Министерство науки и высшего образования 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
Направление подготовки: 01.03.04 Прикладная математика

РАБОТА ПРОВЕРЕНА
Рецензент, старший преподаватель
кафедры ИТвЭ
_____/В.В. Костерин
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____/А.А. Замышляева
«____»_____20___ г.

Прогнозирование нелинейных процессов на примере валютного рынка Forex

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.04.2019.84.ПЗ ВКР

Руководитель работы, доцент, к.х.н.
_____/Е.Ю. Алексеева
«____»_____2019 г.
Автор работы
Студент группы ЕТ-413
_____/ Н.С. Сергеев
«____»_____2019 г.
Нормоконтролер, доцент, к.т.н.
_____/Т.Ю. Оленчикова
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Сергеев Н.С. Прогнозирование нелинейных процессов на примере валютного рынка Forex. – Челябинск: ЮУрГУ (НИУ), ЕТ-413, 64 с., 35 ил., 11 табл., библиогр. список – 40 наим., 2 прил.

В работе рассмотрены методы прогнозирования нелинейных процессов на примере валютного рынка Forex.

В виде нелинейных процессов использованы финансовые временные ряды, т.е. котировки валютных пар рынка Forex. Для прогнозирования нелинейных процессов выбрана рекуррентная нейронная сеть. Так как рекуррентные нейронные сети страдают проблемой долгосрочной зависимости, использован тип рекуррентной нейронной сети LSTM. При тестировании прогнозирования обученные нейронные сети LSTM показали хорошие результаты, из чего можно сделать вывод, что они могут быть использованы в дальнейшем прогнозировании. Реализован интерфейс пользователя, подобраны оптимальные системные требования для запуска приложения и составлена инструкция для пользователя. Применение построенной нейронной сети LSTM к реальным данным показало, что среднеквадратичная ошибка достигает 3-5%.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С каждым днем международная торговля развивается все больше и больше, поэтому торговля валютой стала одной из самых распространенных видов деятельности. Рынок Forex постоянно меняется и возникают ситуации, когда поведение цены можно предсказать и заработать на этом.

Для предсказания используются методы фундаментального и технического анализа. Если фундаментальный анализ подразумевает сбор данных влияющих на последующую цену, например, данные об уровне инфляции, то технический анализ – анализ исторических данных для получения прогноза на будущее.

Нейронные сети в торговле относятся к техническому анализу, так как они занимаются поиском закономерностей на временном промежутке исторических данных. На рынке Forex также используются нейронные сети, на которых реализованы торговые советники (роботы).

Целью данной работы является прогнозирование нелинейных процессов (временных рядов) на примере валютного рынка Forex.

Задача работы заключается в анализе временных рядов с помощью нейронных сетей и получении последующего прогноза, который будет являться рекомендацией трейдеру по торговле на рынке Forex.

Для достижения поставленной цели необходимо воспользоваться нейронными сетями как одной из разновидностей нелинейных методов анализа для последующего прогнозирования.

1 МЕТОДЫ И СРЕДСТВА АНАЛИЗА И ПРОЗНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

1.1 Основные понятия рынка Forex

Рынок Forex (от англ. Foreign Exchange – «зарубежный обмен») – это финансовый рынок, суть которого заключается в обмене валют [31]. Данный рынок появился в 70-ые годы, когда все страны перешли на такую систему, в которой курс валют устанавливался рынком, а не государством. Банки всего мира используют валютообменные операции на рынке Forex для того, чтобы совершать межгосударственные расчеты, спекулятивные сделки и т. д. Рынок Forex работает 24 часа в сутки, 5 дней в неделю, и не имеет конкретного места торговли. Все торги происходят через компьютеры в банках всего мира с помощью Сети Интернет. Основными участниками валютного рынка являются центральные банки, коммерческие банки, инвестиционные компании, валютные биржи и брокеры. За счет изменения курса валют, на рынке Forex формируется выручка.

Forex группирует в себе четыре региональных рынка: азиатский, европейский, американский и австралийский, каждый из которых имеет свою валюту. Основными валютами рынка Forex являются:

1) USD – доллар США ($);
2) EUR – евро (€);
3) JPY – японская йена (¥);
4) GBP – британский фунт стерлингов (£);
5) CHF – швейцарский франк (₣);
6) AUD – австралийский доллар ($);
7) CAD – канадский доллар ($);
8) NZD – новозеландский доллар ($).

Основные элементы сделки:

1) валюта;

2) дата совершения сделки;

3) контрагент;

4) обменный курс (Цена одной валюты, которая выражается в цене другой);

5) сумма;

6) дата реального обмена валют и платежные инструкции.

Валютная пара – пара, состоящая из двух валют, например, USD/RUB – пара, состоящая из доллара США и российского рубля. Котировка – количество единиц второй валюты, которые содержатся в одной единице первой. Например, USD/RUB равна 65.41, то есть, формируется отношением: 1 доллар США к 65.41 российским рублям. Котировки валютных пар являются временными рядами.

Пункт – изменение последней цифры в котировке. Например, если USD/RUB = 65.41 вырос на USD/RUB = 65.42, это означает, что доллар поднялся на один пункт.

Тик – это движение котировки в ту или другую сторону [25].

1.2 Способы представления временных рядов Forex

Временной ряд – последовательностью значений, которые получены в определенный момент времени [1].

Каждый временной ряд получается из двух компонент:

1) момент времени;

2) значение в этот момент времени.

Временные ряды также подразделяются на стационарные и нестационарные. Стационарные ряды – такие ряды остаются в равновесии относительно постоянного среднего уровня. Все остальные ряды являются нестационарными.

В широком смысле, стационарный ряд – это ряд, математическое ожидание которого не зависит от времени , а корреляционная функция

зависит от разности . Корреляционной функцией называют математическое ожидание произведения отклонений ряда от среднего. В узком смысле, стационарный временной ряд – одинаковое распределение величин и для любых и .

Одним из наиболее распространенных методов моделирования нестационарных временных рядов называется параметрическое оценивание. Для его реализации необходимо подобрать параметры такой функциональной зависимости (полиномиальную, квазиполиномиальную, с экспоненциальными множителями и т. д.), исключив которую ряд остается стационарным. Тот ряд, который остался, принято считать с доверительной вероятностью. Для проверки на стационарность используют всевозможные тесты. Например, для нормально распределенных случайных величин тест проводится по критериям Фишера и Стьюдента.

Нестационарные ряды могут быть сведены к стационарным, используя:

1) теорему Гливенко о сходимости эмпирической вероятности к распределению генеральной совокупности;

2) критерий согласия Колмогорова о близости выборочной функции распределения и распределения генеральной совокупности.

Предположим, что – количество элементов выборки объема Т:

, (1.1)

где – объем выборки.

Данные элементы из выборки попадают в некоторый отрезок из области значений случайной величины . Также, предположим что существует – вероятность попадания результата наблюдения в .

Тогда, по теореме Гливенко:

— , (1.2)

т. е., отношение — равномерно сходится к .

В случае, если к одной и той же генеральной совокупности с некоторым теоретическим распределением относятся несколько

выборок с эмпирическим распределением, которые отличаются друг от друга, то для нахождения параметров этой теоретической совокупности существует предел:

, (1.3)

. (1.4)

Функция распределения стремится по вероятности к функции Колмогорова , так что критерий выполнен.

, (1.5)

. (1.6)

На рынке Forex, временной ряд представляет собой график, где на оси абсцисс располагаются значения ряда, т.е. количество единиц второй валюты валютной пары, а на оси ординат – момент времени.

Для построения графика используют торговый период – интервал времени, за которое берется среднее значение. Интервалы подразделяются на: Тик, 1 минута, 5 минут, 10 минут, 15 минут, 30 минут, 1 час, 4 часа, 1 день, 1 неделя, 1 месяц. Также, используются характеристики котировок:

1) цена открытия – цена на рынке Forex, получившаяся в начало торгового периода;

2) цена закрытия – цена на конец торгового периода;

3) максимальная цена – самая большая цена за торговый период;

4) минимальная цена – самая низкая цена за торговый период;

5) тиковый объем – количество изменений цены.

Далее представлены виды графиков котировок валют.

Тиковый график – это графическое линейное представление изменений валютного курса с точностью до одного тика. Тиковый график характеризуется тем, что показывает каждое новое значение котировки и дает трейдеру точную и полную информацию о динамике цены. Данный график не привязан к фиксированной временной оси (см. рисунок 1.1).

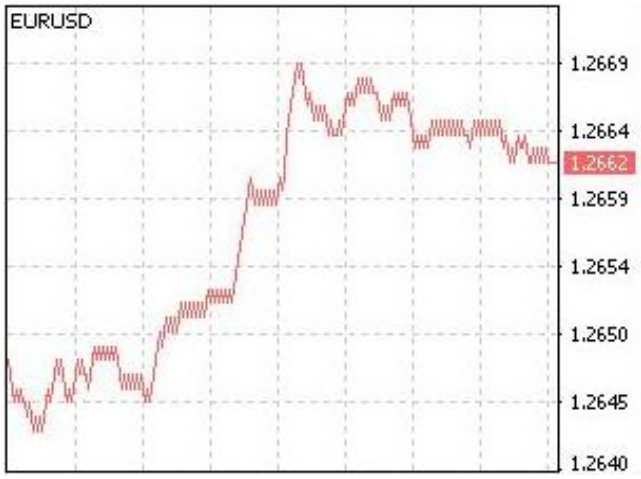

Рисунок 1.1 – Тиковый график

Линейный график – график, в основном, каждая точка которого является ценой закрытия, но эту цену можно сменить любую другую, например, на цену открытия (см. рисунок 1.2).

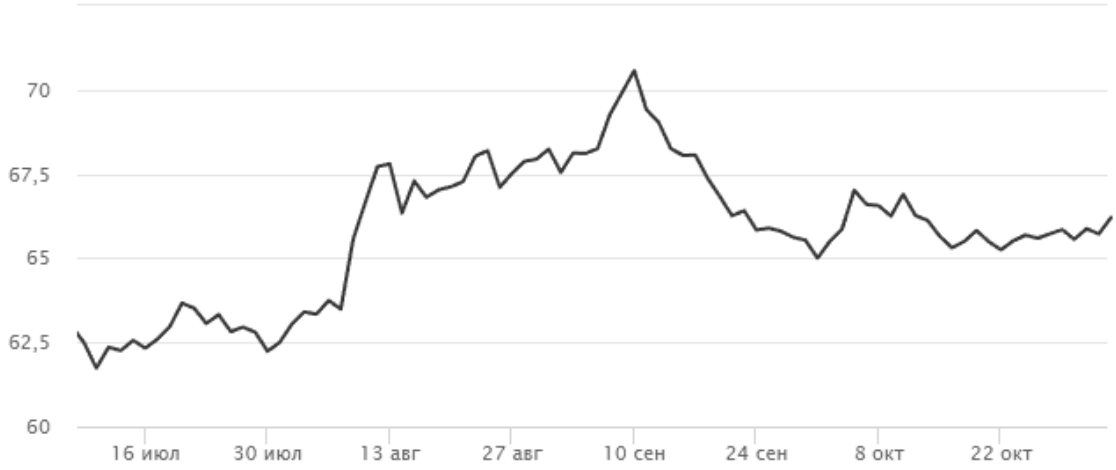

Рисунок 1.2 – Линейный график

График баров – график, состоящий из баров, имеет максимальную и минимальную цены и цены открытия и закрытия. Цвет бара означает рост или падение (см. рисунок 1.3).

Японские свечи – график, похож на график баров, также имеет максимальную и минимальную цены и цены открытия и закрытия. Однако, график японских свечей поддается целому ряду специфических интерпретаций.

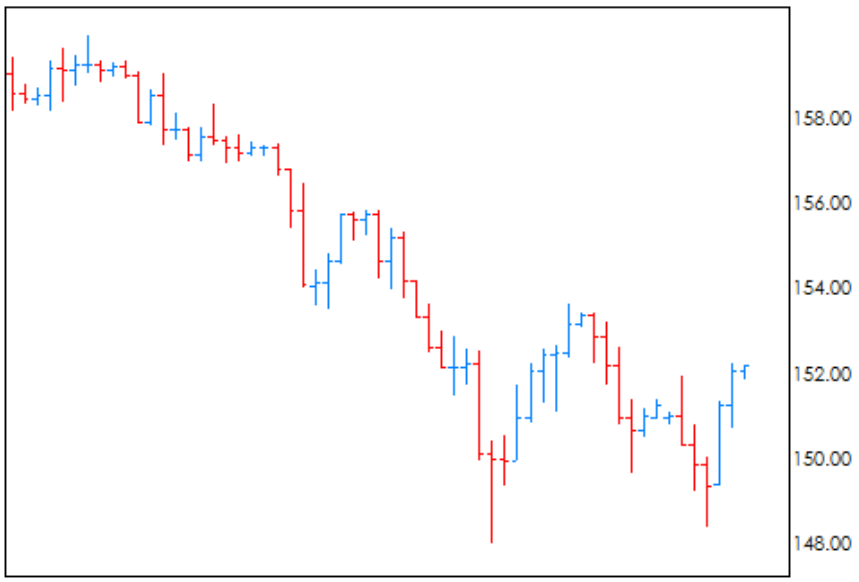

Рисунок 1.3 – Баровый график

Если свеча зеленая– это означает рост валюты, а если красная – падение валюты (см. рисунок 1.4).

Рисунок 1.4 – Японские свечи

1.3 Фундаментальный и технический анализ

1.3.1 Фундаментальный анализ

Фундаментальный анализ – сбор и анализ информации влияющей на цену валюты. В фундаментальном анализе всевозможные слухи и ожидания являются наиболее важными.

Фундаментальный анализ включает в себя оценку факторов:

1) рост денежной массы на внутреннем рынке;

2) уровень инфляции и инфляционные ожидания;

3) уровень процентной ставки.

В фундаментальном анализе происходит сравнение показателей для одной страны в разные интервалы времени.

1.3.2 Технический анализ

Технический анализ использует информацию о ситуации на рынке за предыдущие периоды его работы. Большинство методов технического анализа используют именно цену, т.к. данные о цене являются открытыми и поступают в режиме реального времени. Таким образом, технический анализ сводится к анализу графиков, чтобы предсказать возможное будущее изменение. Также, данный анализ дает трейдеру всевозможные торговые сигналы, с помощью которых он понимает, когда и что нужно покупать или продавать. Основным преимуществом такого анализа является его гибкость: трейдер может использовать одни и те же технические приемы и навыки для разных валют. Инструменты, которые использует трейдер, называются техническими индикаторами, и они обеспечивают объективное представление изменения цен, а также получение торговых сигналов заранее, чем соответствующая информация будет отображена на графике.

Технический анализ отличается от других видов исследований тем, что он основан на математических и статистических расчетах, а не на экономических.

Методы технического анализа подразделяются на категории:

1) графические – основаны на паттернах поведения цены, то есть, поведение цены повторяется и, таким образом, формируются разные фигуры, например, фигура треугольник (см. рисунок 1.5);

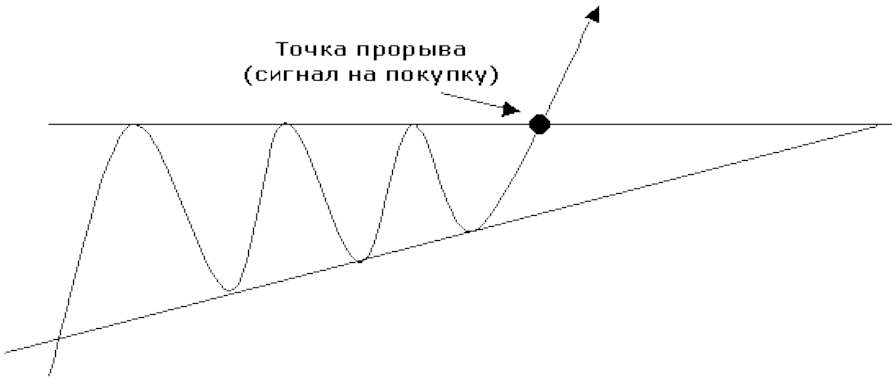

Рисунок 1.5 – Фигура технического анализ «Треугольник»

2) индикаторные – методы, которые используют фильтрацию или математическую аппроксимацию. Подразделяются на подкатегории: Трендовые индикаторы, индикаторы уровней поддержки и сопротивления, индикаторы изменчивости и т. д.;

3) вероятностные – методы, которые используют приемы из теории вероятностей и математической статистики. Такие методы можно использовать для определения силы тренда, коррекции и т. д.

Технический анализ является одним из популярных методов анализа.

1.4 Методы анализа и прогнозирования временных рядов

1.4.1 Основные понятия

Задача прогнозирования подразумевает предсказание последующих значений временного ряда. Для того чтобы решить подобную задачу, нестационарные ряды сводятся к стационарным рядам. Задачи прогнозирования зависят от времени, на которое необходимо предсказать значения временного ряда.

Для того чтобы смоделировать временной ряд, существует огромное количество методов, в чем и заключается проблема, потому каждый метод имеет свои плюсы и минусы.

Наиболее распространенными методами являются:

1) корреляционный анализ;

2) модели авторегрессии и скользящего среднего;

3) нейронные сети.

