Санкт-Петербургский государственный университет

***АРСЛАНОВ Николай Адельевич***

**Выпускная квалификационная работа**

***«Разработка и практическое применение нейронных сетей на фондовом рынке»***

Уровень образования:

Направление *38.03.01 «Экономика»*

Основная образовательная программа *СВ.5104.2016 «Экономико-математические методы»*

Научный руководитель: доцент, Кафедра экономической кибернетики, кандидат экономических наук,

Колесов Дмитрий Николаевич

Рецензент: заместитель генерального директора, партнер, Общество с ограниченной ответственностью «Сбондс.ру», кандидат экономических наук,

Васильев Константин Геннадьевич

Санкт-Петербург

2020

Оглавление

[Введение 3](#_Toc41436488)

[Глава 1 Концептуальная модель нейросетевых систем 5](#_Toc41436489)

[1.1 Общая информация о нейронных сетях 5](#_Toc41436490)

[1.2 Классификация нейронных сетей 7](#_Toc41436491)

[1.3 Поэтапное представление процесса разработки 9](#_Toc41436492)

[Глава 2 Интеллектуальные системы в финансовой сфере экономики 15](#_Toc41436493)

[2.1 Применение нейронных сетей на фондовом рынке 15](#_Toc41436494)

[2.2 Выбор и анализ финансовых активов 17](#_Toc41436495)

[Глава 3 Интеллектуальная обработка данных 21](#_Toc41436496)

[3.1 Общая информация об аномалиях 21](#_Toc41436497)

[3.2 Методы обнаружения аномалий 24](#_Toc41436498)

[3.3 Алгоритм поиска аномалий 27](#_Toc41436499)

[3.4 Программа для обработки данных 29](#_Toc41436500)

[Глава 4 Разработка и практическое применение нейросетевых систем 34](#_Toc41436501)

[4.1 Разработка интеллектуальных систем 34](#_Toc41436502)

[4.1.1 Многослойная нейронная сеть 34](#_Toc41436503)

[4.1.2 Рекуррентная нейронная сеть 44](#_Toc41436504)

[4.2 Статистический анализ временных рядов 50](#_Toc41436505)

[Заключение 58](#_Toc41436506)

[Список использованных источников 61](#_Toc41436507)

[Приложение 1 64](#_Toc41436508)

## Введение

Эффективное управление ресурсами является основной экономической задачей современного мира, вследствие этого повышение показателей роста производительности труда является ключевой проблемой в экономике. Вышеуказанная задача всегда решается путем внедрения новых технологий. В процессе технологической эволюции на первый план выходят интеллектуальные системы управления, которые в свою очередь находятся в постоянном состоянии непрерывного развития. Важным свойством таких систем является способность обучаться и использовать накопленный опыт для принятия решений. Существует достаточно много алгоритмов, которые применяются для обучения таких систем. Но наиболее известным и часто используемым из них является класс алгоритмов под названием “нейронные сети”.

Нейронная сеть - это математическая модель, которая состоит из большого числа простых элементов, соединенных и взаимодействующих между собой. Обучение в этом случае представляет собой поиск оптимальной структуры модели, при которой будет достигаться минимальное расхождение получаемого результата со значениями реальных данных.

На данный момент интеллектуальные системы, построенные на базе нейронных сетей, находят широкое применение во многих областях жизнедеятельности человека, например: в медицине, в IT-сфере и в различных производственных бизнес-процессах. В рамках этой работы будут рассматриваться применение ИИ в финансовой сфере экономики, частью которой является фондовый рынок.

Область применения ИИ на фондовом рынке достаточно широка. Например, он может использоваться при прогнозировании стоимости активов, оценки финансовых рисков, а также он может исполнять часть торгового функционала при операционной деятельности участников рынка.

С ростом доступности цифровых технологий повышается актуальность исследуемой темы. В настоящий момент технологии, использующие нейронные сети, проникают практически во все сферы нашей жизни, поэтому возникает потребность в глубоком исследовании данного вопроса. Поэтому цель данной научно-исследовательской работы заключается в определении и анализе эффективности методик, применяемых в процессе разработки нейронных сетей.

