01.04.02.2018.185.ПЗ ВКР

Руководитель работы,
доцент кафедры ПМиП, к.т.н.
_____/ М.Ю. Катаргин
« ____ » _____ 2018 г.

Автор работы
студент группы ЕТ-222
_____/ А.Р. Сайтхужина
« ____ » _____ 2018 г.

Нормоконтролер,
доцент кафедры ПМиП,
к.э.н.
_____/ Д.А. Дрозин
« ____ » _____ 2018 г.

АННОТАЦИЯ

Сайтхужина А.Р. Анализ временных рядов Forex. – Челябинск: ЮУрГУ (НИУ), ЕТ-222, 73 с., 41 ил., 40 табл., библиогр. список – 50 наим., 8 прил.

В работе рассмотрены методы технического анализа временных рядов Forex.

В результате классического статистического анализа было установлено, что ряд первых разностей исходных данных временных рядов не вполне соответствует нормальному распределению и является стационарным. На основании расчетов и построений автокорреляционных функций было выяснено, что исследуемый процесс весьма похож на белый шум, что говорит о том, что анализ линейными методами бесперспективен.

Выбор нейронных сетей в качестве нелинейных методов анализа позволил построить модель, правильно предсказывающую и классифицирующую значения приращений первых разностей котировок в 50-60% случаев. На основе этой модели построен алгоритм торгового робота. Результаты тестирования обученных нейронных сетей показали превышение вероятности выигрыша над проигрышем порядка 3-5%, что принято считать хорошим результатом.

Оглавление

ВВЕДЕНИЕ

Международная торговля стремительно развивается с каждым днем. В связи с этим объемы операций мирового валютного рынка постоянно растут. Поэтому торговля валютой сегодня стала одним из самых распространенных видов деятельности: около трех триллионов долларов достигает дневной оборот рынка Forex [1, с. 24].

Валютный курс оказывает большое влияние на многие макроэкономические процессы, происходящие в обществе. Динамика валютного курса, степень и частота его колебаний являются показателями экономической и политической стабильности общества [2]. Именно поэтому так важен анализ финансовых временных рядов.

Рынок Forex меняется, и меняется циклично. Это значит, что регулярно возникают ситуации, когда поведение цены становится максимально прогнозируемым. А если есть шанс спрогнозировать цену, значит, есть возможность на рынке заработать.

Технический анализ – именно это помогает трейдерам предугадывать поведение рынка и увеличивать свой депозит.

Нейронные сети в трейдинге (пер. с англ. trading – торговля) без преувеличения можно отнести к методам технического анализа, поскольку они ищут закономерности на определённом временном промежутке, отталкиваясь от исторических данных.

Сегодня на рынке Форекс можно найти торговые советники (роботы), в основе которых заложена нейронная сеть и технология, позволяющая обучать её по конкретной торговой системе, чтобы в дальнейшем прогнозировать движение активов на рынке. Применение таких интеллектуальных помощников является очень актуальным для трейдеров.

Целью данной работы является анализ временных рядов Forex методами технического анализа.

Задача заключается в том, что на основании анализа с помощью нейронных сетей удастся предложить модель для временных рядов Forex и сформулировать рекомендации по торговле на рынке Forex.

Для достижения поставленной цели требуется решить следующие задачи: провести классический анализ временных рядов Forex; определить возможность предсказания рядов на основе методов классического анализа; при невозможности поспользоваться нейронными сетями, как одной из разновидностей нелинейных методов анализа.

1 МЕТОДЫ И СРЕДСТВА АНАЛИЗА ВРЕМЕННЫХ РЯДОВ

Анализ временных рядов — совокупность математико-статистических методов анализа, предназначенных для выявления структуры временных рядов и для их прогнозирования [3]. Для того чтобы построить математическую модель процесса анализируемого временного ряда необходимо выявить структуру данного ряда. С помощью полученной модели можно построить прогноз последующих значений временного ряда. Полученный прогноз можно использовать для эффективного принятия решений.

1.1 Основные понятия валютного рынка Forex

Forex (от англ. Foreign Exchange — «зарубежный обмен») — это международный финансовый рынок, на котором производится обмен валют. Он был основан в 1976 году, когда все страны мира отказались от золотого стандарта и перешли на ямайскую систему, при которой курсы валют устанавливаются не государством, а рынком. Forex стал просто необходим для нормального функционирования мировой экономики и обеспечения обмена капиталов между разными странами [4].

Прибыль на Forex формируется за счет разницы и изменения курсов валют. Рынок Forex, в отличие от фондового рынка, работает 24 часа в сутки 5 дней в неделю, за исключением выходных.

Операции по обмену валюты, совершаемые в мире на рынке Forex, называются *валютным дилингом*. Самыми распространенными и востребованными мировыми валютами являются:

- USD – доллар США;
- EUR – евро;
- CHF – швейцарский франк;
- JPY – японская йена;
- GBP – английский фунт стерлингов.

Котировка означает, сколько единиц той валюты, которая содержится в паре на втором месте (она называется "валютой котировки"), содержится в единице валюты, которая идет первой (эта валюта называется "базовой"). Например, запись котировки EURUSD=0.8467 означает, что в одной единице евро содержится 0.8467 доллара, так как евро - базовая валюта.

Понятия "купить" и "продать" на рынке всегда применяются в отношении базовой валюты.

Под *пунктом* подразумевается минимальное изменение цены, то есть изменение последней цифры в записи котировки. Если котировка EURUSD=0.8467 изменилась на EURUSD=0.8466, это значит, что евро упал на один пункт.

Тик — это событие, характеризующееся новой ценой по финансовому инструменту в некоторый момент времени.

Бар является основной единицей торговли рынка Forex. Он показывает цену открытия (OPEN), закрытия (CLOSE), а также максимум (HIGH) и минимум (LOW) периода [5]. Иногда бар содержит еще и объем (VOL). Пример ценового бара изображен на рисунке 1.1:

Рисунок 1.1 – Ценовой бар

1.2 Способы представления временных рядов Forex

Для анализа рынка Forex строятся графики, где по вертикальной оси расположена цена и/или тиковый объем, а по горизонтальной оси – время.

Торговым периодом называют интервал времени, который используется для построения графика.

В качестве временного периода ценовых графиков традиционно используются следующие интервалы: месяц, неделя, день, 4 часа, 1 час, 30 минут, 15 минут, 10 минут, 5 минут, 1 минута, тик.

Тиком является единичное изменение цены маркетмейкером рынка переданное форме новых цен покупки и продажи (Ask и Bid – пер. с англ. «спрос» и «предложение»).

Для построения графиков (за исключением тиковых) используются следующие характеристики котировок:

- *цена открытия периода (OPEN)* - цена на рынке Forex, сложившаяся на начало торгового периода ((Ask+Bid)/2);
- *цена закрытия периода (CLOSE)* - цена на рынке Forex, сложившаяся на конец торгового периода ((Ask+Bid)/2);
- *максимальная цена периода (HIGH)* - самая высокая цена на рынке Forex, сложившаяся за период (обычно используется Ask);
- *минимальная цена периода (LOW)* - самая низкая цена на рынке Forex, сложившаяся за период (обычно используется Bid);
- *тиковый объем (tick volume)* - количество тиков (изменений цены маркет-мейкеров) за период.

Существуют следующие виды графиков [1, с. 65]:

1) Тиковый график (Tick chart).

Тиковый график представляет собой график самого мелкого масштаба – единичных котировок цены (рисунок 1.2). Так как в отличии от бирж на Forex отсутствует информация о количестве сделок и их объеме, то тиковый график представляет собой котировки Bid и Ask маркетмейкеров рынка.

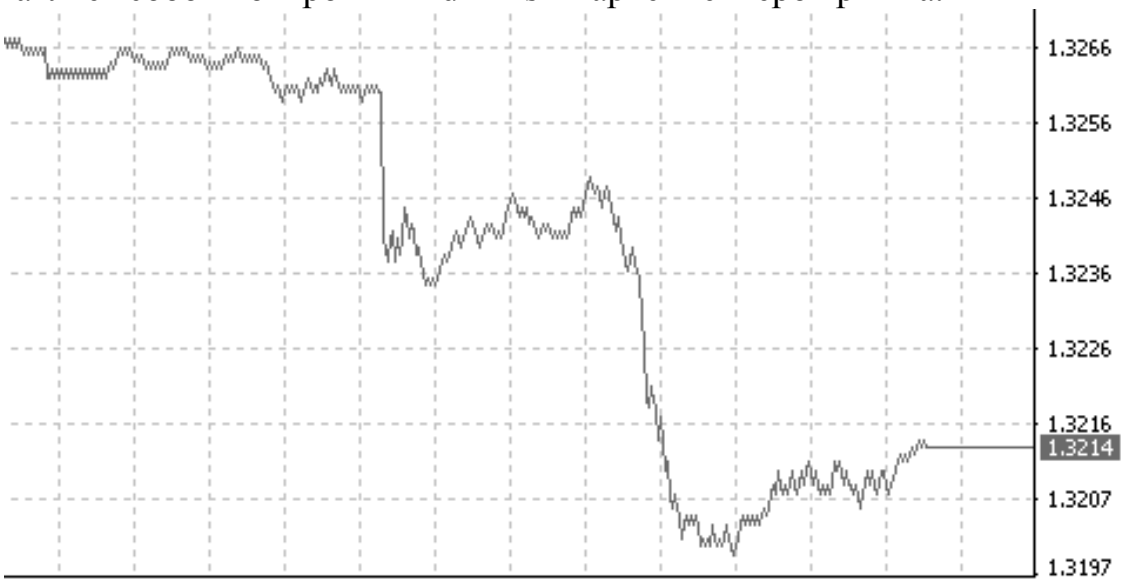

Рисунок 1.2 – Тиковый график

Верхняя часть каждой черты представляет собой котировки – Ask, а нижняя – Bid. Тиковый график используется трейдерами Forex в основном не для анализа рынка, а для определения конкретной точки входа в рынок, когда решение о входе в рынок (со всеми соответствующими параметрами) уже принято.

2) Линейный график (Line Chart)

Для построения линейного графика цены в большинстве случаев используется цена закрытия (Close), в более редких случаях используют цену открытия (Open), наивысшую цену за период (High) и минимальную цену за период (Low) (рисунок 1.3).

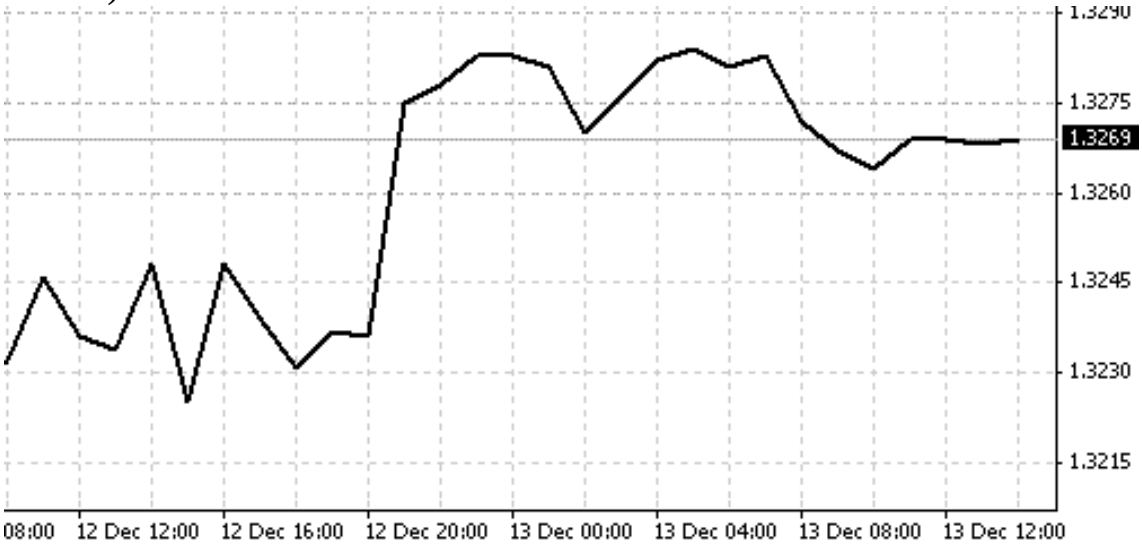

Рисунок 1.3 – Линейный график

3) График баров (Bar chart)

График баров выглядит следующим образом:

Рисунок 1.4 – График баров

Вертикальная черта бара – это показатель того диапазона цен, в котором торговалась валютная пара. Самая низкая точка бара по вертикали – это уровень самой низкой цены за рассматриваемый временной период, а вершина бара соответствует самой высокой цене за этот же период. Горизонтальная отметка слева на баре – это уровень цены открытия, а справа – закрытия [5].

Если левая черта выше правой это означает, что цены в начале базового периода были выше, чем в конце, другими словами цена на рынке падала. Если ниже, то цена росла. Недостатки: несмотря на то, что график баров позволяет видеть диапазон изменения цены внутри периода, он не позволяет видеть, какой характер носили движения цены внутри периода. Для этого нужно переключиться на более мелкие интервалы.

4) Японские свечи (Japanese candlesticks)

График японских свечей выглядит следующим образом:

Рисунок 1.5 – График японских свечей

График японских свечей достаточно похож на баровый график, но более удобен для визуального восприятия. Как и баровый график он тоже строится по ценам open, high, low, close. Свеча изображена на рисунке 1.6.

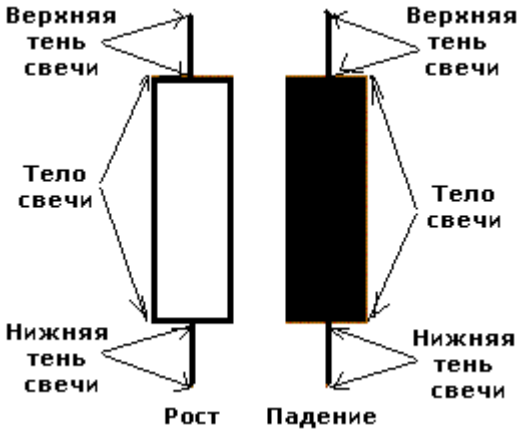

Рисунок 1.6 – Японская свеча

Расстояние между ценой открытия и ценой закрытия образует тело свечи (прямоугольник - джиттай), который закрашивается в зависимости от соотношения цен открытия и закрытия [5]. Традиционно действует следующее правило: если цена открытия выше цены закрытия (т.е. цена упала за торговый период), то тело свечи окрашивается в темный цвет, если же цена открытия ниже цены закрытия (цена за торговый период выросла), то тело свечи закрашивается в светлый цвет или не закрашивается. В большинстве программ технического анализа цвета тела свечи можно менять, поэтому комбинации могут быть и другими, например, красный и зеленый и др.

Расстояние между телом свечи и максимальной ценой дня рисуется в виде вертикальной линии, которую называют верхней тенью свечи. Расстояние между телом свечи и минимальной ценой периода также рисуется в виде линии и называется нижней тенью свечи.

Также как и на баровом графике, график японских свечей не позволяет видеть характер движения внутри торгового периода, а показывает лишь диапазон колебаний цены.

1.3 Фундаментальный и технический анализ

1.3.1 Фундаментальный анализ

Фундаментальный анализ занимается оценкой ситуации с точки зрения политической, экономической и финансово-кредитной политики. Информация об учетных ставках центральных банков, экономический курс правительства, возможные перемены в политической жизни страны, а также всевозможные слухи и ожидания являются наиболее важными в фундаментальном анализе [6].

Фундаментальный анализ включает оценку следующих факторов, влияющих на валютные курсы:

- Показатели экономического роста (валовой национальный продукт, объемы промышленного производства и др.)
- Состояние торгового баланса, степень зависимости от внешних источников сырья
- Рост денежной массы на внутреннем рынке
- Уровень инфляции и инфляционные ожидания
- Уровень процентной ставки
- Платежеспособность страны и доверие к национальной валюте на мировом рынке
- Спекулятивные операции на валютном рынке
- Степень развития других секторов мирового финансового рынка, например, рынка ценных бумаг, конкурирующего с валютным рынком.

Успех трейдера в проведении фундаментального анализа зависит от понимания законов финансовых рынков, умении сопоставлять между собой абсолютно несвязанные на первый взгляд события. Также фундаментальный анализ предполагает знание общей картины, только уточняемой новостями. Поэтому при фундаментальном анализе зачастую происходит сравнение показателя для одной страны в разные промежутки времени, и редко сравнивается, например, уровень безработицы в США с зоной евро.

1.3.2 Технический анализ

Технический анализ - метод прогнозирования цен исключительно за счет данных истории рынка - цен, объема и открытого интереса. Большинство методов технического анализа используют именно цену, поскольку данные о цене обычно являются общедоступными и, как правило, поступают в режиме реального времени [7]. На рынке Forex технический анализ использует данные только о ценах и тиковом объеме (tick volume - количестве котировок), поскольку данные о реальных объемах недоступны, а открытый интерес не рассчитывается.

Технический анализ является средством статистической оценки массовой человеческой психологии, на основе исторических данных. Считается, что технические методы анализа должны одинаково работать на всех видах финансовых рынках [8].

Технический анализ отличается от других видов исследований также и тем, что он основан на математических, статистических, а не экономических расчетах. Все методы технического анализа разрабатывались отдельно, поэтому никакой строгой системы не существует, есть лишь набор методов или методик, от самых простых (графический анализ), до сверхсложных (например, нейронные алгоритмы). Более того, часто различные отдельные методы технического анализа противоречат друг другу. Единственное, что связывает эти отдельные методики – общие принципы и аксиомы.

В [9] методы технического анализа разделены на ряд категорий:

1) Графические методы технического анализа.

В графических методах для анализа используется обычное или трансформированное изображение рынка (различные виды графика цены и/или объема). Обычно эти методы основаны на каких-либо повторяющихся шаблонах (паттернах) поведения цены.

2) Индикаторные методы.

Методы, использующие фильтрацию или математическую аппроксимацию, весьма условно разделяются на ряд категорий:

– *Трендовые индикаторы.* Трендовые индикаторы показывают силу основной тенденции рынка и ее направление, например, Moving Average, Directional Movement, Parabolic, Kaufman Efficiency Ratio и т.д.

– *Индикаторы волатильности (изменчивости).* Указывают на силу изменчивости рынка, силу движений, не зависящих от основного тренда, например, Standard Deviation, Bollinger Bands и т.д.

– *Индикаторы ускорения (момента).* Этот очень широкий класс методов используется для определения текущей скорости изменения цены (ускорения). К ним относятся большая часть осцилляторов, например, Momentum, Relative Strength Index (RSI) и Price Rate-Of-Change (ROC), Stochastic Oscillator и т.д.

– *Методы определения цикла.* К ним относятся различные индикаторы, использующие в своей основе спектральный анализ Фурье или построение тригонометрических кривых.

– *Методы исследования объема.* Это класс индикаторов использует в качестве основных данных временные ряды объема торговли и используется для определения силы рыночного движения, поскольку считается, что чем больше объем торгов, тем больше инвесторов «верит» в движение, а значит, оно не является ложным.

– *Индикаторы уровней поддержки и сопротивления.* Методы, указывающие, что текущее движение, вероятно, остановится на определенном уровне, например, Moving Average Envelopes, Fibonacci Retracements и т.д.

3) Вероятностные методы технического анализа.

Методы, использующие приемы теории вероятностей и математической статистики для определения силы тренда, вероятности коррекции и т.д., например, профиль рынка.

Существует также большое разнообразие методов, которые нельзя причислить ни к одной группе. Достаточно часто встречаются торговые системы, торгующие на основе данных по электромагнитному полю земли, солнечной активности, лунным циклам. Существуют компании, которые поставляют эти данные в режиме реального времени.

По оценкам технический анализ является наиболее популярным классом методов. В той или иной мере его используют около 50% трейдеров.

1.4 Методы анализа временных рядов

Методы анализа временных рядов определяются целями анализа и вероятностной природой формирования значений ряда. Наиболее распространенными методами являются:

1) *Спектральный анализ.* Позволяет находить периодические составляющие временного ряда.

2) *Корреляционный анализ.* Позволяет находить существенные периодические зависимости и соответствующие им задержки (лаги) как внутри одного ряда (автокорреляция), так и между несколькими рядами. (кросскорреляция).

3) *Модели авторегрессии и скользящего среднего*. Модели ориентированы на описание процессов, проявляющих однородные колебания, возбуждаемые случайными воздействиями. Позволяют предсказывать будущие значения ряда.

4) *Нейронные сети.* Позволяют на основе подобранной модели поведения временного ряда предсказывать его значения в будущем.

1.4.1 Спектральный анализ

Колебания относительно тренда выявляются применением спектрального анализа, для исследования таких процессов используют гармонические модели и модели авторегрессии либо скользящего среднего.

Спектральный анализ - это разновидность обработки данных, связанная с преобразованием их частотного представления или спектра.

Классическим примером спектрального анализа является разложение временных рядов с циклическими компонентами на несколько гармонических функций с различными частотами с помощью решения задачи линейной множественной регрессии, где зависимая переменная – наблюдаемый временной ряд, а независимые переменные или регрессоры – функции синусов всех возможных (дискретных) частот:

(1.1)

где – круговая частота, выраженная в радианах в единицу времени ( );

и – коэффициенты регрессии, показывающие степень, с которой соответствующие функции коррелируют с данными [10].

Спектральный анализ определяет корреляционную функцию синусов и косинусов различной частоты с наблюдаемыми данными. Если найденная корреляция (коэффициент при определенном синусе или косинусе) велика, то можно заключить, что существует строгая периодичность на соответствующей частоте в данных.

Преимуществом методов спектрального анализа является возможность оценить вклад колебательной компоненты без вычисления других компонент ряда.

1.4.2 Корреляционный анализ

Корреляционный анализ необходим для выявления корреляций и их лагов — задержек их периодичности. Связь в одном процессе получила название *автокорреляции*, а связь между двумя процессами, характеризуемыми рядами — *кросскорреляции*. Высокий уровень корреляции может служить индикатором причинно-следственных связей, взаимодействий внутри одного процесса, между двумя процессами, а величина лага указывает временную задержку в передаче взаимодействия [11].

Зависимостью между соседними уровнями ряда называют *коэффициент автокорреляции*.

Автокорреляционная функция – это изменение коэффициентов автокорреляции в зависимости от лага:

(1.2)

(1.3)

где τ–порядок коэффициента корреляции;
n – количество наблюдений;
x_i – значение i-го признака одного события;
y_i – значение i-го признака другого события;
– средние значения x и y соответственно.

Максимумы модуля автокорреляционной функции показывают наличие лагов, т.е. промежутков времени, на которых проявляется скрытая зависимость случайных величин. К примеру, в рядах с существенным влиянием циклической компоненты лаги выражены на графике выборочной автокорреляционной функции особенно сильно. Модели, использующие лаговую автокорреляцию, называются автокорреляционными (АМ) или авторегрессионными.

Для применения автокорреляционных моделей (АМ) желательно иметь временной ряд, автокорреляционная функция которого имеет небольшое число максимумов и достаточно быстро спадает с ростом шага автокорреляции. Если имеется цикличность данных, которая меняется со временем, то полностью исключить ее различными методами сглаживания, как правило, не удается. В этом случае автокорреляционные модели применяются на этапе качественного анализа, точность которого должна быть улучшена с использованием других подходов.

Последовательное усложнение автокорреляционных моделей привело к появлению т.н., адаптивных моделей прогнозирования, которые базируются на объединении двух схем – скользящего среднего (СС) и авторегрессии. Модель СС состоит в том, что для определения свойств временного ряда с целью краткосрочного прогноза берется выборка последних данных за некоторый промежуток времени Т.

1.4.3 Модели авторегрессии и скользящего среднего

Достаточно часто временные ряды имеют сложную структуру. Моделирование таких рядов путем построения модели тренда, сезонности и периодической составляющей не приводит к удовлетворительным результатам, помимо этого, необходимо учитывать автокоррелированность временного ряда.

В этих случаях могут быть использованы модели авторегрессии - скользящего среднего (АРИСС-модели).

Выделяют следующие модели:

- авторегрессионная модель, АР (AR – англ. autoregressive model);
- модель скользящего среднего, СС (MA);
- смешанная модель с авторегрессией и скользящим средним, АРСС (ARMA);
- интегрированная модель авторегрессии-скользящего среднего, АРИСС, используемая для нестационарных процессов (ARIMA).

Временные ряды, в которых последовательные измерения слишком зависимы, целесообразно рассматривать как генерируемые последовательностью независимых импульсов, которые представляют собой реализации случайных величин с фиксированным распределением, которое обычно предполагается нормальным с нулевым средним, с отсутствием автокорреляции [12]. Такая последовательность называется белым шумом.

Рассмотренные модели основываются на гипотезе, что изучаемый процесс является выходом линейного фильтра, на вход которого подан процесс белого шума (рисунок 1.7).

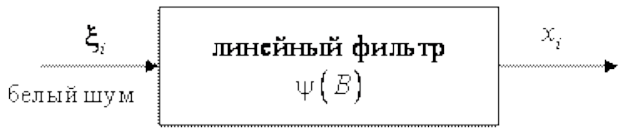

Рисунок 1.7 – Представление временного ряда
с помощью линейного фильтра

Авторегрессионная модель – модель временных рядов, в которой значения временного ряда в данный момент линейно зависят от предыдущих значений этого же ряда. Авторегрессионный процесс порядка p (*AR(p)*-процесс) определяется следующим образом:

(1.4)

где – параметры модели (коэффициенты авторегрессии);

– постоянная (часто для упрощения предполагается равной нулю);

– белый шум.

Модель скользящего среднего порядка q (*MA(q)* – модель временного ряда) вида:

(1.5)

где – белый шум;

– параметры модели (можно считать равным 1 без ограничения общности).

Моделью *ARMA(p, q)*, где *p* и *q* — целые числа, задающие порядок модели, называется следующий процесс генерации временного ряда :

(1.5)

где – константа;

– белый шум, то есть последовательность независимых и одинаково распределённых случайных величин (как правило, нормальных), с нулевым средним;

и – действительные числа, авторегрессионные коэффициенты и коэффициенты скользящего среднего, соответственно.

Модель ARIMA(p,d,q) (модель Бокса Дженкинса) [3] для нестационарного временного ряда имеет вид:

(1.5)

где – стационарный временной ряд;

– параметры модели.

– оператор разности временного ряда порядка d (последовательное взятие d раз разностей первого порядка - сначала от временного ряда, затем от полученных разностей первого порядка, затем от второго порядка и т.д.)

1.4.4 Нейронные сети

До недавнего времени самыми распространенными методами прогнозирования были однофакторные прогнозы по временным рядам, основанные на регрессионных методах. Однако такие прогнозы неспособны учитывать влияние на валютные курсы таких нерегулярных факторов, как учетные ставки центральных банков, экономический курс правительства, возможные перемены в политической жизни страны, поэтому на практике следует применять многофакторное прогнозирование, позволяющее строить прогноз с точностью, значительно превышающей точность по временным рядам [13].

Среди всех многофакторных подходов особо выделяется метод на базе искусственных нейронных сетей, позволяющий устанавливать связи между выходными характеристиками системы и входными факторами [14]. Такие связи позволяют вычислить будущие значения параметров и достаточно точно

классифицировать группы объектов, например, группы выигрышных и проигрышных ситуаций на рынке Forex.

Именно поэтому нейронные сети можно считать последним нововведением, которое было сделано участниками торгов.

Нейронные сети в трейдинге без преувеличения можно отнести к методам технического анализа, поскольку они ищут закономерности на определённом временном промежутке, отталкиваясь от исторических данных.

1.4.4.1 Искусственная нейронная сеть

Искусственная нейросеть построена по принципу биологических нейронных сетей, то есть, она копирует организацию нервных клеток живого организма и состоит из искусственных нейронов. Она представляет собой математическую модель и ее воплощение в аппаратном и программном обеспечении для осуществления сложных логических вычислений.

Простейшая нейросеть состоит из трех слоев нейронов:

- входного (input layer), элементами которого являются входные данные;
- скрытого (hidden layer), состоящего из вычислительных узлов, называемых нейронами (neurons);
- выходного (output layer), который состоит из одного или нескольких нейронов, выходы которых являются выходами всей сети.

Эта последовательность называется перцептроном. Более сложные сети могут содержать более 3 слоев.

На входы подаются данные задаваемые пользователем, например, выборки временного ряда. Смысл выходных данных также задаётся пользователем, например, сигналы 1=buy и 0=sell.

Все узлы соседних слоёв связаны между собой. Эти связи называются синапсами (synapses). Каждый синапс имеет вес (weight w[i,j,k]), на которой умножаются данные передаваемые по синапсу. Данные передвигаются слева направо, т.е. от входов сети к её выходам. Отсюда и название, "сеть прямого распространения". Общий пример этой сети изображён на рисунке 1.8.

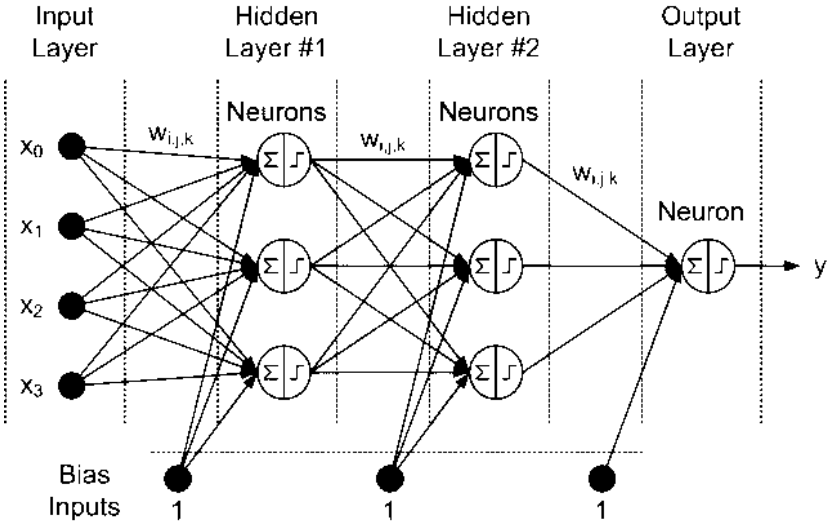

Рисунок 1.8 – Сеть прямого распространения

Данные перерабатываются нейронами за два шага:

1. Все входы, помноженные на соответствующие веса, сначала суммируются

2. Затем получившиеся суммы обрабатываются функцией активации нейрона (activation or firing function) и посылаются на единственный выход.

Смысл функции активации нейрона заключается в моделировании работы нейрона мозга: нейрон срабатывает только после того как информация достигла определённого порога. В математическом аспекте, эта функция как раз и придаёт нелинейность сети. Без неё, нейронная сеть была бы линейной авторегрессионной моделью (linear prediction model). Наиболее часто используемыми функциями активации нейрона являются сигмоидальная функция, гиперболический тангенс и рациональная функция (рисунок 1.9):

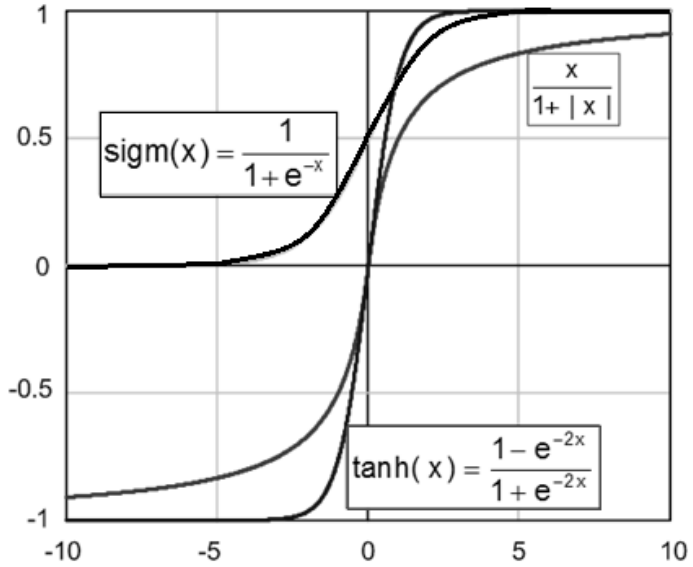

Рисунок 1.9 – Функции активации

Порог активации этих функций равен 0. Этот порог может быть сдвинут по горизонтальной оси за счёт дополнительного входа нейрона, называемым входом смещения (bias input), которому приписан определённый вес таким же образом, как и к другим входам нейрона.