1.4.2 Корреляционный анализ

Корреляционный анализ – анализ, с помощью которого выявляются корреляции и задержки их периодичности. Автокорреляция – связь в одном процессе. Коэффициент автокорреляции – зависимость между соседними уровнями ряда.

Автокорреляционная функция – изменение коэффициентов автокорреляции в зависимости от задержки (лага):

, (1.7)

, (1.8)

где – порядок коэффициента корреляции;

– количество наблюдений;

– значение i-го признака одного события;

– значения i-го признака другого события;

– средние значения x и y.

Модели, которые используют лаговую автокорреляцию, называют автокорреляционными моделями. Для применения таких моделей нужно иметь временной ряд, у которого функция автокорреляции имеет небольшое число максимумов и быстро спадает с ростом шага автокорреляции.

1.4.3 Модели авторегрессии и скользящего среднего

Выделяют следующие модели:

1) авторегрессионная модель – AR;

2) модель скользящего среднего – MA;

3) смешанная модель: авторегрессия и скользящее среднее – ARMA;

4) интегрированная смешанная модель – ARIMA.

Рассмотренные модели основываются на гипотезе, что изучаемый процесс является выходом линейного фильтра, на вход которого подан процесс белого шума (см. рисунок 1.6).

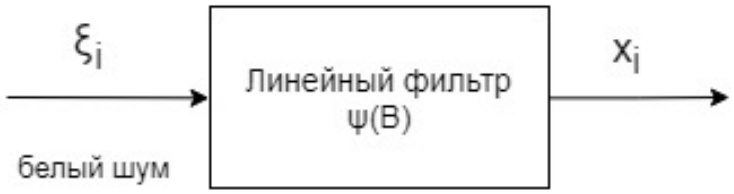

Рисунок 1.6 – Представление ряда с помощью линейного фильтра

Авторегрессионная модель – модель, значения временного ряда в которой линейно зависят от предыдущих значений этого же ряда [20]. Определяется следующим образом:

, (1.9)

где ,…, – коэффициенты авторегрессии;

– константа;

– белый шум.

Модель скользящего среднего порядка вида:

, (1.10)

где ,…, – параметры модели ( );

– белый шум.

Моделью ARMA (p, q) называет процесс генерации временного ряда. Временной ряд { }находятся по формуле:

, (1.11)

где – константа;

– целые числа, которые задают порядок модели;

– белый шум;

,… , ,…, – авторегрессионные коэффициенты и коэффициент скользящего среднего.

Модель ARIMA (p, d, q) для нестационарного ряда:

, (1.12)

где – константа;

– стационарный временной ряд;

,… , ,…, – параметры модели;

– оператор разности временного ряда порядка .

1.4.4 Нейронные сети

1.4.4.1 Искусственные нейронные сети

Существуют два вида прогнозов по временным рядам: однофакторные и многофакторные. Их различие заключается в количестве факторов, которые используются для прогноза. Например, однофакторные прогнозы – один из самых распространенных методов прогнозирования, основан на регрессионных методах. Но у такого прогноза есть недостаток: он не способен учитывать влияние нерегулярных факторов на валютные курсы, поэтому на практике необходимо использовать многофакторные прогнозы.

Одним из многофакторных подходов являются искусственные нейронные сети, которые могут устанавливать связи между входными и выходными значениями. Данный метод относиться к техническому анализу, так как, нейронные сети устанавливают связи на определенном промежутке, используя исторические данные.

Искусственные нейронные сети – математические модели, построенные по аналогии с сетью нервных клеток живого организма [12]. Имеет программную или аппаратную реализацию. Представляет собой систему искусственных нейронов, которые взаимодействуют между собой.

Нейронная сеть состоит из трех слоев:

1) входной слой – на данный слой подаются входные данные;

2) скрытый слой – состоит из нейронов, единиц нейронной сети, выполняющих роль сумматора.

3) выходной слой – один или несколько выходных параметров нейронной сети.

Такая последовательность называется перцептроном. Также, каждая нейронная сеть имеет синапсы – связи между соседними слоями, которые имеют веса.

Способы связи нейронов:

1) однослойные сети прямого распространения – в такой нейронной сети данные отсутствует скрытый слой;

2) многослойные сети прямого распространения – содержат в себе несколько скрытых слоев;

3) рекуррентные сети – это многослойные сети, в которых разрешены циклы. Нейроны в такой сети могут передавать часть информации себе на вход.

Нейронная сеть прямого распространения выглядит следующим образом (см. рисунок 1.7).

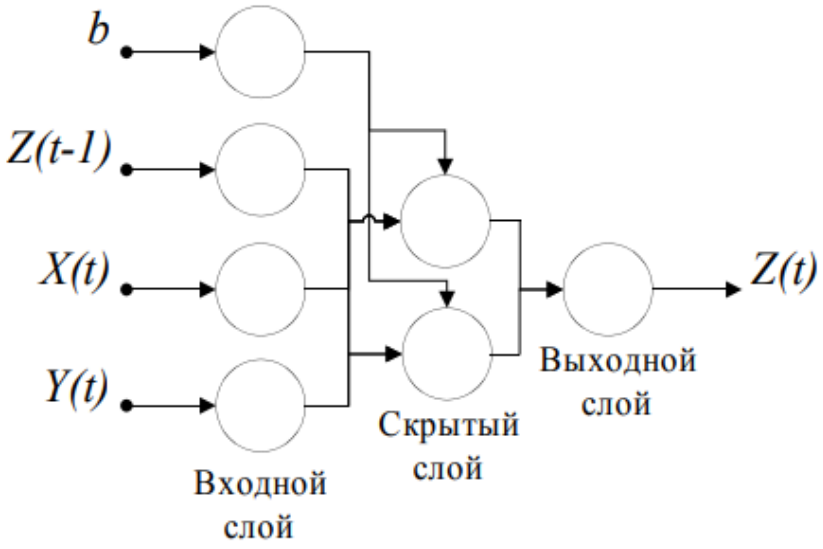

Рисунок 1.7 – Трехслойная нейронная сеть прямого распространения

Под нейронной сетью прямого распространения понимают передачу данных по синапсу слева направо, т.е. от входов к выходам.

Модель нейрона можно задать парой уравнений:

, (1.13)

, (1.14)

где , …, – входные сигналы;

,…, – синоптические веса нейрона;

– порог;

– функция активации.

Нейрон можно изобразить следующим образом (см. рисунок 1.8).

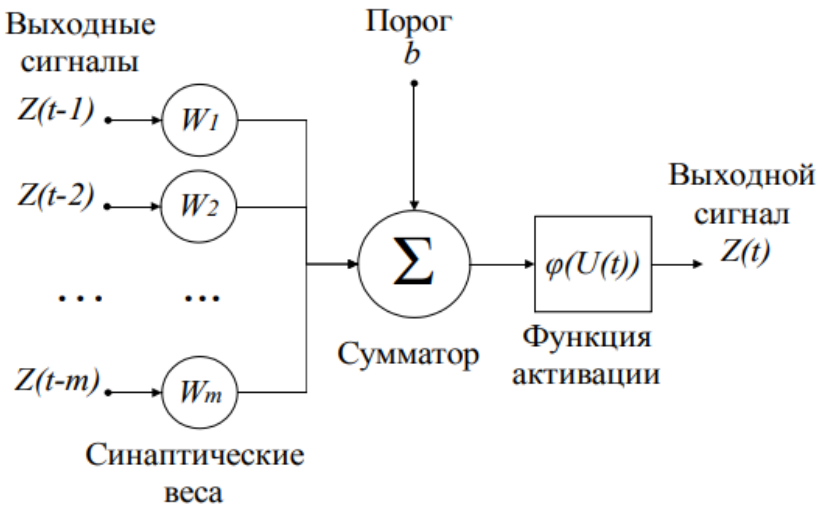

Рисунок 1.8 – Нелинейная модель нейрона

Виды функций активации:

1) функции единичного скачка;

2) кусочно-линейная функция;

3) сигмоидальная функция.

Нейронные сети нуждаются в обучении с дальнейшей целью их использования. В сравнении с традиционными алгоритмами, обучение нейронных сетей – неоспоримое преимущество. С математической точки зрения, обучение нейронной сети – многопараметрическая задача нелинейной оптимизации. Технически, обучение представляет собой поиск весов между нейронами. Если нейронная сеть обучена успешно, то он должна вернуть правильный результат, на основании данных, которых нет в

обучающей выборке. Такой процесс называется тренировкой нейронной сети [13].

Типы обучения нейронных сетей:

1) сети, обучаемые с учителем;

2) сети смешанного обучения.

Сети, обучаемые с учителем используют для получения информации о связях между входами и выходами нейронной сети. В роли учителя для нейронной сети выступает набор выходных данных, а на вход подается такой набор данных, который бы соответствовал выходным. Например, если нужно определить кто изображен на фотографии (кошка, собака и т. д.), то на вход подается набор фотографий разных животных, а также ответ, кто конкретно изображен на фотографии. Таким образом, такие сети учатся строить связи между начальными данными и результатами адаптивного итерационного процесса.

Сети, обучаемые без учителя относятся к адаптивным нейронным сетям, у которых метод обработки данных численный, а не символьный. Преимуществом подобных сетей является то, что можно проводить расчеты с высокой степенью точности. Благодаря этому преимуществу, нейронная сеть может обобщать большие наборы данных и находить в них взаимосвязи. Подобные наборы данных присущи финансовыми временным рядам.

Задачи, решаемые с помощью нейронных сетей:

1) задача классификации – в подобной задаче необходимо классифицировать данные по каким-либо параметрам;

2) задача прогнозирования – предсказание какого-либо последующего значения, например, курса валюты;

3) распознавание образов – задача идентификации объекта по каким-либо свойствам;

4) кластеризация – задача разбиения на кластеры.

1.4.4.2 Преимущества и недостатки

К преимуществам нейронных сетей можно отнести следующее:

1) можно комбинировать данные фундаментального и технического анализа;

2) для получения более точного прогноза, нейронные сети в Forex имеют возможность определять на рынке неучтенные факторы и использовать их в прогнозировании;

3) постоянное обучение за счет новых данных и имеющихся прогнозов;

4) могут быть полезны для анализа долгосрочных процессов.

Недостатки:

1) нейронные сети работают по принципу черного ящика: получая результат проанализированных сетью данных, трейдер не понимает принципов, благодаря которым сеть принимает решение. Таким образом, трейдер не будет доверять свои деньги такой сети;

2) у таких сетей есть проблемы статистического анализа: успешный анализ исторических данных не дает гарантий получения успешного прогноза на будущее;

3) чем нейронная сеть более усложненная, тем количество вычислений больше, и оно растет по экспоненте;

4) не эффективны в краткосрочной торговле.

1.4.4.3 Программное обеспечение для реализации нейронных сетей

Для реализации нейронных сетей существует огромное множество программных пакетов, которые могут быть использованы, как и для рынка Forex, так и для решения других проблем. 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 расширений, который добавляет функционал в MS Excel, связанный с нейротехнологиями.

Продолжение таблицы 1.1

Trading Solutions	Позволяет трейдерам создавать и отлаживать нейронные сети.
Statistica	ПО для статистического анализа, в том числе, с применением нейросетей.
Matlab Neural Network Toolbox	Пакет средств для проектирования, моделирования, разработки и визуализации нейронных сетей.
Python	Имеет библиотеки Keras, TensorFlow и др.

1.5 Выбор программных средств для создания нейросетей с целью трейдинга

В данной работе выбран язык программирования Python с библиотеками Keras, TensorFlow, Scikit-learn и другими, по причине того, что это язык является одним из популярных инструментов в анализе данных и машинном обучении. Также, этот язык очень прост в использовании.

Библиотеки, используемые для статистического анализа:

1) Matplotlib – реализует возможность построения графиков. С помощью данной библиотеки можно разные виды графиков;

2) NumPy – работа с многомерными массивами и матрицами. Имеет в себе пакет функций для операций над ними;

3) Pandas – работа со специальными структурами данных и операции над такими структурами. Имеет возможность загружать файлы с данными;

4) SciPy – пакет модулей для линейной алгебры, оптимизации, интеграции и статистики. Совместим с NumPy.

1.6 Вывод по разделу

В данном разделе рассмотрены понятия валютного рынка Forex, а также рассмотрены виды графиков для визуализации данных временных

рядов. В частности, рассмотрены следующие виды графиков: тиковый, линейный, баровый и японские свечи.

Методы анализа временных рядов можно разбить на две категории: методы фундаментального анализа и методы технического анализа. Технический анализ также можно разбить на методы: Графические, вероятностные и индикаторные. Данная работа посвящена методам технического анализа.

Сделан обзор методов анализа данных, среди которых рассмотрены корреляционный анализ, модели авторегрессии и скользящего среднего, нейронные сети. Одним из преимуществ нейронной сети является то, что они могут определять неучтенные в прогнозировании факторы и использовать их.

В качестве средства реализации нейронных сетей выбран язык Python с библиотеками Keras, TensorFlow, Scikit-learn и другими, потому что данный язык является распространенным как инструмент для анализа данных и машинного обучения.

2 ПОСТРОЕНИЕ МОДЕЛИ ВРЕМЕННОГО РЯДА

2.1 Подготовка данных к прогнозированию

2.1.1 Формат представления данных

Перед тем, как использовать данные для последующего прогнозирования, их необходимо проверить на наличие разного рода проблем, например, на пропуски в последовательности данных. Ниже рассматриваются этапы подготовки данных. В качестве исходных данных выбраны котировки валютной пары USD/RUB с интервалами от 1 минуты до 60 минут. Данные загружены в виде csv-файлов с полями:

1) <DATE> – Дата, представленная в виде ДД/ММ/ ГГГГ (День/Месяц/Год);

2) <TIME> – Время с форматом ЧЧ:ММ:СС (Часы:Минуты:Секунды);

3) <OPEN> – Цена открытия;

4) <HIGH> – Максимальная цена;

5) <LOW> – Минимальная цена;

6) <CLOSE> – Цена закрытия.

В таблицах 2.1 – 2.3 приведены данные с интервалами 1, 30, 60 минут.

Таблица 2.1 – Данные для интервала 1 минута

DATE	TIME	OPEN	HIGH	LOW	CLOSE
03.08.2018	5:10:00	63.3055	63.3231	63.3055	63.3055
03.08.2018	5:11:00	63.3141	63.3301	63.3055	63.3055
03.08.2018	5:12:00	63.3055	63.3055	63.3055	63.3055
03.08.2018	5:13:00	63.3055	63.3229	63.3055	63.3229
03.08.2018	5:14:00	63.3055	63.3055	63.3055	63.3055

Таблица 2.2 – Данные для интервала 30 минут

DATE	TIME	OPEN	HIGH	LOW	CLOSE
01.01.2018	0:30:00	57.647	57.647	57.647	57.647
01.01.2018	1:00:00	57.647	57.647	57.647	57.647

Продолжение таблицы 2.2

01.01.2018	1:30:00	57.677	57.677	57.677	57.677
01.01.2018	2:00:00	57.677	57.677	57.647	57.647
01.01.2018	2:30:00	57.647	57.647	57.647	57.647

Таблица 2.3 – Данные для интервала 60 минут

DATE	TIME	OPEN	HIGH	LOW	CLOSE
01.01.2018	1:00:00	57.647	57.647	57.647	57.647
01.01.2018	2:00:00	57.677	57.677	57.647	57.647
01.01.2018	3:00:00	57.647	57.647	57.647	57.647
01.01.2018	4:00:00	57.647	57.647	57.647	57.647
01.01.2018	5:00:00	57.647	57.647	57.647	57.647

Построены графики японских свечей для полученных данных. Каждая свеча данного графика состоит из значений OHLC (Open, High, Low, Close). Если тело свечи имеет черный цвет, это означает, что цена закрытия ниже цены открытия, и торги в этот период времени закрылись с понижением. А если тело свечи белое – цена закрытия больше цены открытия и торги завершились с понижением (см. рисунок 2.1).

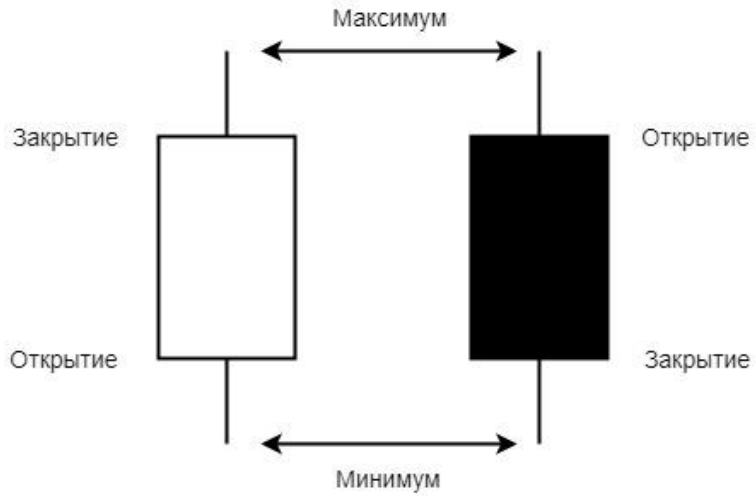

Рисунок 2.1 – Японские свечи

Далее изображен график японских свечей для интервала в 1 минута (см. рисунок 2.2) и его увеличенная версия (см. рисунок 2.3).