Для реализации данной цели потребовалось решить следующие исследовательские задачи, а именно:

1. Дать характеристику нейронным сетям;
2. Составить поэтапный план разработки;
3. Рассмотреть применение нейронных сетей на фондовом рынке;
4. Выбрать и проанализировать финансовые активы;
5. Выполнить предобработку исходных данных;
6. Разработать и протестировать программы с нейросетевой структурой;
7. Провести оценку методами статистического анализа данных;
8. Сравнить полученные результаты.

Научно-исследовательская работа основана на трудах Т.Теплова и Т.Соколова «Исследовательские поля облигационных рынков», Ф.Фабоцци «Рынок облигаций. Анализ и стратегии» и А.Буренина «Рынок ценных бумаг и производных финансовых инструментов». Помимо этого, были использованы работы Ф.Гафарова и А.Галимянова «Искусственные нейронные сети и их приложения», В.Круглова и В.Борисова «Искусственные нейронные сети. Теория и практика», Т.Рашида «Создаем нейронную сеть», Ф.Шолле «Глубокое обучение на Python».

Реализация вышеуказанных цели и задач обусловила структуру и логику данной работы. Она состоит из четырех глав, введения, заключения, списка используемых источников и приложений. В первой главе представлена общая характеристика нейронных сетей, и выделен поэтапный план их разработки. Во второй главе приводится информация об использовании нейронных сетей на фондовом рынке. Помимо этого, здесь осуществляется выбор и анализ финансовых активов, выделяются исходные данные и определяются задачи для нейронной сети. В третьей главе рассматриваются методы, применяемые для поиска аномальных экземпляров данных. Далее на их основе составляется алгоритм обработки данных, который затем реализуется на программном уровне. В четвертой главе происходит разработка и тестирование программ с нейросетевой структурой. Затем осуществляется оценка данных с помощью методов анализа временных рядов. Наконец, проводится сравнительный анализ между результатами программ и статистической оценки данных, делаются выводы.

## Глава 1 Концептуальная модель нейросетевых систем

### 1.1 Общая информация о нейронных сетях

В настоящее время все большее внимание уделяется тем областям науки, которые занимаются развитием и внедрением систем с искусственным интеллектом в различные сферы деятельности человека. Такие системы должны самостоятельно анализировать имеющиеся данные и решать поставленные задачи. Для этого были разработаны специальные алгоритмы, которые дают возможность ИИ обучаться и использовать накопленный опыт для принятия решений. Одним из таких классов алгоритмов являются искусственные нейронные сети.

Искусственная нейронная сеть (ИНС) – это математическая модель, которая состоит из большого числа простых элементов – нейронов, определенным образом соединенных и взаимодействующих между собой [30].

Интеллектуальные системы, построенные на базе нейронных сетей, способны решать задачи, непосредственно связанные с процессом обучения. Например, сюда можно отнести задачи прогнозирования, оценки рисков, анализа данных, оптимизации и прочее.

Применение искусственных нейронных сетей несет в себе как положительные, так и отрицательные последствия. Поэтому можно выделить преимущества и недостатки, которые оказывают наибольшее влияние на функционирование системы ИИ в целом. Так, к преимущественным факторам относят:

1. **Способность ИНС решать задачи в условиях неопределенности реальных зависимостей между входными и выходными данными.** Это возможно благодаря тому, что в процессе обучения нейронная сеть регулирует свои внутренние параметры таким образом, чтобы конечный результат наилучшим образом совпадал со значением тренировочных данных. При этом реальные зависимости между данными могут оставаться неизвестными.
2. **Уменьшение влияния шума входных данных**. Обучение ИНС происходит постепенно, и с каждой новой итерацией уменьшается величина ошибки. Одновременно с этим нейронная сеть ограничивает влияние искаженных данных, делая его несущественным. Поэтому независимо оттого, какая информация будет на входе, нейронная сеть отфильтрует и, по необходимости, отбросит ее.
3. **Адаптация к внешней среде.** Поскольку внешняя среда постоянно изменяется, то возникает необходимость в обновлении структуры и параметров нейронной сети для получения более точных результатов. Для этого требуется обучить сеть, но уже на новых тренировочных данных. Отсюда следует, что одной из ключевых особенностей нейронной сети является возможность ее переобучения.
4. **Высокая эффективность обработки информации.** Искусственная нейронная сеть способна быстро анализировать получаемую информацию, так как она осуществляет параллельную обработку данных.
5. **Надежность работы ИНС на программном уровне.** Нейронные сети достаточно устойчивы к повреждениям структурных элементов. Это связано с тем, что информация распределяется по всем элементам ИНС, и при незначительном повреждении структуры теряется лишь некоторая ее часть. Как следствие, происходит малозначимое снижение производительности и качества выдаваемых результатов.