Таким образом, количество входов, слоев, нейронов в каждом слое и веса входов нейронов полностью определяют нейронную сеть, т.е. нелинейную модель, которую она создаёт. Чтобы пользоваться этой моделью необходимо знать веса. Веса вычисляются путём обучения сети на данных, результат для которых известен: на входы сети подаются множество наборов входных и соответствующих выходных данных и рассчитывается среднеквадратичная ошибка отклонения выхода сети от ожидаемого. Цель обучения сети заключается в уменьшении этой ошибки путём оптимизации весов. Существуют несколько методов оптимизации, среди которых основными являются метод Обратного Распространения Ошибки (ОРО) и метод генетической оптимизации.

Ключевой способностью нейронной сети является способность к обучению. Основные задачи, решаемые с помощью нейросетей:

- классификация – распределение данных по параметрам;
- прогнозирование – например, курса валют исходя из имеющихся данных;
- распознавание образов;
- кластеризация.

Наиболее успешно нейронные технологии применяются для распознавания образов.

1.4.4.2 Программное обеспечение для создания нейросетей для целей трейдинга

Уже довольно давно на рынке присутствуют достаточно мощные пакеты и программные комплексы, позволяющие как проектировать нейронные сети для рынка Forex самостоятельно, так и включающие в себя готовые решения для торговли. В таблице 1.1 приведены наиболее распространенные пакеты.

Таблица 1.1

Программные пакеты и комплексы для работы с нейронными сетями

Название пакета	Описание
ExcelNeuralPackage	Пакет, разработанный российскими специалистами расширяющий возможности MS Excel в области нейротехнологий.
Nero Brainmaker	Несложная программа для проектирования сетей с большим количеством слоев.
NeuroLab	Приложение для Wealth-Lab.
NeuralWorks	Семейство продуктов для разработки сетей.
NeuroShell	Продукты для решения широкого ряда задач, в том числе и в трейдинге.
Trading Solutions	Специализированная программа для трейдеров, позволяющая им создавать и отлаживать нейронные сети.
Statistica	Продукты компании StatSoft для статистического анализа, в том числе, с применением нейросетей.
Matlab	Neural Network Toolbox - это пакет расширения MATLAB, содержащий средства для проектирования, моделирования, разработки и визуализации нейронных сетей.
Python	Библиотеки Keras, TensorFlow, PyBrain

Для Forex-трейдеров существует возможность писать роботов и советников на языках MQL4 и MQL5. Для работы в этом направлении есть соответствующие библиотеки. Также можно использовать пакет NeuroSolutions, который позволяет не только создавать нейронные сети, но и подключаться к MetaTrader.

1.4.4.3 Преимущества и недостатки нейронных сетей

Стоить отметить важные преимущества нейронных сетей.

1) Существенным плюсом нейросетей является тот факт, что обучение происходит на постоянной основе за счёт новых данных и уже имеющихся прогнозов.

2) Существует возможность комбинировать технические и фундаментальные данные, что позволяет их оптимально применять.

3) Нейронные сети в Forex обладают определёнными навыками, позволяющими определять на рынке неучтённые факторы и применять их в прогнозировании, добиваясь максимально точного результата.

К сожалению, на сегодняшний день нейронные сети в трейдинге показывают противоречивые результаты. Это связано со следующими причинами:

1) Нейросети предоставляют еще одну возможность статистического анализа и поэтому им свойственны все проблемы и болезни статистических методов: успешный анализ исторических данных не гарантирует успеха в будущем – это утверждение в полной мере справедливо и для нейросетей.

2) По мере усложнения сети, количество вычислений растет по экспоненте.

3) Нейронные сети работают по принципу черного ящика: загружая в сеть данные и получая результат, трейдер не понимает принципов, на основании которых она принимает решение, значит он не склонен доверять ей свои деньги, тем более, на таком рынке, как Forex.

В краткосрочной торговле, и, в частности, на Forex, нейросети показывают слабую эффективность, тем самым подтверждая утверждение нобелевского лауреата Юджина Фама о хаотическом характере изменения цены и невозможности предсказаний в краткосрочном плане [15]. Однако они могут быть полезны для анализа долгосрочных процессов и выработки инвестиционных прогнозов, а также при анализе инвестиционных рисков.

Уже сейчас банки и инвестиционные компании активно применяют нейротехнологии. Возможность обрабатывать большие массивы и способность к обучению, позволяют нейронным сетям на Forex идентифицировать более сложные паттерны, чем это возможно с помощью вероятностных, индикаторных и графических методов.

1.5 Выбор программных средств

Среди упомянутых выше средств существует возможность создания нейронной сети для трейдинга с помощью языка Python с его библиотеками такими, как Keras, TensorFlow, PyBrain, Scikit-learn и др.

В данной работе отдано предпочтение именно этому языку, т.к. он является одним из лидирующих инструментов в анализе данных и машинном обучении и весьма прост в использовании [16].

Для статистического анализа временных рядов использованы следующие библиотеки:

1) Numpy – необходима для работы с многомерными массивами и матрицами, вместе с большим пакетом высокоуровневых математических функций для операций над этими массивами (матрицами) [17].

2) Pandas – высокоуровневая библиотека Python, построенная поверх низкоуровневой Numpy. Pandas предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами [18].

3) Scipy – содержит модули для линейной алгебры, оптимизации, интеграции и статистики [19]. SciPy работает совместно с NumPy, что позволяет ей значительно расширить функциональность.

4) Matplotlib – является основным инструментом для визуализации данных на языке Python [20]. С ее помощью можно создавать линейные графики, гистограммы, графики рассеяния и др.

Существует множество средств для работы с нейронными сетями на Python. Анализ рейтингов библиотек Python в сети интернет [21, 22] показал, что наиболее удобной и простой в использовании является библиотека Keras. Это средство представляет собой высокоуровневый интерфейс для создания нейронных сетей [23]. Keras написан на Python и работает поверх таких более низкоуровневых решений, как TensorFlow [24] и Theano. Благодаря высокой скорости обучения нейронной сети и большой популярности среди разработчиков [25] была выбрана именно эта библиотека.

Выводы по разделу

В данном разделе изучены основные понятия рынка Forex. Рассмотрены 4 способа визуализации временных рядов: тиковый график, линейный график, график баров и график японских свечей.

Анализ временных рядов можно проводить с помощью методов и средств фундаментального и технического анализа. Данная работа посвящена методам технического анализа.

Выполнен краткий обзор методов анализа данных, представленных в виде временных рядов. В частности, рассмотрен спектральный анализ, корреляционный анализ, анализ с помощью модели авторегрессии и скользящего среднего, а также анализ с помощью нейронных сетей.

В работе проведен классический статистический анализ временных рядов Forex, позволяющий определить наличие тренда, зависимостей как между элементами одного ряда, так и между значениями двух рядов. Для решения задач прогнозирования и классификации временных рядов использованы нейросетевые методы технического анализа.

В качестве средств анализа временных рядов выбран язык Python с его библиотеками, так как в последнее время он набирает популярность среди разработчиков в качестве инструмента для анализа данных и машинного обучения.

2 ПОДГОТОВКА ИСХОДНЫХ ДАННЫХ К АНАЛИЗУ

Источниками данных являются опубликованные в сети Интернет истории рядов Forex за весьма длительные периоды.

Прежде чем начать работать с данными, необходимо проверить их на наличие так называемых «дефектов» (пропуски в последовательности данных, избыточные отсчеты, несопоставимые недельные отрезки). Ниже рассматриваются этапы преобразования исходных данных для последующего использования их в анализе.

2.1 Формат представления данных

В качестве начальных исходных данных были взяты финансовые временные ряды котировок валютной пары EUR/USD рынка Forex с интервалами от 1 минуты до 240 минут (таблица 2.1) в виде csv-файлов (формат 1) [26], которые содержат следующие поля:

<DTYYYYMMDD><TIME>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>

где DTYYYYMMDD – формат даты (Год-Месяц-День), в котором представлены исходные данные;
TIME – время в виде Часы-Минуты;
OPEN – цена открытия;
HIGH – максимальное значение цены за период времени, который содержит бар;
LOW – минимальное значение цены за период времени, который содержит бар;
CLOSE – цена закрытия;
VOL – объем.

Таблица 2.1

Наборы данных для различных интервалов времени

Интервал, мин	Дата начала периода (ГГГГ-ММ-ДД)	Дата окончания периода (ГГГГ-ММ-ДД)
1	2017-01-02	2017-10-15
15	2012-11-08	2017-10-15
30	2008-01-01	2017-10-15
60	2012-10-29	2017-10-15
240	2012-10-29	2017-10-15

В таблицах 2.2-2.4 приведены данные для интервалов 1 минута, 60 и 240 минут, вид которых обозначим как «данные в формате 1»:

Таблица 2.2

Данные для интервала 1 минута в формате 1

DTYYYYMMDD TIME	OPEN	HIGH	LOW	CLOSE	VOL
02.01.2017 0:05	1,05185	1,05185	1,05164	1,05165	6
02.01.2017 0:06	1,05164	1,05165	1,05164	1,05165	3
02.01.2017 0:07	1,05164	1,05165	1,05164	1,05165	3
02.01.2017 0:08	1,05166	1,05166	1,05166	1,05166	2

Таблица 2.3

Данные для интервала 60 минут в формате 1

DTYYYYMMDD TIME	OPEN	HIGH	LOW	CLOSE	VOL
29.10.2012 0:00	1,29407	1,29447	1,29406	1,29419	191
29.10.2012 1:00	1,29419	1,29431	1,2924	1,29254	905
29.10.2012 2:00	1,29252	1,29309	1,29184	1,29214	747
29.10.2012 3:00	1,29214	1,29284	1,29144	1,29195	968
29.10.2012 4:00	1,29195	1,29293	1,29179	1,29293	686

Таблица 2.4

Данные для интервала 240 минут в формате 1

DTYYYYMMDD TIME	OPEN	HIGH	LOW	CLOSE	VOL
29.10.2012 4:00	1.29195	1.29363	1.29179	1.29286	2488
29.10.2012 8:00	1.29285	1.29342	1.28859	1.28967	6176
29.10.2012 12:00	1.28970	1.29154	1.28886	1.28969	7802
29.10.2012 16:00	1.28972	1.29230	1.28847	1.28954	6904
29.10.2012 20:00	1.28942	1.29067	1.28913	1.29047	1767

Построены OHLC (*Open*, *High*, *Low*, *Close*) графики выгруженных данных, представленные в виде японских свечей, где тело свечи черного цвета означает, что цена закрытия ниже цены открытия, и торги в этот период времени закрылись понижением, а зеленое тело говорит о том, что цена закрытия свечи больше цены открытия и торги закрылись с повышением (рисунок 2.1).

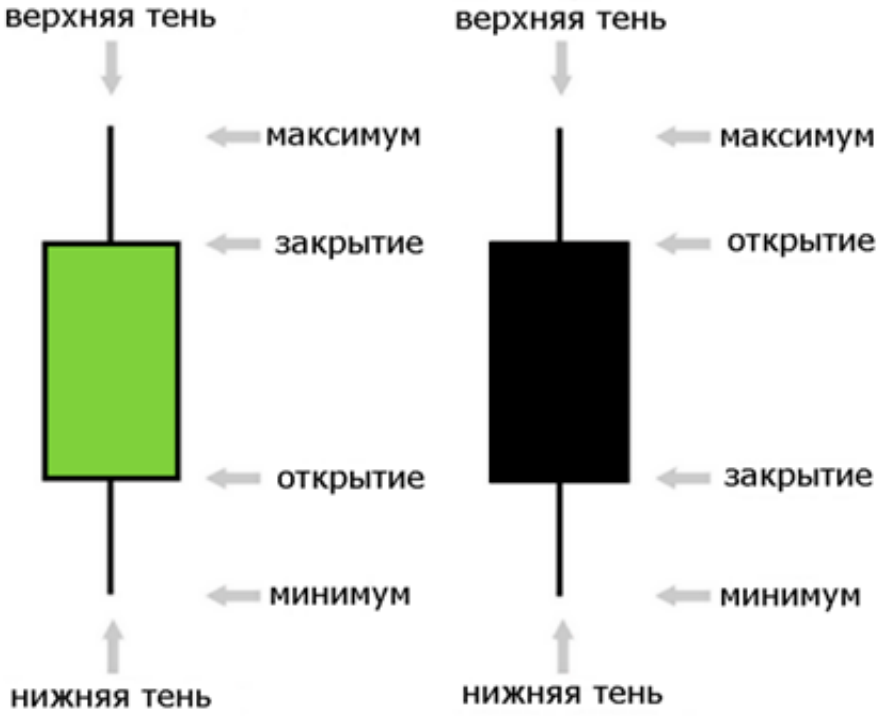

Рисунок 2.1 – Японские свечи

Графики японских свечей для интервалов 1 минута, 60 и 240 минут изображены на рисунках 2.2-2.4.

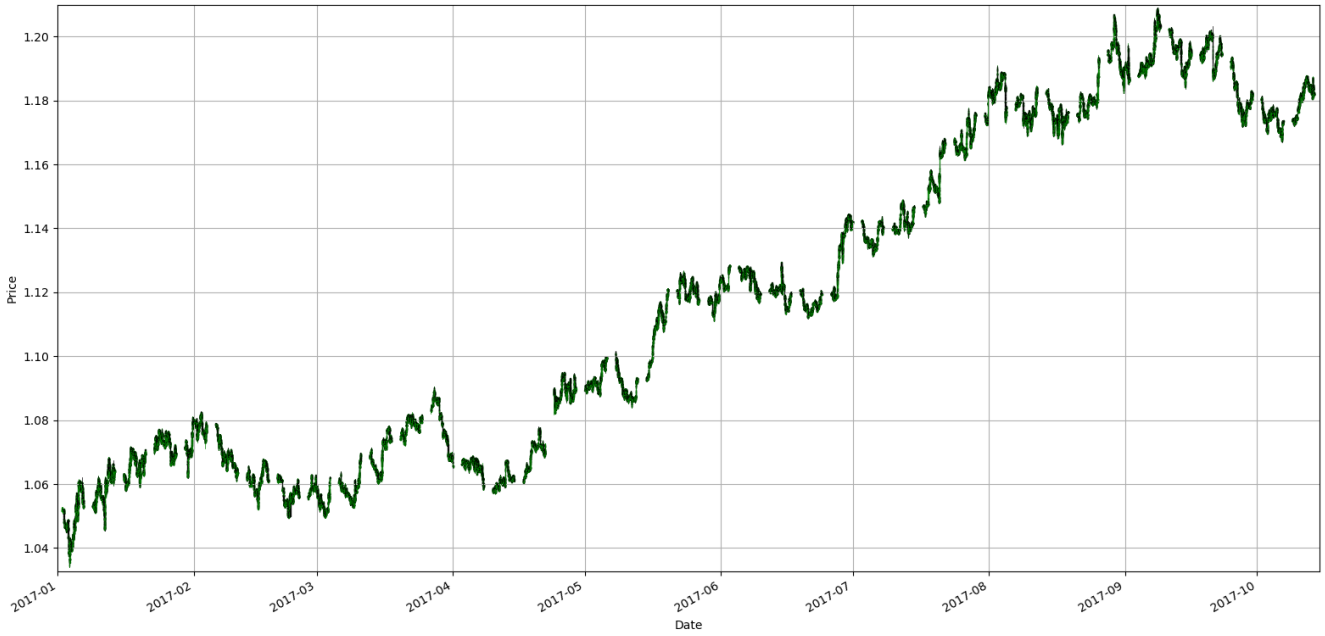

Рисунок 2.2 – График японских свечей для интервала 1 минута

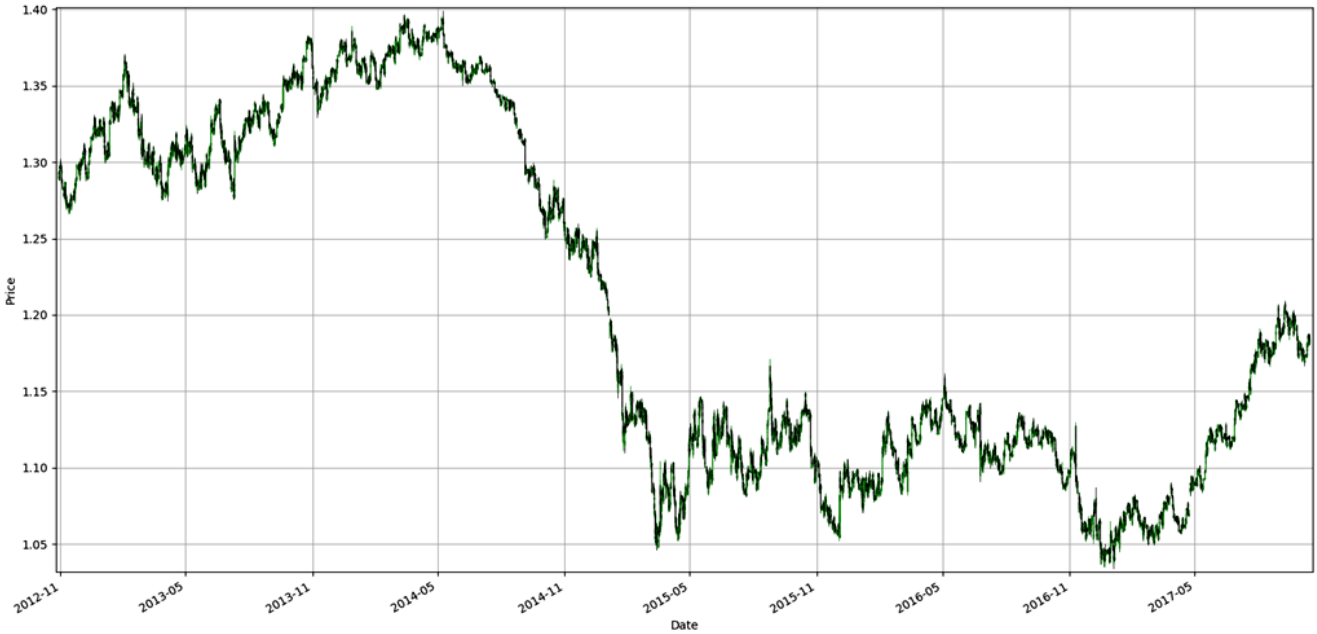

Рисунок 2.3 – График японских свечей для интервала 60 минут

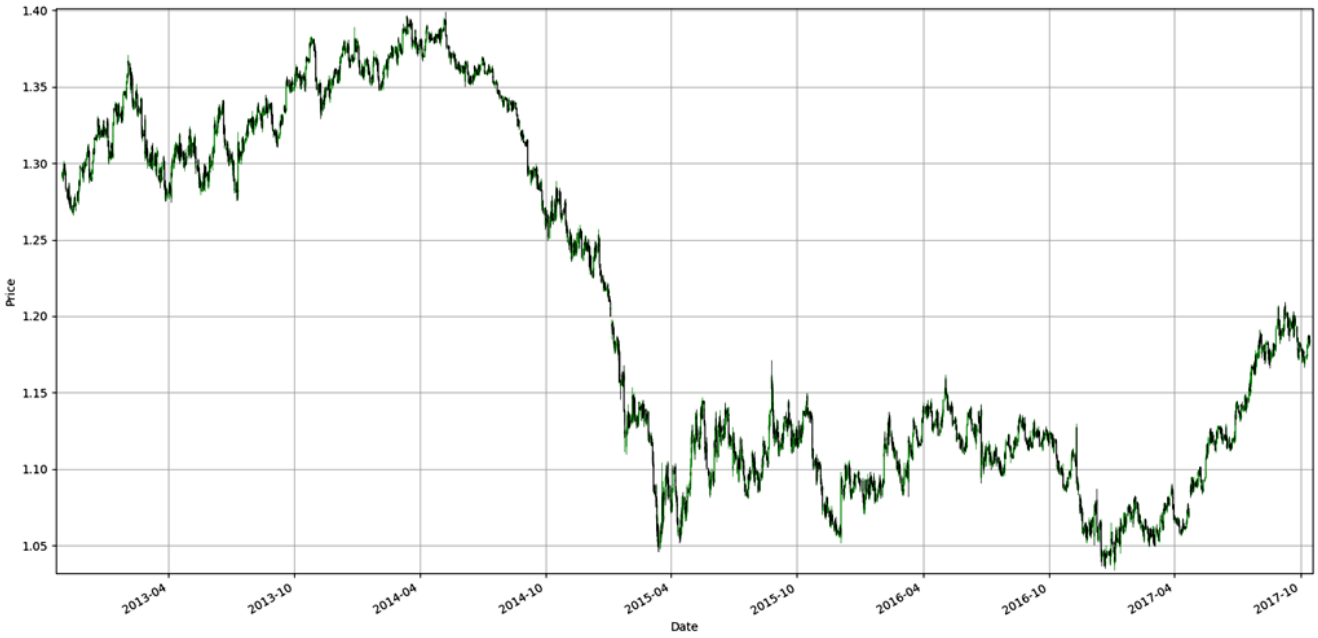

Рисунок 2.4 – График японских свечей для интервала 240 минут

На всех трех графиках (см. рисунки 2.2-2.4), а особенно на рисунке 2.2, видно, что имеются пропуски в данных, о чем свидетельствуют разрывы (промежутки) в графиках. Для наглядности увеличим масштаб графика OHLC для интервала 240 минут (рисунок 2.5):

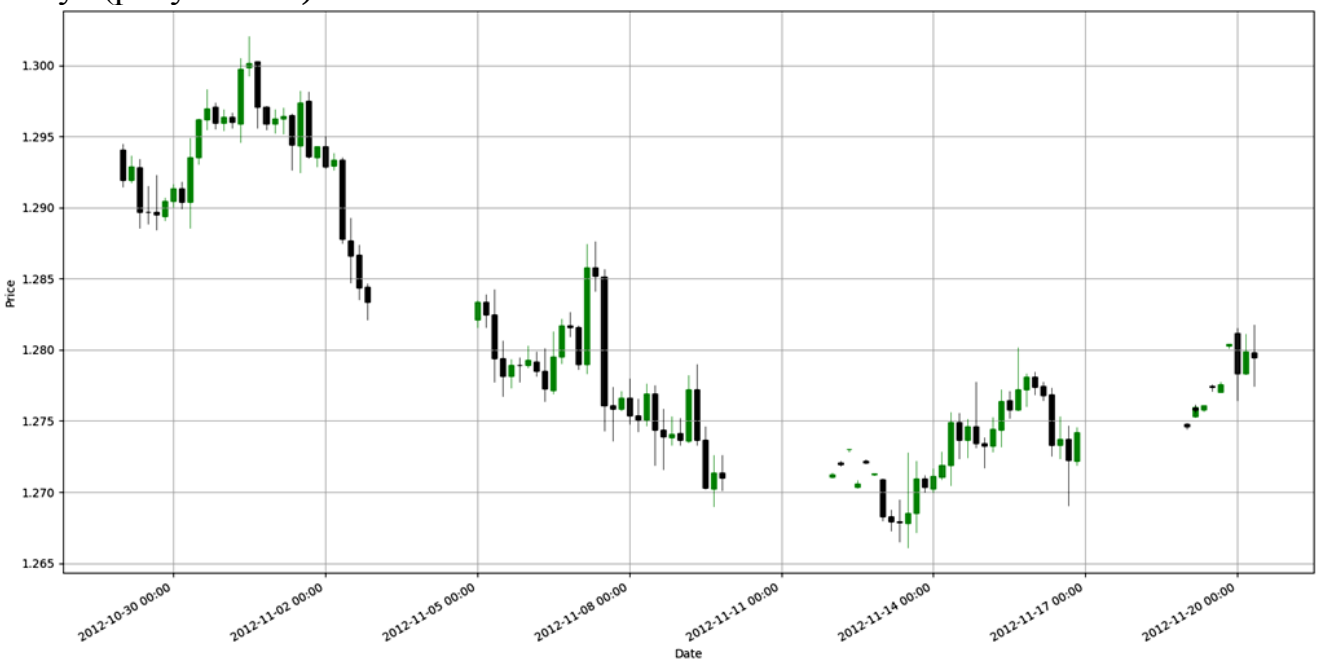

Рисунок 2.5 – Увеличенный график японских свечей
для интервала 240 минут

На рисунке 2.5 отчетливо видно пропуски в данных. В основном это связано с тем, что в выходные и праздничные дни рынок Forex не ведет торги.

2.2 Преобразование данных

При просмотре данных и графиков выгруженных котировок было выявлено следующее:

 Имеются пропуски в последовательности данных.

 Имеются избыточные отсчеты, например, для 15 минутного интервала имеются отрезки, идущие через 5, а потом 10 минут.

 Пропущены выходные дни.

 Недельные отрезки должны быть сопоставимы, т.е. иметь примерно одинаковое число отсчетов.

Этапы преобразования данных.

1) Не все методы анализа временных рядов допускают работу с неравномерными отсчетами. Поэтому перед анализом данные необходимо обработать, чтобы устранить подобные дефекты. Для этого была разработана программа на языке Python под названием «Preparation.py» (см. приложение №1).

Исходные данные программно преобразуются к промежуточному формату – формат 2. Данные в таком формате записываются в новые файлы. Графа с датой и временем бара <DTYYYYMMDD TIME> преобразуется в графу <Datetime> в виде целого числа минут, прошедших от некоторой условной стартовой даты (было принято Start = 01.01.2001 00:00). Остальные поля остаются неизменны и аналогичны полям в формате 1. Пример данных для интервала 240 минут в формате 2 показан в таблице 2.5.

Таблица 2.5

Данные для интервала 240 минут в формате 2

Datetime	OPEN	HIGH	LOW	CLOSE	VOL
6219600	1.29195	1.29363	1.29179	1.29286	2488
6219840	1.29285	1.29342	1.28859	1.28967	6176
6220080	1.2897	1.29154	1.28886	1.28969	7802
6220320	1.28972	1.2923	1.28847	1.28954	6904
6220560	1.28942	1.29067	1.28913	1.29047	1767

2) Далее происходит устранение “дыр” во времени таким образом, чтобы шаг времени был равен интервалу.

3) Чтобы исключить влияние временного разрыва в выходные и праздничные дни, предполагается работать исключительно с недельными интервалами. Выполняется исключение избыточных отсчетов из временного ряда и добавление недостающих с помощью линейной интерполяции по формуле:

(2.1)

где – это предыдущее время бара,

– это текущее время бара,

x – время, которые необходимо прибавить, чтобы получить нужный отсчет,

– предыдущее значение одного из полей *Open, High, Low, Close, Vol*,

– текущее значение одного из полей *Open, High, Low, Close, Vol*,

y – значение одного из полей *Open, High, Low, Close, Vol*, которое получим нужный интервал времени.

4) Временные ряды в чистом виде зачастую не соответствуют нормальному закону распределения и не являются стационарными, что мешает корректному анализу.

В данной работе предлагается не брать для анализа сами значения котировок *C(t)*. Так как эти изменения очень часто «гораздо меньше по амплитуде, чем сами котировки, между последовательными значениями курсов имеется большая корреляция – наиболее вероятное значение курса в следующий момент равно его предыдущему значению» [13].

Наличие таких статистических взаимосвязей приводит к некорректному анализу данных. Для устранения подобного рода корреляций рекомендуется вычислить изменения котировок по формуле

(2.2)

То есть взять ряд первых разностей, который является нормальным и стационарным (обоснование см. в разделе 3).

Для построения линейного графика цены в большинстве случаев используется цена закрытия (CLOSE). На рисунках 2.6-2.11 поочередно изображены графики цен закрытия CLOSE в формате 3 и графики первых разностей этой цены с интервалами 1 минута, 60 и 240 минут.