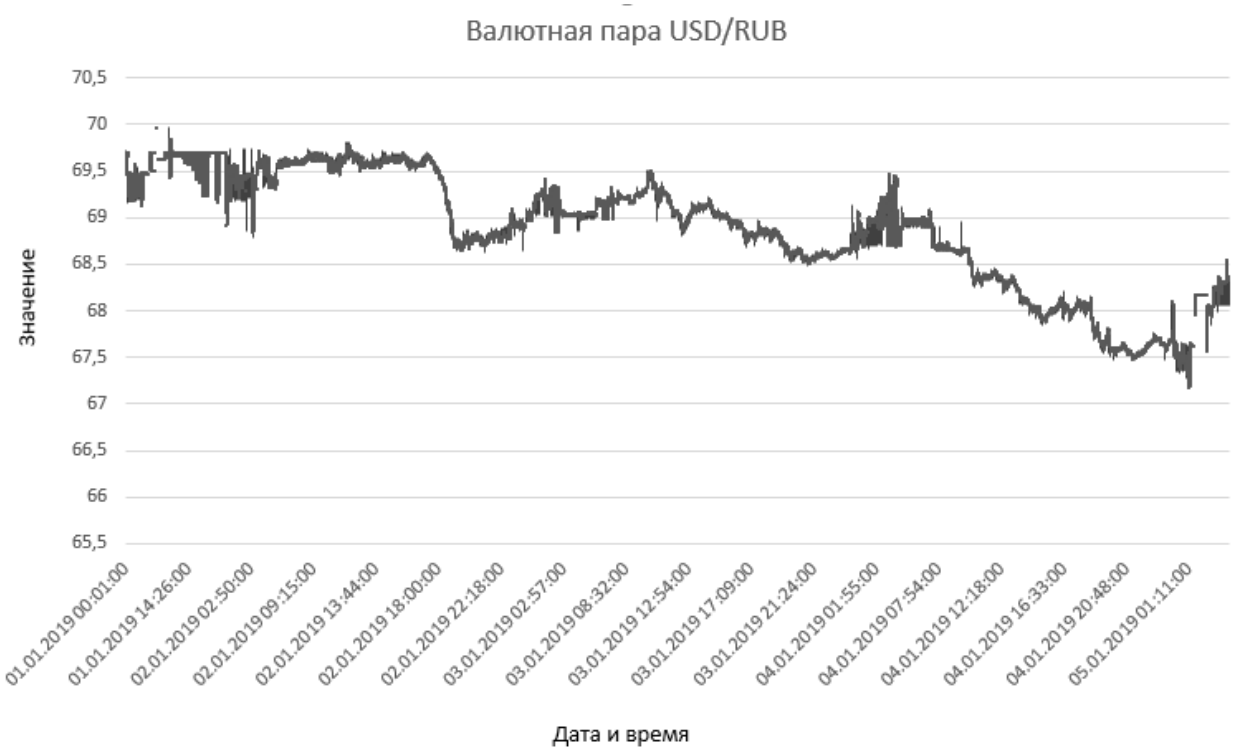

Рисунок 2.2 – График японских свечей для интервала 1 минут

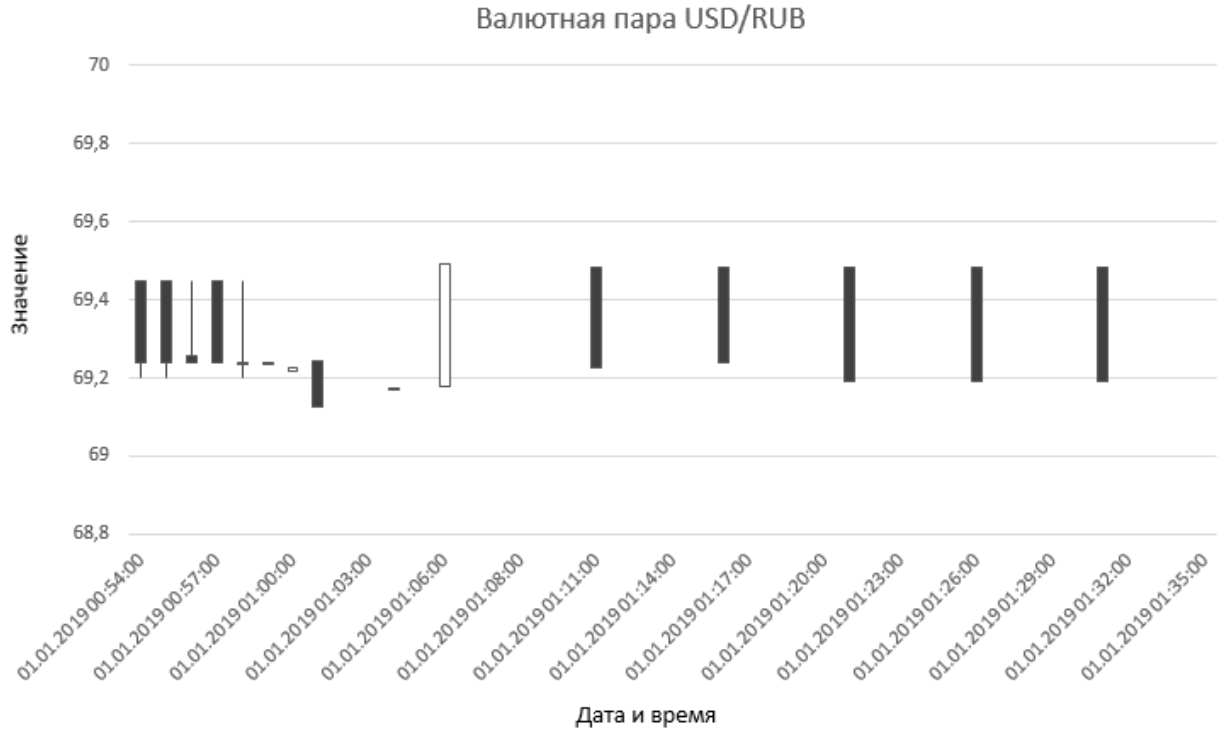

Рисунок 2.3 – Увеличенный график японских свечей
для интервала 1 минута

2.1.2 Преобразование данных

При просмотре данных и графиков обнаружено следующее:

1) имеются пропуски в последовательностях данных;

2) отрезки должны быть сопоставимы, т.е. иметь примерно одинаковое число отчетов;

3) пропущены выходные дни.

Этапы преобразования.

1. Так как, не все методы анализа временных рядов допускают работу с неравномерными данными, принято решение обработать их. Исходные данные преобразуются к новому формату. Данные столбцов <DATE> и <TIME> объединяются и преобразуются в графу <DateTime>, которая имеет тип числовой тип данных. В этой графе указывается прошедшее количество минут с некоторой условной начальной даты, которой является 01.01.1990 00:00. Остальные поля не изменились. Пример данных для 60 минут представлен в таблице 2.4.

Таблица 2.4 – Преобразованные данные для интервала 60 минут

DateTime	OPEN	HIGH	LOW	CLOSE
10227.04	57.647	57.647	57.647	57.647
10227.08	57.677	57.677	57.647	57.647
10227.13	57.647	57.647	57.647	57.647
10227.17	57.647	57.647	57.647	57.647
10227.21	57.647	57.647	57.647	57.647

2. Далее происходит устранение дыр во времени таким образом, чтобы шаг равнялся интервалу.

3. В основном, временные ряды в чистом виде не соответствуют нормальному закону распределения и не являются стационарными. Это мешает их корректному анализу, поэтому предлагается брать для анализа сами значения котировок.

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

—— , (2.1)

где – элемент подпоследовательности.

Эта формула используется для каждой последовательности, на которые разбиты все данные. Для того, чтобы проверить имеющийся ряд на стационарность, будет использован тест Дики-Фуллера.

2.1.3 Проверка ряда на стационарность

Тест Дики-Фуллера – это средство, используется в статистике и эконометрике для проверки временных рядов на стационарность. Является одним из тестов единичного корня.

Временной ряд имеет единичный корень, если его первые разности образуют стационарный ряд. При помощи этого теста проверяют значение коэффициента *d* в авторегрессионном уравнении первого порядка:

, (2.2)

где – коэффициент авторегрессионного уравнения;

– временной ряд;

– ошибка.

Если , то процесс имеет единичный корень и временной ряд не является стационарным. Если – ряд стационарный. Для финансово-экономических процессов значение не свойственно, т.к. как финансово-экономическая среда достаточно инерционная.

В тесте Дики-Фуллера используется две гипотезы: нулевая и альтернативная. Нулевая гипотеза говорит о существовании единичного корня. В этом случае ряд – нестационарный. Альтернативная гипотеза – единичного корня нет, ряд – стационарный.

Также существует расширенный тест Дики-Фуллера, который подразумевает, что если в тестовые регрессии добавить лаги первых разностей временного ряда, то распределение DF-статистики не изменится.

Проверив имеющие данные расширенным тестом Дики-Фуллера, получили результаты, которые записаны в таблице 2.5.

Таблица 2.5 – Тест Дики-Фуллера для данных с интервалом 1 день

Наименование	Значение
Статистика теста ADF	-0.467539
Вероятность	0.898197
1%	-3.432
5%	-2.862
10%	-2.567

Таким образом, ряд является стационарным и может использоваться для прогнозирования. Программа для проверки ряда на стационарность находится в приложении 2.

2.2 Построение модели временного ряда с помощью нейронных сетей

Для построения модели временного ряда использовалась рекуррентная нейронная сеть с долгой краткосрочной памятью.

Рекуррентная нейронная сеть – это класс моделей машинного обучения, основанный на использовании предыдущих состояний сети вычисления текущего состояния. Нейроны могут передавать часть информации себе на вход и за счет этого запоминать события, которые произошли некоторое время назад. В отличие от сверточных нейронных сетей, которые могут заниматься анализом изображений, видео и прочего, рекуррентная нейронная сеть может анализировать последовательности, например, обработка текста на естественном языке.

Рекуррентную нейронную сеть можно изобразить следующим образом (см. рисунок 2.4).

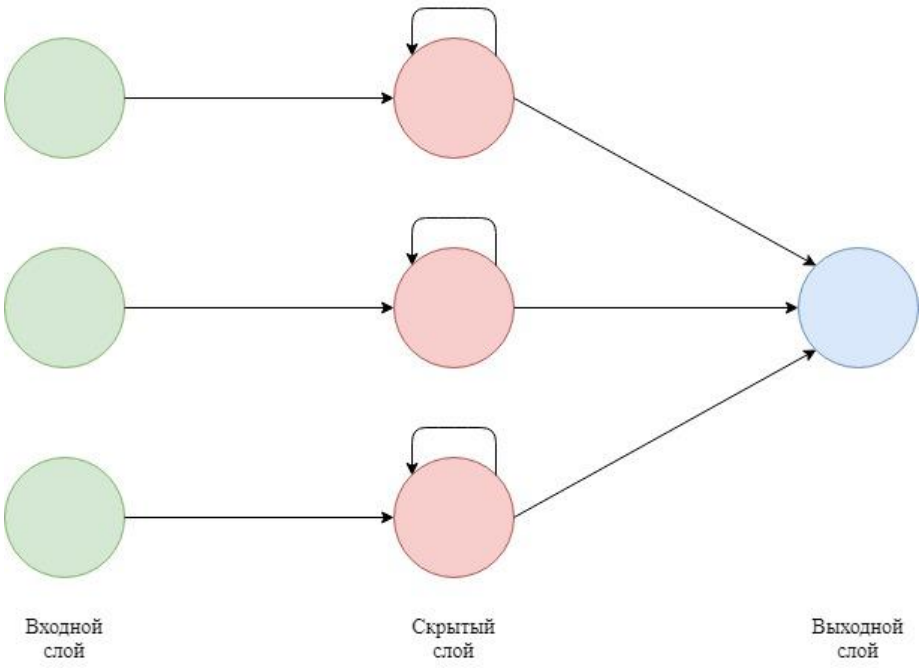

Рисунок 2.4 – Рекуррентная нейронная сеть

Рекуррентную нейронную сеть можно представить в виде сетей с прямым распространением сигнала развернутую во времени. Обучение такой нейронной сети происходит следующим образом: На вход нейронной сети подается входной сигнал, после чего она выдает сигнал, а также значение, которое подается на вход точно такой же нейронной сети на следующем этапе вместе со следующим значением. Используя метод обучения «с учителем», мы знаем, какой правильный ответ сеть должна выдать на последнем этапе работы, следовательно, возможно посчитать ошибку на этом этапе. Данный метод обучения называется методом обратного распространения с разворачиванием сети ко времени. Также, с помощью этого метода можно рассчитать, какая будет ошибка на предыдущих слоях. Обучение такой сети представлено ниже (см. рисунок 2.5).

Сети долгой краткосрочной памяти – это тип рекуррентной нейронной сети, способной запоминать данные на долгое время. Такой тип сети разработан для устранения проблемы долгосрочной зависимости. Сеть долгой краткосрочной памяти уменьшает или увеличивает количество информации для ячейки. Для этого используется фильтр.

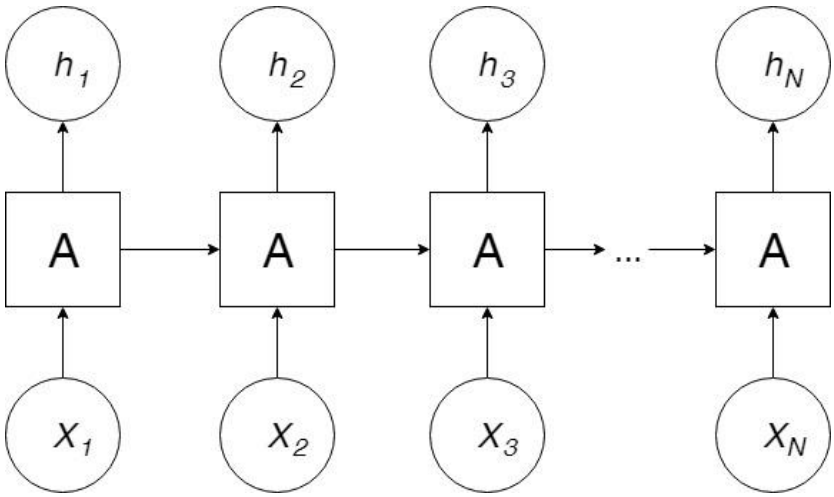

Рисунок 2.5 – Обучение рекуррентной нейронной сети

Фильтр – так называемые «ворота», которые пропускают или не пропускают информацию (см. рисунок 2.6). Значения фильтра устанавливаются нейронами сети, когда в этом возникает необходимость. Аналогом фильтра в реальной жизни является вентиль от крана. Фильтр состоит из сигмовидного слоя и операции поточечного умножения. Значения фильтра от нуля до единицы, т.е. если это число равно нуля – это означает, что значение, которое поступает на вход нейрона, не влияет на его содержание.

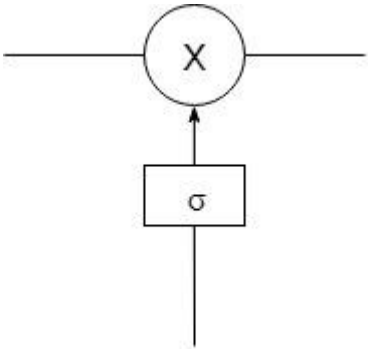

Рисунок 2.6 – Фильтр

Для контроля состояния ячейки, сеть с долгой краткосрочной памятью содержит в себе 3 фильтра.

1. Фильтр забывания: На первом этапе сети следует решить, какая информация не будет использована в дальнейшем. Определяется это слоем фильтра утраты. Если он получает на вход единицу – то информация будет

полностью сохранена, иначе – полностью удалена. Математически, фильтр забывания можно представить так:

, (2.3)

где – состояние сети;

– входное значение;

– обучаемые параметры рекуррентной нейронной сети.

2. Входной фильтр: На следующем шаге необходимо определить, какая информация сохраниться в состоянии ячейки. Для этого нужен слой фильтра входа, который решает, какие значения требуется обновить.

, (2.4)

где – состояние сети;

– входное значение;

– обучаемые параметры рекуррентной нейронной сети;

– сигмоидальная функция.

3. Выходной фильтр: Если он получает на вход единицу – то информация будет полностью сохранена и передана на входной фильтр, иначе – полностью удалена.

, (2.5)

где – состояние сети;

– входное значение;

– обучаемые параметры рекуррентной нейронной сети.

На основе всех данных, поступающих в момент времени , вычисляется состояние ячейки памяти на текущем шаге, используя следующие фильтры:

, (2.6)

, (2.7)

где – состояние сети;

– входное значение;

– обучаемые параметры рекуррентной нейронной сети.

Итоговое значение LSTM-слоя определяется выходным фильтром и нелинейной трансформацией над состоянием блока памяти:

, (2.8)

где – состояние сети;

– выходной фильтр;

– состояние ячейки памяти.

Фильтры нейронной LSTM-сети можно изобразить следующим образом (см. рисунок 2.7).