К существенным недостаткам можно отнести следующее:

1. **Проблемы, возникающие при интерпретации результатов работы ИНС.** Это объясняется тем, что алгоритмы работы системы, построенной на базе нейронных сетей, скрыты от конечного пользователя, и поэтому становится сложно понять, почему система выдала тот или иной результат, приняла то или иное решение.
2. **Неоднозначность получаемых результатов.** Поскольку нейронная сеть постоянно корректируется под воздействием внешней среды, то невозможно гарантировать получение одинаковых результатов.
3. **Сложности подбора данных для обучающей выборки.** Эта проблема характеризуется тем, что эффективность обучения ИНС, прежде всего, зависит от обучающей выборки. Правильно подобранные тренировочные данные могут значительно повысить скорость обучения и улучшить качество выдаваемых результатов.
4. **Сложность применения систем ИИ в динамичной внешней среде.** В этом случае, основная проблема заключается в том, что при работе в динамичной среде необходимо часто переобучать нейронную сеть для получения более точных выходных значений. Но при этом стоит учитывать тот факт, что процесс обучения может занимать довольно продолжительное время. Отсюда следует, что применение нейронных сетей не всегда целесообразно.
5. **Большие трудовые, ресурсные и временные затраты.** Разработка нейронных сетей включает в себя целый комплекс различных затрат. Например, сюда можно отнести затраты на формирование структуры, поддержание работоспособности и обучение сети. В зависимости от сложности решаемых задач определяется величина расходов. Соответственно, чем проще поставленная задача, тем меньше требуется ресурсов на ее исполнение, и наоборот.

Таким образом, в этом параграфе было раскрыто понятие искусственных нейронных сетей и было показано, какие преимущества и недостатки получают системы ИИ при их использовании.

### 1.2 Классификация нейронных сетей

Ранее было отмечено, что искусственная нейронная сеть представляет собой модель, которая состоит из большого числа простых элементов, называемых нейронами, и связей между ними. Нейроны выполняют различные функции в сети, и в зависимости от этого выделяют следующие их виды [31]:

1. **Входные нейроны**, которые вводят в ИНС поступающую из внешней среды информацию.
2. **Выходные нейроны**, задача которых заключается в формировании итоговых результатов.
3. **Скрытые нейроны**, в которых осуществляются основные расчеты сети.

В общем случае, вид нейрона можно определить по его расположению в сети. Так, если нейрон имеет связи только на выходе, то он называется входным, и наоборот – выходным, если у него есть связи лишь на входе. Скрытым называется нейрон, у которого имеются связи как на входе, так и на выходе.

В процессе функционирования ИНС осуществляется преобразование входного вектора данных в выходной. При этом способы обработки информации могут отличаться друг от друга, поскольку элементы нейронной сети могут по-разному генерировать выходной сигнал. Более того, они образуют различные структуры, от которых зависит скорость обучения, эффективность расчетов сигналов, точность выходных значений и прочее. Исходя из этого, выделяют различные классификации и архитектуры нейронной сети [32].

В зависимости от топологии межнейронных связей искусственные нейронные сети делятся на:

1. **Полносвязные сети,** в которых нейроны напрямую связаны друг с другом. В процессе функционирования сети каждый нейрон получает, генерирует и передает сигналы всем остальным нейронам. Выходным значением такой сети является вектор, полностью или частично состоящий из сигналов элементов ИНС.
2. **Многослойные сети,** в которых происходит выделение слоев – групп, состоящих из нейронов с одинаковыми входными и выходными связями. Каждый слой может содержать любое количество нейронов, и оно не зависит от числа нейронов других слоев. Исходя из того, как нейроны одного слоя связаны с нейронами другого слоя, аналогичным образом можно выделить входные, выходные и скрытые слои. Входной слой вводит в систему данные из внешних источников, затем скрытые слои осуществляют обработку этих данных, и в конце выходной слой формирует итоговые результаты.
3. **Слабосвязные сети,** в которых осуществляется взаимодействие только между соседними элементами сети. В ходе работы нейроны производят обмен сигналами с соседними нейронами. В результате этой работы система формирует вектор выходных сигналов сети.