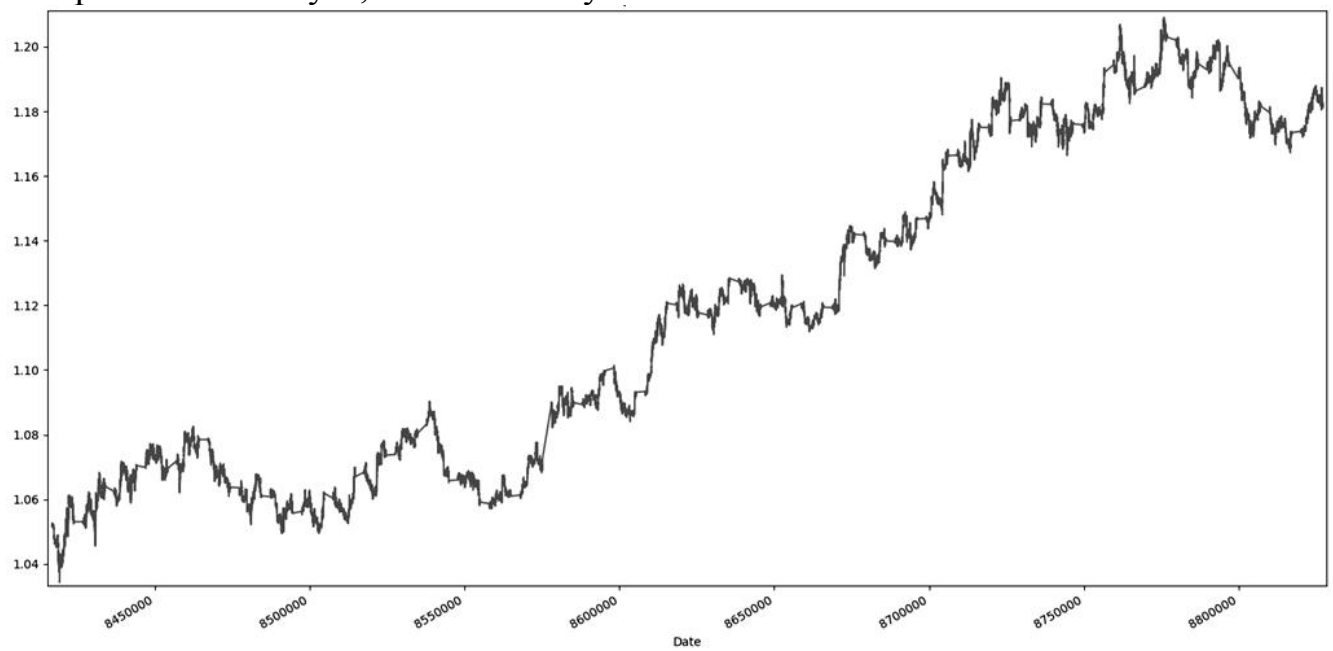

Рисунок 2.6 – График цены закрытия для интервала 1 минута

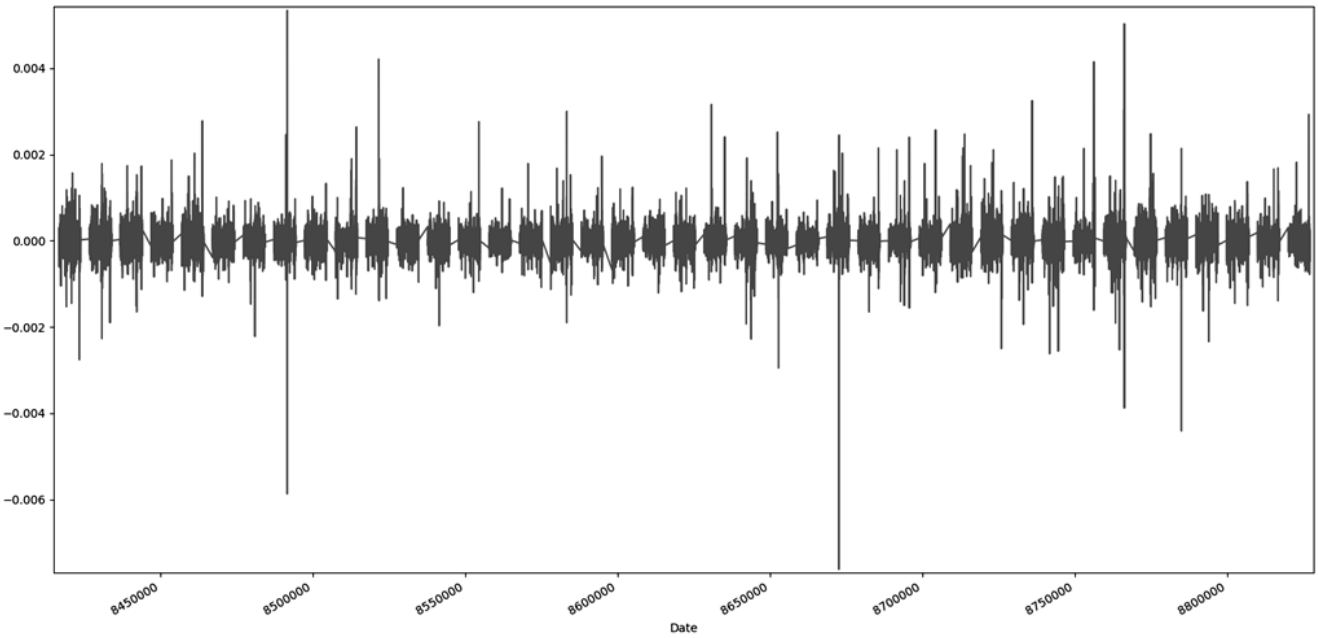

Рисунок 2.7 – График первых разностей
цены закрытия для интервала 1 минута

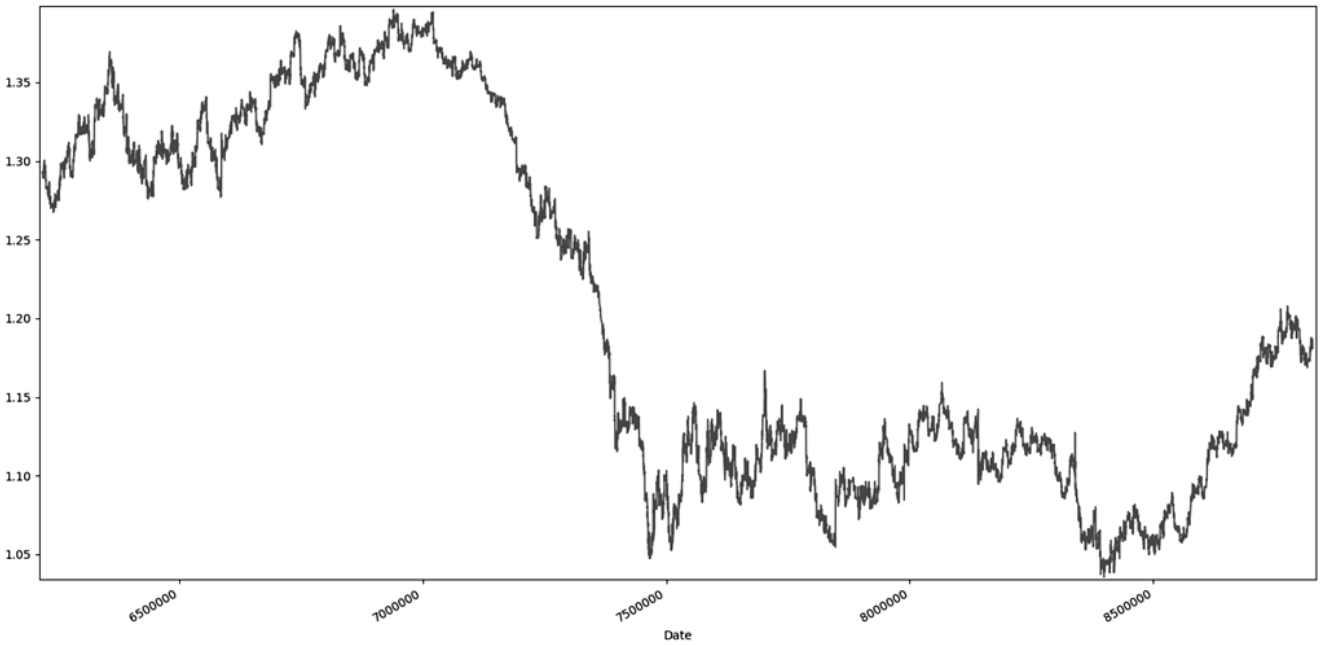

Рисунок 2.8 – График цены закрытия для интервала 60 минут

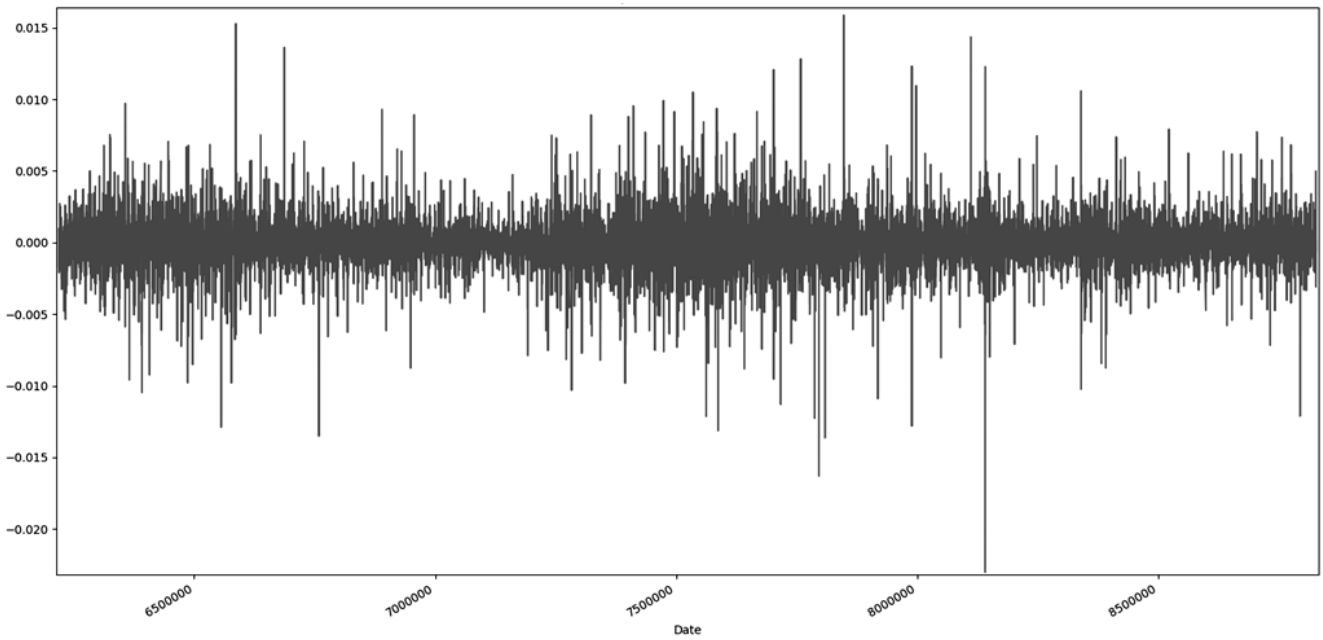

Рисунок 2.9 – График первых разностей
цены закрытия для интервала 60 минут

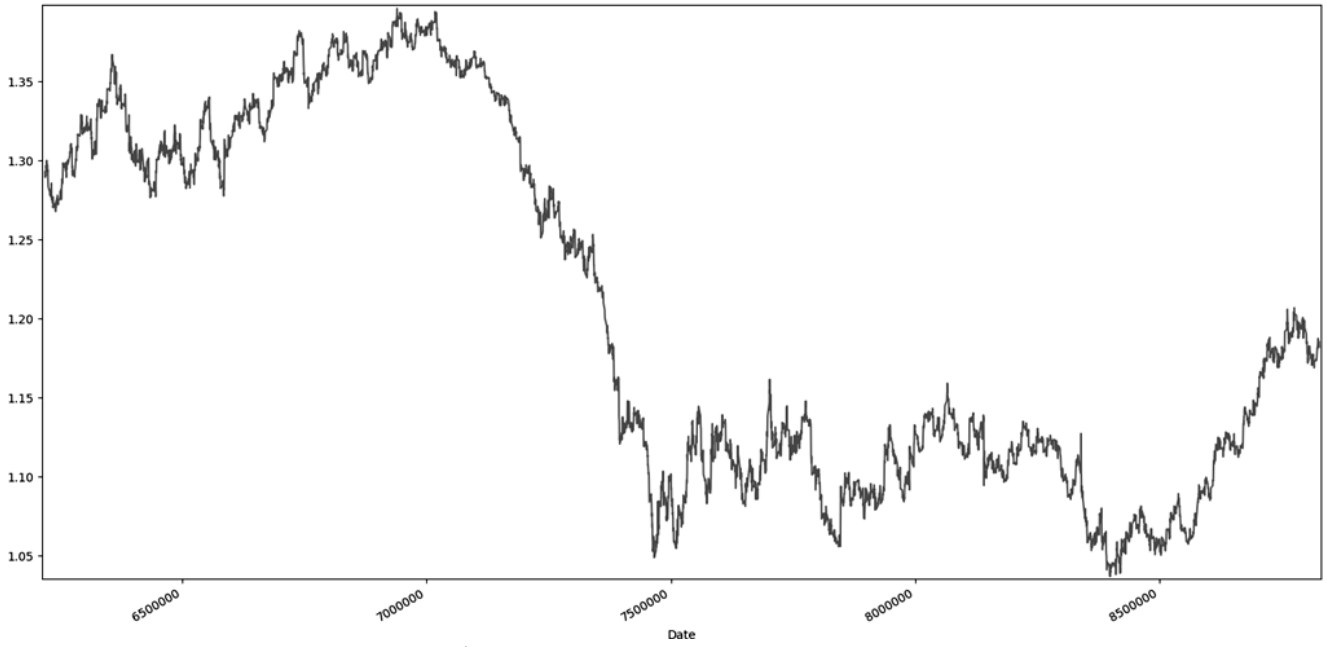

Рисунок 2.10 – График цены закрытия для интервала 240 минут

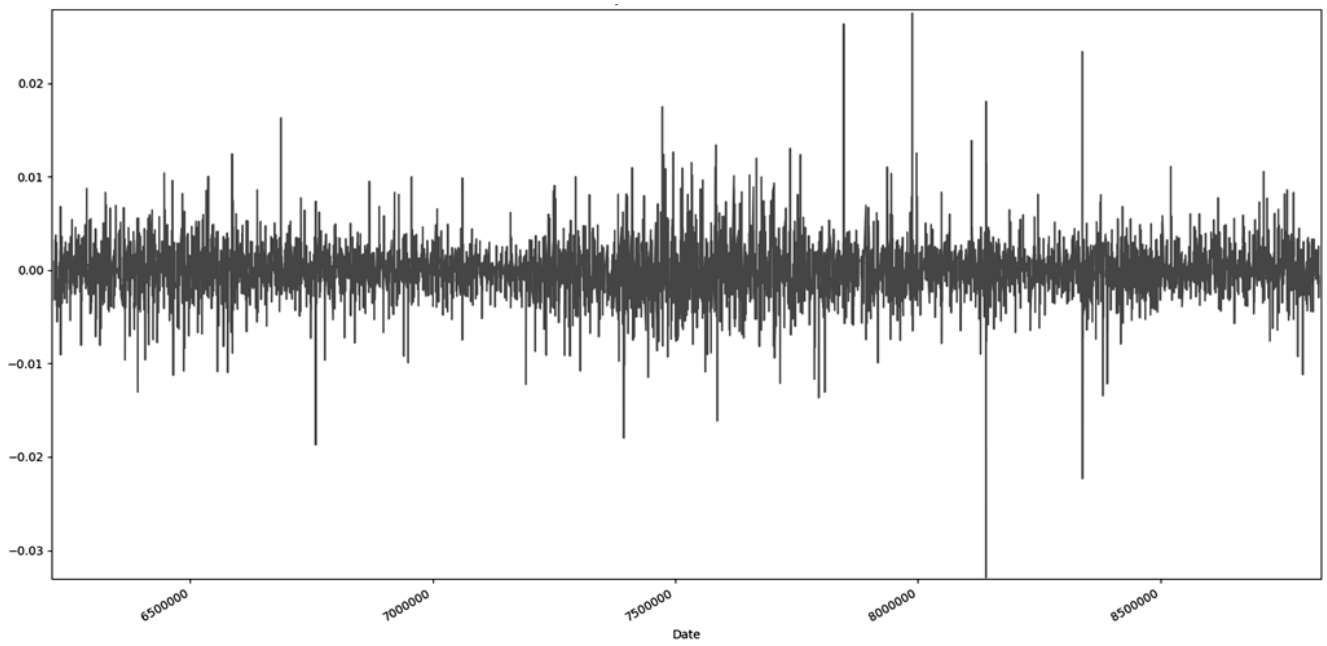

Рисунок 2.11 – График первых разностей
цены закрытия для интервала 240 минут

Полученные величины (первые разности временных рядов) добавляются в конец записи для каждого временного отсчета.

После всех операций, проделанных в пунктах 2-4, выполняется преобразование из формата 2, к окончательному виду формата 3, пример показан в таблице 2.6.

Таблица 2.6

Данные для интервала 240 минут в формате 3

Date time	OPEN	HIGH	LOW	CLOSE	VOL	dOPEN	dHIGH	dLOW	dCLOSE	HIGH-LOW
6219600	1.29195	1.29363	1.29179	1.29286	2488	-0.00212	-0.00084	0.00035	0.00091	0.00184
6219840	1.29285	1.29342	1.28859	1.28967	6176	0.0009	-0.00021	-0.0032	-0.00319	0.00483
6220080	1.2897	1.29154	1.28886	1.28969	7802	-0.00315	-0.00188	0.00027	0.00002	0.00268
6220320	1.28972	1.2923	1.28847	1.28954	6904	0.00002	0.00076	-0.00039	-0.00015	0.00383
6220560	1.28942	1.29067	1.28913	1.29047	1767	-0.0003	-0.00163	0.00066	0.00093	0.00154

Данные в формате 3 сохраняются в отдельные файлы.

5) Создание 3-мерных списков z[i][j][k], где i-номер недели, j-номер записи от начала недели, k-номер элемента, и запись полученных списков по всем периодам в новые файлы. Для этого была написана программа «d3List.py» (см. приложение №2).

В выходные дни рынок не работает, но события в мире происходят, что может привести к скачкам на границе недели. Для того чтобы эти скачки не искажали картину, выполнено разбиение данных на недельные отрезки и исключение неполных недель, в которых число записей меньше минимального количества записей в неделе в зависимости от интервала.

Выводы по разделу

В данном разделе описана подготовка выгруженных данных котировок валютной пары EURUSD рынка Forex для последующего анализа. Данные представлены в виде csv-файлов. Значения котировок выгружены с различными интервалами времени – 1, 5, 15, 30, 60 и 240 минут – в среднем за 5 лет.

При просмотре исходных данных были обнаружены пропуски и избыточные отсчеты в последовательности данных. Для того чтобы не усложнять анализ из-за неравномерности отсчетов, была написана программа (см приложение №1), устраняющая подобные дефекты.

Рассчитаны первые разности временных рядов, так как в чистом виде исходные данные зачастую не соответствуют нормальному закону распределения и не являются стационарными.

Созданы трехмерные списки с количеством недель в выбранном периоде, количеством отсчетов в неделе и непосредственно с самими отсчетами (см. приложение №2). Это необходимо для того, чтобы возможные скачки курса при переходе от недели к неделе (во время выходных и праздничных дней курс не меняется) не искажали картину. Также исключены неполные недели с минимальным количеством записей.

Результаты каждого этапа преобразования данных записываются в новые файлы.

3 КЛАССИЧЕСКИЙ СТАТИСТИЧЕСКИЙ АНАЛИЗ ВРЕМЕННЫХ РЯДОВ

Целью анализа статистических характеристик котировок на рынке FOREX, и, в частности, котировок EURUSD, является соответствие временного ряда нормальному закону распределения и определение его стационарности.

3.1 Описательные статистики

Следующий этап после подготовки исходных данных предполагает получение из них общих характеристик, закономерностей, и их последующую интерпретацию. Исходный массив данных — не самый удобный объект для формулировки предположений, данные необходимо охарактеризовать набором удобно интерпретируемых с точки зрения статистики параметров [27]. Таковыми являются:

1. среднее значение,
2. медиана,
3. максимальное и минимальное значения,
4. дисперсия,
5. стандартное отклонение,
6. коэффициент асимметрии,
7. эксцесс.

3.1.1 Основные понятия

Среднее (mean, average) или *среднее выборки* представляет собой арифметическое среднее всех значений массива:

(3.1)

$$\bar{x}$$

Среднее значение также может называться *математическим ожиданием*. Применимо не для всех распределений и наиболее популярно для нормальных распределений, для которых совпадает с медианой [28].

Медиана – это число, которое является серединой выборки и делит все наблюдения на две части: с одной стороны, все наблюдения меньше по значению медианы, а с другой – больше. Медиана существует для любого распределения и не чувствительна к выбросам [29]. Если средняя равна (близка по значению) медиане, то это один из признаков нормального закона распределения, то есть плотность распределения симметрична.

Дисперсия (variance) – это сумма квадратов отклонений каждого значения в массиве от среднего, деленная на размер выборки минус 1:

(3.2)

$$\overline{\qquad\qquad}$$

Дисперсия выборки равна 0, только в том случае, если все значения равны между собой и, соответственно, равны среднему значению. Чем больше величина дисперсии, тем больше разброс значений в массиве относительно среднего.

Стандартное отклонение выборки (Standard Deviation) – это мера того, насколько широко разбросаны значения в выборке относительно их среднего. Корень квадратный из дисперсии и есть среднеквадратичное (стандартное) отклонение:

(3.3)

Стандартное отклонение и дисперсия не устойчивы к выбросам.

Вычисление *асимметрии* и *эксцесса* позволяет установить симметричность распределения случайной величины *X* относительно математического ожидания

.

Коэффициент асимметрии (skewness – скос) – это безразмерная величина, являющаяся показателем степени *несимметричности* кривой плотности распределения:

(3.5)

где μ – математическое ожидание;

– среднеквадратическое отклонение, цифра 3 означает третий центральный момент (т.е. μ_3 имеет размерность случайной величины в кубе), характеризующий асимметрию закона распределения случайной величины.

Коэффициент асимметрии нормального распределения равен нулю. Положительное значение коэффициента асимметрии означает, что распределение имеет длинный правый «хвост», а отрицательное – длинный левый (относительно среднего) [30, с. 150].

График плотности вероятностей при различных значениях асимметрии изображен на рисунке 3.1 ниже:

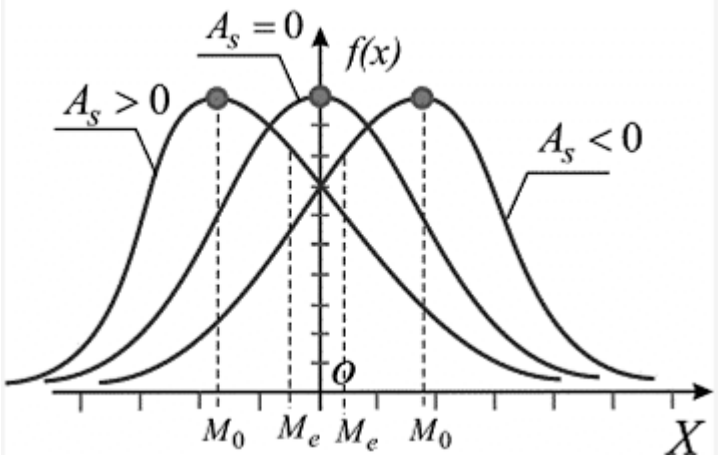

Рисунок 3.1 – График плотности вероятностей при различных значениях асимметрии

Другой величиной, характеризующей плотность распределения, является *коэффициент эксцесса*.

Коэффициент эксцесса (Kurtosis) — числовая мера, отображающая «остроту пика» распределения случайной величины, характеризует относительную остроконечность или сглаженность распределения по сравнению с нормальным распределением.

Коэффициент эксцесса распределения случайной величины *x* определяется формулой

(3.6)

где – среднее значение случайной величины *x*, – среднеквадратическое отклонение случайной величины *x*.

Данную формулу можно записать по-другому:

(3.7)

где μ – математическое ожидание;

– среднеквадратическое отклонение;

цифра 4 означает центральный момент четвертого порядка.

«Минус три» в конце формулы введено для того, чтобы коэффициент эксцесса стандартного нормального распределения был равен нулю.

Нормальному распределению плотности вероятности соответствует нулевой эксцесс . В случае если , то плотность вероятности имеет положительный эксцесс, что соответствует тому, что график плотности распределения имеет более острую и высокую вершину, а хвосты распределения находятся выше, чем у нормального распределения. Если хвосты распределения находятся ниже, чем у нормального распределения, а пик более низкий и плоский, то плотность вероятности имеет отрицательный эксцесс и [30, с. 151].

График плотности вероятностей при различных значениях эксцесса изображен на рисунке 3.2 ниже:

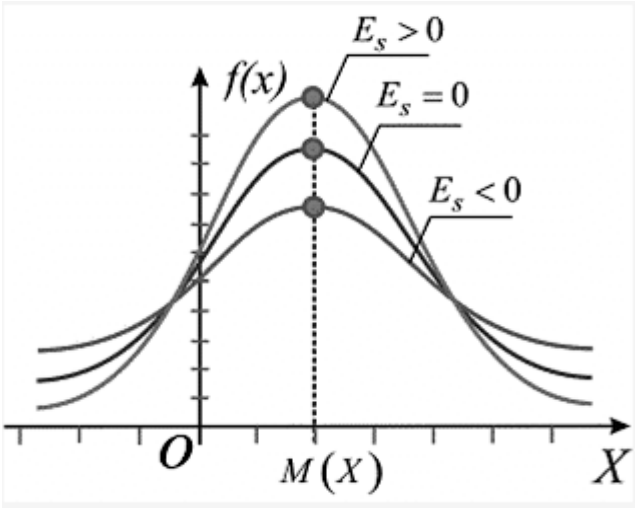

Рисунок 3.2 – График плотности вероятностей
при различных значениях эксцесса

Нахождение таких показателей, дающих максимально полное представление об имеющимся наборе данных, и составляет то, что называют описательной статистикой. Из статистики известно, что нормальное распределение имеет вид симметричной колоколообразной кривой, распространяющейся по всей числовой оси. Математическое ожидание и медиана стандартного нормального распределения равны $\mu=0$, а дисперсия равна $\sigma^2=1$. Кривая плотности вероятности симметрична относительно математического ожидания. Коэффициент асимметрии и эксцесс равны .

3.1.2 Расчет описательных статистик исходных данных

Для расчета описательных статистик была написана программа «DescriptionStatistics.py» (см приложение №3).

В таблицах 3.1-3.3 отображены описательные статистики для исходного временного ряда для интервалов 1 минута, 60 и 240 минут.

Таблица 3.1

Описательные статистики временного ряда
для интервала 1 минута

	OPEN	HIGH	LOW	CLOSE
Длина выборки	294589	294589	294589	294589
Среднее значение	1,1167	1,1168	1,1167	1,1167
Медиана	1,1154	1,1155	1,1153	1,1154
Минимальное значение	1,0343	1,0346	1,0340	1,0343
Максимальное значение	1,2091	1,2092	1,2088	1,2091
Дисперсия	0,0025	0,0025	0,0025	0,0025
Стандартное отклонение	0,0501	0,0501	0,0501	0,0501
Коэффициент асимметрии	0,2643	0,2646	0,2641	0,2644
Коэффициент эксцесса	-1,4433	-1,4431	-1,4434	-1,4433

Таблица 3.2

Описательные статистики временного ряда
для интервала 60 минут

	OPEN	HIGH	LOW	CLOSE
Длина выборки	30359	30359	30359	30359
Среднее значение	1,2050	1,2058	1,2042	1,2050
Медиана	1,1517	1,1527	1,1505	1,1517
Минимальное значение	1,0356	1,0370	1,0340	1,0356
Максимальное значение	1,3963	1,3993	1,3949	1,3963
Дисперсия	0,0129	0,0128	0,0129	0,0129
Стандартное отклонение	0,1134	0,1133	0,1135	0,1134
Коэффициент асимметрии	0,2551	0,2543	0,2560	0,2552
Коэффициент эксцесса	-1,5754	-1,5761	-1,5746	-1,5753

Таблица 3.3

Описательные статистики временного ряда
для интервала 240 минут

	OPEN	HIGH	LOW	CLOSE
Длина выборки	7199	7199	7199	7199
Среднее значение	1,2048	1,2067	1,2031	1,2048
Медиана	1,1512	1,1535	1,1483	1,1512
Минимальное значение	1,0369	1,0391	1,0340	1,0369
Максимальное значение	1,3963	1,3993	1,3937	1,3963
Дисперсия	0,0129	0,0128	0,0129	0,0129
Стандартное отклонение	0,1134	0,1132	0,1135	0,1134
Коэффициент асимметрии	0,2564	0,2549	0,2585	0,2568
Коэффициент эксцесса	-1,5746	-1,5759	-1,5727	-1,5742

По данным таблиц 3.1-3.3 можно сделать вывод о том, что исходные данные отклоняются от нормального распределения, так как:

1) математическое ожидание не равно нулю,

2) дисперсия не равна единице,

3) коэффициент асимметрии больше нуля, а значит имеется длинный правый «хвост»,

4) коэффициент эксцесса меньше нуля, следовательно, вершина более пологая в отличии от нормального распределения.

Для наглядности построим гистограммы нормального распределения котировок EURUSD по всем характеристикам (OPEN, HIGH, LOW, CLOSE) для интервалов 1 минута, 60 и 240 минут (рисунки 3.3-3.5).

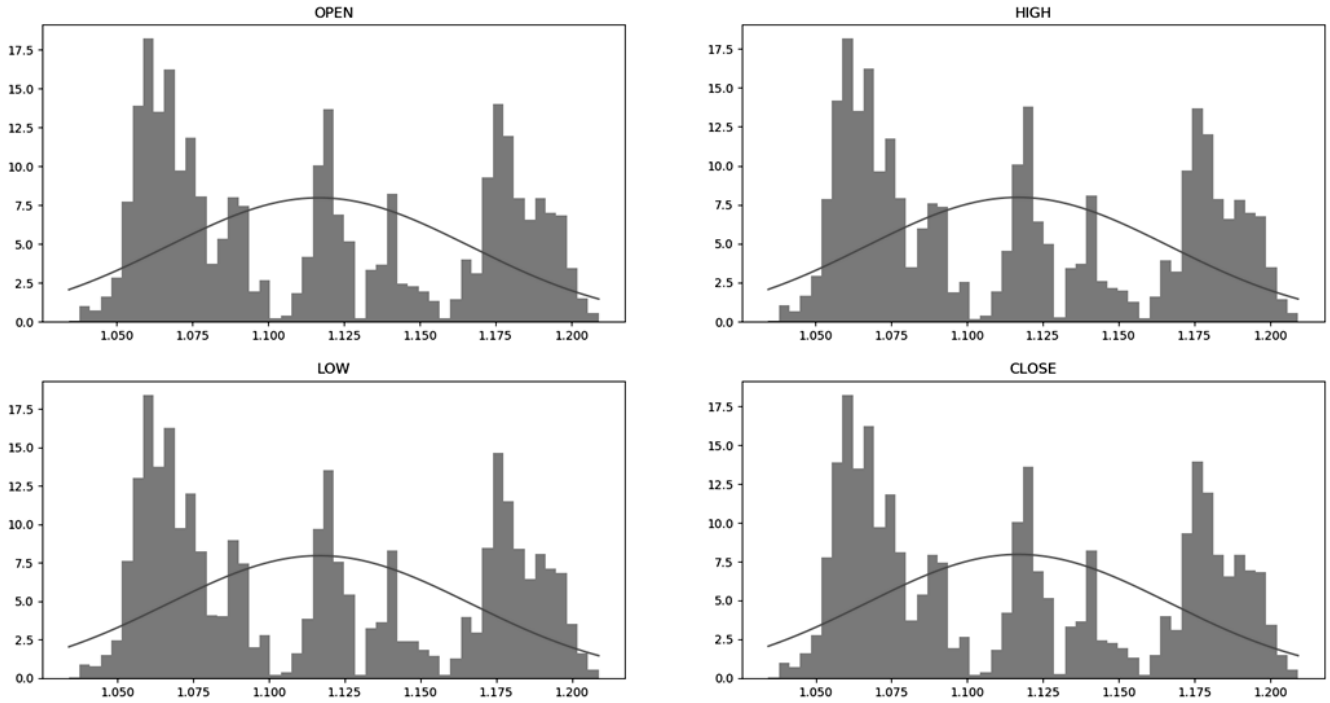

Рисунок 3.3 – Гистограмма котировок EURUSD для интервала 1 минута

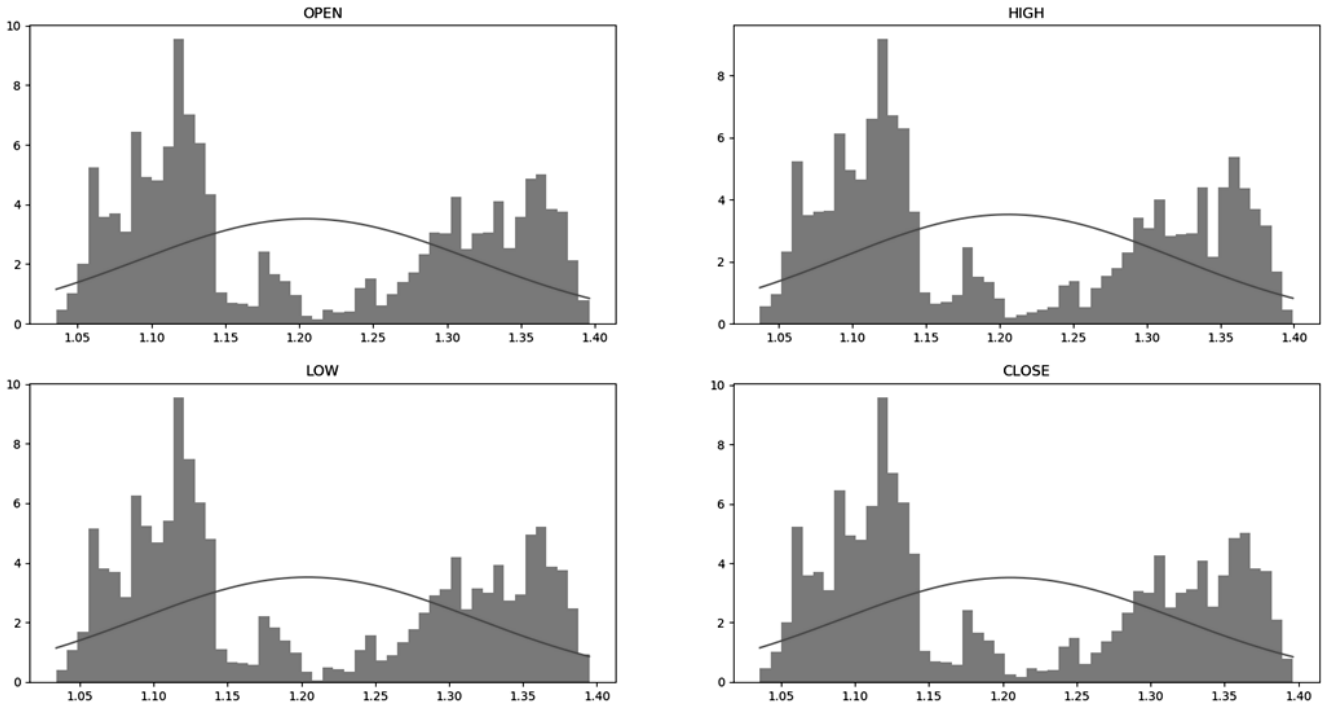

Рисунок 3.4 – Гистограмма котировок EURUSD для интервала 60 минут

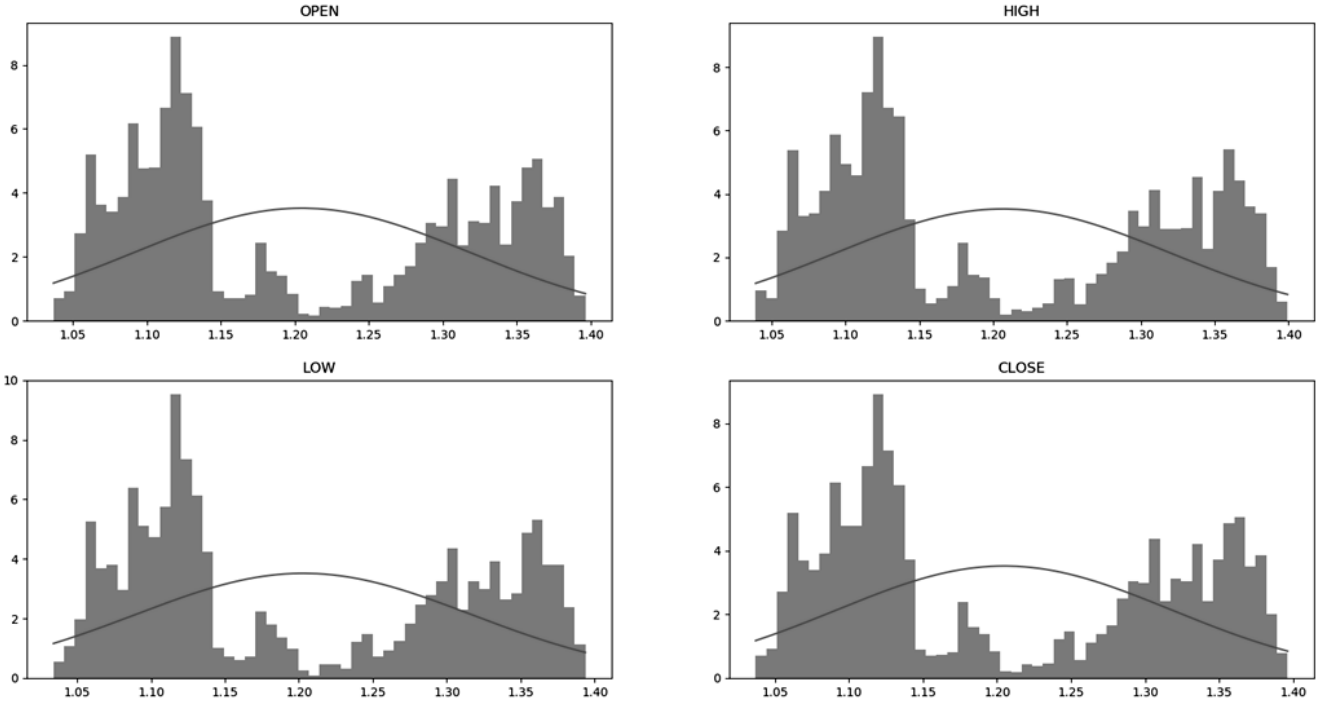

Рисунок 3.5 – Гистограмма котировок EURUSD для интервала 240 минут

Гистограмма показывает, какое количество раз встретилась конкретная цена в выбранном нами интервале. Рисунки 3.3-3.5 подтверждают наше предположение о том, что распределения исходных данных не являются нормальными, очень портит картину большое количество вершин, пологость и асимметричность.