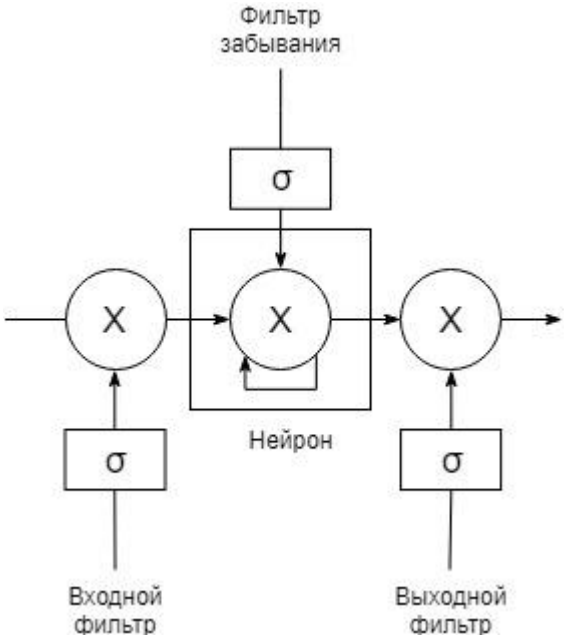

Рисунок 2.7 – Фильтры LSTM-сети

Фильтр Калмана – это рекурсивный фильтр, который оценивает вектор состояния динамической системы, используя ряд неполных измерений. Фельдкамп использовал методы фильтра Калмана для улучшения производительности рекуррентной нейронной сети. Поскольку он использовал «производный коэффициент дисконтирования, наложенный на экспоненциальное разложение эффектов прошлых динамических производных», то нет никаких оснований полагать, что его обученная рекуррентная сети с фильтром Каламана будет очень полезной при очень длительных минимальных задержках по времени.

Для простоты будут использоваться только цены закрытия. Keras понадобится для отладки и отслеживания модели. Цены представляют собой

временные ряды длины , где – это серия элементов, где каждый элемент имеет индекс от нуля до . Представим, что есть скользящее окно фиксированного размера , перемещаем его вправо так, чтобы не происходило перекрытия между данными во всех скользящих окнах.

В качестве основных скрытых элементов модель рекуррентной нейронной сети имеет LSTM – ячейки. Используя значения с самого начала обучения начиная с первого скользящего окна до окна , в момент времени , и для прогнозирования цен в следующем окне :

, (2.9)

, (2.10)

, (2.11)

. (2.12)

Таким образом происходит поиск функции аппроксимации:

. (2.13)

Наша нейронная сеть имеет 1 выход и 3 скрытых слоя. Количество входов варьируется от подаваемых данных. Затем модель компилируется с оптимизатором Adam, функцией ошибки MSE (среднеквадратическая ошибка) и метрикой MSE, для оценки качества.

2.3 Математическая модель

Для того, чтобы построить архитектуру, которая допускает постоянный поток ошибок через специальные, самостоятельно подключенные блоки без недостатков наивного подхода, произведено расширение карусели постоянной ошибки CEC, воплощенная автономным линейным -тым блоком добавив дополнительные функции. Мультипликативный входной блок введен для защиты содержимого памяти, хранящегося в -том блоке, от возмущения нерелевантными входными сигналами. Аналогично, вводится мультипликативный выходной блок защищает другие блоки от возмущения текущими нерелевантным содержимым памяти, хранящемся в -том блоке.

В итоге, более сложная единица называется ячейкой памяти. Каждая ячейка памяти построена вокруг центрального линейного блока с фиксированным самоподключением. В дополнении к , получает входные данные от мультипликативного блока и от другого мультипликативного блока , активация в момент времени обозначается как , а – как . В итоге получаем формулы:

(2.14)
,

(2.15)
,

где ,

,

.

Индексы суммирования могут обозначать входные блоки, блоки фильтров, ячейки памяти или даже обычные скрытые блоки, если таковые имеются. Все эти различные типы блоков могут передавать полезную информацию о текущем состоянии сети. Например, входной фильтр может использовать входы из других ячеек памяти, чтобы решить, сохранять ли определенную в своей ячейке памяти.

В момент , выход блока , вычисляется как:

(2.16)
,

где «внутреннее состояние» задает подобным образом:

(2.17)
,

для $t > 0$. (2.18)

Дифференцируемая функция применяется к . Дифференцируемая функция масштабирует выходные данные ячейки памяти, вычисленные из внутреннего состояния .

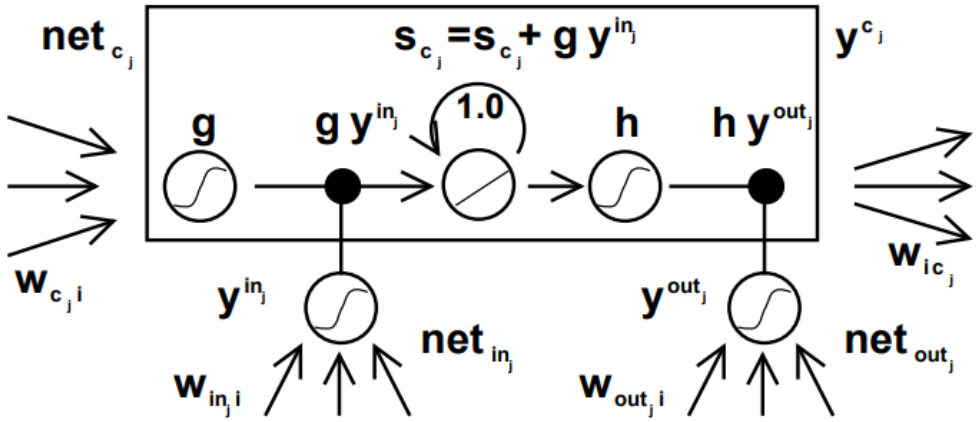

Рисунок 2.8 – Архитектура ячейки памяти и фильтр блоков ,

Как замечено ранее, рекуррентную нейронную сеть с долгой краткосрочной памятью необходимо обучать с помощью метода обратного распространения. Этот метод делится на 3 этапа:

1) прямой проход – вычисление состояния слоев;

2) обратный проход – вычисление ошибки слоев;

3) вычисление изменения весов.

Прямой проход – для каждого вектора последовательности вычисляем состояние скрытого слоя и выхода скрытого слоя

, (2.19)

, (2.20)

где – весовая матрица выходного слоя;

– входной вектор номер ;

– весовая матрица распределительного слоя;

– состояние скрытого слоя для входа .

Найдем выход сети :

, (2.21)

где – функция активации выходного слоя;

– весовая матрица обратных связей скрытого слоя;

– состояние скрытого слоя для входа .

Обратный проход – вычисляем ошибку выходного слоя , ошибку скрытого слоя в конечном состоянии в конечном состоянии и ошибку скрытого слоя в промежуточном состоянии :

, (2.22)

, (2.23)

, (2.24)

где – весовая матрица обратных связей скрытого слоя;

– выход сети для входа x(t);

– исключающее ИЛИ-НЕ;

– функция активации скрытого слоя;

– состояние скрытого слоя.

Вычисление изменения весов:

, (2.25)

, (2.26)

, (2.27)

, (2.28)

, (2.29)

где – ошибка выходного слоя;

– весовая матрица обратных связей скрытого слоя;

– состояние скрытого слоя для входа ;

– вектор сдвигов выходного слоя;

– весовая матрица выходного слоя;

– входной вектор номер t;

– весовая матрица распределительного слоя;

– ошибка скрытого слоя в промежуточном состоянии.

Также следует упомянуть о экспоненциально убывающей ошибке. Цель выходного блока в момент времени обозначается . Используя среднеквадратичную ошибку, значение ошибки :

, (2.30)

где – функция активации -го блока без входа с дифференцируемой функцией активации ;

– текущей входной сигнал -го блока;

– вес соединения от -го блока до -го блока.

Значение ошибки обратного распространения -го блока без выхода:

, (2.31)

где – вес соединения от -го блока до -го блока;

– текущей входной сигнал -го блока;

– среднеквадратичная ошибка.

Соответствующим вкладом в обновление общего веса является:

, (2.32)

где – скорость обучения;

– произвольный блок связанный с -ым блоком.

2.4 Вывод по разделу

В данной главе рассмотрена подготовка данных к последующему прогнозированию, проверка их на стационарность после подготовки, и построение модели нейронной сети.

С целью исправления таких проблем как пропуск данных и прочих, выполнено преобразование данных, которое привело имеющийся временной ряд к стационарному временному ряду.

Для проверки на стационарность преобразованных данных использован расширенный тест Дики-Фуллера, который является одним из тестов на единичные корни. В итоге, после проверки получены результаты, которые показывают, что имеющие преобразованные данные – стационарные.

Для построения модели временного ряда использована рекуррентная нейронная сеть. Основным свойством данной нейронной сети является то, что она может запоминать выходные данные и учитывать их в будущем.

Однако, на практике стандартные рекуррентные нейронные сети не могут справиться с долгосрочными зависимостями, поэтому выбран специфический тип нейронной сети для решения этой проблемы – рекуррентная нейронная сеть с долгой краткосрочной памятью. Такая нейронная сеть может запоминать информацию в течении длительных периодов времени, поэтому такую сеть практически не нужно обучать.

Описана математическая модель, которая включает себя модель нейронной LSTM-сети, метод обучения такой сети – метод обратного распространения, а также информация о экспоненциально убывающей ошибке.

3 СОЗДАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

3.1 Проектирование интерфейса

Необходимо разработать интерфейс для приложения, прогнозирующего нелинейные процессы на примере валютного рынка Forex. Для реализации графического пользовательского интерфейса будет использоваться библиотека «Tkinter».

После запуска программы появляется окно, в котором находится главное меню программы (см. рисунок 3.1).

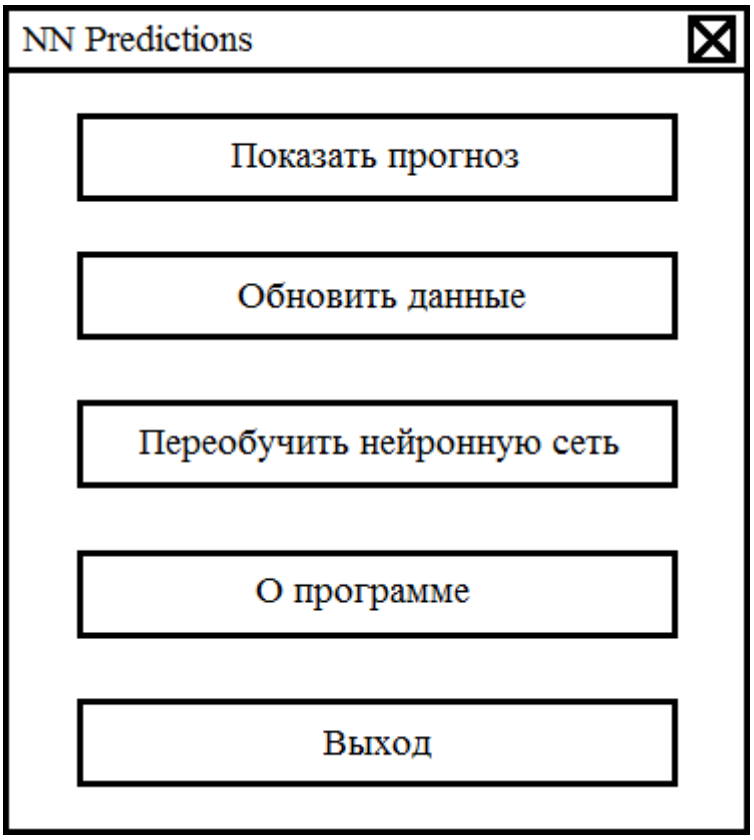

Рисунок 3.1 – Главное меню

Меню состоит из пунктов:

1) показать прогноз;

2) обновить данные;

3) обучить нейронную сеть;

4) о программе;

5) выход.

При выборе пункта меню «Обучить нейронную сеть» будет открыто окно с параметрами для обучения (см. рисунок 3.2).

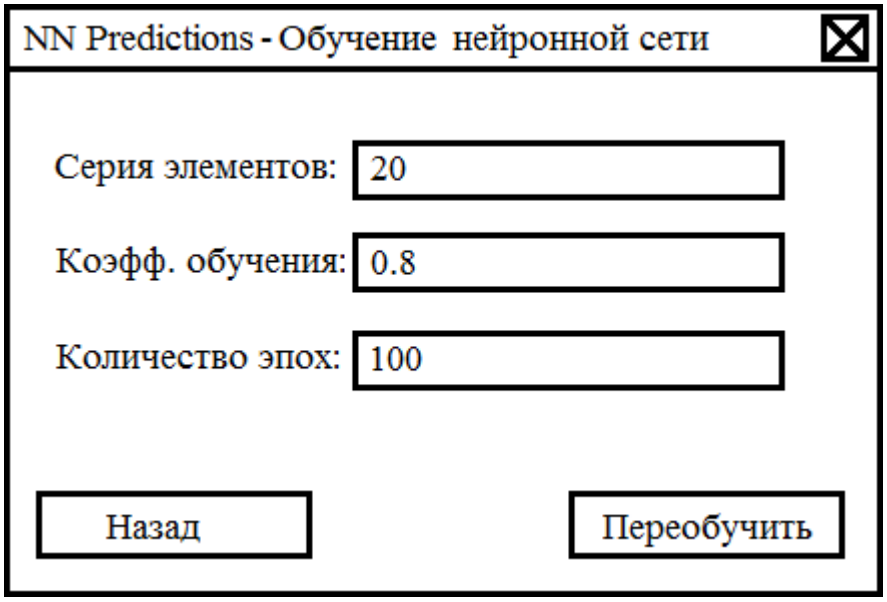

Рисунок 3.2 – Обучение нейронной сети

В данном окне можно задать такие параметры для обучения нейронной сети, как:

1) серия элементов – некоторое количество элементов, которые будут использоваться для нахождения следующего элемента;

2) коэффициент обучения – коэффициент обучаемых данных к тестовым;

3) количество эпох – количество эпох, требуемых для обучения. С каждой эпох ошибка обучения нейронной сети падает.

Если все параметры указаны правильно, то после запуска будет выдано сообщение о начале обучения. За процессом обучения можно наблюдать в окне консоли. По окончанию обучения будет предоставлен график качества обучения.

При выборе пункта «Обновить данные» будет открыто окно с настройками для обновления данных (см. рисунок 3.3).

Для того чтобы обновить данные, необходимо задать следующие параметры:

1) валютную пару;

2) период, за который необходимо получить данные (Дата начала и дата конца);

3) интервал – временной интервал, которому будет соответствовать одно значение из полученных данных.

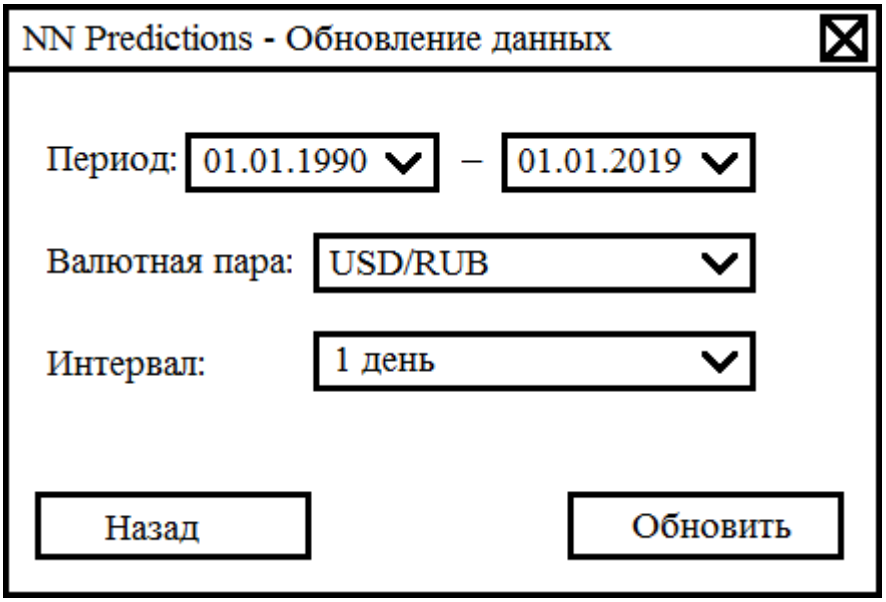

Рисунок 3.3 – Обновление данных

После установки всех параметров и нажатия кнопки «Скачать», начнется процесс скачивания. Если данные успешно получены – они сохраняются в файл data.csv.

При выборе пункта «О программе» появится сообщение, содержащее информацию о разработчике программы (см. рисунок 3.4).

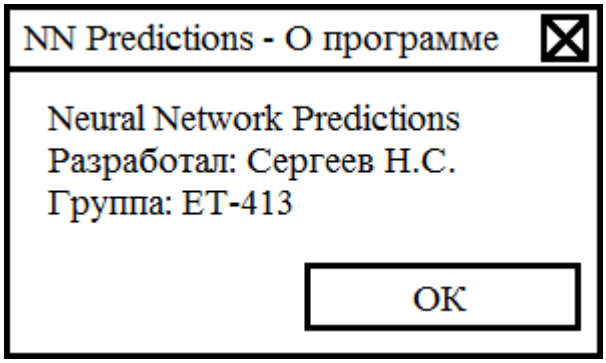

Рисунок 3.4 – О программе

При выборе пункта «Показать прогноз» будет показана форма с вероятным прогнозом, а также оценкой качества нейронной сети.