В зависимости от типа структур нейронов искусственные нейронные сети можно поделить на:

1. **Гомогенные сети**, которые состоят из однотипных нейронов. Другими словами, элементы ИНС используют одинаковый механизм расчета сигналов.
2. **Гетерогенные сети**, которые состоят из нейронов разного типа. В этом случае, каждый элемент ИНС использует свой собственный алгоритм для расчета выходных сигналов.

Нейронные сети можно классифицировать по виду оперируемых сигналов. Исходя из этого, выделяют:

1. **Бинарные сети**, в которых сигналы принимают только два значения: логический ноль или логическая единица. Предположим, что нейрон использует единичную ступенчатую функцию для расчета выходного значения. Если нейрон находится в заторможенном состоянии, то есть входных сигналов не хватает для совершения “скачка”, тогда выходным значением будет являться логический ноль. Если же произойдет возбуждение нейрона, то есть будет совершен “скачок”, тогда на выходе будет логическая единица.
2. **Аналоговые сети,** в которых сигналы представлены в виде действительных чисел. В этом случае, нейрон применяет сигмоидальную функцию для определения выходного значения.

В следующей классификации происходит деление нейронных сетей в зависимости от того, как меняется состояние системы с течением времени. Сюда можно отнести:

1. **Синхронные сети**, основная идея которых заключается в том, что сначала система запоминает состояние каждого нейрона по отдельности, и при переходе в следующий момент времени она синхронно меняет их состояния на новые.
2. **Асинхронные сети,** которые аналогичным образом учитывают и запоминают состояния нейронов, но с каждой новой итерацией происходит изменение только одного нейрона или целой группы нейронов.

Таким образом, были рассмотрены классификации, связанные с основными элементами нейронной сети. Так, ИНС могут отличаться друг от друга способами организации структуры сети, методами обновления данных, видами нейронов и генерируемых ими сигналов. В дополнение к этому, было определены виды нейронов и выполняемые ими функции.

### 1.3 Поэтапное представление процесса разработки

Перейдем к обсуждению того, как осуществляется процесс разработки искусственной нейронной сети. Он может быть представлен в виде последовательной реализации следующих этапов [38]:

1. Сбор, подготовка и нормализация данных;
2. Выбор топологии сети;
3. Подбор характеристик сети и параметров обучения;
4. Обучение сети;
5. Проверка адекватности;
6. Регулировка параметров.

Разберем подробно каждый перечисленный пункт.

Первоначальный этап занимает ключевую позицию в процессе разработки сети, поскольку именно здесь происходит выбор исходных данных, от которых будет зависеть дальнейшая работоспособность сети. Существуют критерии, удовлетворение которых приведет к правильному выбору начальных данных. Например, к ним относят критерий непротиворечивости и репрезентативности данных. Первый из них используется для того, чтобы данные, входящие в одну выборку, были согласованы друг другом. В противном случае, когда критерий не выполняется, может произойти снижение эффективности работы сети. Второй критерий необходим для того, чтобы исходные данные наилучшим образом соответствовали характеристикам своей предметной области. Из этого следует, что при наличии репрезентативности нейронная сеть будет лучше решать поставленную ей задачу.

Предположим, что выбранные исходные данные удовлетворяют вышеназванным критериям. Теперь необходимо осуществить преобразования, которые позволят нейронным сетям использовать их. Для этого данные записываются в виде вектора, значения которого будут поступать на входы нейронной сети. Но здесь может возникнуть проблема, связанная c тем, что сеть будет неправильно оценивать входные данные. Это происходит из-за того, что они могут отличаться по размерности и, как следствие, по-разному влиять на функционирование системы ИИ в целом. Для устранения этой проблемы применяется нормировка данных, которая приводит все значения к единому масштабу.

На втором этапе осуществляется выбор топологии сети. Здесь важно отметить, что от структуры зависит скорость обучения сети, качество генерируемых сигналов, точность выходных значений и так далее. При этом стоит понимать, что для решения той или иной задачи лучше всего подойдет какая-то одна определенная топология сети, а ее неправильный выбор может привести к снижению эффективности работы ИНС. Исходя из этого, можно сделать вывод о том, что топология сети является важным элементом разработки и выбор следует делать, прежде всего, исходя из постановки задачи, имеющихся данных и ресурсов.