Дополнительно проведем тест на нормальность Жарка-Бера [29] с нулевой гипотезой H0: распределение является нормальным при уровне значимости больше 0,05 (p>0,05).

Для проверки по критерию Жака-Бера используется тот факт, что у нормального распределения коэффициент асимметрии равен нулю, а эксцесс равен 3, отклонение этих величин от нормальных значений служит мерой отклонения распределения от нормального. Статистика Жарке-Бера вычисляется на основе коэффициентов асимметрии и эксцесса, т.е. учитываются лишь 2 характеристики нормального распределения:

— ——— (3.8)

где – коэффициент асимметрии;

– коэффициент эксцесса, n – количество наблюдений.

Python предоставляет доступ к библиотеке statsmodels, в которой существует функция jarque_bera(), возвращающая значения данной статистики:

- значение статистики Жарке-Бера (Jarke-Bera) с соответствующим значением вероятности (Probability),
- коэффициенты асимметрии (Skewness),
- коэффициенты эксцесса (Kurtosis) (считается без вычета 3).

Результаты теста Жарка-Бера для временных рядов EURUSD для интервалов 1 минута, 60 и 240 минут приведены в таблицах 3.4-3.6.

Таблица 3.4

Результат теста Жарка-Бера для интервала 1 минута

	OPEN	HIGH	LOW	CLOSE
Статистика теста JB	28998,5785	29000,3639	28998,9016	28999,4294
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	0,2643	0,2646	0,2641	0,2644
Эксцесс	1,5567	1,5569	1,5566	1,5567

Таблица 3.5

Результат теста Жарка-Бера для интервала 60 минут

	OPEN	HIGH	LOW	CLOSE
Статистика теста JB	3468,7571	3469,5094	3467,7766	3468,7380
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	0,2551	0,2543	0,2560	0,2552
Эксцесс	1,4246	1,4239	1,4254	1,4247

Таблица 3.6

Результат теста Жарка-Бера для интервала 240 минут

	OPEN	HIGH	LOW	CLOSE
Статистика теста JB	822,5739	822,8589	822,0281	822,4602
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	0,2564	0,2549	0,2585	0,2568
Эксцесс	1,4254	1,4241	1,4273	1,4258

По результатам теста Жарка-Бера также следует вывод о несоответствии исходного временного ряда нормальному распределению, так как:

- асимметрия не равна к нулю, а ее положительное значение означает, что распределение имеет длинный правый хвост;
- эксцесс меньше 3, что говорит о плосковершинности;
- поскольку величины тестовой статистики Жарка Бера очень велики, а уровень значимости оказался равен нулю, то, следовательно, нужно отвергнуть гипотезу о нормальном распределении исходного ряда.

3.1.3 Расчет описательных статистик первых разностей временных рядов

Убедившись в том, что исходные данные не соответствуют нормальному распределению, перейдем к анализу характеристик первых разностей временных рядов.

В таблицах 3.7-3.9 отображены описательные статистики для первых разностей временных рядов для интервалов 1 минута, 60 и 240 минут.

Таблица 3.7

Описательные статистики первых разностей временного ряда для интервала 1 минута

	dOPEN	dHIGH	dLOW	dCLOSE
Длина выборки	294589	294589	294589	294589
Среднее значение	0,0000004	0,0000004	0,0000004	0,0000004
Медиана	0,0000000	0,0000000	0,0000000	0,0000000
Минимальное значение	-0,0076000	-0,0060200	-0,0078200	-0,0076200
Максимальное значение	0,0050100	0,0042800	0,0075800	0,0053400
Дисперсия	0,0000000	0,0000000	0,0000000	0,0000000
Стандартное отклонение	0,0001415	0,0001304	0,0001310	0,0001426
Коэффициент асимметрии	0,0897148	1,0452451	-0,7824385	0,0367625
Коэффициент эксцесса	69,1171277	82,7106622	165,2657955	84,1288152

Таблица 3.8

Описательные статистики первых разностей временного ряда для интервала 60 минут

	dOPEN	dHIGH	dLOW	dCLOSE
Длина выборки	30359	30359	30359	30359
Среднее значение	-0,0000001	-0,0000013	0,0000006	-0,0000008
Медиана	0,0000000	-0,0000400	0,0000300	0,0000000
Минимальное значение	-0,0230100	-0,0206300	-0,0183900	-0,0230100
Максимальное значение	0,0158700	0,0206900	0,0142600	0,0158700
Дисперсия	0,0000017	0,0000016	0,0000015	0,0000017
Стандартное отклонение	0,0012965	0,0012637	0,0012070	0,0012951
Коэффициент асимметрии	-0,1449233	0,5821715	-0,5809222	-0,1448078
Коэффициент эксцесса	17,6184197	26,5186379	21,6802268	17,5893084

Таблица 3.9

Описательные статистики первых разностей
временного ряда для интервала 240 минут

	dOPEN	dHIGH	dLOW	dCLOSE
Длина выборки	7199	7199	7199	7199
Среднее значение	-0,0000010	-0,0000019	-0,0000005	-0,0000039
Медиана	0,0000100	-0,0000900	0,0000700	0,0000100
Минимальное значение	-0,0329200	-0,0194100	-0,0357100	-0,0329200
Максимальное значение	0,0275400	0,0375400	0,0268400	0,0275000
Дисперсия	0,0000069	0,0000068	0,0000063	0,0000068
Стандартное отклонение	0,0026210	0,0026077	0,0025039	0,0026171
Коэффициент асимметрии	0,0602651	0,8723997	-0,3254407	0,0605290
Коэффициент эксцесса	12,4688864	15,8807100	15,2266658	12,5109544

Значения в таблицах 3.7-3.9 говорят о том, что первые разности исходных данных не соответствуют нормальному распределению, так как коэффициент эксцесса значительно больше нуля, следовательно, вершина имеет «острый пик».

Построим гистограммы нормального распределения котировок EURUSD, взяв их первые разности по всем характеристикам (OPEN, HIGH, LOW, CLOSE), для интервалов 1 минута, 60 и 240 минут (рисунки 3.6-3.8).

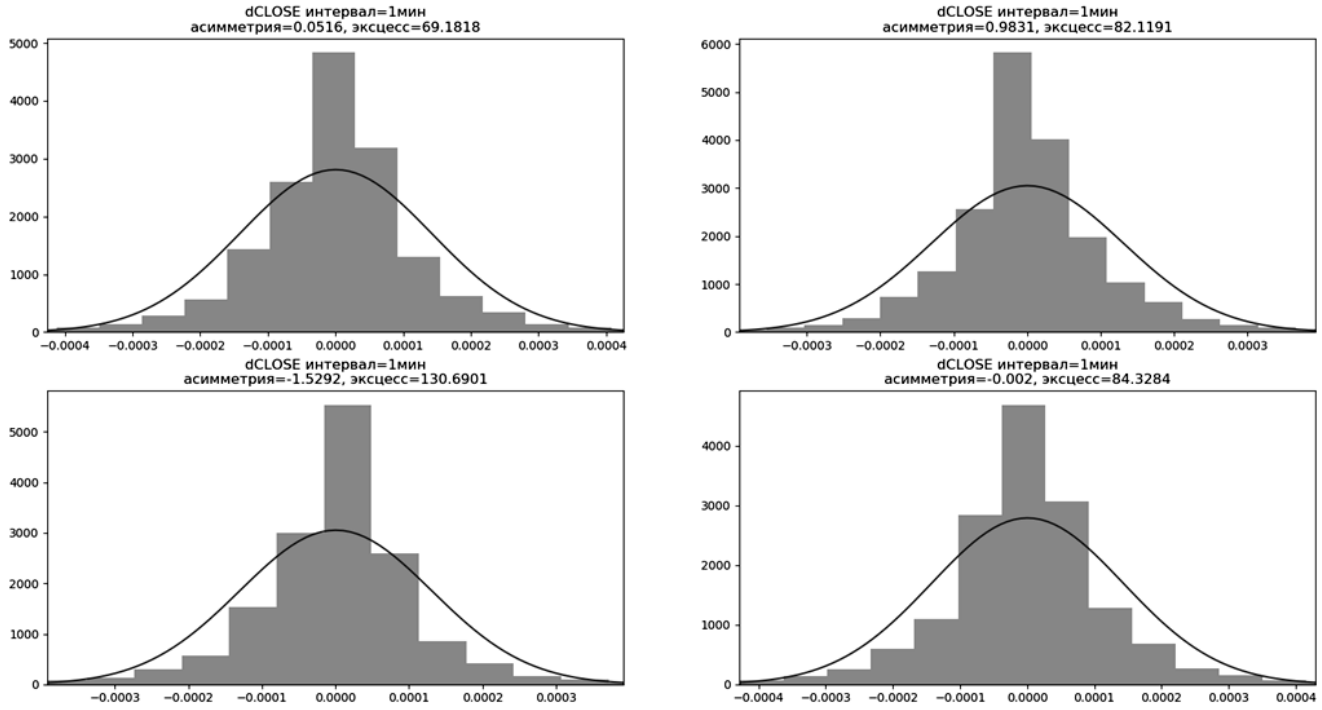

Рисунок 3.6 – Гистограмма первых разностей котировок EURUSD
для интервала 1 минута

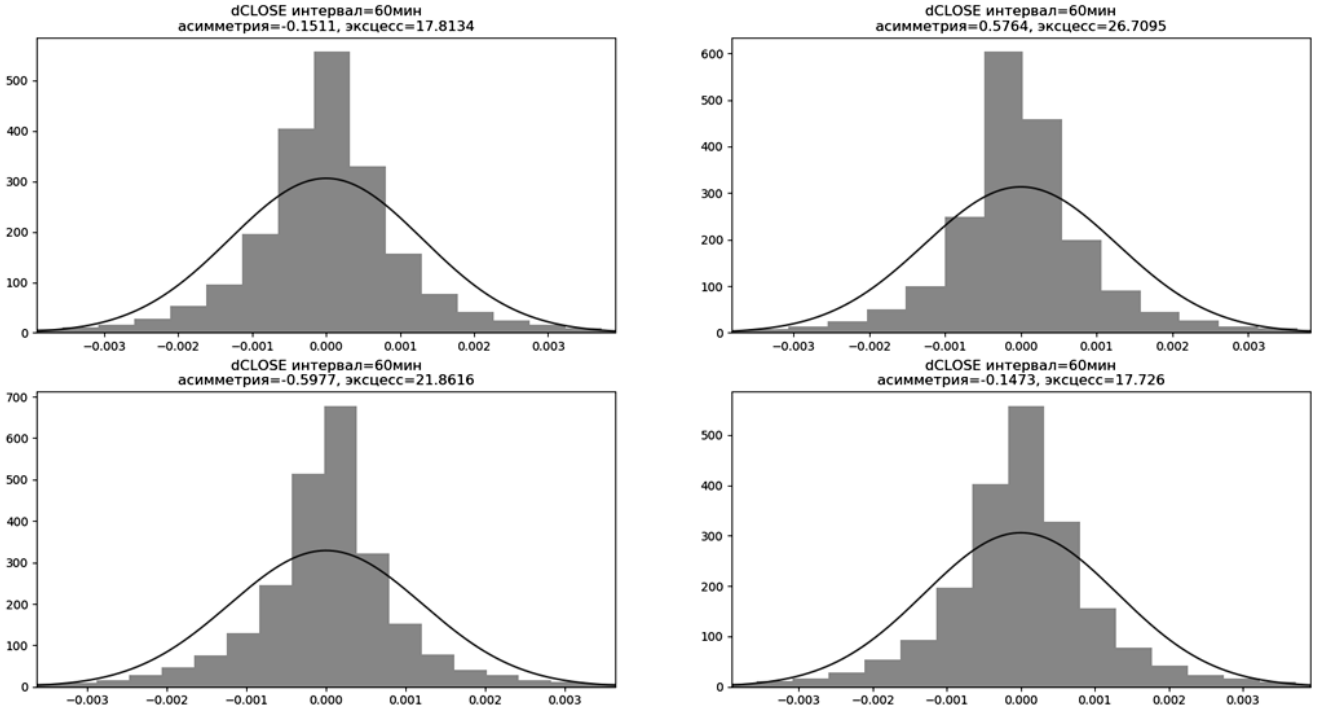

Рисунок 3.7 – Гистограмма первых разностей котировок EURUSD для интервала 60 минут

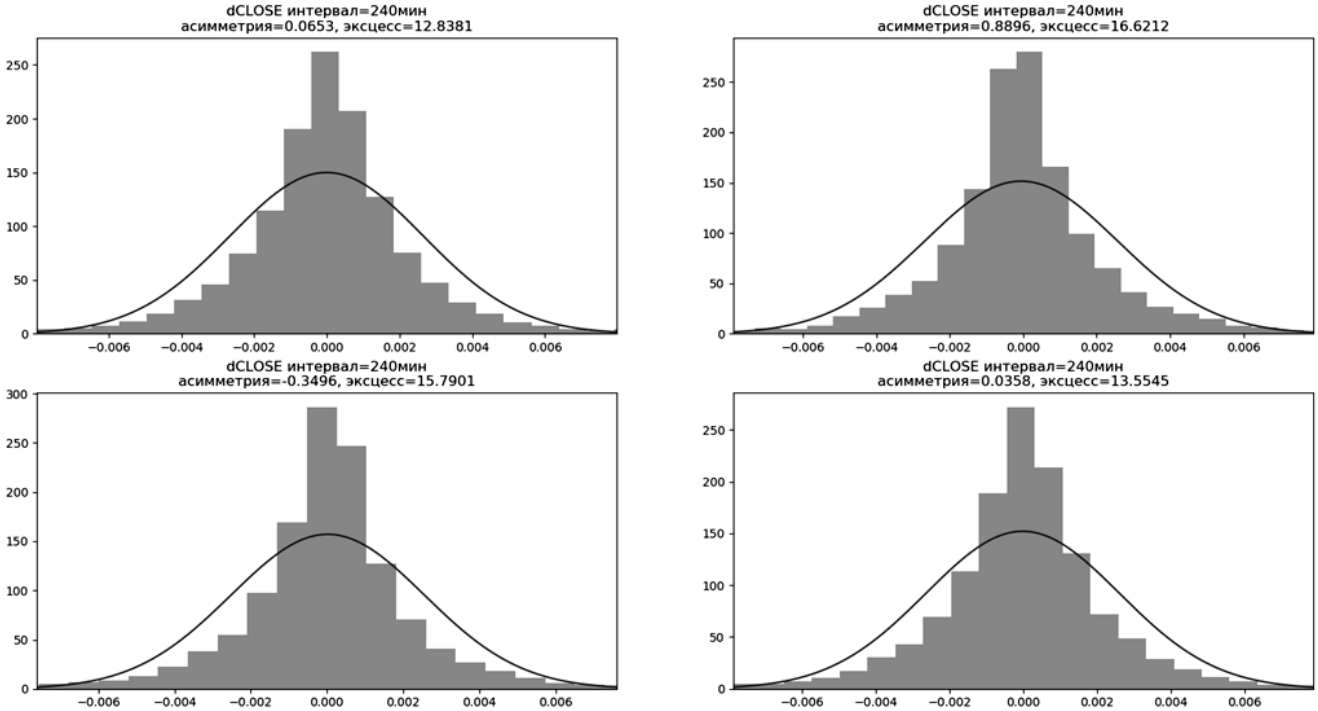

Рисунок 3.8 – Гистограмма первых разностей котировок EURUSD для интервала 240 минут

На гистограммах можно заметить наличие значительных выбросов, которые могут быть вызваны некоторыми фундаментальными событиями. Если удалить все значения, превосходящие , то практически ничего не меняется, кроме

коэффициента эксцесса, который уменьшается примерно в 10 раз (рисунки 3.9-3.11).

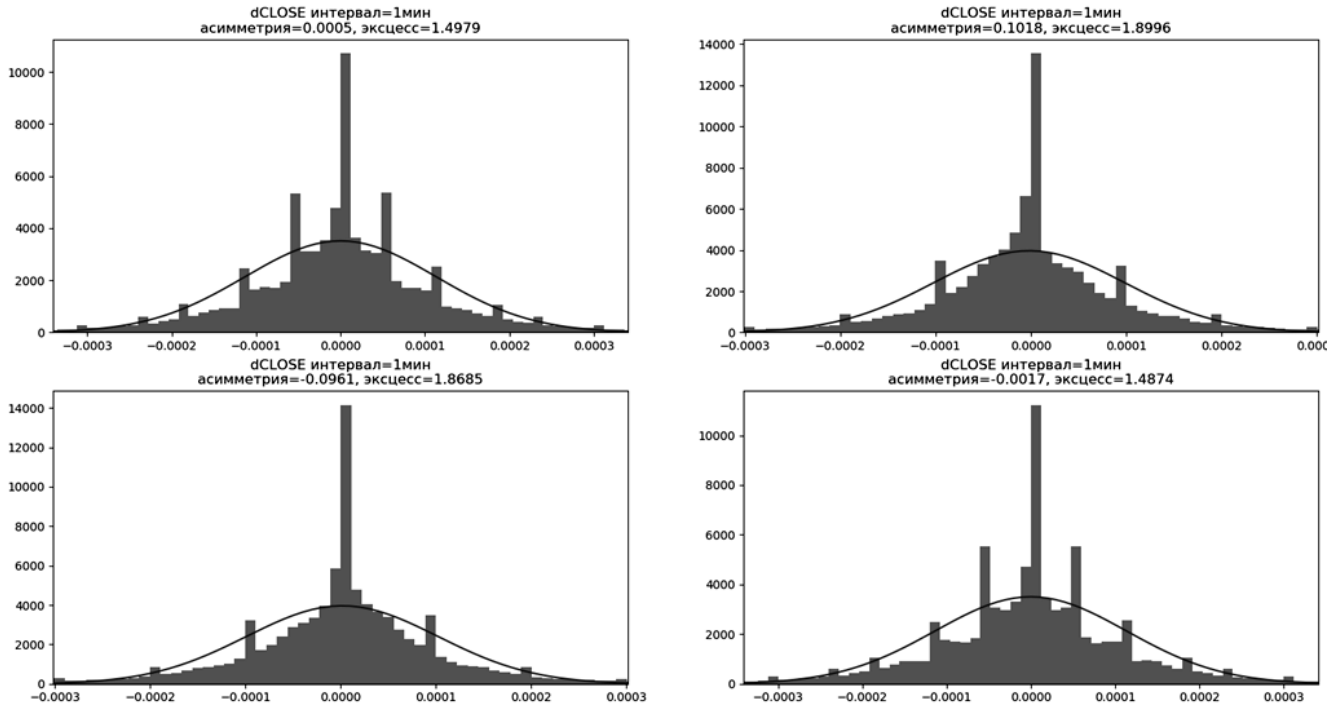

Рисунок 3.9 – Гистограмма первых разностей котировок EURUSD для интервала 1 минута без значений за пределами

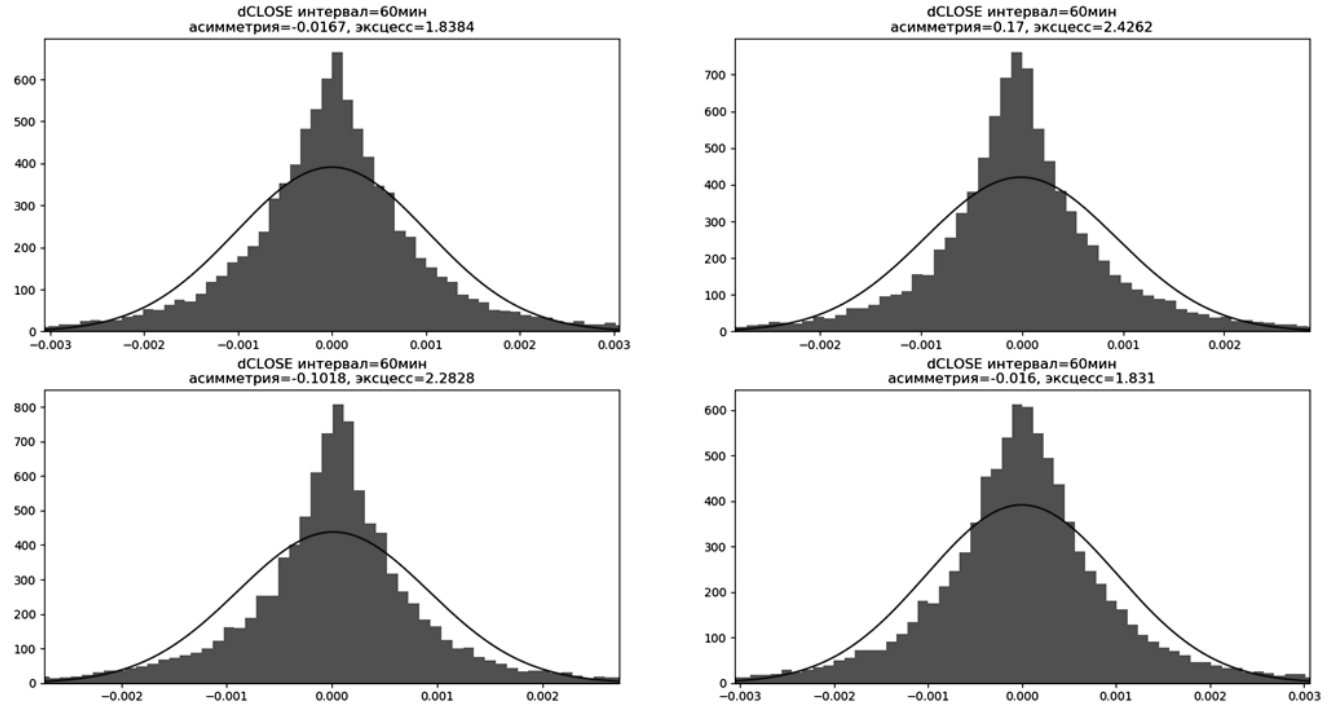

Рисунок 3.10 – Гистограмма первых разностей котировок EURUSD для интервала 60 минут без значений за пределами

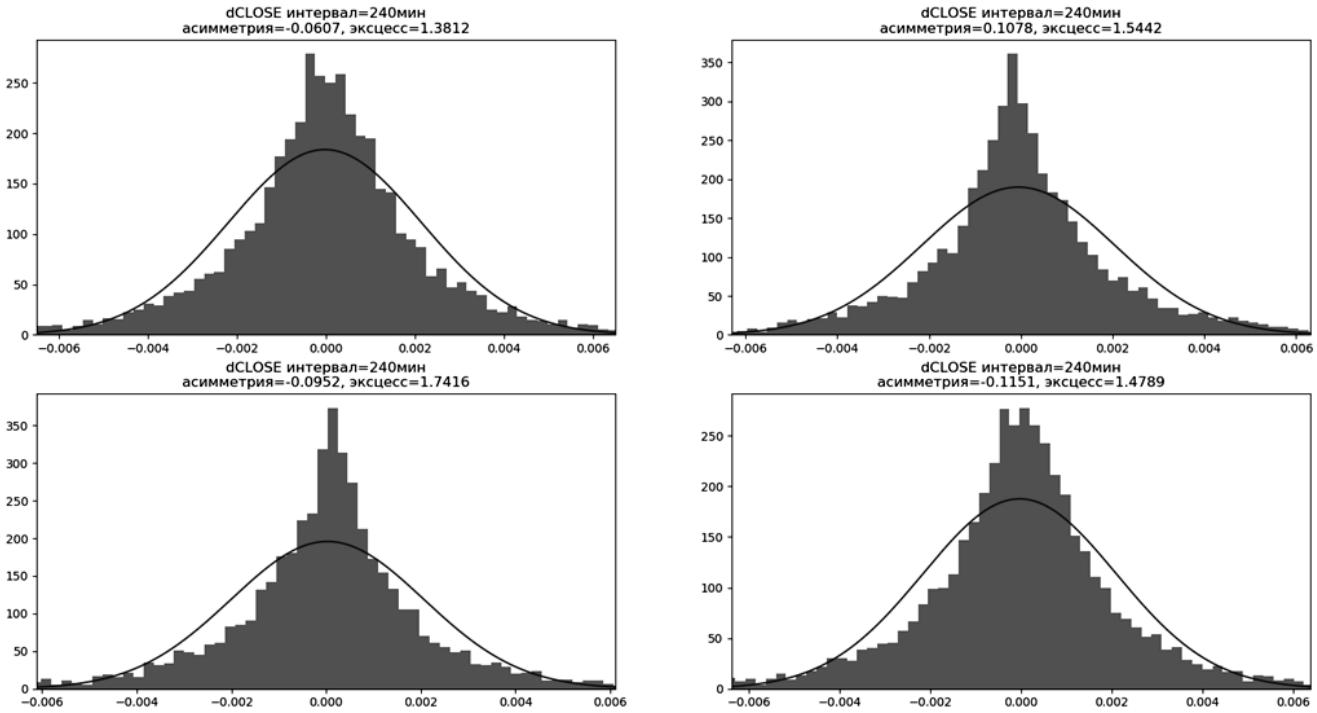

Рисунок 3.11 – Гистограмма первых разностей котировок EURUSD для интервала 240 минут без значений за пределами

По результатам построения гистограмм можно сделать вывод о том, что первые разности временных рядов EURUSD не вполне соответствуют нормальному распределению. Результаты теста Жарка-Бера для первых разностей временных рядов EURUSD для интервалов 1 минута, 60 и 240 минут, приведенные в таблицах 3.10-3.12, подтверждают это.

Таблица 3.10

Результаты теста Жарка-Бера для интервала 1 минута

	dOPEN	dHIGH	dLOW	dCLOSE
Статистика теста JB	58638057,4522	84024439,4551	335281953,4076	86875068,9129
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	0,0897	1,0452	-0,7824	0,0368
Эксцесс	72,1171	85,7107	168,2658	87,1288

Таблица 3.11

Результаты теста Жарка-Бера для интервала 60 минут

	dOPEN	dHIGH	dLOW	dCLOSE
Статистика теста JB	392760,3602	891281,8629	596278,7340	391463,6778
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	-0,1449	0,5822	-0,5809	-0,1448
Эксцесс	20,6184	29,5186	24,6802	20,5893

Таблица 3.12

Результаты теста Жарка-Бера для интервала 240 минут

	dOPEN	dHIGH	LOW	dCLOSE
Статистика теста JB	46639,8178	76561,7476	69672,8218	46955,0682
Вероятность	0,0000	0,0000	0,0000	0,0000
Асимметрия	0,0603	0,8724	-0,3254	0,0605
Эксцесс	15,4689	18,8807	18,2267	15,5110

3.2 Анализ стационарности котировок

Перед тем, как перейти к моделированию, прогнозированию и классификации, стоит сказать о таком важном свойстве временных рядов, как стационарность.

Под стационарностью понимают свойство процесса не менять своих статистических характеристик с течением времени, а именно среднее и дисперсия наблюдений постоянны, ковариационная функция не должна зависеть от времени (должна зависеть только от расстояния между наблюдениями). Также не должны присутствовать тренд и сезонность в рядах.

По стационарному ряду можно легко построить прогноз, так как мы полагаем, что его будущие статистические характеристики не будут отличаться от наблюдаемых текущих. Большинство моделей временных рядов, так или иначе, моделируют и предсказывают эти характеристики (например, математическое ожидание или дисперсию), поэтому в случае нестационарности исходного ряда предсказания окажутся неверными.

Классический анализ временных рядов и методы прогнозирования связаны с созданием стационарных данных временных рядов путем выявления и устранения тенденций и устранения сезонных эффектов.

Реальные котировки на рынке Forex не являются стационарными, а имеют следующие отклонения:

- наличие трендов, порождаемых зависимостью между наблюдениями во времени. Наличие зависимости является характерной чертой, в частности, котировок валют, и вообще экономических наблюдений;
- наличие цикличности;
- наличие переменной дисперсии.

Существует несколько способов проверить, является ли временной ряд стационарным или нет:

1. **Посмотреть на графики**. Можно просмотреть график временных рядов данных и визуально проверить, есть ли какие-либо очевидные тенденции или сезонность.
2. **Сводная статистика**. Можно просмотреть сводную статистику для данных на наличие сезонности или случайных выбросов и проверить очевидные или существенные различия.
3. **Статистические тесты**. Можно использовать статистические тесты, чтобы проверить, соблюдены ли ожидания стационарности или были ли они нарушены.

Существует несколько тестов на стационарность временных рядов, основными из которых являются тесты на наличие единичного корня (unitroottest). Временной ряд имеет единичный корень, или порядок интеграции один, если его первые разности образуют стационарный ряд. Наиболее известным является тест Дики-Фуллера [31].

- **Нулевая гипотеза (H0)**: На *N*-процентном (*N*=1%, 5%, 10%) уровне значимости временной ряд *имеет единичный корень*, то есть он *нестационарен*, если значение полученной статистики больше критического ( ).
- **Альтернативная гипотеза (H1)**: нулевая гипотеза отклоняется на *N*-процентном (*N*=1%, 5%, 10%) уровне значимости при ; предполагается, что временной ряд *не имеет единичного корня*, что означает, что он является *стационарным*. Он не имеет структуры, зависящей от времени.

Будет логичным начать проверку на стационарность с помощью теста Дики-Фуллера с ряда первых разностей временных рядов валютной пары EURUSD для интервалов 1 минута, 60 и 240 минут.

Python предоставляет доступ к библиотеке statsmodels, в которой существует функция adfuller(), возвращая значения статистики Дикки-Фуллера для различны:

- значение статистики Дики-Фуллера с соответствующим значением вероятности (Probability),
- критические значения при разных значениях вероятности 1%, 5% и 10%.

Результаты теста Дики-Фуллера для временных рядов EURUSD для интервалов 1 минута, 60 и 240 минут приведены в таблицах 3.13-3.15.