3.2 Построение алгоритма

3.2.1 Выбор требуемых алгоритмов

Для разработки приложения для прогнозирования нелинейных процессов с помощью нейронных сетей необходимо выделить следующие модели, которые должны реализовывать определенную часть функций и выполнять следующие действия:

– получать данные из интернета;

– обучать нейронную сеть;

– демонстрировать прогноз для текущих данных;

– реализовывать интерфейс приложения.

Далее представлены схемы алгоритмов для каждого из пунктов, которые необходимо реализовать.

3.2.2 Схемы алгоритмов

3.2.1.1 Основные алгоритмы

Основной алгоритм отвечает за работу всего приложения (см. рисунок 3.5), а алгоритм обработки сообщений – за работоспособность реализованного интерфейса (см. рисунок 3.6). Алгоритм обработки сообщений меню содержит обработку нажатия клавиш главного меню. После нажатия на выбранную кнопку – будет вызвано вспомогательное окно, которое относиться к выбранному пункту. При нажатии кнопки «Выход» приложение завершит работу.

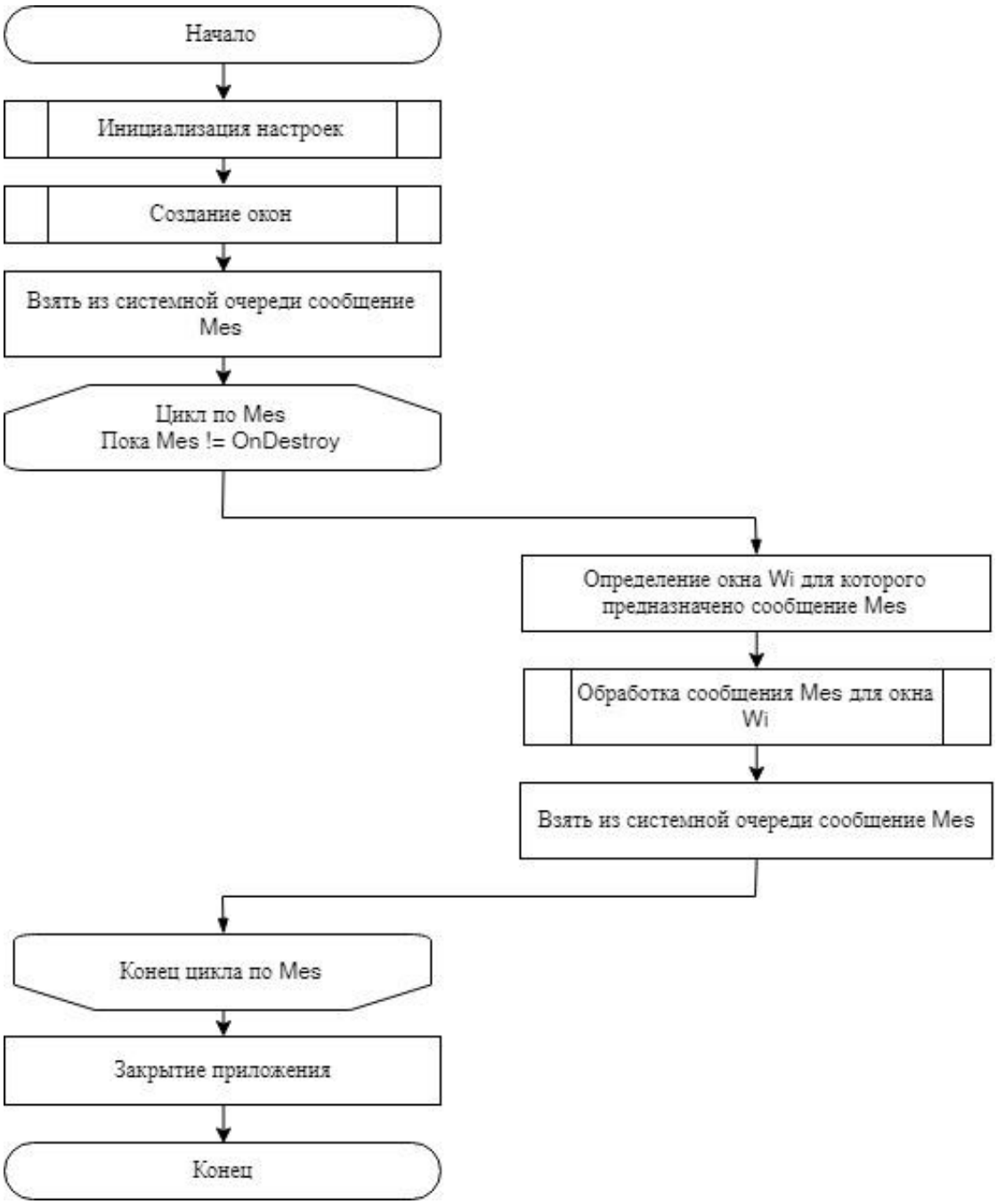

Рисунок 3.5 – Основной алгоритм

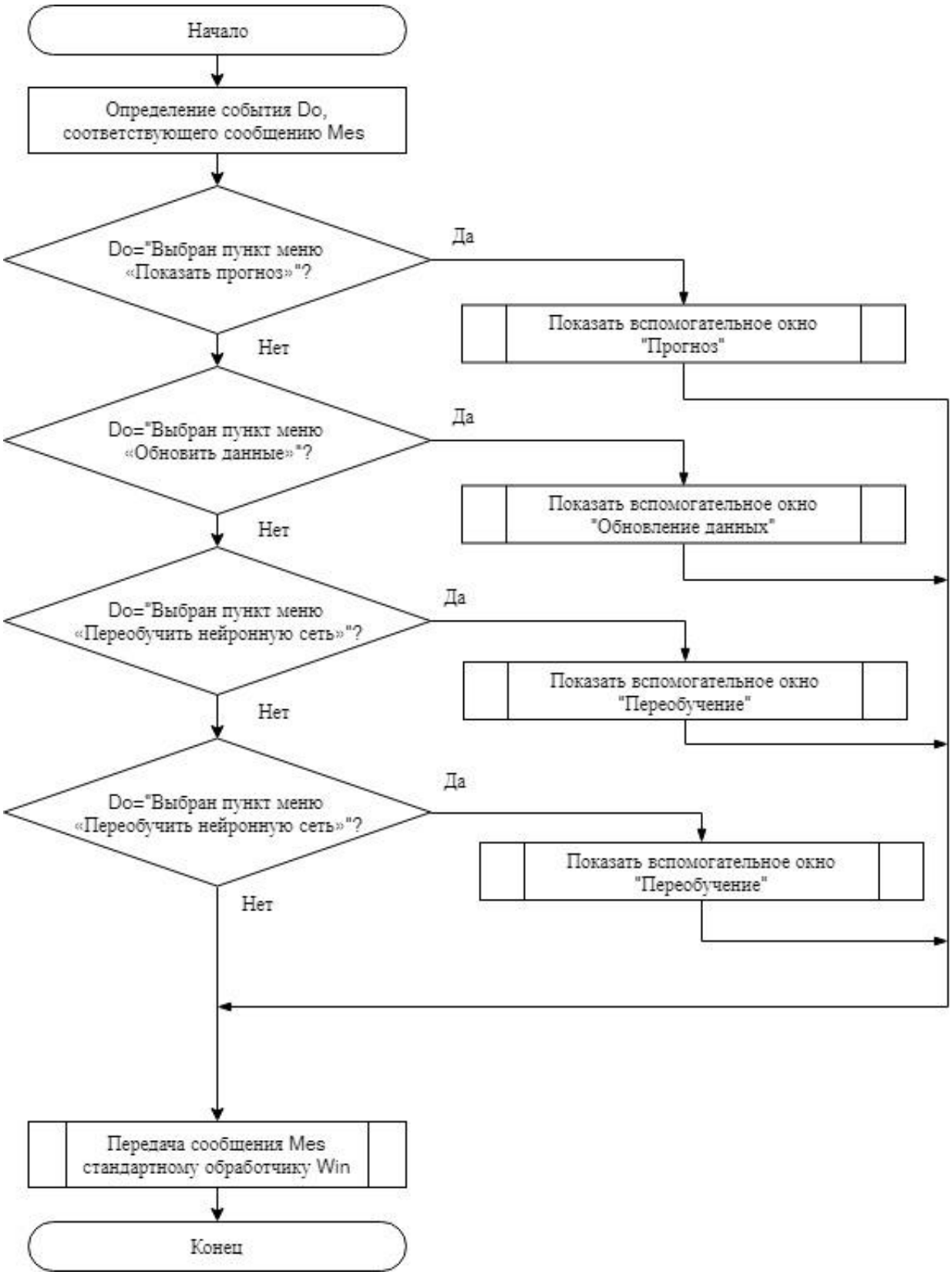

Рисунок 3.6 – Алгоритм обработки сообщений

3.2.1.2 Дополнительные алгоритмы

При выборе пункта меню «Обновить данные» появляется окно, в котором необходимо настроить такие параметры, как:

– дата начала – дата, начиная с которой будет происходить получение данных;

– дата окончания – дата, до которой будет происходить получение данных;

– валютная пара – пара валют, выражение 1 единицы первой валюты в единицах второй;

– временной интервал (1 минута, 5 минут, и т. д.).

После настройки данных, вспомогательная функция проверяет их на корректность и в случае успеха – она вызывает функцию для получения данных (см. рисунок 3.7).

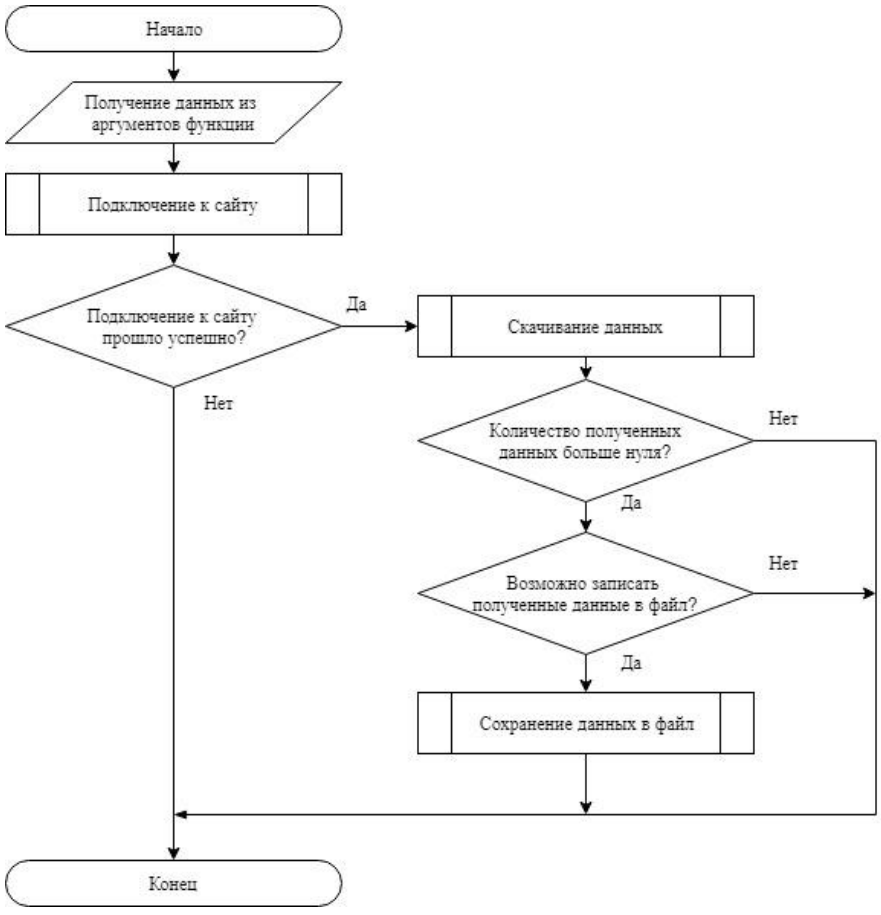

Рисунок 3.7 – Алгоритм получения данных из интернета

Если выбран пункт меню «Обучить нейронную сеть», то будет показано окно, в котором необходимо настроить такие параметры, как:

– серия элементов – некоторое количество элементов, которые будут использоваться для нахождения следующего элемента;

– коэффициент – коэффициент данных для обучения к данным для проверки, лежит в диапазоне [0;1];

– количество эпох.

После настройки данных, вспомогательная функция проверяет их на корректность и в случае успеха – вызывает функцию для обучения нейронной сети (см. рисунок 3.8).

Рисунок 3.8 – Алгоритм обучения нейронной сети

Имея обученную нейронную сеть, а также данные для прогнозирования, программа может приступить к прогнозированию. Прогнозирование вычисляется на несколько шагов вперед, т.е. можно посмотреть прогноз для данных, которых еще нет. При выборе пункта меню

«Показать прогноз» будет вызвана функция для прогнозирования на имеющихся данных с использованием нейронной сети (см. рисунок 3.9).

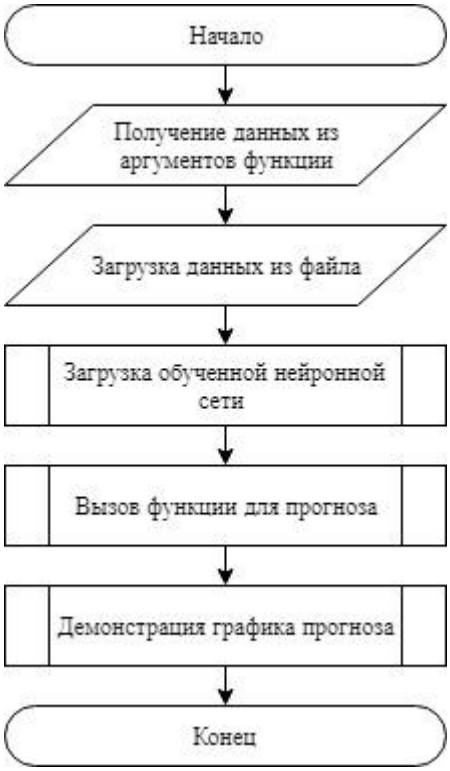

Рисунок 3.9 – Алгоритм демонстрации прогноза

3.2.2 Описание параметров среды

Python 3 – это инструмент для создания программ самого разнообразного назначения. С его помощью можно решать задачи различных типов. Библиотеки компонентов в различных вариантах практически одинаковы.

Реализованная программа использует 3 модуля для решения поставленных задач.

3.2.2.1 Модуль «gui.py»

Данный модуль является главным и используется для загрузи интерфейса приложения. Помимо функций интерфейса, модуль содержит дополнительные вспомогательные функции, которые помогают передать данные из интерфейса в другие функции, например, в функцию для получения данных из интернета.

Таблица 3.1 – Назначение и параметры методов модуля «gui.py»

Метод	Назначение
Retraining_window(event)	Окно с настройками для обучения нейронной сети
GetCode(name)	Получение кода валютной пары
Update_window(event)	Окно с настройками для обновления данных
CheckReqData_Download(ds,ms,ys,de,me,ye,cur,period)	Передача значений и вызов функции обновления данных
CheckReqData_Retrain(series,k,epochs)	Передача значений и вызов функции обучения нейронной сети
period_val(per)	Получение кода временного периода
About_msg(event)	Вызов сообщения с данными о разработчике
Quit(event)	Выход из программы
ShowMain(event)	Основное окно
HideAndShow(i)	Настройка видимости окон
AddToDate(dates,c1)	Сдвиг для спрогнозированных данных
DatetimeIndex(name)	Выбор индекса для прогноза или обучения
x,y	Координаты для центрирования окна

Таблица 3.2 – Используемые переменные модуля «gui.py»

Переменная	Назначение
currency_name	Наименования валютных пар
period_name	Наименования временных периодов
Root	Основное окно
Window	Окно для обучения нейронной сети
Upd	Окно для обновления данных

3.2.2.2 Модуль «nn.py»

Данный модуль содержит функции для работы с нейронными сетями, а также функции для реализации графиков.

Таблица 3.3 – Назначение и параметры методов модуля «nn.py»

Метод	Назначение
load_data (datasetname, column, seq_len, normalise_window, k)	Загрузка и разбиение данных из входного набора данных
normalise_windows(window_data)	Преобразование данных
plot_results (predicted_data, true_data, datelist, evaluate)	Построение графика с результатами прогнозирования
FixHours(str1)	Приведение часов к двухзначному варианту (00 – 23).
ConvToDateTime(dataset, count1, tr)	Преобразование столбцов даты и времени в один столбец
func_UpgradeNN(days, k, epochs)	Обучение нейронной сети
func_LoadNN()	Загрузка нейронной сети и прогнозирование

3.2.2.3 Модуль «download.py»

Данный модуль содержит одну функцию для скачивания данных из интернета. Данные скачиваются с сайта finam.ru. Исходный код данного модуля находится в приложении 1.

3.3 Экспериментальная проверка работы программы

После разработки программы было выполнено тестирование модулей «nn.py» и «download.py» на предмет ошибок (см. таблицу 3.4, 3.5).

Таблица 3.4 – Тестирование модуля «nn.py»

Тест	Результат
Отсутствие файла «data.csv».	Ошибка: Файл не найден.