Третий этап заключается в определении характеристик сети и подборе параметров обучения. Так, после выбора топологии сети необходимо определить механизм расчета сигналов, задать количество нейронов / слоев и установить связи между ними. Следует понимать, что слишком большое число нейронов (слоев, связей) может привести к снижению производительности сети, а их малое количество может существенно понизить качество выдаваемых результатов.

Также стоит обратить внимание на подбор параметров обучения, поскольку от них зависит, как быстро нейронная сеть сможет выдавать правильные результаты [39]. Неправильный выбор параметров может привести к тому, что сеть окажется переобученной или недообученной. В первом случае сеть “запоминает” данные обучающей выборки и из-за этого не может правильно оценивать реальные данные. Во втором случае сеть имеет недостаточно опыта, и ею результаты оказываются неточными. Таким образом, из всего вышесказанного следует, что от подбора параметров зависит работоспособность сети, и поэтому необходимо найти их оптимальное соотношение.

На четвертом этапе происходит обучение искусственной нейронной сети [45]. Этот процесс занимает ключевую позицию в разработке, и именно от него зависит точность выходных значений. Обучение непосредственно связано с сигналами, которые возникают и передаются между нейронами.

Нейрон представляет собой элемент (рис.1.1), который по определенному алгоритму формирует новый сигнал. Этот алгоритм можно представить в виде последовательной реализации следующих действий:

1. Прием сигналов, поступающих от других элементов ИНС;
2. Объединение полученных сигналов;
3. Расчет значения выходного сигнала;
4. Передача нового сигнала следующим нейронам.

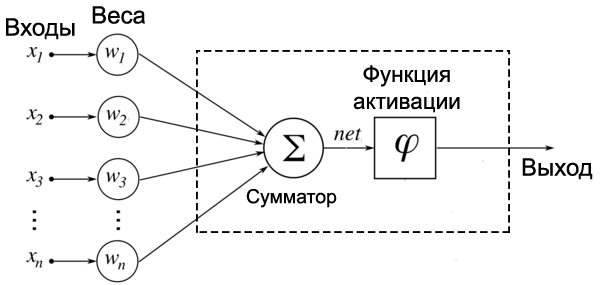


Рисунок 1.1 Схема строения нейрона сети

*Источник*: Основы ИНС // URL: <https://neuralnet.info/chapter/основы-инс/>

(Дата обращения: 05.10.2019)

Начнем с того, что нейроны получают сигналы от предшествующих элементов НС по линиям связи. На каждой линии устанавливается весовой коэффициент, который усиливает либо ослабевает поступающий в нейрон сигнал. Затем полученные сигналы объединяются в комбинированный сигнал, который представляет собой сумму взвешенных значений.

Далее осуществляется генерация нового сигнала. Для этого используется функция активации – математическая функция, аргументом которой является комбинированный сигнал. На практике чаще всего применяются пороговые и логистические функции. Особенность первого вида функций заключается в том, что пока комбинированный сигнал не превысит некоторое пороговое значение, выходной сигнал не будет сформирован. В этом случае сигнал может принимать только 2 значения: 0 или 1. Очевидно, что в этом случае получается единичная ступенчатая функция. Ко второму виду функций относятся сигмоидальные функции (сигмоиды). По своей сути, они представляют собой сглаженную версию ступенчатой функции. Здесь сигнал может принимать любые значения в промежутке от 0 до 1. После расчетов новый сигнал передается следующим нейронам по линиям связи.

Как было сказано ранее, нейроны определенным образом соединяются между собой, образуя структуру нейронной сети. В процессе функционирования сети осуществляется преобразование входных данных в выходные. Предположим, что нейронная сеть выдала результат, который отличается от ожидаемого. В результате будет получена некоторая ошибка вычислений. Для минимизации этой ошибки необходимо провести обучение нейронной сети.

Обучение представляет собой процесс, в ходе которого осуществляется подбор коэффициентов на линиях связи таким образом, чтобы результат, выдаваемый нейронной сетью, был максимально приближен к действительности. Отметим, что обучение сети происходит итеративно: сначала сеть проводит расчеты, затем определяется величина ошибки, и после этого осуществляется калибровка весов.