Таблица 3.13

Результаты теста Дики-Фуллера для интервала 1 минута

	dOPEN	dHIGH	dLOW	dCLOSE
Статистика теста DF	-60,0031	-62,0769	-57,3823	-59,4933
Вероятность	0,0000	0,0000	0,0000	0,0000
1%	-3,4304	-3,4304	-3,4304	-3,4304
5%	-2,8615	-2,8615	-2,8615	-2,8615
10%	-2,5668	-2,5668	-2,5668	-2,5668

Таблица 3.14

Результаты теста Дики-Фуллера для интервала 60 минут

	dOPEN	dHIGH	dLOW	dCLOSE
Статистика теста DF	-79,3185	-25,2055	-25,1401	-79,2832
Вероятность	0,0000	0,0000	0,0000	0,0000
1%	-3,4306	-3,4306	-3,4306	-3,4306
5%	-2,8616	-2,8616	-2,8616	-2,8616
10%	-2,5668	-2,5668	-2,5668	-2,5668

Таблица 3.15

Результаты теста Дики-Фуллера для интервала 240 минут

	dOPEN	dHIGH	dLOW	dCLOSE
Статистика теста DF	-61,8171	-15,1291	-14,2853	-85,6094
Вероятность	0,0000	0,0000	0,0000	0,0000
1%	-3,4313	-3,4313	-3,4313	-3,4313
5%	-2,8619	-2,8619	-2,8619	-2,8619
10%	-2,5670	-2,5670	-2,5670	-2,5670

Результаты теста показали статистическое значение Дики-Фуллера во всех случаях меньше критических значений. Чем более негативна эта статистика, тем больше вероятность отклонения нулевой гипотезы. Поэтому можно утверждать, что ряд первых разностей котировок EURUSD является стационарным, а исходный ряд имеет один единичный корень и называется *интегрированным рядом первого порядка*.

3.3 Распределения вероятностей

В результате расчетов для всех 3х-мерных списков получены усредненные по множеству недель значения стандартного отклонения σ и размаха (*High-Low)*.

Если предположить отсутствие корреляции между отсчетами, то следует ожидать линейного роста дисперсии временного ряда с ростом длины временного интервала бара. Или, что то же самое:

(3.9)

где k – коэффициент аппроксимации;

T – длина временного интервала бара.

Если предположение подтвердится, то это будет означать, что поведение процесса аналогично интегралу от белого шума. А значит, возможность анализа на основе линейных методов отсутствует.

Исходя из полученных данных, для всех временных интервалов получены усредненные значения коэффициента аппроксимации. В таблицах 3.16-3.20 приведены данные, позволяющие сравнить расчетные значения стандартного отклонения с экспериментальными для первых разностей временных рядов Forex для всех интервалов.

Таблица 3.16

Расчеты стандартного отклонения для dOPEN

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1399	0,1399	0,1571	-0,0172	-12,30%
5	0,3672	0,1642	0,3514	0,0158	4,30%
15	0,6228	0,1608	0,6086	0,0142	2,29%

Окончание таблицы 3.16

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
30	0,8712	0,1591	0,8607	0,0106	1,21%
60	1,2387	0,1599	1,2171	0,0215	1,74%
240	2,4616	0,1589	2,4343	0,0273	1,11%
	среднее	0,1571			

Таблица 3.17

Расчеты стандартного отклонения для dCLOSE

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1410	0,1410	0,1570	-0,0160	-11,33%
5	0,3675	0,1643	0,3510	0,0165	4,48%
15	0,6231	0,1609	0,6080	0,0150	2,42%
30	0,8708	0,1590	0,8599	0,0109	1,26%
60	1,2382	0,1598	1,2160	0,0221	1,79%
240	2,4300	0,1569	2,4320	-0,0020	-0,08%
	среднее	0,1570			

Таблица 3.18

Расчеты стандартного отклонения для dHIGH

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1286	0,1286	0,1502	-0,0216	-16,81%
5	0,3439	0,1538	0,3360	0,0079	2,31%
15	0,5940	0,1534	0,5819	0,0121	2,03%
30	0,8442	0,1541	0,8229	0,0213	2,52%
60	1,1998	0,1549	1,1638	0,0360	3,00%
240	2,4268	0,1566	2,3275	0,0993	4,09%
	среднее	0,1502			

Таблица 3.19

Расчеты стандартного отклонения для dLOW

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1286	0,1286	0,1460	-0,0174	-13,54%
5	0,3345	0,1496	0,3265	0,0081	2,41%
15	0,5771	0,1490	0,5655	0,0117	2,02%
30	0,8121	0,1483	0,7997	0,0124	1,53%
60	1,1493	0,1484	1,1310	0,0183	1,59%
240	2,3574	0,1522	2,2619	0,0955	4,05%
	среднее	0,1460			

Таблица 3.20

Расчеты стандартного отклонения для HIGH_LOW

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка
1	0,00017	0,14247	0,00017	0,14247
5	0,00046	0,37381	0,00020	0,16717
15	0,00082	0,63170	0,00021	0,16310
30	0,00118	0,87759	0,00021	0,16023
60	0,00169	1,22466	0,00022	0,15810
240	0,00356	2,26703	0,00023	0,14634

Как видно из приведенных выше таблиц, для 1-минутного интервала получается заметное расхождение, что заставляет предположить наличие корреляции между отсчетами для малых интервалов времени.

На рисунке 3.12 представлены графики зависимости расчетного и экспериментального стандартного отклонения от шага времени T для ряда первых разностей цен закрытия (dCLOSE) по всем интервалам.

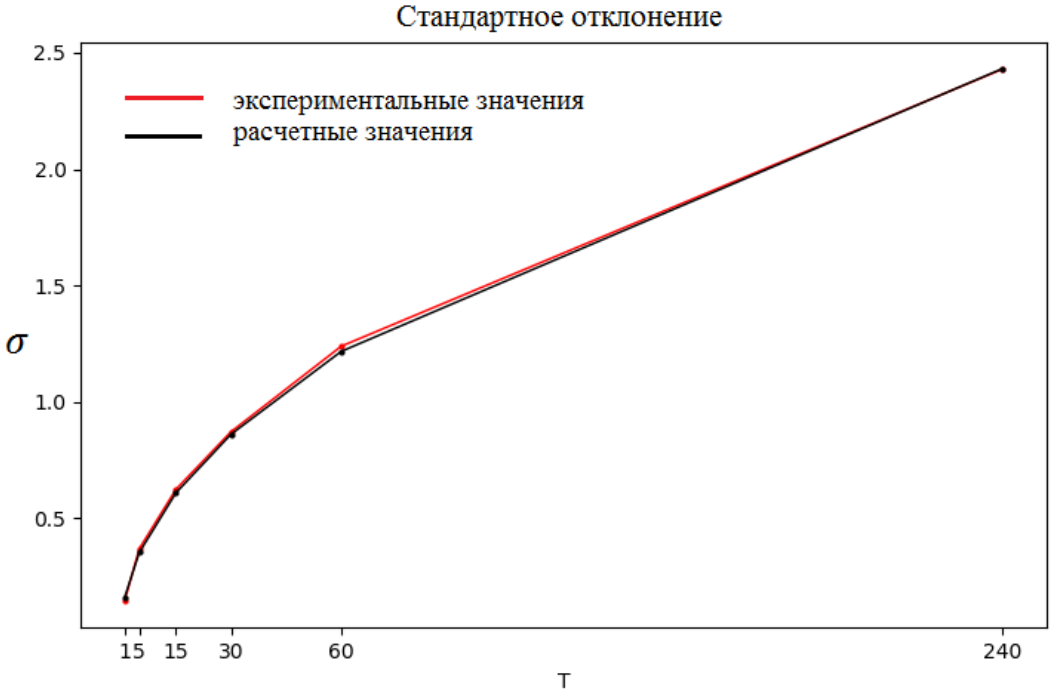

Рисунок 3.12 – График зависимости расчетного и экспериментального стандартного отклонения от периода для dCLOSE

Расчетная и экспериментальная зависимости квадрата сигмы (дисперсии) от периода T ряда первых разностей цен закрытия (dCLOSE) по всем интервалам для изображены на рисунке 3.13.

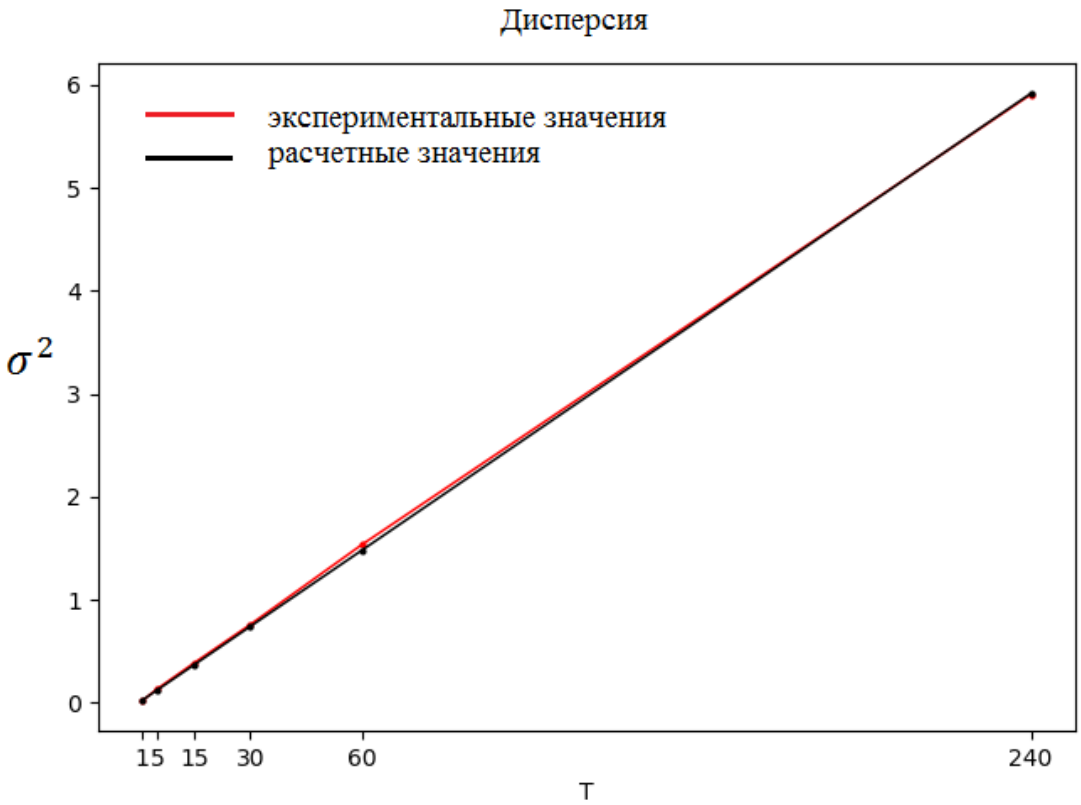

Рисунок 3.13 – График зависимости расчетной и экспериментальной дисперсии от периода для dCLOSE

Ниже приводятся аналогичные таблицы 3.21-3.24, в которых коэффициент k рассчитывается без участия 1-минутного интервала.

Таблица 3.21

Расчеты стандартного отклонения для dOPEN

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1399	0,1399	0,1606	-0,0207	-14,76%
5	0,3672	0,1642	0,3591	0,0081	2,21%
15	0,6228	0,1608	0,6219	0,0009	0,14%
30	0,8712	0,1591	0,8795	-0,0083	-0,95%
60	1,2387	0,1599	1,2438	-0,0051	-0,41%
240	2,4616	0,1589	2,4876	-0,0260	-1,06%
среднее без 1 мин		0,1606			

Таблица 3.22

Расчеты стандартного отклонения для dCLOSE

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
1	0,1410	0,1410	0,1606	-0,0207	-14,76%
5	0,3675	0,1643	0,3591	0,0081	2,21%
15	0,6231	0,1609	0,6219	0,0009	0,14%

Окончание таблицы 3.22

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппроксимация	Абсолютная ошибка	Ошибка, %
30	0,8708	0,1590	0,8795	-0,0083	-0,95%
60	1,2382	0,1598	1,2438	-0,0051	-0,41%
240	2,4300	0,1569	2,4876	-0,0260	-1,06%
среднее без 1 мин		0,1602			

Таблица 3.23

Расчеты стандартного отклонения для dHIGH

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппрокси-мация,	Абсолютная ошибка	Ошибка, %
1	0,1286	0,1286	0,1546	-0,0259	-20,17%
5	0,3439	0,1538	0,3456	-0,0017	-0,50%
15	0,5940	0,1534	0,5986	-0,0047	-0,78%
30	0,8442	0,1541	0,8466	-0,0024	-0,28%
60	1,1998	0,1549	1,1973	0,0025	0,21%
240	2,4268	0,1566	2,3945	0,0323	1,33%
среднее без 1 мин		0,1546			

Таблица 3.24

Расчеты стандартного отклонения для dLOW

Шаг (минут)	Sigma*10^3	Коэффициент аппроксимации	Аппрокси-мация,	Абсолютная ошибка	Ошибка, %
1	0,1286	0,1286	0,1495	-0,0209	-16,25%
5	0,3345	0,1496	0,3343	0,0003	0,08%
15	0,5771	0,1490	0,5790	-0,0018	-0,32%
30	0,8121	0,1483	0,8188	-0,0066	-0,82%
60	1,1493	0,1484	1,1579	-0,0087	-0,75%
240	2,3574	0,1522	2,3159	0,0415	1,76%
среднее без 1 мин		0,1495			

Полученные результаты еще больше укрепляют подозрение в том, что рассматриваемые временные ряды разностей имеют характеристики, близкие к белому шуму.

Подозрителен тот факт, что ошибка для 1-минутного интервала все равно возрастает, а для всех прочих интервалов становится малой. Это говорит о наличии возможной корреляции для коротких интервалов, и предположение о наблюдаемом процессе как о белом шуме нарушается. Полное подтверждение этому можно будет установить после расчета автокорреляционных функций.

3.4 Автокорреляция элементов временного ряда

Автокорреляция элементов временного ряда – корреляционная зависимость между последовательными элементами временного ряда.

Лаг – число периодов, по которым рассчитывается коэффициент автокорреляции между парами элементов ряда. С увеличением лага число пар значений, по которым рассчитывается коэффициент автокорреляции, уменьшается. Максимальный лаг должен быть не больше (n/4).

Коэффициент автокорреляции уровней ряда первого порядка, измеряющий зависимость между соседними уровнями ряда и , т.е. при лаге 1, рассчитывается по формуле:

(3.10)

где

Аналогично определяются коэффициенты автокорреляции второго и более высоких порядков.

Автокорреляционная функция временного ряда – последовательность коэффициентов автокорреляции с лагами, равными 1, 2, 3 ….

График зависимости значений автокорреляционной функции от величины лага называется *коррелограммой*.

Свойства коэффициента корреляции:

- Если наиболее высоким оказался коэффициент автокорреляции первого порядка, исследуемый ряд содержит только тенденцию.
- Если наиболее высоким оказался коэффициент автокорреляции порядка t, ряд содержит циклические колебания с периодичностью в t моментов времени.

Если ни один из коэффициентов автокорреляции не является значимым, то:

1. Ряд не содержит тенденции и циклических колебаний и имеет случайную структуру.
2. Ряд содержит сильную нейтральную тенденцию, для выявления которой нужно провести дополнительный анализ.

Для всех 3-мерных списков были рассчитаны усредненные по всем неделям автокорреляционные функции (см. приложение №2) и построены коррелограммы для первых разностей котировок для всех временных интервалов (см. приложение №4). Также обозначены доверительные границы: 95% – сплошная черная линия, 99% – пунктирная зеленая линия. На рисунках 3.14-3.16 по оси X откладываются номера лагов, а по оси Y значения соответствующих функций.

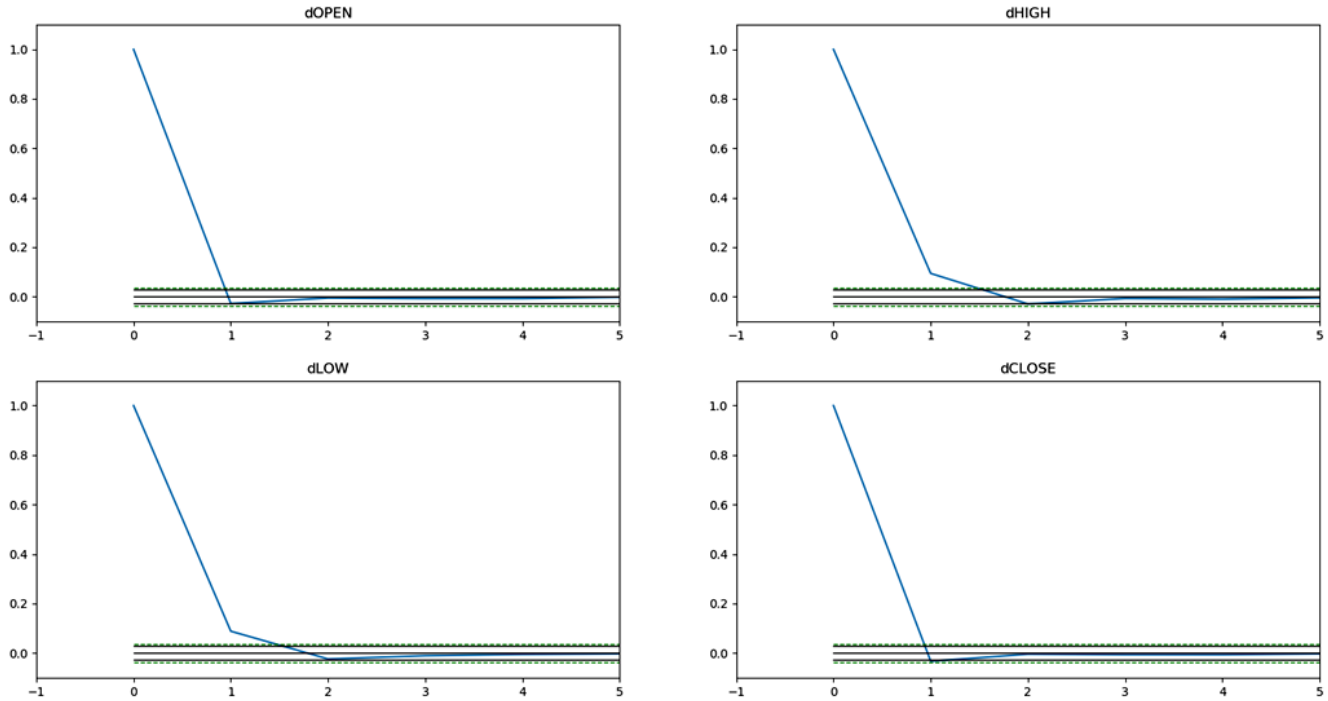

Рисунок 3.14 – Коррелограмма для интервала 1 минута

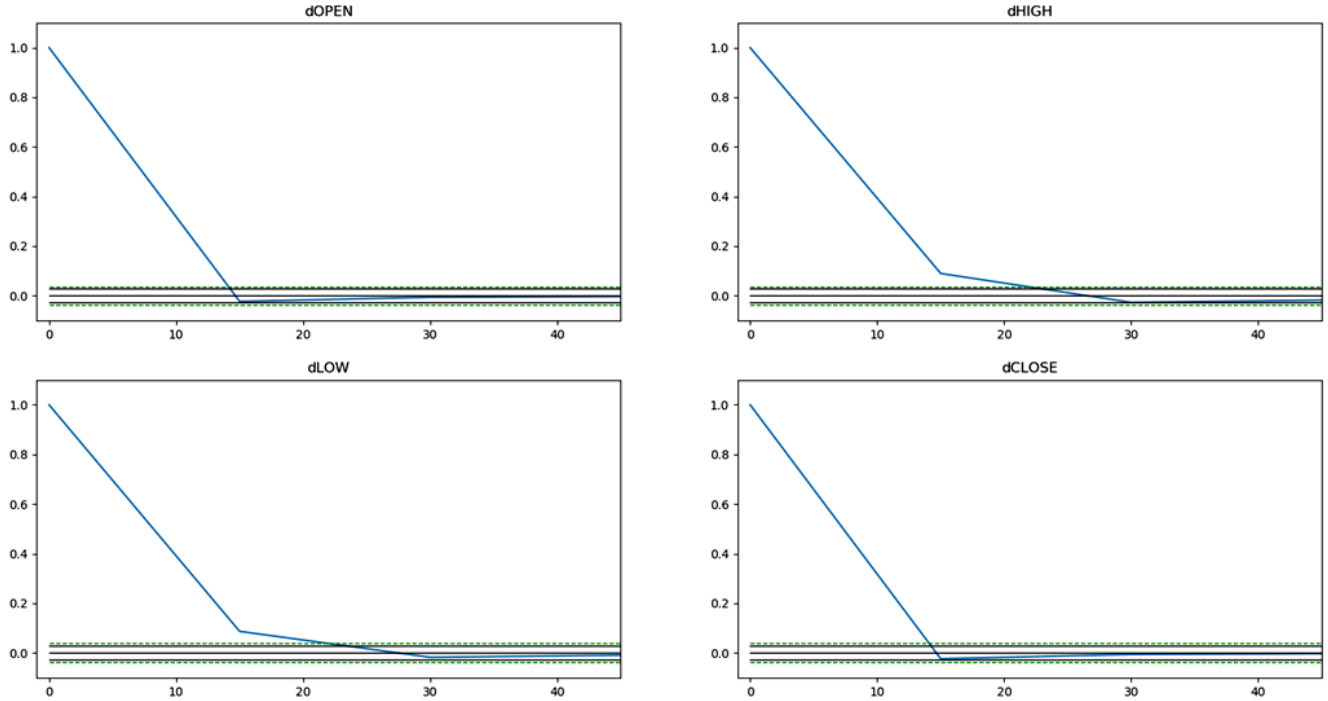

Рисунок 3.15 – Коррелограмма для интервала 15 минут

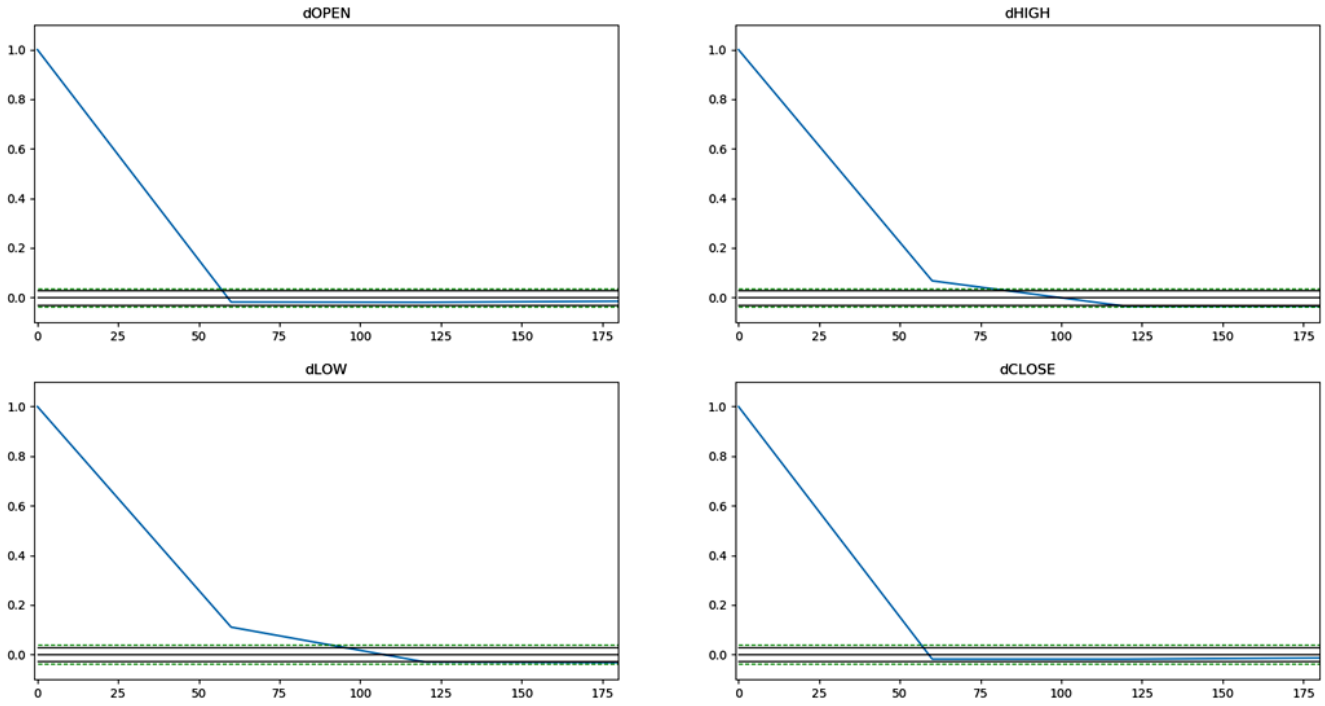

Рисунок 3.16 – Коррелограмма для интервала 60 минут

На построенных автокорреляционных функциях (рисунки 3.14-3.16) для всех характеристик бара по всем интервалам заметных значений корреляций не наблюдается, что говорит о непригодности линейных методов предсказания.

Построим автокорреляционные функции размаха значений (High-Low) для всех интервалов (рисунок 3.17)

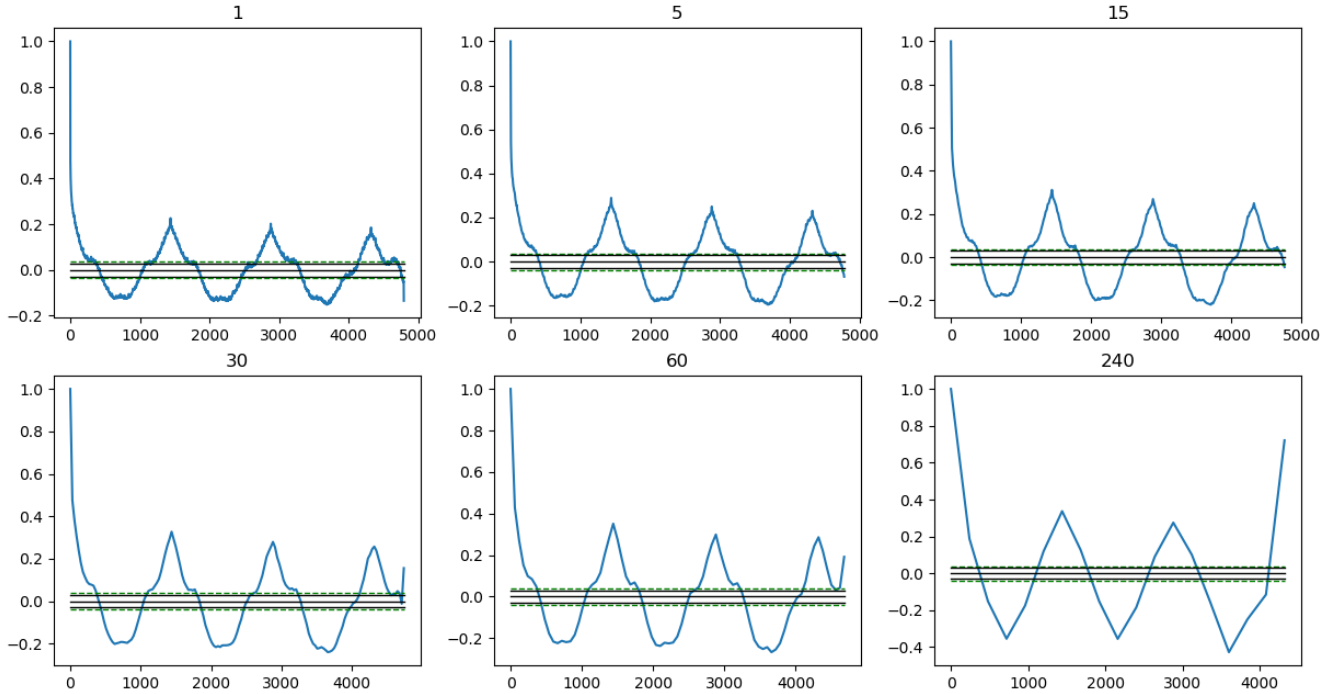

Рисунок 3.17 – Коррелограмма размаха значений (High-Low) для всех интервалов

Коррелограммы для (High-Low) на рисунке 3.17 являются значимыми, т.к. наблюдается суточная периодичность. Этот факт весьма интересен. Однако найти подходящее объяснение этому факту не удалось.

Для любых промежутков времени автокорреляционные функции похожи на дельта-функции. Это означает, что исследуемый процесс весьма похож на белый шум. Следовательно, анализ временных рядов линейными методами невозможен.

Выводы по разделу

После проведения статистического анализа исходных данных были выявлены следующие факты:

1) По расчетам описательных статистик, по гистограммам и результатам теста Жарка-Бера ряд первых разностей временных рядов Forex отклоняется от нормального распределения, так как коэффициент эксцесса значительно превышает установленное значение для нормального распределения.

2) По результатам теста Дики-Фуллера ряд первых разностей стационарен, а исходный ряд является интегрированным рядом первого порядка.

3) Линейность дисперсии говорит о том, что характер поведения временных рядов похож на белый шум, а значит применение линейных методов анализа невозможно.

4) Расчет автокорреляционных функций показал, что заметных зависимостей между значениями временных рядов не наблюдается. Полученные функции автокорреляции лишний раз подтверждают близость ряда первых разностей к белому шуму.

5) Автокорреляционная функция величины (High-Low) представляет собой периодическую функцию с суточным периодом и весьма слабым затуханием. Этот факт в литературе не упоминается. Объяснение ему предложить не удалось.

6) Сделанные выводы заставляют предположить, что классические методы анализа не смогут привести к успеху. В связи с чем принято решение воспользоваться нелинейными методами технического анализа, а именно применить методы нейронных сетей.

4 НЕЙРОННЫЕ СЕТИ В АНАЛИЗЕ ВРЕМЕННЫХ РЯДОВ

Проведение статистического анализа временных рядов показало, что применение линейных методов анализа бесперспективно, аналогичного мнения придерживаются и другие исследователи [32,33].

Остается возможность воспользоваться нелинейными методами, такими как нейронные сети, генетические алгоритмы и др. В настоящей работе все дальнейшие исследования проводятся с помощью нейронных сетей.

Принятие решения о том вступать или не вступать в игру может быть основано на результатах либо задачи предсказания, либо задачи классификации.

1. *Задача предсказания*. Решением задачи предсказания является прогнозирование будущих значений временного ряда.

2. *Задача классификации*. Рассматривается следующая стратегия. Отрезок из M подряд следующих отсчетов временного ряда может быть отнесён к одному из непересекающихся классов, а именно:

   a) в течение ближайших последующих *N* отсчетов первым произойдет прирост значения курса на величину +*delta*;
   b) первым произойдет прирост на величину –*delta*;
   c) не произойдет ни a), ни b).

Выходной слой нейронной сети для задачи классификации содержит 3 нейрона, имеющие на выходе оценки вероятностей принадлежности входа нейронной сети каждому из классов.

Если одна из вероятностей заметно превышает 50%, то можно вступать в игру.

В случае a) принимается решение делать ставку на повышение, в случае b) – на понижение, в случае c) мы не вступаем в игру.

Все возможные исходы можно представить в виде матрицы, где строки – прогнозы, столбцы – реальные данные:

	a	**b**	**c**
$\mathbf{a}_{\text{прогноз}}$	выигрыш	проигрыш	0
$\mathbf{b}_{\text{прогноз}}$	проигрыш	выигрыш	0
$\mathbf{c}_{\text{прогноз}}$	0	0	0

Необходимо оценить вероятности исходов игры.

4.1 Построение модели нейронной сети. Задача прогнозирования

Программы тренировки и тестирования нейронных сетей сначала отлаживались на некоторых регулярных функциях, для которых ожидаемый результат был более или менее известен. После того, как удавалось получить правдоподобные результаты для регулярных функций, можно было сделать вывод, что программы написаны правильно.

После этого они применялись для исследования реальных временных рядов.

4.1.1 Экспериментальные данные

Чтобы убедиться в том, что построенная далее нейронная сеть работает корректно, в качестве модели экспериментальных данных принята следующая функция смеси синусоид с некратными периодами, которая моделирует некий хаотичный процесс, похожий на временной ряд:

(4.1)

Значения аргумента *x* определялись следующим образом:

, (4.2)

где – случайное число;

– количество значений аргумента *x;*

d_x = 0,1 – шаг аргумента.

График функции смеси синусоид представлен на рисунке 4.1.