Продолжение таблицы 3.4

Отсутствие файла «nn.json» или файла «nn.h5».	Ошибка: Файл не найден.
Обучение: 0 серий.	Ошибка: Количество серий должно быть больше или равно 5.
Обучение: Коэффициент = 5.	Ошибка: Коэффициент обучаемых данных к тестовым должен быть в диапзоне [0;1].
Обучение: 0 эпох.	Ошибка: Минимальное количество эпох = 10.
Обучение: Количество серий больше количества данных.	Ошибка: Количество серий должно быть меньше количества данных.
Отсутствие файла «settings.bin».	Ошибка: Файл не найден.

Таблица 3.5 – Тестирование модуля «download.py»

Тест	Результат
Отсутствие подключения к сети Интернет.	Ошибка: Невозможно подключиться к серверу.
Дата начала больше даты конца.	Ошибка: Данные отсутствуют.
Файл «data.csv» открыт.	Ошибка: Невозможно записать данные.

Данные ошибки исправлены. Также произведено тестирование прогнозирования на разных данных (см. рисунок 3.10 – 3.11).

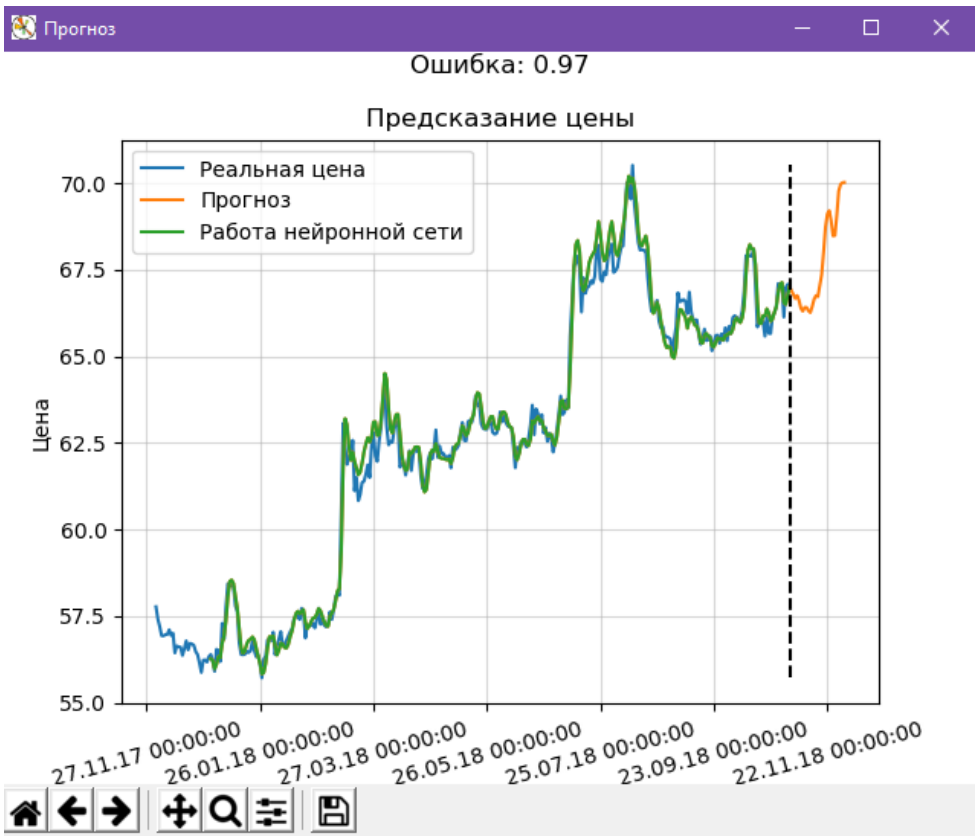

Рисунок 3.10 – Тестирование прогнозирования (USD/RUB, 1 день, 01.01.18 – 01.01.19)

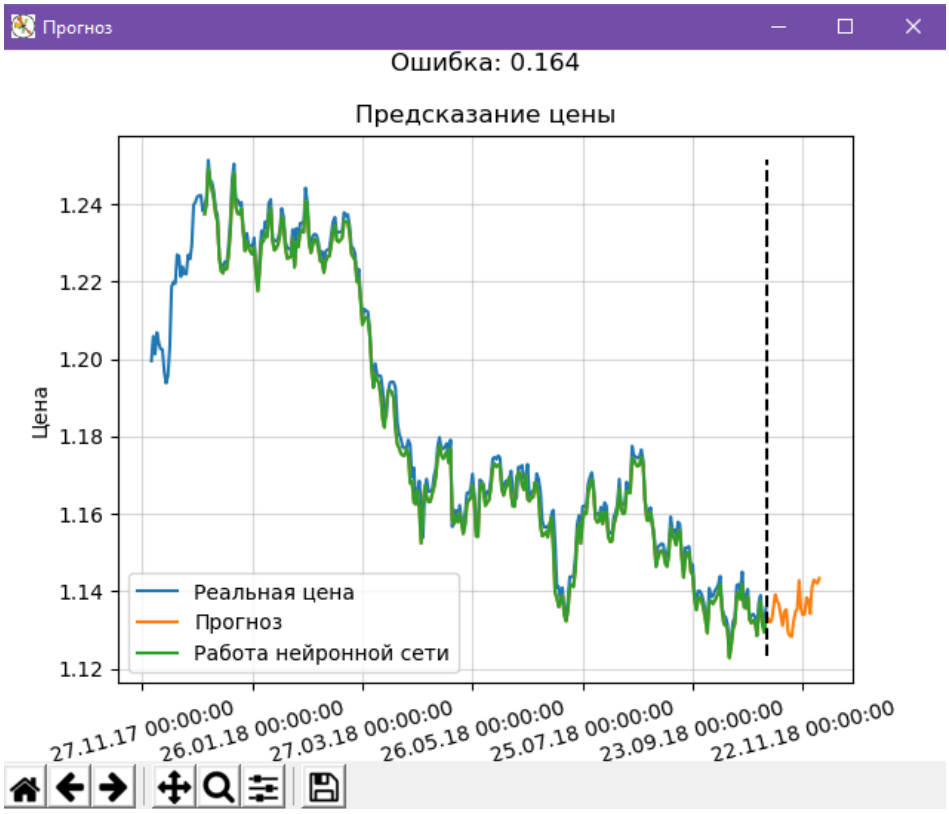

Рисунок 3.11 – Тестирование прогнозирования (EUR/USD, 1 день, 01.01.18 – 01.01.19)

3.4 Системные требования

Для запуска программы требуются следующие минимальные системные требования:

– процессор CPU – 2.2GHz или выше;

– 64-разрядная система Microsoft Windows 7;

– 2 ГБ ОЗУ;

– скорость Интернет-соединения – 256 килобит в секунду;

– 1 ГБ свободного места на жестком диске.

3.5 Инструкция пользователя

Для начала необходимо открыть файл program.exe. После запуска появится окно консоли, а спустя некоторое время – главное окно программы (см. рисунок 3.12).

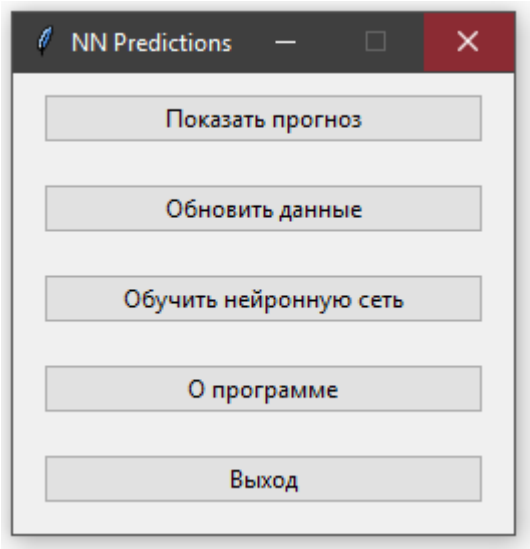

Рисунок 3.12 – Главное окно программы

Это окно содержит несколько пунктов:

– показать прогноз;

– обновить данные;

– обучить нейронную сеть;

– о программе;

– выход.

При выборе пункта «Показать прогноз» будет показан прогноз для имеющихся данных (см. рисунок 3.13).

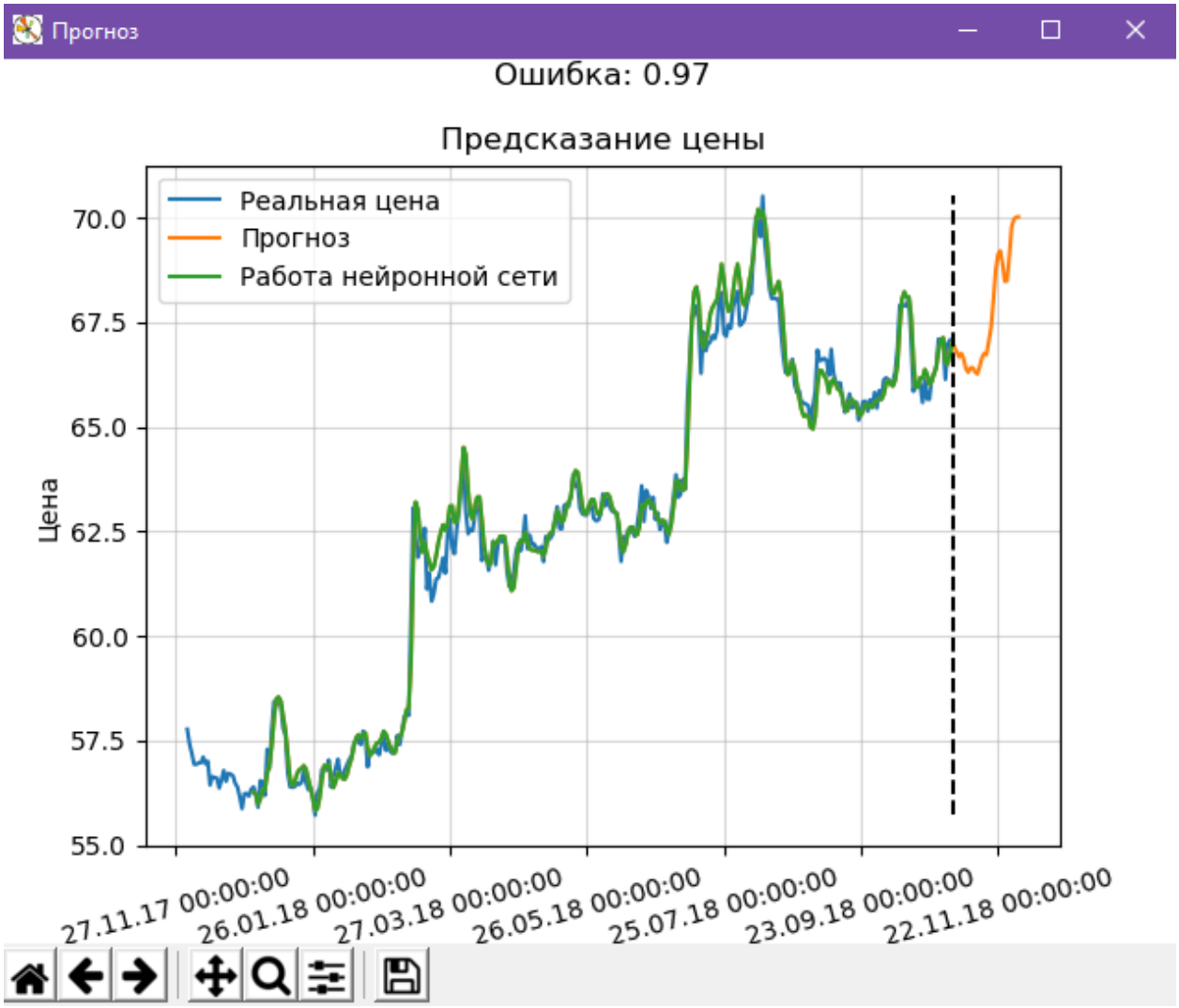

Рисунок 3.13 – Прогноз

Данный график можно перемещать, масштабировать, сохранить и т. д. Для этого есть специальная панель инструментов под графиком (см. рисунок 3.14).

Рисунок 3.14 – Панель инструментов

При выборе пункта «Обновить данные» появится окно с настройками для обновления данных для прогнозирования. Для обновления необходимо подключение к сети Интернет. После настройки необходимо нажать кнопку

«Обновить». По окончании обновления будет выдано сообщение об успехе, или сообщение об ошибке, если данные не удалось обновить (рисунок 3.15).

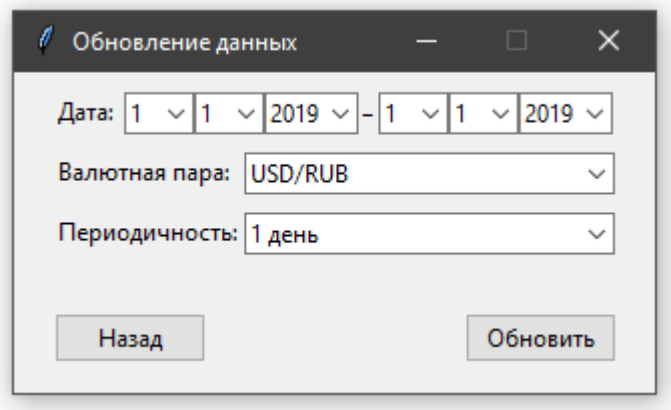

Рисунок 3.15 – Обновление данных

При выборе пункта меню «Обучить нейронную сеть» будет показано окно с настройками для обучения (рисунок 3.16).

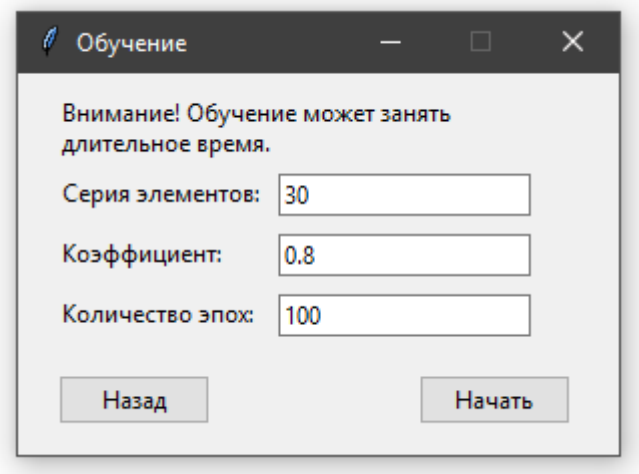

Рисунок 3.16 – Обучение нейронной сети

Для начала обучения необходимо нажать кнопку «Начать». После нажатия, появится уведомление о начале обучения. За ходом обучения можно наблюдать через консоль приложения (см. рисунок 3.17).

По окончании обучения нейронной сети, появится график, который демонстрирует качество обучения на обучающей и тестовой выборке (см. рисунок 3.18).

```
1536/5025 [========>.....................] - ETA: 16s - loss: 8.2407e-05 - mean_
2048/5025 [==========>...................] - ETA: 13s - loss: 7.6093e-05 - mean_
2560/5025 [============>.................] - ETA: 11s - loss: 7.5940e-05 - mean_
3072/5025 [==============>...............] - ETA: 9s - loss: 8.2980e-05 - mean_a
3584/5025 [================>.............] - ETA: 6s - loss: 8.2492e-05 - mean_a
4096/5025 [==================>...........] - ETA: 4s - loss: 8.1095e-05 - mean_a
4608/5025 [====================>.........] - ETA: 1s - loss: 8.4565e-05 - mean_a
5025/5025 [==============================] - 25s 5ms/step - loss: 8.2483e-05 - m
ean_absolute_error: 0.0046 - val_loss: 7.7642e-05 - val_mean_absolute_error: 0.0
060
Epoch 53/100
 512/5025 [==>...........................] - ETA: 20s - loss: 5.7444e-05 - mean_
1024/5025 [=====>........................] - ETA: 18s - loss: 5.9522e-05 - mean_
1536/5025 [========>.....................] - ETA: 16s - loss: 7.7706e-05 - mean_
2048/5025 [==========>...................] - ETA: 13s - loss: 7.5162e-05 - mean_
2560/5025 [============>.................] - ETA: 11s - loss: 7.2427e-05 - mean_
3072/5025 [==============>...............] - ETA: 8s - loss: 7.3414e-05 - mean_a
3584/5025 [================>.............] - ETA: 6s - loss: 7.4396e-05 - mean_a
4096/5025 [==================>...........] - ETA: 4s - loss: 7.4968e-05 - mean_a
4608/5025 [====================>.........] - ETA: 1s - loss: 7.9458e-05 - mean_a
5025/5025 [==============================] - 25s 5ms/step - loss: 8.1799e-05 - m
ean_absolute_error: 0.0046 - val_loss: 7.6937e-05 - val_mean_absolute_error: 0.0
059
Обучение остановлено на эпохе:52
```

Рисунок 3.17 – Ход обучения

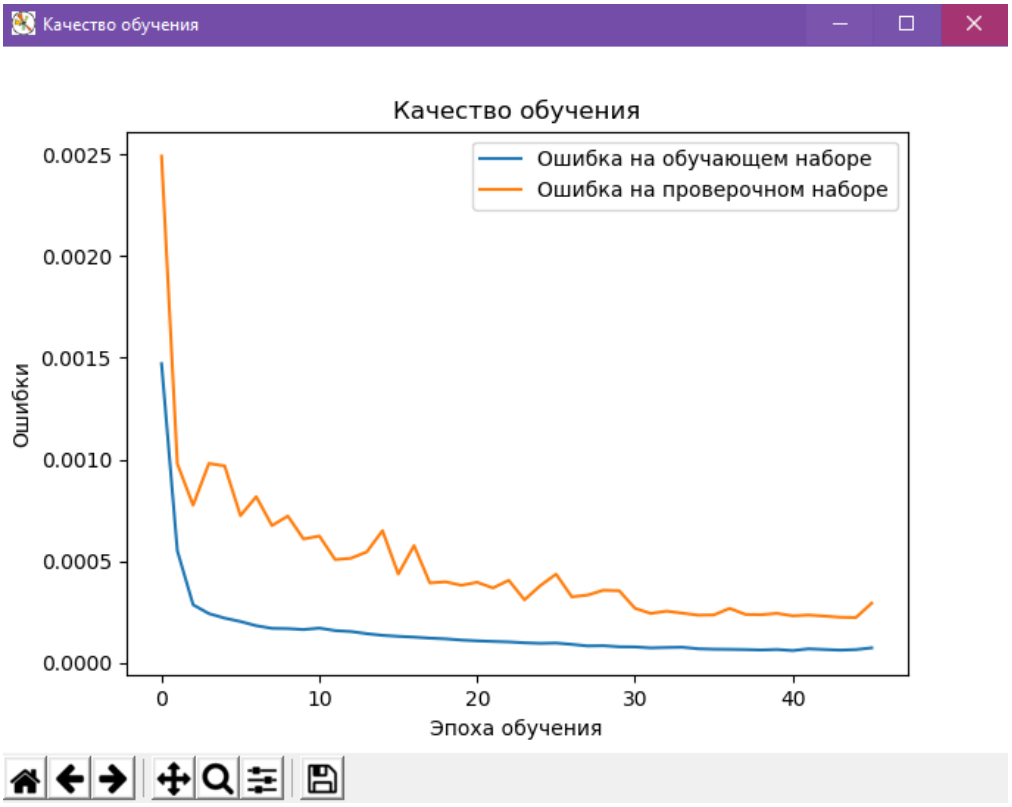

Рисунок 3.18 – График качества обучения

При выборе пункта меню «О программе» будет выдано сообщение с информацией о разработчике (см. рисунок 3.19).