Добавим, что нейронные сети могут использовать разные алгоритмы обучения [50]. Так, например, выделяют обучение с учителем, которое предполагает, что нейронная сеть будет использовать размеченный набор данных при проведении расчетов. В противоположность этому ставят обучение без учителя, которое подразумевает то, что сеть будет использовать неразмеченный набор данных. Кроме того, выделяют обучение с подкреплением, представляющее собой нечто среднее между первыми двумя способами. В этом случае, набор включает в себя как размеченные, так и неразмеченные данные.

Предположим, что в распоряжении есть многослойная ИНС и некоторый размеченный набор данных. Также допустим, что в выходные значения сети не совпадают с ожидаемыми значениями, то есть существует ошибка вычислений. Тогда для минимизации этой ошибки можно применить метод обратного распространения ошибки [46]. Его суть заключается в том, что выходная ошибка распространяется обратно по нейронам сети пропорционально их вкладам, величина которых определяется весовыми коэффициентами соответствующих связей.

После распределения итоговой ошибки по нейронам сети необходимо обновить весовые коэффициенты линий связей таким образом, чтобы последующие расчеты были более точными. В этом случае довольно часто применяют метод градиентного спуска. С помощью него осуществляется минимизация функции ошибки, аргументами которой выступают весовые коэффициенты. Основная идея этого метода заключается в том, что на каждом шаге находится градиент функции ошибки, и на его основе делаются поправки весов линий сети. Для построения градиента необходимо найти частные производные, значения которых можно определить по формуле 1:

Где:

– весовой коэффициент между -м и -м нейроном двух смежных слоев;

– выходная ошибка *j*-го нейрона;

– выходной сигнал *i*-го нейрона;

E – величина ошибки сети.

*Источник*: Рашид Т. Создаем нейронную сеть. Стр. 116.

Каждое полученное значение представляет собой величину поправки, которую необходимо добавить к соответствующему весовому коэффициенту для уменьшения ошибки системы. Для этого используется формула 2:

Где:

– весовой коэффициент между -м и -м нейроном двух смежных слоев;

– коэффициент обучения;

E – величина ошибки сети.

*Источник*: Рашид Т. Создаем нейронную сеть. Стр. 119.

Можно заметить, что в приведенной выше формуле поправка записывается с отрицательным знаком. Это сделано для того, чтобы весовой коэффициент увеличивался при отрицательной производной и уменьшался при положительной. Помимо этого, к этой поправке добавляется коэффициент обучения, задача которого заключается в сглаживании величины изменений во избежание перескоков через минимум.

В конечном итоге, через некоторое число шагов ошибка сети станет минимальной, и нейронная сеть начнет выдавать достаточно точные результаты.

На пятом этапе осуществляется проверка адекватности нейронной сети. Она необходима для того, чтобы понять, справляется ли сеть с решением поставленной задачи. Данная проверка подразумевает тестирование на примерах, которые не использовались в процессе обучения. Если в результате будет выявлено, что нейронная сеть не удовлетворяет требованиям решаемой задачи, то возникает необходимость в регулировке изначальных параметров.

На шестом этапе происходит настройка параметров сети [25]. Так, например, можно регулировать число слоев, количество нейронов, скорость обучения и так далее. При изменении параметров сеть начнет работать по-другому. В первом случае, она будет функционировать ровно так, как и планировал разработчик. Другими словами, будут устранены недочеты, которые негативно воздействовали на работоспособность сети. Во втором случае, нейронная сеть станет работать только хуже, поскольку возникнут новые проблемы, которые придется также решать.

Таким образом, в этом параграфе была представлен поэтапный процесс разработки. В дополнение к этому, было показано, как функционирует искусственная нейронная сеть.

## Глава 2 Интеллектуальные системы в финансовой сфере экономики

### 2.1 Применение нейронных сетей на фондовом рынке

На текущий момент системы ИИ, построенные на базе искусственных нейронных сетей, находят свое применение во многих областях жизнедеятельности человека, в том числе и в финансовой сфере, частью которой является фондовый рынок. Их можно использовать для решения задач, например, связанных с прогнозированием цены актива и значений различных финансовых показателей, с оценкой возможного риска и так далее. Более того, работа с использованием таких систем выполняется значительно быстрее и эффективнее по сравнению с традиционными методами. Поэтому можно сказать, что участникам рынка предоставляется достаточно мощный инструмент, использование которого сопровождается повышением производительности труда.