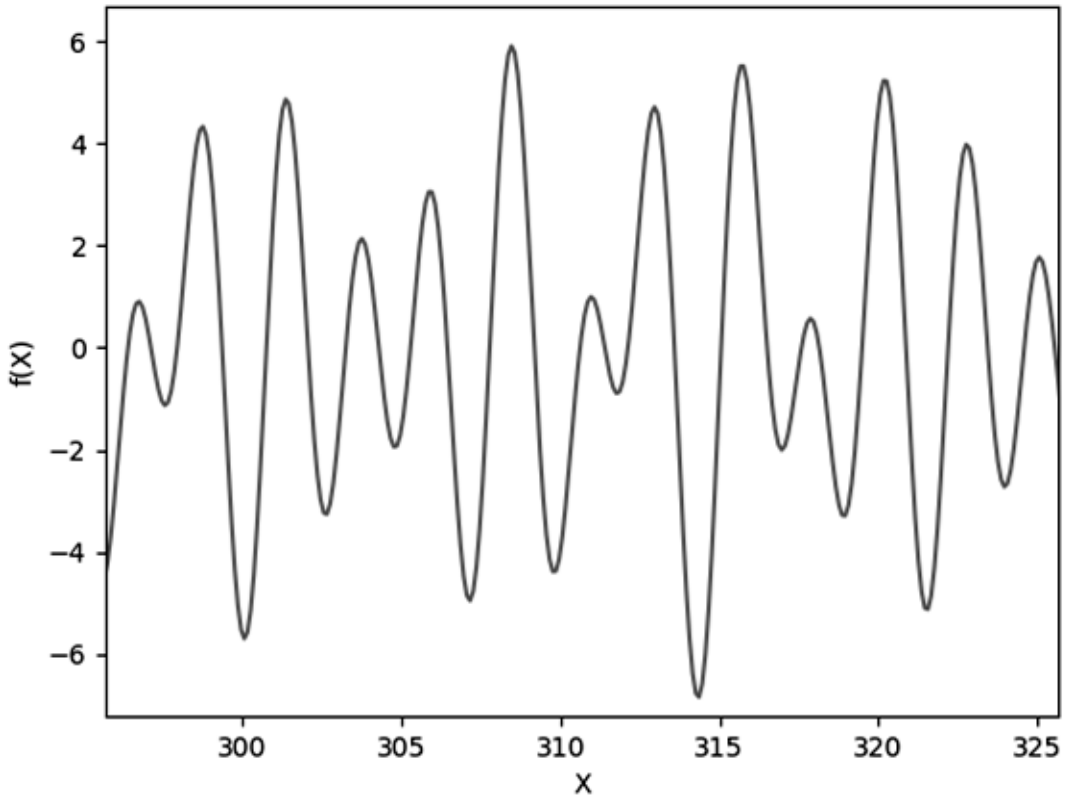

Рисунок 4.1 – Смесь синусоид с некратными периодами

4.1.2 Подготовка данных

Значения экспериментальной функции, на которых будет обучаться и тестироваться построенная нейронная сеть, представляются в виде матрицы *m[nRow, nInp+1]* (реализацию см. в приложении №5), в которой строки *nRow* – это количество значений модельной функции, столбцы *(nInp+1)* – это значения функции, разбитые на окна по *nInp* штук, последним значением в строке является максимальный прирост функции на *nFuture* последующих точках.

Данные (матрица *m[nRow, nInp+1]*) разделяются на матрицу признаков *inp=m[: , 1:nInp+1]* и вектор целевой переменной *out=m[: , nInp+1]*. Здесь nInp – число значений в строке, используемых для предсказания, оно же – число входов для нейронной сети.

4.1.3 Создание модели и настройка гиперпараметров

Написана программа (см. приложение №5), реализующая нейронную сеть с помощью библиотеки Keras, которая позволяет создавать нейронные сети с минимальными затратами на программирование.

В качестве модели нейронной сети используется последовательная модель Sequential из модуля keras.models с заданием слоев keras.layers типа Dense.

Определяетя входной, выходной и скрытые слои. Нейронная сеть имеет плотную (Dense) структуру – каждый нейрон связан со всеми нейронами следующего слоя. Выходной слой состоит из единственного нейрона, определяющего предсказанные значения экспериментальной функции (4.1). Количество скрытых слоев – 2. Число нейронов в каждом слое, кроме выходного, предлагается взять 24.

Экспериментальным путем было принято решение использовать в качестве функции активации для всех слоев, кроме выходного, функцию *"tanh"* (рисунок 4.2а). Для выходного слоя было решено воспользоваться линейной функцией *"linear"* (рисунок 4.2б) для определения предсказанных значений функции (4.1).

$$Out = th(NET) \text{ или } Out = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}} \qquad Out = NET$$

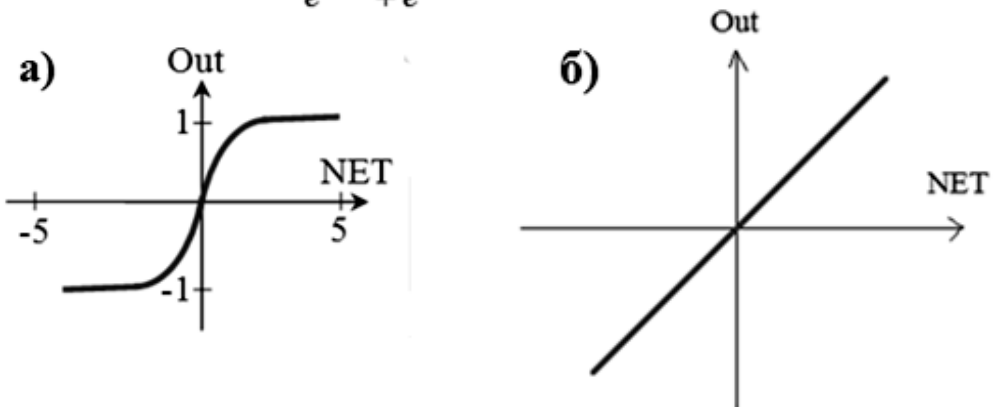

Рисунок 4.2 – Функции активации:
а) гиперболический тангенс, б) линейная функция

В модели используется нейрон смещения (bias), который суммируется со взвешенными входами нейрона, образуя текущее состояние нейрона – входную величину (аргумент) для функции активации нейрона.

Методом оптимизации модели нейронной сети был выбран метод RMSProp (Root Mean Square Propagation)

«Эпохи» – количество проходов нейронной сети по всем данным (выбирается исходя из того, насколько быстро модель с каждым новым проходом

приближается к желаемой предсказательной точности). Путем настройки и отладки модели было решено взять 100 эпох.

4.1.4 Прогнозирование и оценка результатов

В результате тренировки нейронной сети на смеси синусоид с некратными периодами задача предсказания решалась с такими параметрами (таблица 4.1):

Таблица 4.1

Параметры настройки нейронной сети.

Количество значений аргумента функции	1000
Шаг аргумента функции	0,1
Количество входных точек на каждом шаге обучения	40
Количество точек, непосредственно следующих за входными, на которых фиксируется прирост	12
Число нейронов в слое (кроме выхода, на всех слоях одинаковое)	24
Количество эпох обучения	100

Были получены следующие результаты предсказания приращений на N последующих значениях функции (рисунок 4.3, таблица 4.2):

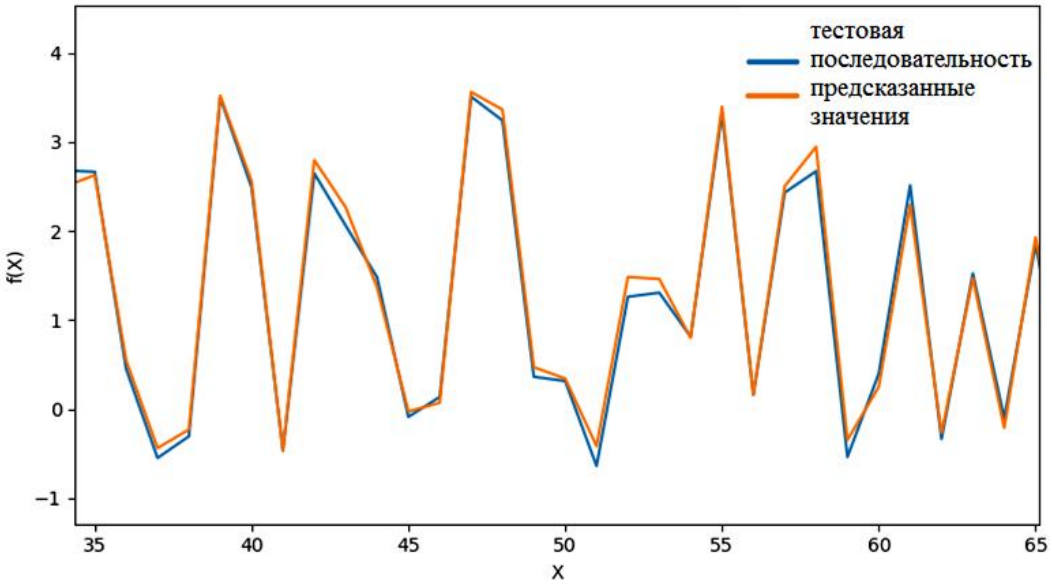

Рисунок 4.3 – Результат работы нейронной сети

Таблица 4.2

Результаты тренировки нейронной сети

Предсказанные значения	Ожидаемые значения	Разность
0.8608	0.8280	0.0328
9.8999	9.8019	0.0978
1.7048	1.8451	-0.1403
2.8557	3.2370	-0.3813
10.0227	10.2939	-0.2712

Окончание таблицы 4.2

8.1522	8.2942	-0.1420
5.2669	5.6408	-0.3739
3.5651	3.3513	0.2138
3.6349	3.5629	0.0720
1.6460	1.7725	-0.1265

Оптимизируемым критерием в задаче предсказания является среднеквадратичная ошибка MSE:

— (3.10)

где *Z(t)* – фактическое значение временного ряда, а – прогнозное.

В работе получена ошибка .

Задача прогнозирования для функции смеси синусоид с некратными периодами решена успешно, о чем свидетельствует таблица 4.1 и рисунок 4.3. А значит, построенную модель нейронной сети можно использовать в целях построения стратегии игры.

4.2 Задача классификации

На основании данных функции смеси синусоид написана программа (см. приложение №6) для классификации отрезков ряда приращений значений этой функции. Каждый отрезок ряда должен быть отнесен к одному из следующих классов:

a) в течение ближайших N отсчетов первым произойдет прирост на назначенную величину +delta;

b) первым произойдет прирост на величину –delta;

c) не произойдет ни a), ни b).

Обучающая последовательность нейронной сети представляет собой таблицу, в каждой строке которой размещен отрезок ряда. Для каждой строки определяется, к какому классу она должна быть отнесена. Это можно сделать, просмотрев продолжение функции за пределами этого отрезка. Выходными значениями нейронной сети являются три значения 0 или 1 для пометки принадлежности отрезка тому или иному классу.

В процессе тестирования нейронной сети на нее подаются случайные отрезки ряда значений функции, выходом являются оценки вероятности принадлежности отрезков к тому или иному классу.

При решении задач классификации оптимизируемым критерием является кроссэнтропия. Использование этого критерия позволяет получить оценки вероятностей принадлежности отрезка ряда классу.

Результатом тестирования нейронной сети является матрица оценок вероятностей. Для рассматриваемой функции смеси синусоид были получены следующие значения:

	a	b	c
$a_{прогноз}$	0.9982	0.0000	0.0018
$b_{прогноз}$	0.0000	0.9968	0.0032
$c_{прогноз}$	0.0264	0.0265	0.9471

Построенная на модельной функции нейронная сеть показала, что выигрыш трейдера на основании предсказанных значений на прирост +delta составляет 99% и на прирост –delta 99%.

Полученные результаты свидетельствуют о работоспособности построенных нейронных сетей и предположительно хорошем качестве разработанной программы.

4.3 Эксперименты с реальными временными рядами

Убедившись в том, что нейронная сеть дает хорошие результаты для рассмотренной выше функции, на следующем этапе переходим к реальным временным рядам Forex.

4.3.1 Задача прогнозирования

Написана программа для прогнозирования последующих значений временных рядов Forex с помощью нейронных сетей (см. приложении №7). Обучение нейросети производилось на 70% от исходных данных, соответственно, тестирование делалось на 30%. Задача прогнозирования решена со следующими результатами для интервалов 30, 60 и 240 минут (рисунки 4.4-4.6).

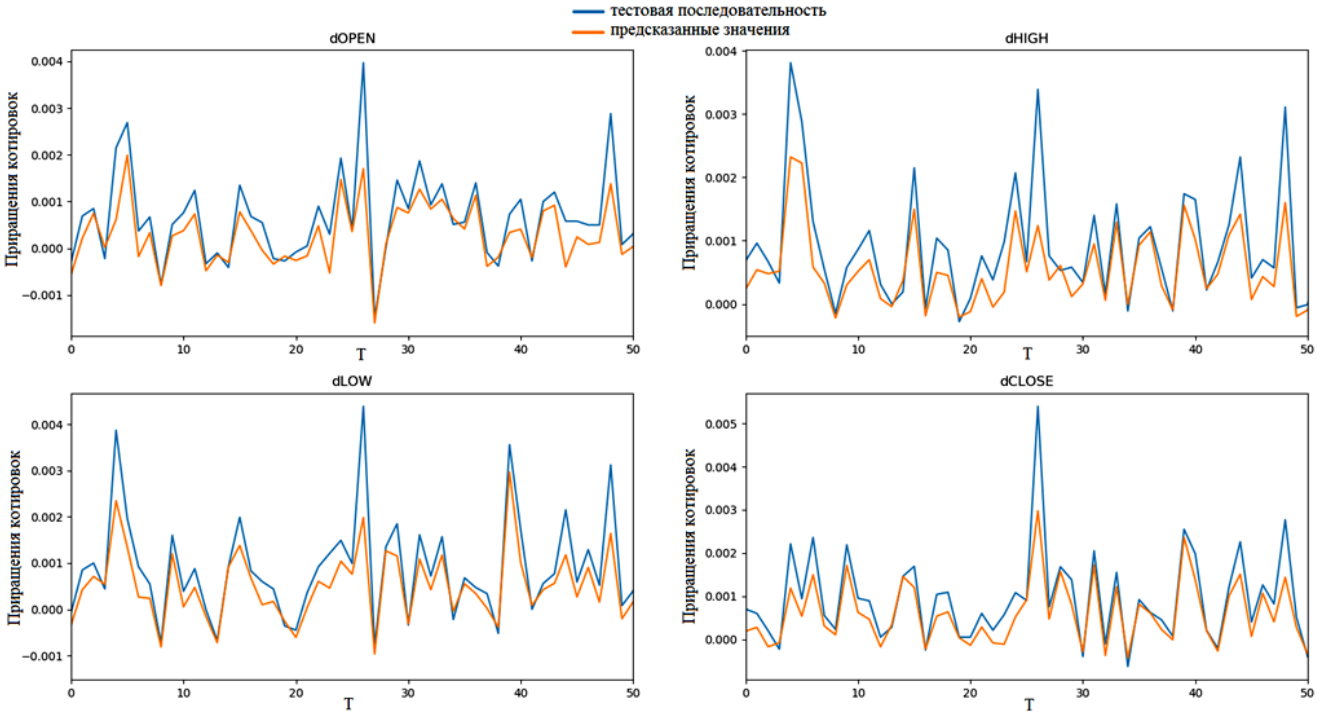

Рисунок 4.4 – Результат работы нейронной сети
для интервала 30 минут

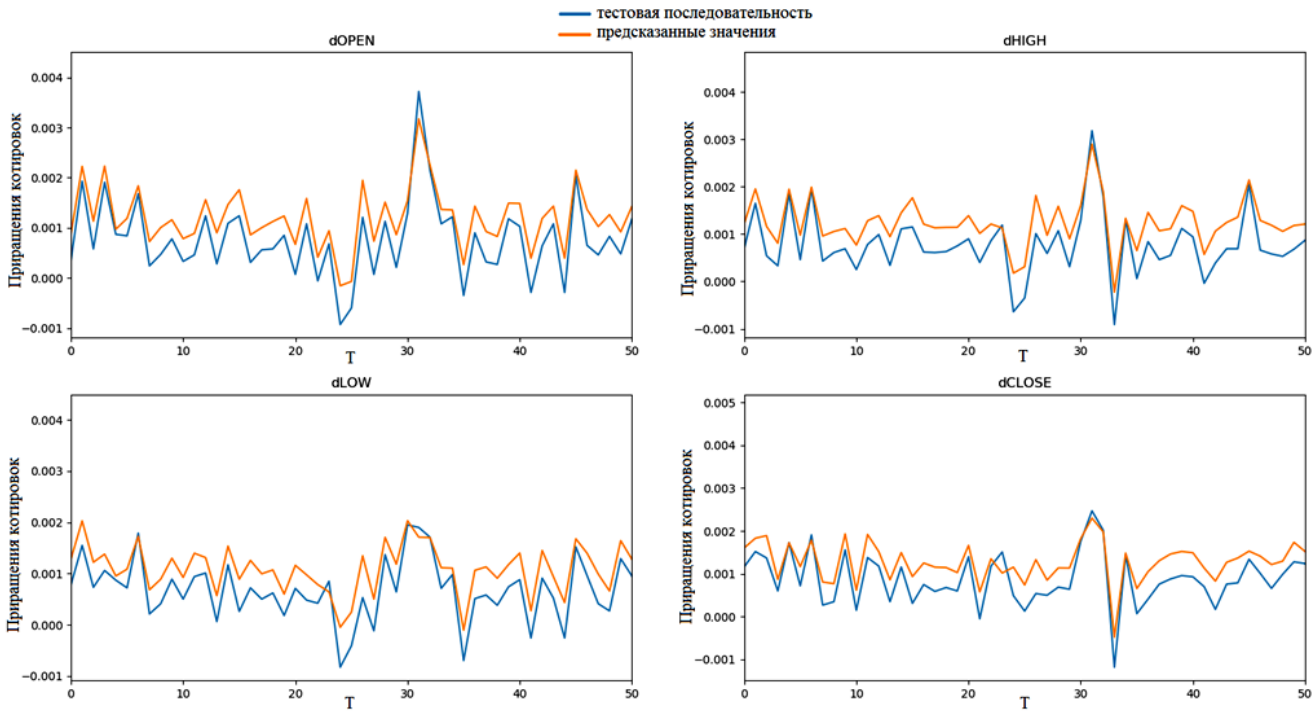

Рисунок 4.5 – Результат работы нейронной сети
для интервала 60 минут

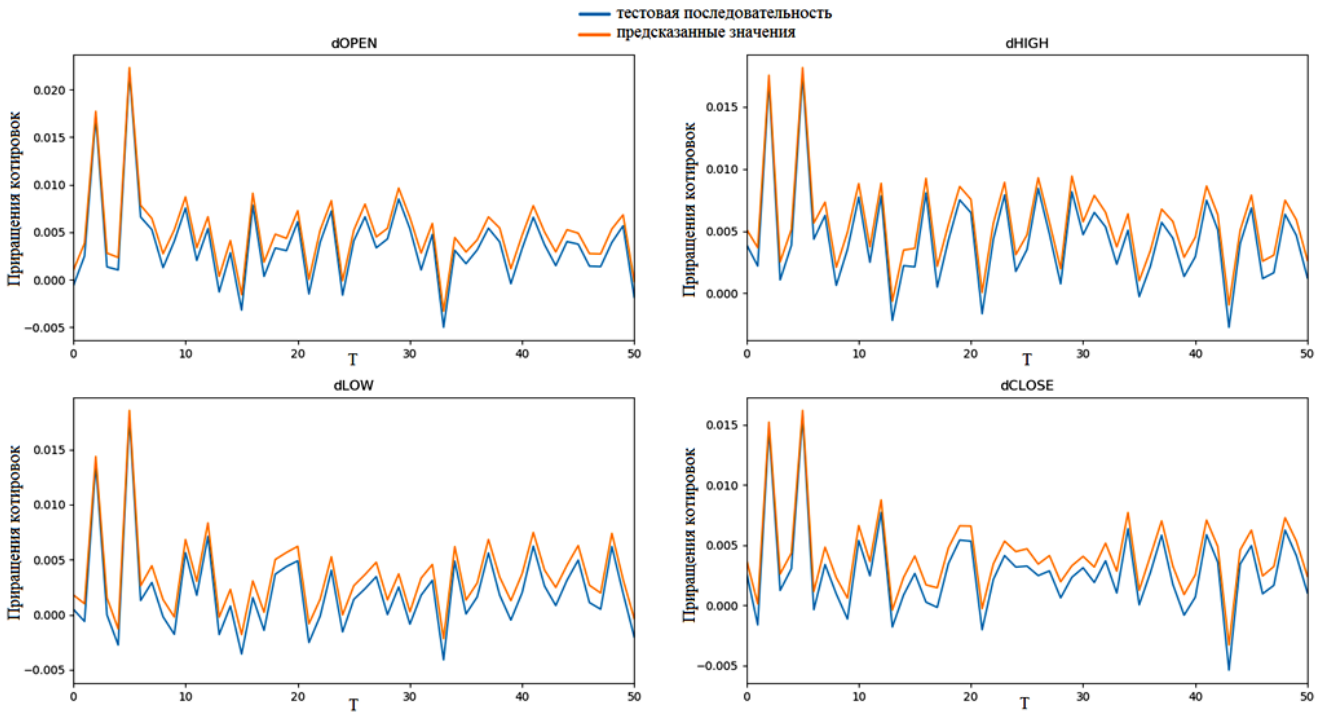

Рисунок 4.6 – Результат работы нейронной сети
для интервала 240 минут

В таблице 4.3 представлены средние квадратичные ошибки для данных с интервалами 30, 60 и 240 минут:

Таблица 4.3

Результаты тренировки нейронной сети

timeframe	Name	dOpen	dHigh	dLow	dClose
30	RMSE	0,000253	0,000247	0,000253	0,000250
	Диапазон	0,002947	0,002694	0,002799	0,002999
60	RMSE	0,000306	0,000306	0,000350	0,000340
	Диапазон	0,004488	0,004120	0,004058	0,004527
240	RMSE	0,000812	0,000884	0,000769	0,000842
	Диапазон	0,011638	0,011618	0,010812	0,010864

4.3.2 Задача классификации

Алгоритм рассматриваемого ниже торгового робота, базирующегося на обученной нейронной сети, выполняющей классификацию, заключается в следующем: на вход нейронной сети подается некоторый отрезок или набор отрезков (несколько характеристик – OPEN, HIGH, LOW, CLOSE) временного ряда. Если максимальная из оценок вероятностей на выходе нейронов выходного слоя превышает некоторое заданное пороговое значение, то робот принимает решение сделать ставку. Этот процесс повторяется каждый раз, когда открывается новый таймфрейм.

Результатом моделирования нейронной сети являются количественная матрица и матрица оценок вероятностей.

Количественная матрица M:

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}
m_{31}	m_{32}	m_{33}

В количественной матрице величины m_{11} и m_{22} показывают, сколько раз ожидался и произошел прирост на величину и соответственно.

Количество выигрышей на повышении рассчитывается, как .

Количество выигрышей на понижении рассчитывается, как .

Сумма выигрышей в неделю: .

Число попыток в неделю:

Число выигрышей в неделю рассчитывается по следующей формуле:

где n – количество недель.

Затем эти величины нормировались, в результате чего получили оценки вероятностей исходов игры.

Описанный выше алгоритм реализован в приложении №8.

После тестирования работы нейронной сети при различных гиперпараметрах задача классификации была решена со следующими результатами, представленными ниже.

1) Тест №1

Первый этап тестирования проводился со следующими параметрами (таблица 4.4):

Таблица 4.4

Гиперпараметры для теста №1

Значения характеристик котировок, подаваемые на вход	dHIGH dLOW dCLOSE
Процент недель для тренировки	70%
Число нейронов в каждом слое, кроме выходного	40
Количество входных точек на каждом шаге обучения	24
Пороговое значение вероятности	0,55
Граница приращения	0,003
Количество эпох обучения	100

Результаты тестирования приведены в таблице 4.5.

Таблица 4.5

Результаты решения задачи классификации
на реальных временных рядах

Интервал в минутах	30			60			240		
Количественная матрица	2395	857	2	1084	744	0	360	262	0
	740	913	1	250	386	0	174	216	0
	7472	6460	0	3326	2907	1	481	424	0
Матрица оценок вероятностей исходов	0,5992	0,3999	0.0009	0,5930	0,4070	0.0000	0,5788	0,4212	0.0000
	0,4474	0,5520	0.0006	0,3931	0,6069	0.0000	0,4462	0,5538	0.0000
	0,5363	0,4637	0.0000	0,5335	0,4663	0.0002	0,5315	0,4685	0.0000
Сумма выигрышей в неделю	600			476			140		
Число попыток в неделю	15,13			9,82			4,05		
Число выигрышей в неделю	2,39			1,90			0,56		

2) Тест №2

Второй этап тестирования проводился со следующими параметрами (таблица 4.6):

Таблица 4.6

Гиперпараметры для теста №2

Значения характеристик котировок, подаваемые на вход	dHIGH dLOW dCLOSE
Процент недель для тренировки	70%
Число нейронов в каждом слое, кроме выходного	40
Количество входных точек на каждом шаге обучения	24
Пороговое значение вероятности	0,55
Граница приращения	0,002
Количество эпох обучения	100

Результаты тестирования приведены в таблице 4.7.

Таблица 4.7

Результаты решения задачи классификации
на реальных временных рядах

Интервал в минутах	30	60	240
Количественная матрица	3519 2636 1 2016 2131 0 3806 3620 0	2030 1569 0 1531 1588 0 1041 939 0	519 374 0 372 398 0 120 134 0
Матрица оценок вероятностей исходов	0,5716 0,4282 0.0000 0,4861 0,5139 0.0000 0,5125 0,4875 0.0000	0,5640 0,4360 0.0000 0,4909 0,5091 0.0000 0,5258 0,4742 0.0000	0,5812 0,4188 0.0000 0,4831 0,5169 0.0000 0,4724 0,5276 0.0000
Сумма выигрышей в неделю	998	518	171
Число попыток в неделю	41,048	26,765	6,652
Число выигрышей в неделю	3,976	2,064	0,684

3) Тест №3

Третий этап тестирования проводился со следующими параметрами (таблица 4.8):

Таблица 4.8

Гиперпараметры для теста №3

Значения характеристик котировок, подаваемые на вход	dHIGH dLOW dCLOSE
Процент недель для тренировки	70%
Число нейронов в каждом слое, кроме выходного	24
Количество входных точек на каждом шаге обучения	24
Пороговое значение вероятности	0,55
Граница приращения	0,002
Количество эпох обучения	100

Результаты тестирования приведены в таблице 4.9.

Таблица 4.9

Результаты решения задачи классификации
на реальных временных рядах

Интервал в минутах	30			60			240		
Количественная матрица	2605	1788	1	1740	1244	0	527	385	0
	1118	1442	0	1202	1351	0	344	384	0
	5618	5157	0	1660	1501	0	140	137	0
Матрица оценок вероятностей исходов	0,5929	0,4069	0.0002	0,5831	0,4169	0.0000	0,5779	0,4221	0.0000
	0,4367	0,5633	0.0000	0,4708	0,5292	0.0000	0,4725	0,5275	0.0000
	0,5214	0,4786	0.0000	0,5252	0,4748	0.0000	0,5054	0,4946	0.0000
Сумма выигрышей в неделю	1141			645			182		
Число попыток в неделю	27,705			22,06			6,56		
Число выигрышей в неделю	4,546			2,57			0,728		

Результаты тестирования говорят о том, что наилучшие гиперпараметры были выбраны в тесте №1. Нейронная сеть на реальных временных рядах показала, что вероятность правильной классификации для интервала 30 минут составляет 58%, для интервала 60 минут – 58%, для интервала 240 минут – 57%.

Выводы по разделу

Выбор нейронных сетей в качестве средства для решения задачи прогнозирования и задачи классификации привел к следующим результатам:

1. На экспериментальной функции смеси синусоид с некратными периодами, моделирующей хаотичный процесс, смоделированная нейронная сеть показала хороший результат, как в задаче прогнозирования, так и в задаче классификации.

2. Применение построенной нейросети к реальным данным оказалось возможным. Результаты могут показаться недостаточно хорошими. Рассмотренная стратегия обеспечивает правильность классификации отрезков ряда в 50-60% случаев.

3. Однако, чтобы окончательно убедиться в успешности предлагаемой стратегии торговли, требуется около 1-2 лет ее тестирования в режиме реального времени, путем подключения к платформе MetaTrader, чтобы собрать статистику игры и проанализировать полученные результаты.

ЗАКЛЮЧЕНИЕ

В результате анализа временных рядов Forex было получено следующее:

1) Разработаны программы на языке Python для анализа временных рядов Forex.

2) В теоретической части исследования были рассмотрены два подхода к анализу рынка: фундаментальный и технический. В практической части работа проводилась линейными (классический статистический анализ, корреляционный анализ) и нелинейными (нейронные сети) методами технического анализа

3) На первом этапе анализа выгруженные котировки валютной пары EURUSD подготавливались для дальнейшей работы в связи с тем, что были обнаружены пропуски и избыточные отсчеты в имеющихся данных.

4) Для того чтобы скачки в значениях, вызванные выходными днями на рынке Forex, не искажали картину, выполнено разбиение данных на недельные отрезки и исключение неполных недель, в которых число записей меньше минимального количества записей в неделе в зависимости от интервала.

5) После того, как данные были приведены в рабочий вид, на втором этапе был проведен классический статистический анализ временных рядов, в результате которого выяснилось, что первые разности не вполне соответствуют нормальному распределению, но являются стационарными.

6) Поведение корреляционных функций говорит о том, что статистические характеристики первых разностей близки к белому шуму. Откуда следует, что классические методы анализа едва ли приведут к успеху.

7) Автокорреляционная функция величины (High-Low) представляет собой периодическую функцию с суточным периодом и весьма слабым затуханием. Этот факт в литературе не упоминается. Объяснение ему предложить не удалось.

8) Выводы, сделанные в п. 5-6, заставляют предположить, что классические методы анализа не смогут привести к успеху. В связи с чем принято решение воспользоваться нелинейными методами технического анализа, а именно применить методы нейронных сетей.

Следующим этапом в работе стал выбор нейронных сетей в качестве средства для решения задачи прогнозирования и задачи классификации, который привел к таким результатам:

1) На экспериментальной функции смеси синусоид с некратными периодами, моделирующей некий хаотичный процесс, построенная нейронная сеть показала хороший результат, как в задаче прогнозирования, так и в задаче классификации, что означает применимость использования разработанных программ к анализу рядов Forex и разработке стратегии торговли.

2) Применение построенной нейросети к реальным данным оказалось возможным. Результаты тестирования обученных нейронных сетей показали превышение вероятности выигрыша над проигрышем порядка 3-5%, что принято считать хорошим результатом.

Проведённые исследования поставили большое количество новых вопросов. Работа может быть продолжена по следующим направлениям:

1) Создание методики подбора параметров нейронной сети.
2) Модифицирование стратегии – нейронная сеть продолжает обучаться в процессе работы.
3) При больших значениях прогнозируемых вероятностей возможно следует ожидать обнаружение типовых образов, что может обеспечить выигрыш.
4) Создание робота-трейдера, представляющего собой пару приложений – клиент и сервер, где клиент (на стороне MetaTrader) обращается к приложению на сервере, на котором реализована обученная нейроннная сеть и передает ему отрезок ряда котировок. Нейронная сеть вычисляет оценки вероятностей принадлежности отрезка к одному из рассмотренных в работе классов, после чего передаёт роботу приказ о вступлении или не вступлении в игру.