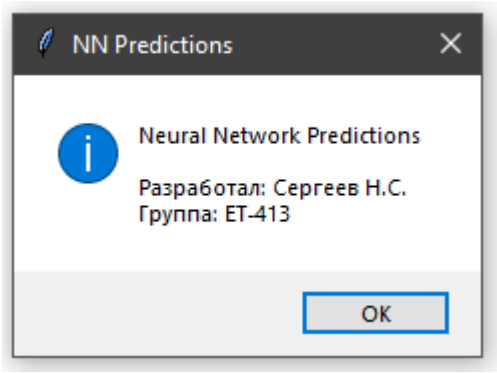

Рисунок 3.19 – О программе

При нажатии на пункт меню «Выход» приложение завершит работу.

3.6 Вывод по разделу

В данном разделе описана разработка и тестирования приложения для прогнозирования нелинейных процессов. Спроектирован интерфейс приложения. Для упрощения разработки, приложение разделено на модули:

– модуль gui.py;

– модуль nn.py;

– модуль download.py.

Для каждого из модулей разработан алгоритм, а также проведено тестирование этих алгоритмов. Также предоставлены минимальные системные требования и инструкция пользователя. В итоге был получен готовый программный продукт. В дальнейшем его можно развивать, например, прогнозирование для графика японских свечей.

ЗАКЛЮЧЕНИЕ

В результате выполненной работы получены следующие пункты.

1. Разработаны программы на языке Python для прогнозирования нелинейных процессов на примере валютного рынка Forex, а также для проверки временного ряда на стационарность.

2. В теоретической части исследованы два подхода к анализу валютного рынка: фундаментальный и технический. Фундаментальный анализ подразумевает оценку всевозможных параметров, влияющих на рынок. Технический анализ разделяется на линейные (классический статистический анализ, корреляционный анализ и т. д.) и нелинейные (нейронные сети).

3. На первом этапе анализа выгруженные котировки валютной пары USD/RUB подготавливались к дальнейшей работе, т.к. обнаружены проблемы, связанные с имеющимися данными.

4. После подготовки, полученный временной ряд проверен на стационарность с помощью теста Дики-Фуллера, который дал положительные результаты.

5. Для прогнозирования нелинейных процессов использована модель рекуррентной нейронной сети с долгой краткосрочной памятью. Одним из плюсов данной нейронной сети является то, что она может запоминать данные на долгий промежуток времени, из-за чего эту нейронную сеть не нужна часто переобучать.

6. Реализован интерфейс пользователя, разработаны алгоритмы, продемонстрированные в виде схем, подобраны оптимальные системные требования для запуска приложения и составлена инструкция для пользователя. Также проводилось тестирование на предмет ошибок. Найденные ошибки были описаны и исправлены.

Применение построенной нейронной сети к реальным данным показало, что среднеквадратичная ошибка достигает 3-5%, что принято считать хорошим результатом.

Работа над программой может быть продолжена по следующим направлениям:

1) обновление данных и прогноз в режиме реального времени (данные и прогноз обновляются автоматически).

2) оптимизация программы для работы с большими объемами данных.

3) прогноз на выбранное пользователем количество шагов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Авдеев, А.С. Разработка адаптивных моделей и программного комплекса прогнозирования экономических временных рядов – дис. канд. техн. наук. / А.С. Авдеев – Барнаул, 2010. – 196 с.

2 Авдеев, А.С. Разработка программного комплекса нейропрогнозирования / А.С. Авдеев, О.И. Пятковский // Программные продукты и системы. – Тверь, 2010. – №1. – С. 106-109.

3 Автоматизированные нейронные сети – Режим доступа: http://www.statsoft.ru/products/STATISTICA_Neural_Networks/ (Дата обращения: 04.11.2018).

4 Айвазян, С.А. Прикладная статистика. Основы эконометрики: Учебник для вузов: В 2 т. 2-е изд. – Т.1. / С.А. Айвазян, В.С. Мхитарян // М.: ЮНИТИ-ДАНА, 2001. – 656 с.

5 Аляутдинов, М.А. Нейрокомпьютеры: от программной к аппаратной реализации. / М.А. Аляутдинов, А.И. Галушкин, П.А. Казанцев, Г.П. Остапенко // М.: Горячая линия – Телеком, 2008. – 152 с.

6 Анненков, А.П. Разработка модели эволюции валютных котировок – дис. канд. экон. наук. / А.П. Анненков – Москва, 2011. – 182 с.

7 Балошников, А.М. Моделирование динамики обменных курсовоснов-ных валют / А.М. Балошников, В.А. Балошникова // Прикладная информатика, 2010. – № 1(25). – с. 15-20.

8 Беляков, С.С. Использование агрегирования в методах нелинейной динамики для анализа и прогнозирования временных рядов котировки акций: автореф. дис. канд. экон. наук / С.С. Беляков – Ставрополь, 2005. – 24 с.

9 Беляков, С.С. Использование агрегирования в методах нелинейной динамики для анализа и прогнозирования временных рядов котировки акций: автореф. дис. канд. экон. наук / С.С. Беляков – Ставрополь, 2005. – 24 с.

10 Галушкин, А.И. Нейронные сети, как последовательностные машины / А.И. Галушкин, Ю.И. Фомин – М.: МАИ, 1991. – 157 с.

11 Горбань, А.Н. Нейроинформатика / Горбань А.Н., Дунин-Барковский В.Л., Кирдин А.Н. и др. – Новосибирск: Наука. Сибирское предприятие РАН, 1998. – 296 с.

12 Горбань, А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев – Новосибирск: Наука, 1996. – 276 с.

13 Горбань, А.Н. Обучение нейронных сетей / А.Н. Горбань // М.: СССР-США СП "ПараГраф", 1990. – 160 с.

14 Канторович, Г.Г. Анализ временных рядов: Лекционные и методические материалы. Лекции № 1-4 / Г.Г. Канторович // Экономический журнал ВШЭ, 2002. – №1. С. 85-115.

15 Канторович, Г.Г. Анализ временных рядов: Лекционные и методические материалы. Лекции № 5-7 / Г.Г. Канторович // Экономический журнал ВШЭ, 2002. – №2. С. 251-273.

16 Канторович, Г.Г. Анализ временных рядов: Лекционные и методические материалы. Лекции № 8-9 / Г.Г. Канторович // Экономический журнал ВШЭ, 2002. – №3. С. 379-401.

17 Канторович, Г.Г. Анализ временных рядов: Лекционные и методические материалы. Лекции № 10-13 / Г.Г. Канторович // Экономический журнал ВШЭ, 2002. – №4. С. 498 – 523.

18 Канторович, Г.Г. Анализ временных рядов: Лекционные и методические материалы. Лекции № 14-16 / Г.Г. Канторович // Экономический журнал ВШЭ, 2003. – №1. С. 79 – 103.

19 Колодко, Д.В. Нестационарность и самоподобие валютного рынка Forex / Д.В. Колодко // Электронный научный журнал «Управление экономическими системами» №3. – 2012. – URL: http: //www.uecs.ru/ index.php?option=com_flexicontent&view=items&id= 1144 (Дата обращения: 08.11.2018).

20 Осминин, К.П. Алгоритмы построения статистик для анализа и прогнозирования нестационарных временных рядов / К.П. Осминин // Информационные технологии вычислительные системы, 2009. № 1. С. 3 – 15.

21 Осминин, К.П. Нестационарные временные ряды: Методы прогнозирования с примерами анализа финансовых и сырьевых рынков. / Ю.Н. Орлов, К.П. Осминин // М.: Книжный дом «ЛИБРОКОМ», 2011. – 384 с.

22 Ширяев, А.Н. Основы стохастической финансовой математики. Том 1. Факты. Модели. / А.Н. Ширяев // М.: ФАЗИС, 1998.

23 Ширяев, А.Н. Основы стохастической финансовой математики. Том 2. Теория. / А.Н. Ширяев // М.: ФАЗИС, 1998.

24 ТОЛКОВЫЙ ТРЕЙДЕРСКИЙ СЛОВАРЬ – URL: http://economy 24info.com/forex-treyding/tolkovyiy-treyderskiy-slovar-25846.html (Дата обращения: 08.11.2018).

25 Якимкин, В.Н. Как начать зарабатывать на валютном рынке Forex. / В.Н. Якимкин // М.: СмартБук, 2008. – 352 с.

26 Якимкин, В.Н. Кластерный статистический анализ рынков. / В.Н. Якимкин // Финансовая аналитика: проблемы и решения, № 7 (7), июль 2008. – С. 28 – 36.

27 Якимкин, В.Н. Новый подход к прогнозированию на рынке Forex. / В.Н. Якимкин // М.: СмартБук, 2008.

28 Якимкин, В.Н. Фундаментальный анализ. / В.Н. Якимкин // М.: Изд-во Омега-Л, 2008.

29 Яковлев, А. Стохастический подход к решению задач алгоритмической торговли / А. Яковлев, Г. Франгуриди // Журнал D', 2010. – №16. С. 46-49.

30 All About Forex – URL: http://allaboutforextradingmarket.blogspot.com (Дата обращения: 08.11.2018).

31 Callan, E.A Theory of Social Imitation / E. Callan, D. Shapiro // Physics Today, 1974.

32 Cambell, J.Y. The Econometric of Financial Markets / J.Y. Cam-bell // New Jersey: Princeton. University, 1997.

33 Cootner, P. The Random Character of stock Market Prices. / P. Cootner // Cambridge: MIT Press, 1964.

34 Cowles, A. Can Stock Market Forecasters Forecast? / A. Cowles // Econometrica, 1933. – Vol. 1, №3. – Pp. 309-324.

35 Hentshell, H.G.E. Fractal nature of turbulence as manifested in turbulent diffusion. / H.G.E. Hentshell, I. Procaccia // Phys. Rev., 1983. V. A27. P. 1266-1269.

36 Hilborn, R.C. Chaos and Nonlinear Dynamics / R.C. Hilborn // NY.: Oxford University Press, 2000.

37 Hurst, H.E. Long-term Storage of Reservoirs / H.E. Hurst // Transactions of the American Society of Civil Engineers, 1951 – 116 pp.

38 Kendeall, M.G. The analysis of economic time-series. Part I. Prices / M.G. Kendeall // Journal of the Royal Statistical Society, 1953 – V. 96. P.11-25.

39 Kohonen, T. Self-Organization and Associative Memory / T. Kohonen // Springer, second edition, 1988 – 312 pp.

40 Douglas, C. Johnson. Forecasting and time series analysis. / C. Johnson Douglas, A. Gardiner Lynwood, S. John // McGraw-Hill, New York – 1990.

ПРИЛОЖЕНИЕ 1

Код программы

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from tkinter import *
import datetime
import urllib.request
import pandas as pd
import numpy as np
import download as dw
import nn as nn
from keras.models import Sequential

# Валютные пары
currency_name=['USD/RUB','RUB/USD','EUR/RUB','RUB/EUR','EUR/USD']
# Периоды
period_name=['1 мин.','5 мин.', '10 мин.','15 мин.',
             '30 мин.','1 час','1 день','1 неделя','1 месяц']

root=tk.Tk() # Основное окно
window = tk.Toplevel(root) # Переменная окна "Обучение"
upd = tk.Toplevel(root) # Переменная окна "Обновление данных"
# Вычисление середины экрана
x = (root.winfo_screenwidth() - root.winfo_reqwidth()) / 2
y = (root.winfo_screenheight() - root.winfo_reqheight()) / 2

# Загрузка окна "Обучение"
def Retraining_window(event):
    HideAndShow(1)
    window.resizable(width=False, height=False)
    window.geometry("300x190")
    window.title("Обучение")
    button2=ttk.Button(window,text='Начать',command=            lambda:
CheckReqData_Retrain(textbox1,textbox2,textbox3))
    button1=ttk.Button(window,text='Назад')
    button1.bind('<Button-1>',ShowMain)
    button1.place(x=20, y=150)
    button2.place(x=200, y=150)
    txt="Внимание! Обучение может занять\nдлительное время."
    label1 = ttk.Label(window,text=txt)
    label1.place(x=20, y=10)
    label2 = ttk.Label(window,text="Серия элементов:")
    label2.place(x=20, y=50)
    label3 = ttk.Label(window,text="Коэффициент:")
    label3.place(x=20, y=80)
    label4 = ttk.Label(window,text="Количество эпох:")
    label4.place(x=20, y=110)
    textbox1 = ttk.Entry(window)
    textbox1.delete(0, 1000)
    textbox1.insert(0, "30")
    textbox1.place(x=130, y=50)
    textbox2 = ttk.Entry(window)
    textbox2.delete(0, 1000)
    textbox2.insert(0, "0.8")
```

```
    textbox2.place(x=130, y=80)
    textbox3 = ttk.Entry(window)
    textbox3.delete(0, 1000)
    textbox3.insert(0, "100")
    textbox3.place(x=130, y=110)
    window.wm_geometry("+%d+%d" % (x, y))

# Получение кода валютной пары
def GetCode(name):
    if(name=='USD/RUB'):
        return 901
    if(name=='RUB/USD'):
        return 176164
    if(name=='EUR/RUB'):
        return 66860
    if(name=='RUB/EUR'):
        return 176165
    if(name=='EUR/USD'):
        return 83

# Загрузка окна "Обновление данных"
def Update_window(event):
    HideAndShow(2)
    upd.resizable(width=False, height=False)
    upd.geometry("320x160")
    upd.title("Обновление данных")
    combobox           =           ttk.Combobox(upd,values           =
currency_name,height=6,width=27)
    combobox.set(u"USD/RUB")
    combobox.place(x=115, y=40)
    combobox1           =           ttk.Combobox(upd,values           =
period_name,height=6,width=27)
    combobox1.set(u"1 день")
    combobox1.place(x=115, y=70)
    a=[]
    for i in range(1,32):
        a.append(i)
    d_start = ttk.Combobox(upd,values = a,width=2)
    d_start.set(a[0])
    d_start.place(x=55, y=10)
    a=[]
    for i in range(1,13):
        a.append(i)
    m_start = ttk.Combobox(upd,values = a,width=2)
    m_start.set(a[0])
    m_start.place(x=90, y=10)
    a=[]
    for i in range(1979,datetime.datetime.now().year+1):
        a.append(i)
    y_start = ttk.Combobox(upd,values = a,width=4)
    y_start.set(a[len(a)-1])
    y_start.place(x=125, y=10)
    label = ttk.Label(upd,text="_")
    label.place(x=172, y=6)
    a=[]
    for i in range(1,32):
        a.append(i)
```

```
    d_end = ttk.Combobox(upd,values = a,width=2)
    d_end.set(a[0])
    d_end.place(x=182, y=10)
    a=[]
    for i in range(1,12):
        a.append(i)
    m_end = ttk.Combobox(upd,values = a,width=2)
    m_end.set(a[0])
    m_end.place(x=217, y=10)
    a=[]
    for i in range(1979,datetime.datetime.now().year+1):
        a.append(i)
    y_end = ttk.Combobox(upd,values = a,width=4)
    y_end.set(a[len(a)-1])
    y_end.place(x=252, y=10)
    button2=ttk.Button(upd,text='Обновить',command=             lambda:
CheckReqData_Download(d_start,m_start,y_start,d_end,m_end,y_end,combob
ox,combobox1))
    button1=ttk.Button(upd,text='Назад')
    button1.bind('<Button-1>',ShowMain)
    button1.place(x=20, y=120)
    button2.place(x=225, y=120)
    label = ttk.Label(upd,text="Дата:")
    label.place(x=20, y=10)
    label = ttk.Label(upd,text="Валютная пара:")
    label.place(x=20, y=40)
    label = ttk.Label(upd,text="Периодичность:")
    label.place(x=20, y=70)
    upd.wm_geometry("+%d+%d" % (x, y))