Чтобы окончательно убедиться в корректности работы написанной программы, требуется около 1-2 лет ее тестирования в режиме реального времени, путем подключения к платформе MetaTrader, чтобы собрать статистику игры и проанализировать полученные результаты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Куликов, А.А. Форекс для начинающих / А.А. Куликов. – СПб.: Питер, 2003. – 368 с.
2. Миклашевская, Н.А. Международная экономика. Учебник / Н.А. Миклашевская, А.В. Холопов. – «Дело и Сервис», 1997. – 191 с.
3. Бокс, Дж. Анализ временных рядов. Прогноз и управление / Дж. Бокс, Г. Дженкинс. – М.: Мир, 1974. – 406 с.
4. Основы торговли на Форекс. Вводная лекция [электронный ресурс], URL: https://www.fxclub.org/academy_dist_lecture/ (дата обращения 15.05.2018).
5. Нисон, С. Японские свечи. Графический анализ финансовых рынков.: Пер. с англ. / Т. Дроздова, М. Волкова. – М.: Диаграмма, 1998. – 328 с.
6. Кияница, А.С. Фундаментальный анализ финансовых рынков / А.С. Кияница. – СПб.: Питер, 2005. – 288 с.
7. Акелис, Стивен Б. Технически Анализ от А до Я: Пер. с англ. / Стивен Б. Акелис. – М.: Диаграмма, 1999. – 376 с.
8. Швагер, Дж. Технический анализ. Полный курс: Пер. с англ. / Дж. Швагер. – М.: «АЛЬПИНА», 2001. – 768 с.
9. Эрлих, А.А. Технический анализ товарных и финансовых рынков: Прикладное пособие. 2-е изд. / А.А. Эрлих. – М.: Инфра-М, 1996. – 176 с.
10. Гренджер, К. Спектральный анализ временных рядов в экономике: Пер. с англ. / К. Гренджер, М. Хатанака. – М.: Статистика, 1972. – 312 с.
11. Айвазян, С. А. Прикладная статистика: Основы моделирования и первичная обработка данных. Справочное изд. / С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин. — М.: Финансы и статистика, 1983. — 471 с.
12. Афанасьев, В.Н., Юзбашев М.М. Анализ временных рядов и прогнозирование: Учебник. / В. Н. Афанасьев, М. М. Юзбашев. – М.: Финансы и статистика, 2001. – 228 с.
13. Ежов, А.А., Шумский, С.А. Нейрокомпьютинг и его применения в экономике и бизнесе / А.А. Ежов, С.А. Шумский. – М.: МИФИ, 1998. – 224 с.
14. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. — М.: Вильямс, 2008. — с. 1103.
15. Fama, E. Efficient Capital Makers: A Review of Theory and Emperical Work / E. Fama // Journal of Finance. – 1970. – Vol. 5, №2. – P. 383–417.
16. The State of Data Science & Machine Learning [электронный ресурс], URL: https://www.kaggle.com/surveys/2017 (дата обращения 15.05.2018).
17. Документация библиотеки Numpy [электронный ресурс], URL: http://www.numpy.org/ (дата обращения 15.05.2018).
18. Документация библиотеки Pandas [электронный ресурс], URL: https://pandas.pydata.org/ (дата обращения 15.05.2018).
19. Документация библиотеки Scipy [электронный ресурс], URL: https://www.scipy.org/ (дата обращения 15.05.2018).
20. Документация библиотеки Matplotlib [электронный ресурс], URL: https://matplotlib.org/index.html (дата обращения 15.05.2018).

21. My Top 9 Favorite Python Deep Learning Libraries [электронный ресурс], URL: https://www.pyimagesearch.com/2016/06/27/my-top-9-favorite-python-deep-learning-libraries/ (дата обращения 15.05.2018).
22. Choosing framework for building Neural Networks [электронный ресурс], URL: http://danielhnyk.cz/choosing-framework-building-neural-networks-mainly-rrn-lstm/ (дата обращения 15.05.2018).
23. Документация библиотеки Keras [электронный ресурс], URL: https://keras.io/ (дата обращения 15.05.2018).
24. Документация библиотеки TensorFlow [электронный ресурс], URL: https://www.tensorflow.org/ (дата обращения 15.05.2018).
25. Compare PyBrain and Keras's popularity and activity [электронный ресурс], URL: https://python.libhunt.com/compare-pybrain-vs-keras (дата обращения 15.05.2018).
26. Архивы котировок [электронный ресурс], URL: https://www.gkfx.ru/trade_specs/quotes_archive.html (дата обращения 15.05.2018).
27. Крамер, Г. Математические методы статистики / Г. Крамер. – М.: Мир, 1975. – 648 с.
28. Гмурман, В.Е. Теория вероятностей и математическая статистика. 9-е изд. / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.
29. Кобзарь, А. И. Прикладная математическая статистика / А.И. Кобзарь. – М.: Физматлит, 2006. – 816 с.
30. Уилкс, С. Математическая статистика: Пер. с англ. / С. Уилкс. – М.: Наука, 1967. – 632 с.
31. Dickey, D.A., Fuller, W.A. Distribution of the Estimators for Autoregressive Time Series with a Unit Root / D.A. Dickey, W.A. Fuller // Journal of the American Statistical Association. – 1979. – №74 – P. 427-431.
32. Прогнозирование фондового рынка с использованием нейронных сетей [электронный ресурс], URL: https://habr.com/post/396505/ (дата обращения 15.05.2018).
33. Алмазов, А. Фрактальная теория. Как поменять взгляд на рынки [электронный ресурс], URL: http://www.al24.ru/wp-content/uploads/2015/02/%D0%B0%D0%BB%D0%BC_1.pdf (дата обращения 15.05.2018).
34. Найман, Э.–Л. Малая Энциклопедия Трейдера: Пер. с англ. / Э.–Л. Найман. – К.: ВИРА-Р Альфа Капитал, 1999. – 236 с.
35. Мэрфи, Дж. Технический анализ фьючерсных рынков. Теория и практика: Пер. с англ. / Дж. Мэрфи. – М.: Диаграмма, 1998. – 600 с.
36. Уотшем, Т.Дж., Паррамоу К. Количественные методы в финансах: Учеб. Пособие для вузов: Пер. с англ. / под ред. М.Р.Ефимовой / Т. Дж. Уотшем, К. Паррамоу. – М.: Финансы, ЮНИТИ, 1999. – 527 с.
37. Гнеденко, Б.В. Курс теории вероятностей / Б.В. Гнеденко. – М.: Физматлит, 1961. – 406 с.
38. Боровков, А. А. Математическая статистика / А.А. Боровков. – М.: Физматлит, 2007. – 704 с.

39. Эфрон, Б. Нетрадиционные методы многомерного статистического анализа / Б. Эфрон. – М.: Финансы и статистика, 1988. – 263 с.
40. Андерсон, Т. Статистический анализ временных рядов / Т. Андерсон. – М.: Мир, 1976. – 751 с.
41. Бриллинджер, Д. Временные ряды. Обработка данных и теория / Д. Бриллинджер. – М.: Мир, 1980. – 536 с.
42. Четыркин, Е.М. Статистические методы прогнозирования / Е.М. Четыркин. – М.: Статистика, 1977. – 200 с.
43. Лукашин, Ю.П. Адаптивные методы краткосрочного прогнозирования / Ю.П. Лукашин. – М.: Финансы и статистика, 2003. – 416 с.
44. Armstrong, J.S. Forecasting for Marketing / J.S. Armstrong // Quantitative Methods in Marketing. London: International Thompson Business Press. – 1999. – P. 92-119.
45. Jingfei, Yang M. Sc. Power System Short-term Load Forecasting: Thesis for Ph.d degree / Yang M. Jingfei // Elektrotechnik und Informationstechnik der Technischen Universitat. Germany. Darmstadt. – 2006. – 139 P.
46. Rashid, T. Make Your Own Neural Network / T. Rashid. – Kindle Edition, 2016. – 252 p.
47. Nielsen, M. Neural Networks and Deep Learning/ M. Nielson [электронный ресурс], URL: http://neuralnetworksanddeeplearning.com/ (дата обращения 15.05.2018).
48. Segaran, T. Programming Collective Intelligence / T. Segaran. – O'Reilly Media, 2007. — 362 p.
49. Raschka, S. Python Machine Learning / S. Raschka. – Packt Publishing, 2015. – 454 p.
50. McKinney, W. Python for Data Analysis / W. McKinney. – O'Reilly Media, 2012. – 466 p.

Текст программы для подготовки исходных данных

Файл «Preparation.py»

```
import datetime
import csv
import tools
import math from tools
import StringToDatetime
import matplotlib.pyplot as plt

#============================
class HIST:
  DATESTART=datetime.datetime(2001,1,1) # начало отсчета времён
  def __init__(self, HistFile,Format):
    self.Format=Format
    self.inputFile=HistFile
    # индексы полей записи
    if(Format==1):
      # для формата <TICKER>,<DTYYYYMMDD>,<TIME>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>
Формат 1
      self.DAT=1
      self.TIME=2
      self.OPEN=3
      self.HIGH=4
      self.LOW=5
      self.CLOSE=6
      self.VOL=7
    elif(Format==2):
      #для формата <Datetime>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL> Формат 2
      self.DATETIME=0
      self.OPEN=1
      self.HIGH=2
      self.LOW=3
      self.CLOSE=4
      self.VOL=5
    elif(Format==3): # это формат уже преобразованный этим же классом из 1 или 2
      self.TIME=0 # время от self.DATESTART в минутах
      self.OPEN=1
      self.HIGH=2
      self.LOW=3
      self.CLOSE=4
      self.VOL=5
      self.dOPEN=6
      self.dHIGH=7 # первые разности
      self.dLOW=8
      self.dCLOSE=9
      self.HIGH_LOW=10 # разность HIGH-LOW
    else:
      print('HIST __init__: непредусмотренный формат файла')
    # для вычисления моментов - мат ожидание и сигма
    self.avgOPEN=0
    self.avgHIGH=1
    self.avgLOW=2
    self.avgCLOSE=3
    self.avgHL=4
    self.sgmOPEN=5
    self.sgmHIGH=6
```

```
  self.sgmLOW=7
  self.sgmCLOSE=8
  self.sgmHL=9

  self.cs=open(HistFile,newline='')
  self.rdr=csv.reader(self.cs)
  if(Format==1):
    next(self.rdr) # пропустить заголовки полей

#==============================
def ReadOneRec(self): # получить очередную запись в формате 3
  try:
    z=next(self.rdr)
  except StopIteration:
    return 0
  if(self.Format==1):
    dt=DateAndTimeToDatetime(z[self.DAT],z[self.TIME])
    minutes=(dt-self.DATESTART).total_seconds()/60
  elif(self.Format==2):
    dt=StringToDatetime(z[self.DATETIME])
    minutes=(dt-self.DATESTART).total_seconds()/60
  elif(self.Format==3):
    minutes=z[self.TIME]
  else:
    print("непредусмотренный формат файла в ReadOneRec()")
    return

  #первый элемент - число минут от DATESTART
  #d=dt-self.DATESTART;
  Rec=[int(minutes),float(z[self.OPEN]),float(z[self.HIGH]),
       float(z[self.LOW]),float(z[self.CLOSE]),int(z[self.VOL])]
  return Rec

def WriteFileInFormat3(self, outputFile):
  # записать файл в формате 3
  self.cs.close()
  self.cs=open(self.inputFile)
  self.rdr=csv.reader(self.cs)
  if(self.Format==1):
    next(self.rdr) # пропустить заголовки полей
  out=open(outputFile,"w",newline='')
  wrtr=csv.writer(out,delimiter=',')
  while (1):
    row=self.ReadOneRec()
    if(row==0):
          break
    wrtr.writerow(row)
  out.close()

def RemoveDefects(self, interval, outFile):
  # устранить дырки во времени так чтобы шаг времени был interval
  # применимо только к файлу в формате 3 (без разностей)
  # дырки в weekend остаются
  # остальные подменяются линейной интерполяцией
  # лишние отсчеты удаляются

  prev=self.ReadOneRec()
  out=open(outFile,"w",newline='')
  wrtr=csv.writer(out,delimiter=',')
```

```
  wrtr.writerow(prev)
  # индексы всех полей, которые будем интерполировать
  ind=[self.OPEN,self.HIGH,self.LOW, self.CLOSE, self.VOL]
  while(1):
    cur=self.ReadOneRec()
    if(cur==0):
      break #конец файла
    dt=cur[self.TIME]-prev[self.TIME]
    if(dt==interval): #дырки нет
      wrtr.writerow(cur)
      prev=cur
      continue
    if(dt < interval): #лишняя - игнорируем
      continue
    if(dt > 1500):  # weekend - не чиним
      wrtr.writerow(cur) # последняя запись недели
      prev=self.ReadOneRec() # первая запись новой недели
      if(prev==0):
        break
      continue
    # а вот это дырка
    x0=prev[self.TIME]
    x1=cur[self.TIME]
    dx=x1-x0
    # результат интерполяции по всем параметрам
    for x in range(x0+interval,x1,interval):
      # интерполируем все точки в обнаруженной дыре времени
      r=[]
      r.append(x)
      for j in range(len(ind)):
        y0=prev[ind[j]]
        y1=cur[ind[j]]
        y=round(y0+((y1-y0)/dx)*(x-x0),5)
        if(ind[j]==self.VOL):
              y=int(y)
        r.append(y)
      wrtr.writerow(r)
    # кончилась дырка
    wrtr.writerow(cur)
    prev=cur
  out.close()
  return

def AddDiffFields(self,outp):
  # добавить поля первых разностей и high-low к формату 3 без временных дыр
  # outp - выходной файл
  # в запись помещаются разности - текущая минус предыдущая
  # таким образом, первая запись недели в результат не попадает
  out=open(outp,"w",newline='')
  wrtr=csv.writer(out,delimiter=',')
  prev=self.ReadOneRec()
  while(1):
    cur=self.ReadOneRec()
    if(cur==0):
      break #конец файла
    if(cur[self.TIME]-prev[self.TIME]>1500): # новая неделя
      prev=cur
      continue
    cur.append(round(cur[self.OPEN]-prev[self.OPEN],5))
```

```
    cur.append(round(cur[self.HIGH]-prev[self.HIGH],5))
    cur.append(round(cur[self.LOW]-prev[self.LOW],5))
    cur.append(round(cur[self.CLOSE]-prev[self.CLOSE],5))
    cur.append(round(cur[self.HIGH]-cur[self.LOW],5))
    # добавили в запись поля
    # self.dOPEN=6,self.dHIGH=7,self.dLOW=8, self.dCLOSE=9 self.HIGH_LOW=10
    wrtr.writerow(cur)
    prev=cur
  out.close()
  return

def Moments(self):
  # вычисляет моменты для dOPEN,dHIGH,dLOW,dCLOSE и HIGH-LOW
  # исходным брать файл min<сколько минут>a.csv (например, min15a.csv) формат 3
  # индексы моментов определены в __init__
  mo=0
  mh=0
  ml=0
  mc=0
  mhl=0
  self.cs.close()
  self.cs=open(self.inputFile, newline='')
  self.rdr=csv.reader(self.cs)
  nRec=0
  while(1):
    try:
      cur=next(self.rdr)
    except StopIteration:
      break
    nRec=nRec+1
    mo=mo+float(cur[self.dOPEN])
    mh=mh+float(cur[self.dHIGH])
    ml=ml+float(cur[self.dLOW])
    mc=mc+float(cur[self.dCLOSE])
    mhl=mhl+float(cur[self.HIGH_LOW])
  mo=mo/nRec
  mh=mh/nRec
  ml=ml/nRec
  mc=mc/nRec
  mhl=mhl/nRec

  self.cs.close()
  self.cs=open(self.inputFile, newline='')
  self.rdr=csv.reader(self.cs)
  so=0
  sh=0
  sl=0
  sc=0
  shl=0
  while(1):
    try:
      cur=next(self.rdr)
    except StopIteration:
      break
    do=float(cur[self.dOPEN])
    dh=float(cur[self.dHIGH])
    dl=float(cur[self.dLOW])
    dc=float(cur[self.dCLOSE])
    dhl=float(cur[self.HIGH_LOW])
```

```
    p=do-mo
    so=so+p*p

    p=dh-mh
    sh=sh+p*p

    p=dl-ml
    sl=sl+p*p

    p=dc-mc
    sc=sc+p*p

    p=dhl-mhl
    shl=shl+p*p

  so=math.sqrt(so/(nRec-1))
  sh=math.sqrt(sh/(nRec-1))
  sl=math.sqrt(sl/(nRec-1))
  sc=math.sqrt(sc/(nRec-1))
  shl=math.sqrt(shl/(nRec-1))
  mm=[mo,mh,ml,mc,mhl,so,sh,sl,sc,shl]

  self.cs.close()
  return mm

def Normalize(self,outp):
  # нормировать первые разности для high, low и high-low
  # high и low: (значение-среднее)/(nSigma*сигма)
  # high-low: значение/(nSigma*сигма )
  # nSigma определено в классе. Либо 3, либо 1 (сейчас 3)
  mm=self.Moments()
  print("Moments()")
  self.cs=open(self.inputFile, newline='')
  self.rdr=csv.reader(self.cs)

  out=open(outp,"w",newline='')
  wrtr=csv.writer(out,delimiter=',')
  nSigma=3
  while(1):
    try:
      cur=next(self.rdr)
    except StopIteration:
      break

    o=(h-mm[self.avgOPEN])/(nSigma*mm[self.sgmOPEN])
    h=(h-mm[self.avgHIGH])/(nSigma*mm[self.sgmHIGH])
    l=(l-mm[self.avgLOW])/(nSigma*mm[self.sgmLOW])
    c=(h-mm[self.avgCLOSE])/(nSigma*mm[self.sgmCLOSE])
    hl=hl/(nSigma*mm[self.sgmHL])

    tim=int(cur[self.TIME])

    opn=float(cur[self.OPEN])
    high=float(cur[self.HIGH])
    low=float(cur[self.LOW])
    cls=float(cur[self.CLOSE])
    vol=int(cur[self.VOL])

    rec=[tim,opn,high,low,cls,vol,o,h,l,c,hl]
```

```
    wrtr.writerow(rec)
  out.close()

def GetOneCol(self,index):
  # извлекает 1 столбец (index) в список (для гистограмм и не только)
  self.cs.close()
  self.cs=open(self.inputFile, newline='')
  self.rdr=csv.reader(self.cs)
  arr=[]
  while(1):
    try:
      cur=next(self.rdr)
    except StopIteration:
      break
    arr.append(float(cur[index]))
  return arr

def OneWeekPerRow(self):
  # исходным является файл формат 3 с разностями
  # результатом является список (3мерный)
  # z[i][j][k] i-номер недели, j-номер записи от начала недели, k-номер элемента
  self.cs.close()
  self.cs=open(self.inputFile, newline='')
  self.rdr=csv.reader(self.cs)
  z=[]
  i=0
  week=[] # записи за неделю
  j=0
  prev=next(self.rdr)
  week.append(prev)
  while(1):
    try:
      cur=next(self.rdr)
    except StopIteration:
      break
    if(int(cur[self.TIME])-int(prev[self.TIME]) > 1500): # больше суток (1440)
      z.append(week)
      week=[]
      prev=next(self.rdr)
      week.append(prev)
      cur=next(self.rdr)
    else:
      week.append(cur)
      prev=cur
  return z

def RemoveShortWeeks(self,z):
  # z - список записей за недели
  # z[i][j][k] i-номер недели, j-номер записи от начала недели, k-номер элемента
  # удалить недели, в которых число записей меньше minWeek

  # ожидаемое число записей за неделю
  dt=int(z[0][1][self.TIME])-int(z[0][0][self.TIME]) # шаг времени
  mw=5*1440/dt*0.8 # 80% от пятидневки
  for i in range(len(z)-1,-1,-1):
    w=len(z[i])
    if(w < mw):
      z.pop(i)
```

```
  def PrintWeeksLen(self,z):
    nweek=len(z)
    print("Недель=",nweek)
    print("записей в неделях")
    for i in range(nweek):
      print(str(i)+" неделя записей "+str(len(z[i])))

  def TimeHoles(self):
    # дырки во времени
    # возвращает список моментов времени, когда
    # разность времени двух соседних отсчетов больше
    # шага временного ряда
    # шаг вычисляется по двум первым записям
    self.cs.close()
    self.cs=open(self.inputFile, newline='')
    self.rdr=csv.reader(self.cs)
    z=[]
    prev=next(self.rdr)
    cur=next(self.rdr)
    step=int(cur[self.TIME])-int(prev[self.TIME])
    prev=cur
    while(1):
      try:
        cur=next(self.rdr)
      except StopIteration:
        break
      d=int(cur[self.TIME])-int(prev[self.TIME])
      if(d > step):
        z.append([int(prev[self.TIME]), d/1440]) # второе значение - интервал в
сутках
      prev=cur
    return z

def FullTrans(inp,outp,interval):
  # полное преобразование файла из архива котировок в файл
  # где время в минутах, посчитаны разности и устранены дефекты
  tmp=" <расположение файлов с данными> "
  tmp2=" <расположение файлов с данными> "
  h1=HIST(inp,2)
  print("h1=HIST("+inp+",2)")
  h1.WriteFileInFormat3(tmp)
  print("h1.WriteFileInFormat3(tmp)")
  h2=HIST(tmp,3)
  h2.RemoveDefects(interval, tmp2)
  print("h2.RemoveDefects(interval, tmp2)")
  h3=HIST(tmp2,3)
  h3.AddDiffFields(outp)
  print("h3.AddDiffFields("+outp+")")

def FullTransForAllFiles():
  Dir=" <расположение файлов с данными> "
  InFiles=["EURUSD_1_2017-01-02_2017-10-15.csv",
         "EURUSD_5_2013-01-08_2017-10-31.csv",
         "EURUSD_15_2012-11-08_2017-10-15.csv",
         "EURUSD_30_2008-01-01_2017-10-15.csv",
         "EURUSD_60_2012-10-29_2017-10-15.csv",
         "EURUSD_240_2012-10-29_2017-10-15.csv",
         "EURUSD_1440_2012-10-08_2017-10-15.csv"]
  OutFiles=["min1a.csv","min5a.csv","min15a.csv","min30a.csv","min60a.csv",
```

```
        "min240a.csv","min1440a.csv"]
  step=[1,5,15,30,60,240,1440]
  for i in range(len(step)):
    FullTrans(Dir+InFiles[i],Dir+OutFiles[i],step[i])

def CalcMomentsForAllFiles():
  # вычислить моменты для всех временных интервалов
  files=["min1a.csv","min5a.csv","min15a.csv","min30a.csv","min60a.csv",
        "min240a.csv","min1440a.csv"]
  Dir="D:\Python\Архивы котировок\\"
  weight=[1e6,1e6,1e3,1e3,1e3,1e3] # веса (только для приличной печати)
  Mom=[]
  for i in range(len(files)):
    h=HIST(Dir+files[i],3)
    mm=h.Moments()
    Mom.append(mm)
    for j in range(len(mm)):
      mm[j]=round(mm[j]*weight[j],5)
    print(mm)
  return Mom
```

Файл «Tools.py»

```
import datetime

#----------------------------------------
def DatStringToDatetime(s):
  #получить datetime из строки вида '20171014'
  try:
    year=int(s[0:4])
    month=int(s[4:6])
    day=int(s[6:8])
    t = datetime.datetime(year,month,day)
    return t
  except:
    print('DatStringToDatetime(s) некорректный аргумент: ',s)

#-----------------------------------------------
def TimeStringToTimeDeltaSeconds(s):
  # получить timedelta из строки вида '211532'
  try:
    hour=int(s[0:2])
    minu=int(s[2:4])
    sec=0
    if(len(s)>=6):
          sec=int(s[4:6])
    delta=datetime.timedelta(seconds=3600*hour+60*minu+sec)
    return delta
  except:
    print('TimeStringToTimeDeltaSeconds(s) некорректный аргумент: ',s)

#-----------------------------------------------
def TimeStringToTimeDeltaMinutes(s):
  # получить timedelta из строки вида '2115??'
  try:
    hour=int(s[0:2])
    minu=int(s[2:4])
    delta=datetime.timedelta(minutes=60*hour+minu)
```

```
    return delta
  except:
        print('TimeStringToTimeDeltaMinutes(s) некорректный аргумент: ',s)

#-----------------------------------------------
def DateAndTimeToDatetime(d,s):
  return DatStringToDatetime(d)+TimeStringToTimeDeltaMinutes(s)

#----------------------------------------------------
def StringToDatetime(s):
  # строку вида '2012-10-29 04:45' к типу datetime
  t = datetime.datetime.strptime(s, '%Y-%m-%d %H:%M')
  return t
```

Текст программы для создания 3-мерных списков

Файл «d3List.py»

```
import csv
import Hist
import math
#----------------------------------------
def Write3dCSV(z,File):
  # записать 3х мерный список z[i][j][k] в файл File
  # записывается в виде:
  # число 2мерных списков в z
  # число записей в z[0]
  # записи из z[0]
  # ..................
  # число записей в z[1]
  # записи из z[1]
  # ..................
  out=open(File,"w",newline='')
  wrtr=csv.writer(out,delimiter=',')
  n=len(z)
  wrtr.writerow([n])
  for i in range(n):
    wrtr.writerow([len(z[i])])
    wrtr.writerows(z[i])
  out.close()
#-----------------------------------------------
def Read3dCSV(File):
  # читать 3х-мерный список, записанный функцией Write3dCSV
  # и возвратить его как результат
  z=[]
  inp=open(File, newline='')
  rdr=csv.reader(inp)
  r=next(rdr)
  nz=int(r[0])
  for i in range(nz):
    r=next(rdr)
    nzi=int(r[0])
    z.append([])
    for j in range(nzi):
      r=next(rdr)
      z[i].append(r)
  return z
#-----------------------------------------------
def Print3dList(z):
  # печатать 3х мерный список
  n=len(z)
  print(n)
  for i in range(n):
    m=len(z[i])
    print(m)
    for j in range(m):
      print(z[i][j])
#-----------------------------------------------
def WriteOne3dFile(infile,outfile):
  # получить из файла вроде min15a.csv трехмерный список по неделям min15w.csv
  print("Начинаю "+infile);
  h=Hist.HIST(infile,3)
```

```
  print("  перед выполнением OneWeekPerRow")
  z=h.OneWeekPerRow()
  print("  перед выполнением RemoveShortWeeks")
  h.RemoveShortWeeks(z)
  print("  перед выполнением Write3dCSV")
  Write3dCSV(z,outfile)
#-------------------------------------------------
def WriteAll3dFiles():
  # получить все 3хмерные понедельные списки
  # для 1440 что-то неверно и результат не получается
  Dir="<расположение файлов с данными>"
  InFiles=["min1a.csv","min5a.csv","min15a.csv","min30a.csv","min60a.csv",
         "min240a.csv"]
  OutFiles=["min1w.csv","min5w.csv","min15w.csv","min30w.csv","min60w.csv",
         "min240w.csv"]
  for i in range(len(InFiles)):
    infile=Dir+InFiles[i]
    outfile=Dir+OutFiles[i]
    WriteOne3dFile(infile,outfile)

#=================================================
class WEEKS:
  def __init__(self, infile):
    self.infile=infile
    # индексы полей записи
    self.TIME=0 # время от self.DATESTART в минутах
    self.OPEN=1
    self.HIGH=2
    self.LOW=3
    self.CLOSE=4
    self.VOL=5
    self.dOPEN=6
    self.dHIGH=7 # первые разности
    self.dLOW=8
    self.dCLOSE=9
    self.HIGH_LOW=10 # разность HIGH-LOW

    self.WeekArr=Read3dCSV(self.infile) # 3х мерный список
    # шаг времени
    self.Step=int(self.WeekArr[0][1][self.TIME])-
int(self.WeekArr[0][0][self.TIME])

#-------------------------------------------------
  def WeekOneValueMoments(self, WeekNumb, ValIndex):
    # вычисление среднего и сигма за неделю
    # WeekArr[i][j][k] - массив, i-номер недели, j-номер записи от начала недели,
k-номер элемента
    # WeekNumb - номер недели
    # ValIndex - индекс значения для которого выполняются вычисления (dHIGH...)
    mv=0 # среднее за неделю
    d=0 # дисперсия
    nw=len(self.WeekArr[WeekNumb])
    for i in range(nw):
      mv=mv+float(self.WeekArr[WeekNumb][i][ValIndex])
    mv=mv/nw
    for i in range(nw):
      dd=float(self.WeekArr[WeekNumb][i][ValIndex])-mv
      d=d+dd*dd
    sigma=math.sqrt(d/(nw-1))
```

```
    return [mv,sigma]
#-------------------------------------------------
  def AvgSigma(self,ValIndex):
    # усреднённые среднее и сигма за все недели для поля ValIndex
    nWeek=len(self.WeekArr)
    s=0
    m=0
    for i in range(nWeek):
      mm=self.WeekOneValueMoments(i,ValIndex)
      m=m+mm[0]
      s=s+mm[1]
    m=m/nWeek
    s=s/nWeek
    return [m,s]
#-------------------------------------------------
  def AutoCorrOneWeek(self,WeekNumb,ValIndex):
    # автокорреляционная функция для недели
    # WeekNumb и поля ValIndex
    nw=len(self.WeekArr[WeekNumb]) # число записей в неделе
    wm=self.WeekOneValueMoments(WeekNumb,ValIndex)
    arr=self.WeekArr[WeekNumb]
    corr=[]
    for d in range(int(2*nw/3)): # max разница номеров отсчетов - (2/3)nRec
      m=0 # накапливаемая сумма произведений
      i1=0 # индекс первого сомножителя
      n=0 # число слагаемых
      while(1):
        i2=i1+d # индекс второго сомножителя
        if(i2 >= nw):
              break
        m1=float(arr[i1][ValIndex])-wm[0] # вычли среднее
        m2=float(arr[i2][ValIndex])-wm[0] # вычли среднее
        m=m+m1*m2
        n=n+1
        i1=i1+1
      # нормируем
      m=m/(n-1)/(wm[1]*wm[1]) # для d=0 должно получиться 1
      corr.append(m)
    return corr
#-------------------------------------------------
  def AvgAutoCorr(self,ValIndex):
    # усреднённая по всем неделям автокорреляционная функция
    # для поля ValIndex
    nWeek=len(self.WeekArr)
    WeekLen=[]
    a=[]
    maxweeklen=0
    #return nWeek
    for i in range(nWeek):
      w1=self.AutoCorrOneWeek(i,ValIndex)
      #return(w1)
      l=len(w1)
      WeekLen.append(l)
      if(maxweeklen < WeekLen[i]):
        maxweeklen=l
      a.append(w1)
    res=[]
    for d in range(maxweeklen): # d - аргумент автокорр ф-ии
      s=0
```

```
      n=0
      for i in range(nWeek):
        if(d < len(a[i])):
          n=n+1
          s=s+a[i][d]
      s=s/n
      res.append(s)
    return res

#=======================================================
def CalcAllSigma():
  #средние 10^3*sigma по всем разностям и всем файлам
  Dir="<расположение файлов с данными>"

Files=["min240w.csv","min60w.csv","min30w.csv","min15w.csv","min5w.csv","min1w.csv
"]
  w=WEEKS(Dir+Files[0])
  Indexes=[w.dOPEN,w.dHIGH,w.dLOW,w.dCLOSE,w.HIGH_LOW]
  indNames=["dOPEN","dHIGH","dLOW","dCLOSE","HIGH_LOW"]
  for i in range(len(Files)):
    w=WEEKS(Dir+Files[i])
    print("")
    for j in range(len(Indexes)):
      m=w.AvgSigma(Indexes[j])
      print(Files[i]+"  "+indNames[j]+"  "+str(round(1000*m[1],5)))

#=========================================================
def AllAvgAutoCorr():
  # все усреднённые автокорреляционные ф-ии
  # пишем каждую в отдельный файл
  Dir="<расположение файлов с данными>"
  Mins=["240","60","30","15","5","1"]
  Files=[]
  for i in range(len(Mins)):
        Files.append("min"+Mins[i]+"w.csv")
  w=WEEKS(Dir+Files[0])
  Indexes=[w.dOPEN,w.dHIGH,w.dLOW,w.dCLOSE,w.HIGH_LOW]
  indNames=["dOPEN","dHIGH","dLOW","dCLOSE","HIGH_LOW"]
  for i in range(len(Files)):
    w=WEEKS(Dir+Files[i])
    print("")
    print(Files[i])
    for j in range(len(Indexes)):
      r=w.AvgAutoCorr(Indexes[j])
      outputFile=Dir+"AutoCorr_"+Mins[i]+"_"+indNames[j]+".csv"
      out=open(outputFile,"w",newline='')
      wrtr=csv.writer(out,delimiter=',')
      wrtr.writerow(r)
      out.close()
      print("Записан файл "+outputFile)
```

Текст программы для расчета описательных статистик, построения линейных графиков и гистограмм

Файл «DescriptionStatistics.py»

```
import numpy as np
import pandas as pd
import scipy.stats as st
import matplotlib.pyplot as plt

dataset = pd.read_csv('<расположение файлов с данными>',sep=',', \
            names=['Date','OPEN','HIGH','LOW','CLOSE', \
            'VOL','dOPEN','dHIGH','dLOW','dCLOSE','HIGH_LOW'], \
            #usecols=['Date','dOPEN','dHIGH','dLOW','dCLOSE'],
            index_col=0)

df=pd.DataFrame(dataset)
group1 = ['OPEN','HIGH','LOW','CLOSE']
group2 = ['dOPEN','dHIGH','dLOW','dCLOSE']

def Plots():
    fig1=plt.figure(1)
    plt.plot(df['CLOSE'])
    plt.xlabel('Date')
    fig1.autofmt_xdate()
    plt.title('CLOSE plot for interval 1 minute')

    fig2=plt.figure(2)
    plt.plot(df['dCLOSE'])
    plt.xlabel('Date')
    fig2.autofmt_xdate()
    plt.title('dCLOSE plot for interval 1 minute')

##Plots()

def PlotsOHLC():
    plt.figure()
    plt.title("OHLC plot for interval 60 minutes")
    for name in group1:
        data_2=df[name]
        data_2.plot()
    plt.legend(loc='best')

##PlotsOHLC()

# Характеристики
#=========================================================
def Characteristics():
    for name in group1:
        data=df[name]

        length=len(data) # длина выборки
        mean=data.mean() #средняя
        minimum=data.min()
        maximum=data.max()
        median=data.median() #медиана
        std=data.std() #стандартное отклонение
        variance=data.var() #Дисперсия
```

```
        coef_var = std/mean #коэффициент вариации
        skew=st.skew(data) #Ассиметрия (скос)
        kurtosis=st.kurtosis(data) #Куртосиз

        interval=st.t.interval(0.95, len(data)-1, loc=median, scale=st.sem(data))

        print ("Описательная статистика для "+name+":")
        print ("")
        print ("Число элементов выборки: ", length)
        print ("Среднее значение: %0.7f" %mean)
        print ("Медиана: %0.7f" %median)
        print ("Дисперсия: %0.7f" %variance)
        print ("Минимальное и максимальное значения = ( %0.4f" %minimum, ", %0.4f"
%maximum, ")")
        print ("Стандартное отклонение: %0.4f" %std)
        print ("Коэффициент вариации (Пирсона): %0.4f" %coef_var)
        print ("Коэффициент асимметрии: %0.4f" %skew)
        print ("Коэффициент эксцесса: %0.4f" %kurtosis)
        print ("")

##Characteristics()

# Гистограммы
#=========================================================
def Histograms():
    counter = 1
    plt.figure()
    for name in group2:
        data=df[name]
        std=data.std() #стандартное отклонение
        data_sort=sorted(data)
        plt.subplot(2, 2, counter)
        fit=st.norm.pdf(data_sort,np.mean(data_sort),np.std(data_sort))
        plt.plot(data_sort,fit)
        plt.hist(data,bins=30, normed=True)
        plt.xlim(-5*std, 5*std)
        plt.title(name)
        counter += 1

##Histograms()

# Тест Жарка-Бера
#=========================================================
import statsmodels.api as sm
def test_JB():
    for name in group2:
        data=df[name]
        jb_results = sm.stats.stattools.jarque_bera(data)
        jb=jb_results[0] #Жарк-Бера
        p_value=jb_results[1] #Вероятность
        skew=jb_results[2] #Ассиметрия (скос)
        kurtosis=jb_results[3] #Куртосиз

        print ("Тест Жарк-Бера для "+name+":")
        print ("")

##        print (jb)
##        print (p_value)
##        print (skew)
```

```
##        print (kurtosis)

        print ("Жарк-Бера = ", jb)
        print ("Вероятность = ", p_value)
        print ("Асимметрия (скос) = ", skew)
        print ("Куртосиз = ", kurtosis)
        print ("")

##test_JB()

# Среднее значение и дисперсия
#=========================================================
def MeanAndVariance():
    counter=1
    for name in group1:
        data = df[name].values
        p=1000
        n = int(len(data)/p)
        X=np.zeros((n,p))
        mean=[]
        var=[]
        for i in range(n):
            X[i][:]=data[i*p:p*(i+1)]
            m=X[i][:].mean()
            v=X[i][:].var()
            mean.append(m)
            var.append(v)
        ##print(mean)
        ##print(var)
        plt.figure(counter)
        plt.plot(mean)
        plt.title('Mean '+name)
        plt.figure(counter+1)
        plt.plot(var)
        plt.title('Variance '+name)
        counter+=2

##MeanAndVariance()

# Тест Дикки-Фуллера
#=========================================================
def test_DF():
    counter=[1, 2]
    for name in group2:
        data = df[name].values
        print ("Тест Дикки-Фуллера для ряда "+name)
        df_results = sm.tsa.adfuller(data)

        print ("adf: " , df_results[0])
        print ("p-value: " , df_results[1])
        print ("Critical values: " , df_results[4])

        if (df_results[0] > df_results[4]['5%']):
            print ("есть единичные корни, ряд не стационарен")
        else:
            print ("единичных корней нет, ряд стационарен")
        print ("")

##test_DF()
```

```
# Зависимость сигма от шага времени для dCLOSE
#=========================================================
def sigma_t():
    x=[1,5,15,30,60,240]

    y=[0.1410,0.3675,0.6231,0.8708,1.2382,2.4300]
    y1=[0.1570,0.3510,0.6080,0.8599,1.2160,2.4320]

    Y=[i**2 for i in y]
    Y1=[i**2 for i in y1]

    plt.plot(x,Y,'ro-',linewidth=1,markersize=2)
    plt.plot(x,Y1,'ko-',linewidth=1,markersize=2)
    plt.xticks(x)
    plt.title("Variance")
    plt.xlabel('T')
    plt.ylabel('Sigma^2')

##sigma_t()

plt.show()
```

ПРИЛОЖЕНИЕ 4

Текст программы для построения коррелограм

Файл «Correlogram.py»

```
import numpy as np
import pandas as pd
from pandas.tools.plotting import autocorrelation_plot
import matplotlib.pyplot as plt

Dir="<расположение файлов с данными>"

def Interval(n,z95,z99):
    x = [0,n]
    y0=[0,0]
    y95 = [z95,z95]
    y99 = [z99,z99]
    yy95 = [-z95,-z95]
    yy99 = [-z99,-z99]
    plt.plot(x,y99,'g--',linewidth=1)
    plt.plot(x,y95,'k',linewidth=1)
    plt.plot(x, y0, 'k',linewidth=1)
    plt.plot(x,yy95,'k',linewidth=1)
    plt.plot(x,yy99,'g--',linewidth=1)

def Correlogram1():
    Files=["dOPEN", "dHIGH", "dLOW", "dCLOSE"]
    m=240
    data=pd.read_csv(Dir+str(m)+"_dCLOSE"+".csv",sep=',',index_col=0)
    n=(len(data)-1)*m
    z95 = 1.959963984540054 / np.sqrt(n)
    z99 = 2.5758293035489004 / np.sqrt(n)
    counter = 1
    for f in Files:
        dataset = pd.read_csv(Dir+m+f+".csv",sep=',',index_col=0)
        plt.subplot(2, 2, counter)
        plt.plot(dataset)
        Interval(n,z95,z99)
        plt.title(f)
        plt.ylim(-0.1,1.1)
        counter += 1
    plt.show()
Correlogram1()

def Correlogram2():
    Files=[1, 5, 15, 30,60,240]
    counter = 1
    for f in Files:
        dataset = pd.read_csv(Dir+str(f)+"_ds_H_L.csv",sep=',',index_col=0)
        n=(len(dataset)-1)*f
        z95 = 1.959963984540054 / np.sqrt(n)
        z99 = 2.5758293035489004 / np.sqrt(n)
        plt.subplot(2, 3, counter)
        plt.plot(dataset)
        Interval(n,z95,z99)
        plt.title(f)
        counter += 1
    plt.show()
Correlogram2()
```

Текст программы для решения задачи прогнозирования на регулярной функции

Файл «ForecastFunc.py»

```
import matplotlib.pyplot as plt
import numpy as np
import random
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from keras import optimizers
from keras.layers.recurrent import LSTM

PI=3.14159265358979323846

def f(x):
    v=np.sin(np.sqrt(2.)*x*PI)+\
    np.sin(np.sqrt(3.)*x*0.3*PI+0.75*PI)+ \
    np.sin(2*x*np.sqrt(0.3)*PI)
    return v

def plotSin():
    nRow=1000
    dx=0.1
    X=[]
    np.random.seed(17)
    x0=1000*np.random.random()
    for i in range(nRow):
        x=x0+i*dx
        X.append(x)
    X=np.asarray(X)
    Y=f(X)
    plt.plot(X,Y)
    plt.title('Смесь синусоид с некратными периодами')
    plt.xlabel('X')
    plt.ylabel('f(X)')
    plt.show()

def MaxIncrement(x1, nX, dx):
    #возвращает максимальный прирост
    #функции от x1 с шагом dx на nX последующих точках
    y0=f(x1) # от y0 расчёт прироста
    max_dy=-1e38
    for i in range(1,nX+1):
        x=x1+i*dx
        yy=f(x)
        dy=yy-y0
        if dy>max_dy:
            max_dy=dy
    return max_dy

def getRow(x0,dx,nInp,nFuture):
    # получить nInp значений ф-ии f(x) для x от x0 с шагом dx
    # затем посчитать - максимальное приращение на последующих nFuture значениях
ф-ии
    y=[]
```

```
    y.append(x0) # x0 - первое число в строке, дальше идут значения функции
    for i in range(nInp):
        x=x0+i*dx
        y.append(f(x))
    x1=x
    v=MaxIncrement(x1,nFuture,dx)
    y.append(v)

    return np.asarray(y)

def getRows(dx,nInp,nRow,nFuture):
    #получить матрицу nRow строк значений функции f(x),
    #строка от случайного x0, nY значений функции
    #шаг аргумента dx
    r=[]
    for i in range(nRow):
        x0=1000*np.random.random() # 1000 - произвольно
        y=getRow(x0,dx,nInp,nFuture)
        r.append(y)
    return np.asarray(r)

nRow=1000
dx=0.1 # шаг по оси абсцисс
nInp=40 # число точек в строке, используемых для предсказания. Оно же число входов
nFuture=12 # число точек, непосредственно следующих за nInp на которых регистрируется прирост
nDense=24 #число нейронов в слое - здесь, кроме выхода, везде одинаковое
UseBias=True
nEpoch=100

# создание обучающего массива
np.random.seed(17)
m=getRows(dx,nInp,nRow,nFuture)
#делим m на входы и выходы
inp=m[:,1:nInp+1]
out=m[:,nInp+1]

#модель нейросети
def baseline_model():
    model = Sequential()
    model.add(Dense(nDense, use_bias=UseBias, input_dim=nInp, activation="tanh"))
    model.add(Dense(nDense, use_bias=UseBias, activation="tanh"))
    model.add(Dense(nDense, activation="tanh"))
    model.add(Dense(1, activation='linear'))
    sgd = SGD(lr=0.01, momentum=0.9, nesterov=True)
    model.compile(loss='mse', optimizer='rmsprop', metrics=['mse'])
    return model

model=baseline_model()

#обучение

model.fit(inp,out, epochs=nEpoch, verbose=0)

def TestModel(x,y,nToPrint):
    # построчно подаём входы, печатаем что должно было получиться и реальный выход
    n=0 # число ошибок
    nRow=len(x)
    Err=0
```

```
    SumRazmah=0
    predicted=[]
    for i in range(nRow):
        p=np.reshape(x[i],(1,nInp))
        u=model.predict(p,batch_size=1)
        u=np.reshape(u,(1))
        predicted.append(u)
        miny=1e38
        maxy=-1e38
        for j in range(len(x[i])):
            if(x[i][j]>maxy):
                maxy=x[i][j]
            if(x[i][j]<miny):
                miny=x[i][j]
        razmah=maxy-miny
        SumRazmah=SumRazmah+razmah
        d=y[i]-u
        if i<nToPrint:
            print(y[i],u,d)
        Err=Err+d*d
        n=n+1
    predicted=np.reshape(predicted,(len(y),1))
    sigma=np.sqrt(Err/n)
    SredRazmah=SumRazmah/n
    print('среднеквадратичная ошибка=', sigma, '  Диапазон=',SredRazmah)
    print('Ошибка',sigma/SredRazmah)
    plt.plot(y)
    plt.plot(predicted)
    plt.xlabel('X')
    plt.ylabel('f(X)')
    plt.show()
    return predicted

nToPrint=10 # сколько строк печатать

print('\nТест на обучающей последовательности')
print('Ожидаемые значения', 'Предсказанные значения ', 'Разница')
pred1=TestModel(inp,out,nToPrint)

#создадим пробную последовательность
np.random.seed(379)
m=getRows(dx,nInp,nRow,nFuture)
#делим m на входы и выходы
inp=m[:,1:nInp+1]
out=m[:,nInp+1]

print('Тест на пробной последовательности')
print('Ожидаемые значения', 'Предсказанные значения ', 'Разница')
pred2=TestModel(inp,out,nToPrint)
```

Текст программы для решения задачи классификации на регулярной функции

Файл «ClassificationFunc.py»

```
import matplotlib.pyplot as plt
import numpy as np
import random
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers

PI=3.14159265358979323846

dx=0.1 # шаг по оси абсцисс
delta=3. # граница прироста для TakeProfit и StopLoss
nInp=40 # число точек в строке, используемых для предсказания. Оно же число входов
nFuture=12  # число точек, непосредственно следующих за nInp на которых регистрируется прирост
nDense=24 #число нейронов в слое - здесь, кроме выхода, везде одинаковое
UseBias=True

nRow=10000
nEpoch=100
nToPrint=10 # сколько строк печатать

def f(x):
    v=np.sin(np.sqrt(2.)*x*PI)+\
    np.sin(np.sqrt(3.)*x*0.3*PI+0.75*PI)+ \
    np.sin(2*x*np.sqrt(0.3)*PI)
    return v

def getRow(x0,dx,nInp,nFuture):
    # получить nInp значений ф-ии f(x) для x от x0 с шагом dx
    # затем определить в какую сторону приросот на delta
    y=[]
    for i in range(nInp):
        x=x0+i*dx
        y.append(f(x))
    x1=x
    lasty=f(x)

    for j in range(1,nFuture+1):
        v0=0 # будет 1 если за nFuture будет прирост delta
        v1=0 # ---||--- 1 -delta
        v2=0 # будет 1 если ни +delta ни -delta не будет достигнуто

        fx=f(x1+j*dx)
        if fx-lasty>delta:
            v0=1
            break
        if fx-lasty<-delta:
            v1=1
            break
        v2=1
    y.append(v0)
    y.append(v1)
```

```
    y.append(v2)
    return np.asarray(y)

def getRows(dx,nInp,nRow,nFuture):
    #получить матрицу nRow строк значений функции f(x),
    #строка от случайного x0, nY значений функции
    #шаг аргумента dx
    r=[]
    for i in range(nRow):
        x0=1000*np.random.random() # 1000 - произвольно
        y=getRow(x0,dx,nInp,nFuture)
        r.append(y)
    return np.asarray(r)

# создание обучающего массива
np.random.seed(17)
m=getRows(dx,nInp,nRow,nFuture)
#делим m на входы и выходы
inp=m[:,0:nInp]
out=m[:,nInp:nInp+3]

#модель нейросети
def baseline_model(): #, kernel_initializer="glorot_normal"
    model = Sequential()
    model.add(Dense(nDense, use_bias=UseBias, input_dim=nInp, activation="tanh"))
    model.add(Dense(nDense, use_bias=UseBias, activation="tanh"))
    model.add(Dense(nDense, activation="tanh"))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy',            optimizer='rmsprop',
metrics=['accuracy'])
    return model

model=baseline_model()

# training
model.fit(inp,out, epochs=nEpoch, verbose=0)

def TestModel(x,y,nToPrint):
    # построчно подаём входы, печатаем что должно было получиться и реальный выход
    n=0 # число ошибок
    nRow=len(x)
    m=np.zeros((3,3),dtype=np.int32)
    #print(m)
    for k in range(nRow):
        p=np.reshape(x[k],(1,nInp))
        u=model.predict(p,batch_size=1)
        u=np.reshape(u,(3))  # u - вероятности, которые выдаёт модель
        # в y[] то что было на самом деле
        # заполним матрицу m
        for i in range(3):
            for j in range(3):
                if y[k][i]==1 and u[j]>0.5:
                    # 1 добавляется к m[i][j] если мы предсказывали i-й класс, а
получился j-й
                    m[i][j]+=1
        if i<nToPrint:
            print(y[k][0],u[0],'  ',y[k][1],u[1],'  ',y[k][2],u[2],'  ')
    print('Количественная матрица вероятностей\n',m)
    return m
```

```
print('\nТест на обучающей последовательности');
t=TestModel(inp,out,nToPrint)

res=np.zeros((3,3))
t0=t[0][0]+t[0][1]+t[0][2]
t1=t[1][0]+t[1][1]+t[1][2]
t2=t[2][0]+t[2][1]+t[2][2]
for i in range(3):
    res[0][i]=t[0][i]/t0
    res[1][i]=t[1][i]/t1
    res[2][i]=t[2][i]/t2
print('Матрица вероятностей\n', res)

#создадим пробную последовательность
np.random.seed(379)
m=getRows(dx,nInp,nRow,nFuture)
#делим m на входы и выходы
inp=m[:,0:nInp]
out=m[:,nInp:nInp+3]

print('\r\nТест на пробной последовательности');
tt=TestModel(inp,out,nToPrint)
ress=np.zeros((3,3))
tt0=tt[0][0]+tt[0][1]+tt[0][2]
tt1=tt[1][0]+tt[1][1]+tt[1][2]
tt2=tt[2][0]+tt[2][1]+tt[2][2]
for i in range(3):
    ress[0][i]=tt[0][i]/tt0
    ress[1][i]=tt[1][i]/tt1
    ress[2][i]=tt[2][i]/tt2
print('Матрица вероятностей\n', ress)
```

Текст программы для решения задачи прогнозирования на реальных временных рядах Forex

Файл «ForecastForex.py»

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers

PI=3.14159265358979323846

df = pd.read_csv('min30w.csv', sep=',',header=None)
df.columns=['Date','OPEN','HIGH','LOW','CLOSE',
'VOL','dOPEN','dHIGH','dLOW','dCLOSE','HIGH_LOW']
columns=['dOPEN','dHIGH','dLOW','dCLOSE']
column=columns[3]

nRow=len(df)
nInp=20 # число точек в строке, используемых для предсказания. Оно же число входов
nFuture=6  # число точек, непосредственно следующих за nInp на которых регистрируется прирост
nDense=40 #число нейронов в слое - здесь, кроме выхода, везде одинаковое
UseBias=False
nEpoch=100

def MaxIncrement(x1, nX, fx1):
    #возвращает максимальный прирост
    #функции от x1 с шагом dx на nX последующих точках
    y0=fx1 # от y0 расчёт прироста
    max_dy=-1e38
    for i in range(x1,x1+nFuture):
        yy=df[column][i]
        dy=yy-y0
        if dy>max_dy:
            max_dy=dy
    return max_dy

def getRows(column,nRow,nInp,nFuture):
    y=[]
    for i in range(nRow):
        nY=df['Date'][i]
        if(nY==236): # or 26/116/236/476...
            x=i+1+nInp # индекс элемента, следующего за nInp
            data=pd.DataFrame(data=df[column][i+1:i+1+nInp]).as_matrix()
            y_i = np.ravel(data)

            x1=x
            lasty=y_i[-1] # значение последнего элемента в nInp
            v=MaxIncrement(x1,nFuture,lasty)

            y_i=np.append(y_i,v)
            y.append(y_i)
```

```
    return np.asarray(y)

# создание обучающего массива
m=getRows(column,nRow,nInp,nFuture)
#делим m на входы и выходы
nTrain=int(0.7*len(m)) # отсчетов для тренировки
inp=m[0:nTrain,1:nInp+1]
out=m[0:nTrain,-1] # последний элемент в каждой строке

#модель нейросети
def baseline_model(): #, kernel_initializer="glorot_normal"
    model = Sequential()
    model.add(Dense(nDense, use_bias=UseBias, input_dim=nInp, activation="tanh"))
    model.add(Dense(nDense, use_bias=UseBias, activation="tanh"))
    model.add(Dense(nDense, activation="tanh"))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mse', optimizer='rmsprop', metrics=['mse'])
    return model

model=baseline_model()

#обучение
history=model.fit(inp,out, epochs=nEpoch, verbose=0, validation_split=0.1)

def TestModel(x,y,nToPrint):
    # построчно подаём входы, печатаем что должно было получиться и реальный выход
    n=0 # число ошибок
    nRow=len(x)
    Err=0
    SumRazmah=0
    predicted=[]
    for i in range(nRow):
        p=np.reshape(x[i],(1,nInp))
        u=model.predict(p,batch_size=1)
        u=np.reshape(u,(1))
        predicted.append(u)
        miny=1e38
        maxy=-1e38
        for j in range(len(x[i])):
            if(x[i][j]>maxy):
                maxy=x[i][j]
            if(x[i][j]<miny):
                miny=x[i][j]
        razmah=maxy-miny
        SumRazmah=SumRazmah+razmah
        d=y[i]-u
        if i<nToPrint:
            print(y[i],u,d)
        Err=Err+d*d
        n=n+1
    predicted=np.reshape(predicted,(len(y),1))
    sigma=np.sqrt(Err/n)
    SredRazmah=SumRazmah/n
    print('среднеквадратичная ошибка=', sigma, '  Диапазон=',SredRazmah)

    plt.plot(y)
    plt.plot(predicted)
    plt.show()
    return predicted
```

```
nToPrint=10 # сколько строк печатать

print('\nТест на обучающей последовательности');
TestModel(inp,out,nToPrint)

counter = 1
for name in columns:
    #создадим пробную последовательность
    #column=columns[1] # попробуем на данных dHIGH
    m=getRows(name,nRow,nInp,nFuture)
    #делим m на входы и выходы
    inp=m[nTrain:len(m),1:nInp+1]
    out=m[nTrain:len(m),-1] # последний элемент в каждой строке

    print('Тест на пробной последовательности ' + name);
    t=TestModel(inp,out,nToPrint)
    print(history.history.keys())

    plt.subplot(2, 2, counter)_mean_squared_error'], label='test')
    plt.plot(out)
    plt.plot(t)
    plt.xlim(0,50)
    plt.title(name)
    counter += 1
plt.show()
```

Текст программы для решения задачи классификации на реальных временных рядах Forex

Файл «ClassificationForex.py»

```
import numpy as np
import csv
import matplotlib.pyplot as plt
import math
import numpy as np
import random
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
import time

# индексы столбцов массива
iTIME=0 # время от self.DATESTART в минутах
iOPEN=1
iHIGH=2
iLOW=3
iCLOSE=4
iVOL=5
idOPEN=6
idHIGH=7 # первые разности
idLOW=8
idCLOSE=9
iHIGH_LOW=10 # разность HIGH-LOW
#заголовки
ColName=['TIME','OPEN','HIGH','LOW','CLOSE','VOL','dOPEN','idHIGH','idLOW','idCLOS
E','iHIGH_LOW']

#настройки
FileDir='<расположение файлов с данными>'
delta=0.003 # граница приращения
TimeKvant=240 # минут между отсчетами
ColList=[idHIGH,idLOW,idCLOSE]
PercentForTrain=70 # процент недель для тренировки
nDense=40 #число нейронов в слое - здесь, кроме выхода, везде одинаковое
BoundProbability=0.55 # если будет предсказано больше, то играем
nLearnPoint=24 # число отсчетов для обучения
nFuturePoint=12 # число отсчетов по которым определяется достигнуто ли +-delta
nEpoch=100 # число эпох
nToPrint=0 # сколько строк печатать для посмотреть
UseBias=False

def Read3dCSV(File):
      # читать 3х-мерный список, записанный функцией Write3dCSV
      # и возвратить его как результат
      z=[]
      inp=open(File, newline='')
      rdr=csv.reader(inp)
      r=next(rdr)
      nz=int(r[0])
      for i in range(nz):
            r=next(rdr)
            nzi=int(r[0])
```

```
            z.append([])
            for j in range(nzi):
                  r=next(rdr)
                  z[i].append(r)
      return z

def Convert3dTo2d(z):
    #список z содержит эл-ты z[i][j][k]
    # i - номер недели j-номер отсчета внутри недели; k - индекс элемента в
отсчете (Open, Close...)
    # функция сливает недели в v[i][j] i - номер отсчета, j - номер элемента
    v=[]
    for i in range(len(z)):
        for j in range(len(z[i])):
            v.append(z[i][j])
    return np.asarray(v)

def Print_nRows2d(v,Start,n):
      # печать нескольки строк отсчетов
      for i in range(Start,n+Start):
            print(v[i])

def PrepareNeuroInputFrom2d(v,From,ColumnList, nRow, nLearnPoint):
    # берём nRow строк начиная со строки From
    nv=len(v)
    w=v[From:From+nRow][:]
    nw=len(w)
    Rows=[]
    for i in range(nw-nLearnPoint-nFuturePoint): # i - стартовая точка
        Row=[]
        for j in range(nLearnPoint):
            for k in range(len(ColumnList)):
                l=i+j
                Row.append(w[l][ColumnList[k]])
            # теперь вычисляем исход по будущему
            # опеределим достигнут ли прирост +-delta
        dmin=0.
        dmax=0.
        v0=0. # прирост ни в ту ни в другую сторону не достигнут
        vplus=0. # достигнут прирост +delta
        vminus=0. # достигнут прирост -delta

        StartFuturePoint=i+nLearnPoint
        for j in range(StartFuturePoint,nw):
            d=float(w[j][idHIGH])
            dmax=dmax+d
            if dmax > delta:
                vplus=1.
                break
            d=float(w[j][idLOW])
            dmin=dmin+d
            if dmin < -delta:
                vminus=1.
                break
        if vplus==0. and vminus==0:
            v0=1.

        Row.append(vplus)
        Row.append(vminus)
```

```
        Row.append(v0)

        Rows.append(Row)
    return np.asarray(Rows)

#модель нейросети
def baseline_model(nInp):
    kint="glorot_normal"
    model = Sequential()
    model.add(Dense(nDense,            use_bias=UseBias,            input_dim=nInp,
activation="tanh",kernel_initializer=kint))
    model.add(Dense(nDense,                                          use_bias=UseBias,
activation="tanh",kernel_initializer=kint))
    model.add(Dense(nDense, activation="tanh",kernel_initializer=kint))
    model.add(Dense(3, activation='softmax',kernel_initializer=kint))
    model.compile(loss='categorical_crossentropy',            optimizer='rmsprop',
metrics=['accuracy'])
    return model

def IndexOfMaxValueInArray(arr):
    # индекс максимального значения в массиве arr
    k=0
    mx=arr[0]
    for i in range(1,len(arr)):
        if arr[i]>mx:
            k=i
            mx=arr[i]
    return k


def TestModel(model,x,y,nToPrint):
    # построчно подаём входы, печатаем что должно было получиться и реальный выход
    n=0 # число ошибок
    nRow=len(x)
    nInp=len(x[0])
    m=np.zeros((3,3),dtype=np.int32)
    for k in range(nRow):
        p=np.reshape(x[k],(1,nInp))
        u=model.predict(p,batch_size=1)
        u=np.reshape(u,(3))  # u - вероятности, которые выдаёт модель
        # в y[] то что было на самом деле
        # заполним матрицу m
        v=y[k]
        # i - индекс наибольшей вероятности в u
        i=IndexOfMaxValueInArray(u)
        if u[i]<BoundProbability:
            i=2 # не играем
        # j - индекс того события, которое на самом деле имело место (индекс
единицы)
        j=IndexOfMaxValueInArray(v)
        m[i][j]+=1
        if k<nToPrint:
            print(int(float(v[0])),u[0],'                ',int(float(v[1])),u[1],'
',int(float(v[2])),u[2],'  ')
    # нормируем матрицу m
    mm=np.zeros((3,3), dtype=np.float32)
    for i in range(3):
        s=float(sum(m[i]))
        if s>0:
```

```
            for j in range(3):
                mm[i][j]=float(m[i][j])/s
            else:
                mm[i][j]=0
    return m

def Write2dArr(z,File):
      # записать 2х мерный список z[i][j] в файл File
      out=open(File,"w",newline='')
      wrtr=csv.writer(out,delimiter=',')
      for i in range(len(z)):
            wrtr.writerow(z[i])
      out.close()

def Read2dArr(File):
      # читать 2х мерный список z[i][j] из файл File
      inp=open(File,"r",newline='')
      lines=inp.readlines()
      r=[]
      for s in lines:
            s = s.rstrip()
            s=s.split(',')
            s=np.asarray(s)
            #print(s)
            r.append(s)
      return np.asarray(r,dtype=np.float32)

def Print2dArr(arr):
      print('len(arr)=',len(arr))
      for i in range(len(arr)):
            print(arr[i])

##############################################
global_start_time = time.time()
# общая подготовка
Fil=FileDir+'min'+str(TimeKvant)+'w.csv'
z=Read3dCSV(Fil)
v2d=Convert3dTo2d(z)

# подготовка данных для тренировки
nv=len(v2d)
nTrain=int((PercentForTrain/100.0)*nv) # отсчетов для тренировки
nTest=nv-nTrain # строк для тестирования

# ПЕЧАТЬ НАСТРОЕК печать имен используемых столбцов исходного файла
Fields=''
Comma=''
FileDir='<расположение файлов с данными>'
for i in range(len(ColList)):
    Fields=Fields+Comma+ColName[ColList[i]]
    Comma=","

print("Файл ",Fil);
print("Поля вх. файла: ",s)
print('всего недель ',len(z))
print('Всего отсчетов',nv)
print('delta=',delta)
print('TimeKvant=',TimeKvant) # минут между отсчетами
print('PercentForTrain=',PercentForTrain) # процент недель для тренировки
```

```
print('nDense=',nDense,' - число нейронов в слое - здесь, кроме выхода, везде одинаковое')
print('BoundProbability=',BoundProbability,' - граничная вероятность. Если будет предсказано больше, то играем')
print('отсчетов для тренировки nTrain=',nTrain)
print('отсчетов для тестирования nTest=',nTest)
print('nLearnPoint=',nLearnPoint,' - число отсчетов для обучения')
print('nFuturePoint=',nFuturePoint,'  число  отсчетов  по  которым  определяется достигнуто ли +-delta')
print('nToPrint=',nToPrint,' сколько строк печатать для посмотреть')
print('UseBias=',UseBias)
print('nEpoch=',nEpoch)

nScalarInput=nLearnPoint*len(ColList) # всего входов нейросети
model=baseline_model(nScalarInput)

##########################################
# обучение
t=PrepareNeuroInputFrom2d(v2d,0,ColList, nTrain, nLearnPoint)
lent=len(t)
#print('строк в обучающей последовательности ', lent)
# делим t на входы и выходы
inp=t[0:nTrain,0:nScalarInput]
inp=np.asarray(inp, dtype=np.float32)
out=t[0:nTrain,nScalarInput:nScalarInput+3]

# training
model.fit(inp,out, epochs=nEpoch, verbose=0)

##############################################
# прогон на тестовой последовательности
print('\r\nТест на тестовой последовательности');
t=PrepareNeuroInputFrom2d(v2d,nTrain,ColList, nTest, nLearnPoint)
lent=len(t)
print('строк в тестовой последовательности lent=', lent)

# делим t на входы и выходы
inp=t[0:nTest,0:nScalarInput]
out=t[0:nTest,nScalarInput:nScalarInput+3]

#сохраним тестовую последовательность
Write2dArr(inp,"inp.txt")
Write2dArr(out,"out.txt")

m=TestModel(model,inp,out,nToPrint)

def PrintTestResults(m):
      print(m)

      res=np.zeros((3,3),dtype=np.float32)
      m0=m[0][0]+m[0][1]+m[0][2]
      m1=m[1][0]+m[1][1]+m[1][2]
      m2=m[2][0]+m[2][1]+m[2][2]
      for i in range(3):
          res[0][i]=round(m[0][i]/m0,4)
          res[1][i]=round(m[1][i]/m1,4)
          res[2][i]=round(m[2][i]/m2,4)
      print('Матрица вероятностей\n', res)
```

```
        Tries=m[0][0]+m[1][1]+m[0][1]+m[1][0]+m[0][2]+m[1][2]
        Win=m[0][0]+m[1][1]-m[0][1]-m[1][0]
        WinPerWeek=Win/float(len(z)) # выигрышей в неделю
        TriesPerWeek=Tries/float(len(z))
        print('Fields','TimeKvant','nLearnPoint','delta','BoundProbability',\
              'nDense','Win','TriesPerWeek','WinPerWeek',sep=";")
        print(Fields,TimeKvant,nLearnPoint,delta,BoundProbability, nDense,Win,\
              round(TriesPerWeek,3), round(WinPerWeek,3),sep=";")
        print ('')

print('по тестовым данным')
PrintTestResults(m)
print('Training duration (s) : ', time.time() - global_start_time)
```