# Передача данных в функцию обновления данных
def CheckReqData_Download(ds,ms,ys,de,me,ye,cur,period):
    code_val=GetCode(cur.get())
    code_name=cur.get().replace('/','')
    per=period_val(period.get())
    print('Скачивание             данных             для',cur.get(),'за
период',ds.get(),'.',ms.get(),'.',ys.get(),
          '-',de.get(),'.',me.get(),'.',ye.get(),'c
интервалом',period.get(),'...')

dw.func_UpdateDate(code_val,code_name,ds.get(),ms.get(),ys.get(),de.ge
t(),me.get(),ye.get(),per)

# Передача данных в функцию обучения
def CheckReqData_Retrain(series,k,epochs):
    nn.func_UpgradeNN(series.get(),k.get(),epochs.get())

# Получение код периода
def period_val(per):
    if (per=='5 мин.'):return 3
    elif (per=='10 мин.'):return 4
    elif (per=='15 мин.'):return 5
    elif (per=='30 мин.'):return 6
    elif (per=='1 час'):return 7
    elif (per=='1 день'):return 8
    elif (per=='1 неделя'):return 9
    elif (per=='1 месяц'):return 10
```

```
    else: return 2

# Сообщение "О разработчике"
def About_msg(event):
    msg="Neural     Network     Predictions\n\nРазработал:     Сергеев
H.C.\nГруппа: ET-413"
    messagebox.showinfo("NN Predictions",msg)

# Выход из программы
def Quit(event):
    root.destroy()

# Загрузка основного окна
def ShowMain(event):
    HideAndShow(0)
    root.resizable(width=False, height=False)
    root.title("NN Predictions")
    root.geometry("250x230")
    root.wm_geometry("+%d+%d" % (x, y))
    button1=ttk.Button(root,text='Показать   прогноз',width=35,command=
lambda: nn.func_LoadNN())
    button2=ttk.Button(root,text='Обновить данные',width=35)
    button3=ttk.Button(root,text='Обучить нейронную сеть',width=35)
    button4=ttk.Button(root,text='Выход',width=35)
    button5=ttk.Button(root,text='О программе',width=35)
    button2.bind('<Button-1>',Update_window)
    button3.bind('<Button-1>',Retraining_window)
    button5.bind('<Button-1>',About_msg)
    button4.bind('<Button-1>',Quit)
    button1.pack(pady=10, padx=10)
    button2.pack(pady=10, padx=10)
    button3.pack(pady=10, padx=10)
    button5.pack(pady=10, padx=10)
    button4.pack(pady=10, padx=10)
    root.mainloop()

# Выбор текущего окна
def HideAndShow(i):
    root.withdraw()
    window.withdraw()
    upd.withdraw()
    if i==1:
        window.deiconify()
    elif i==2:
        upd.deiconify()
    else:
        root.deiconify()

# Запуск программы
ShowMain(0)


import numpy as np
from numpy import newaxis
import pandas as pd
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM, GRU
```

```
from keras.models import Sequential
from keras import optimizers
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime
from dateutil import parser
from matplotlib.ticker import MultipleLocator, FormatStrFormatter
import random
from keras.models import model_from_json
from tensorflow.keras.callbacks import EarlyStopping
from tkinter import messagebox

# Подготовка данных
def load_data(datasetname, column, seq_len, normalise_window,k):
    data = datasetname.loc[:,column]
    sequence_length = seq_len + 1
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index: index + sequence_length])
    if normalise_window:
        result = normalise_windows(result)
    result = np.array(result)
    row = round(k * result.shape[0])
    train = result[:int(row), :]
    np.random.shuffle(train)
    x_train = train[:, :-1]
    y_train = train[:, -1]
    x_test = result[int(row):, :-1]
    y_test = result[int(row):, -1]

    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],
1))
    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

    return [x_train, y_train, x_test, y_test]

# Преобразование данных под размер окна
def normalise_windows(window_data):
    normalised_data = []
    for window in window_data:
        normalised_window = [((float(p) / float(window[0])) - 1) for p
in window]
        normalised_data.append(normalised_window)
    return normalised_data

# График прогноза
def                                    plot_results(predicted_data,
true_data,datelist1,datelist2,evaluate,pred_c):
    converted_dates1    =    list(map(datetime.strptime,    datelist1,
len(datelist1)*['%d.%m.%y %H:%M:%S']))
    converted_dates2    =    list(map(datetime.strptime,    datelist2,
len(datelist2)*['%d.%m.%y %H:%M:%S']))
    x_axis_pred = converted_dates1
    x_axis_true = converted_dates2[:len(true_data)]
    x_axis_work = converted_dates1[:len(converted_dates1)-pred_c+1]
    work_data = predicted_data[:len(converted_dates1)-pred_c+1]
```

```
    formatter = mdates.DateFormatter('%d.%m.%y %H:%M:%S')
    fig, ax = plt.subplots(facecolor='white')
    fig.canvas.set_window_title('Прогноз')
    plt.xticks(rotation=15)
    ax.xaxis.set_major_formatter(formatter)
    ax.xaxis.set_major_locator(plt.MaxNLocator(8))
    ax.yaxis.set_major_locator(plt.MaxNLocator(8))
    z_txt='\nОшибка: '+str(evaluate)+'\n\nПредсказание цены'
    plt.title(z_txt)
    ax.plot(x_axis_true, true_data, label='Реальная цена')
    ax.plot(x_axis_pred, predicted_data, label='Прогноз')
    ax.plot(x_axis_work, work_data, label='Работа нейронной сети')
    plt.xlabel('День')
    plt.ylabel('Цена')
    ax.grid(which='minor', alpha=0.2)
    ax.grid(which='major', alpha=0.5)
    z_date=x_axis_true[len(x_axis_true)-1]
    xs = [z_date,z_date]
    series1=[min(true_data),max(true_data)]
    plt.plot(xs, series1, linestyle='--',color='k')
    ax.grid(True)
    plt.legend()
    plt.show()

# Исправление формата часов
def FixHours(str1):
    str2=str1.split(':')
    if(len(str2[0])==1):
        str2[0]='0'+str2[0]
    str3=str2[0]+':'+str2[1]+':'+str2[2]
    return str3

# Преобразование столбцов Date и Time в DateTime
def ConvToDateTime(dataset,count1,tr):
    from datetime import datetime
    from dateutil import parser
    datatime1=[]
    close1=[]
    open1=[]
    count=dataset.shape[0]
    for i in range(count):
        datatime1.append(dataset['Date'][i]+'
'+FixHours(dataset['Time'][i]))
        close1.append(dataset['Close'][i])
        open1.append(dataset['Open'][i])
    if(count1>0):
        dt1 = parser.parse(datatime1[count-2])
        dt2 = parser.parse(datatime1[count-3])
        a=dt1-dt2
        dt3=parser.parse(datatime1[count-1])
        for i in range(0,count1):
            dt3+=a
            datatime1.append(dt3.strftime("%d.%m.%y %H:%M:%S"))
            close1.append(close1[len(close1)-1]+random.uniform(-1, 1))
            open1.append(open1[len(open1)-1])
    df2                                                    =
pd.DataFrame({"DateTime":datatime1,"Open":open1,"Close":close1})
```

```
    if(tr==True):
        df2.set_index('DateTime', inplace=True)
    return df2

# Обучение нейронной сети
def func_UpgradeNN(days,k,epochs):
    try:
        Enrol_window=int(days)
        k=float(k)
        epochs=int(epochs)
        if(Enrol_window<5):
            raise ValueError("Количество серий должно быть больше или
равно 5")
        if(Enrol_window>150):
            raise  ValueError("Количество  серий  должно  быть  меньше
150")
        if(k<=0 or k>=1):
            raise ValueError("Коэффициент обучаемых данных к тестовым
должен быть в диапзоне (0;1)")
        if(epochs<10):
            raise ValueError("Минимальное количество эпох = 10")
        dataset                                                        =
pd.read_csv('data.csv',sep=';',index_col=DatetimeIndex('data.csv'))
        if(dataset.shape[0]<=Enrol_window):
            raise  ValueError("Количество  серий  должно  быть  меньше
количества данных")
        feature_train,   label_train,   feature_test,   label_test   =
load_data(dataset, 'Close', Enrol_window, True,k)
        model = Sequential()
        model.add(LSTM(256,                        return_sequences=True,
input_shape=(feature_train.shape[1],1)))
        model.add(Dropout(0.2))
        model.add(LSTM(units=256, return_sequences=True))
        model.add(LSTM(units=256))
        model.add(Dense(units = 1))
        model.compile(optimizer = 'adam', loss = 'mean_squared_error',
metrics=['mae'])
        esc=EarlyStopping(monitor='loss', patience=5)
        cb_list=[esc]
        print('Обучение   началось!\nПосле   завершения   Вам   будет
предоставлен график ошибок обучения.')
        history=model.fit(feature_train,  label_train,  batch_size=512,
epochs=epochs,    validation_data    =    (feature_test,    label_test),
callbacks=cb_list)
        str123="Обучение остановлено на эпохе:"+str(esc.stopped_epoch)
        if(esc.stopped_epoch>0):
            print(str123)
        model_json=model.to_json()
        json_file=open("nn.json","w")
        json_file.write(model_json)
        json_file.close()
        model.save_weights("nn.h5")
        handle = open("settings.bin", "w")
        handle.write(days)
        handle.close()
        print('Обучение завершено!')
        fig, ax = plt.subplots(facecolor='white')
```

```
        fig.canvas.set_window_title('Качество обучения')
        plt.title('Качество обучения')
        plt.plot(history.history['loss'],label="Ошибка   на   обучающем
наборе")
        plt.plot(history.history['val_loss'],label="Ошибка           на
проверочном наборе")
        plt.xlabel('Эпоха обучения')
        plt.ylabel('Ошибки')
        plt.legend()
        plt.show()
    except Exception as e:
        msg="Не удалось обучить нейронную сеть.\n"
        msg+="Причина: "+ str(e)
        messagebox.showinfo("Обучение",msg)

# Добавление сдвига к дате
def AddToDate(dates,c1):
    dates2=dates
    dates_conv=[]
    str1=[]
    for i in range(len(dates)):
        dates_conv.append(parser.parse(dates2[i], dayfirst=True))
    a=dates_conv[len(dates_conv)-1]-dates_conv[len(dates_conv)-2]
    step=a
    for i in range(c1-1):
        step+=a
    for i in range(len(dates)):
        str1.append((dates_conv[i]-step).strftime("%d.%m.%y
%H:%M:%S"))
    return str1

# Выбор индексного столбца
def DatetimeIndex(name):
    data = pd.read_csv(name,sep=';')
    a=parser.parse(data['Date'][data.shape[0]-2], dayfirst=True)
    b=parser.parse(data['Date'][data.shape[0]-1], dayfirst=True)
    d=parser.parse('01.01.2019 00:00:00', dayfirst=True)
    e=parser.parse('02.01.2019 00:00:00', dayfirst=True)
    c=b-a
    f=e-d
    if(c<f): return 'Time'
    else: return 'Date'

# Загрузка нейронной сети и прогнозирование
def func_LoadNN():
    try:
        print('Загрузка прогноза...')
        handle = open("settings.bin", "r")
        days = handle.read()
        handle.close()
        pred_c=0
        Enrol_window=int(days)
        dataset                                                    =
pd.read_csv('data.csv',sep=';',index_col=DatetimeIndex('data.csv'))
        dataset3  = pd.read_csv('data.csv',sep=';')
        z2=ConvToDateTime(dataset3,pred_c,True)
        z3=ConvToDateTime(dataset3,pred_c,False)
```

```
        feature_train2,  label_train2,  feature_test2,  label_test2  =
load_data(z2, 'Close', Enrol_window, True,0)
        feature_train3,  label_train3,  feature_test3,  label_test3  =
load_data(dataset, 'Close', Enrol_window, True,0)
        json_file = open('nn.json', 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        model = model_from_json(loaded_model_json)
        model.load_weights("nn.h5")
        model.compile(optimizer = 'adam', loss = 'mean_squared_error',
metrics=['mae'])
        predicted_stock_price = model.predict(feature_test2)
        score = model.evaluate(feature_test3,label_test3,verbose=1)
        evaluate=round(score[0]*10000,3)
        dates=[]
        true=[]
        pred=[]
        values=[]
        count=0
        for i in range(len(feature_test2)):
            dates.append(z3['DateTime'][i])
        dates1=AddToDate(dates,0)
        dates2=AddToDate(dates,int(Enrol_window))
        for i in range(len(label_test3)):
            values.append(dataset3['Close'][i])
        for i in range(len(label_test3)):
            true.append((feature_test2[i][0]+1)*values[i])
            pred.append((predicted_stock_price[i]+1)*values[i])
            count+=1
        for i in range(pred_c):

pred.append((predicted_stock_price[i+count]+1)*pred[i+count-1])
        plot_results(pred,true,dates1,dates2,evaluate,Enrol_window)
    except Exception as e:
        msg="Не удалось получить прогноз.\n"
        msg+="Причина: "+ str(e)
        messagebox.showinfo("Прогноз",msg)


import urllib.request
import numpy as np
from urllib.error import URLError, HTTPError
from tkinter import messagebox

# Загрузка данных с сайта Finam
def func_UpdateDate(code_val,code_name,d_s,m_s,y_s,d_e,m_e,y_e,per):
    msg=""
    try:
        fp                                                          =
urllib.request.urlopen("http://export.finam.ru/abc.txt?market=5&em="+s
tr(code_val)+"&code="+str(code_name)+"&apply=0&df="+str(d_s)+"&mf="+st
r(int(m_s)-1)+

"&yf="+str(y_s)+"&from="+str(d_s)+"."+str(m_s)+"."+str(y_s)+"&dt="+str
(d_e)+"&mt="+str(int(m_e)-1)+"&yt="+str(y_e)+"&to="+str(d_e)+
```

```
"."+str(m_e)+"."+str(y_e)+"&p="+str(per)+"&f=abc&e=.txt&cn=abc&dtf=4&t
mf=3&MSOR=1&mstime=on&mstimever=1&sep=1&sep2=2&datf=5&at=1")
        mybytes = fp.read()
        mystr =str(mybytes.decode('cp1251'))
        fp.close();
        if len(mystr)<=0:
            raise ValueError('Количество полученных значений = 0')
        else:
            if mystr.index("<DATE>")>=0:
                mystr=mystr.replace('\r','').split('\n')[1:len(mystr)]
                mystr=mystr[0:len(mystr)-1]
                for i in range(len(mystr)):
                    mystr[i]=mystr[i].replace('/','.').split(',')
                dates = []
                times = []
                v_open = []
                v_close= []
                for i in range(len(mystr)):
                    dates.append(mystr[i][0])
                    times.append(mystr[i][1])
                    v_open.append(mystr[i][2])
                    v_close.append(mystr[i][5])
            else:
                raise ValueError(mystr)
    except Exception as e:
        msg+="Не удалось скачать данные.\n"
        msg+="Причина: "+ str(e)
        messagebox.showinfo("Обновление",msg)
    except HTTPError as e:
        msg+="Не удалось скачать данные.\n"
        msg+="Код ошибки: "+ e.code
        messagebox.showinfo("Обновление",msg)
    except URLError as e:
        msg+="Не удалось связаться с сервером.\n"
        msg+="Причина: "+ e.reason
        messagebox.showinfo("Обновление",msg)
    else:
        try:
            f = open("data.csv", "w")
            f.write("{};{};{};{}\n".format("Date",   "Time",   "Open",
"Close"))
            for x in zip(dates, times, v_open,v_close):
                f.write("{};{};{};{}\n".format(x[0],    x[1],    x[2],
x[3]))
            f.close()
        except IOError as e:
            msg+="Данные успешно получены!\n"
            msg+="Не удалось записать данные на диск.\n"
            msg+="Причина: "+ e.strerror+"\n"
            messagebox.showinfo("Обновление",msg)
        else:
            msg+="Данные успешно получены и записаны в файл data.csv!"
            messagebox.showinfo("Обновление",msg)
```

ПРИЛОЖЕНИЕ 2

Тест Дики-Фуллера

```
import pandas as pd
import math
# Загрузка файла
df = pd.read_csv('Z2.csv',sep=';')
z=[]
z2=[]
for i in range (df.shape[0]):
    z2.append(df.Close.values[i])
# Преобразование данных по формуле
for i in range(1,len(z2)):
    z.append((z2[i]/z2[0])-1)
#ADF
import statsmodels.tsa.stattools as st
adf = st.adfuller(z, maxlag = 5)
print('ADF Statistic: %f' % adf[0])
print('p-value: %f' % adf[1])
print('Critical Values:')
for key, value in adf[4].items():
    print('\t%s: %.3f' % (key, value))
```