Стоит отметить, что применение искусственных нейронных сетей на фондовом рынке по-разному воздействует на функционирование системы ИИ. С одной стороны, возникает целый ряд преимуществ, реализация которых положительно влияет на решение поставленных задач. С другой стороны, нейронные сети несут в себе недостатки, которые могут негативно повлиять на ход работы сети, на итоговый результат. Поэтому возникает необходимость в выделении преимуществ и недостатков, наибольшим образом влияющих на работу системы ИИ. Так, к положительным сторонам можно отнести:

1. **Возможность объединения технического и фундаментального анализа**. Это преимущество становится реализуемым благодаря тому, что нейронные сети могут использовать данные разного типа, поскольку они будут преобразованы в сигналы. Получается, что нейронная сеть может получать и обрабатывать информацию, поступающую в результате проведения вышеперечисленных видов анализа. В итоге это позволяет использовать больше данных для расчетов и, как следствие, сделать выходное значение более точным.
2. **Возможность одновременной обработки нескольких информационных потоков.** Нейронная сеть способна учитывать информацию, одновременно поступающую из разных источников. Поскольку для расчетов используется больший массив данных, то результирующее значение сети становится более достоверным.

К недостаткам, которые возникают при работе нейронных сетей на фондовом рынке, можно отнести следующее:

1. **Необходимость в постоянном переобучении сети**. Эта проблема связана с тем, что рынок подвержен влиянию большого числа как внешних, так и внутренних экономических факторов. Например, это может быть изменение соотношения спроса и предложения, поведение участников рынка, доступность информации и так далее. Отсюда следует, что рынок не может находиться в одном и том же состоянии, он постоянно изменяется под их воздействием. Исходя из этого, требуется периодически заниматься переобучением нейронной сети для того, чтобы качество и точность ее выходных значений оставались на прежнем уровне.
2. **Трудности оценки эмоционального состояния участников рынка.** Очевиден тот факт, что человек довольно часто принимает решения под воздействием своих внутренних убеждений и эмоций. Помимо этого, он может действовать исходя из информации, получаемой из новостных источников. В результате у него формируется определенный эмоциональный настрой, который влияет на общее состояние рынка. Одной из слабых сторон нейронных сетей является то, что они не способны в полной мере учитывать в своих расчетах эмоциональное состояние участников рыночных отношений. Хотя на практике существуют алгоритмы, в которых осуществляется анализ новостных данных, они не могут быть полностью обеспечены той информацией, которая была бы пригодна для оценки влияния вышеперечисленного фактора.

Тем не менее, если удастся разработать нейронную сеть, в которой будут максимально эффективно использоваться вышеназванные преимущества, а влияние недостатков будет снижено до минимума, то в результате получится достаточно мощный инструмент, при помощи которого можно будет выполнять самые разные задачи. Скорее всего, его разработка потребует большое количество ресурсов и времени. Но в конце будет получено средство, которое сможет дать его владельцу конкурентное преимущество перед другими участниками рынка и упростить выполняемую им работу.

Таким образом, в этом параграфе была приведена информация об использовании искусственных нейронных сетей на фондовом рынке. Так, была выявлена их область применения, указан круг решаемых задач, определены положительные и отрицательные стороны.

### 2.2 Выбор и анализ финансовых активов

Ранее было сказано, что системы ИИ на базе искусственных нейронных сетей представляют собой достаточно мощный класс инструментов, которые используются для решения различных практических задач, связанных с анализом и оценкой финансовых активов. Поэтому имеет смысл продемонстрировать работу данного инструмента на конкретном примере. Для начала необходимо выбрать финансовый актив и определить конкретную задачу, которую будет выполнять нейронная сеть.

Итак, далее будем рассматривать государственные облигации, эмитентом которых выступает Министерство финансов РФ. В качестве основной цели будет прогнозирование цены облигации на некоторый срок.

Государственная облигация – это долговая ценная бумага, выпускаемая правительством определенного государства с целью привлечения денежных средств, необходимых для покрытия дефицита бюджета, финансирования государственных проектов [28]. Этот вид облигаций считается одним из самых “надежных” финансовых инструментов, так как вероятность того, что государство не сможет рассчитаться по долгам, минимальна. При этом стоит понимать, что их доходность будет тоже минимальна, так как плата за риск практически отсутствует.

Выделим основные характеристики облигации. Так, сюда можно включить:

1. Номинальная стоимость – это стоимость, установленная эмитентом облигации;
2. Ставка купона – это ставка, по которой осуществляются платежи по облигации;
3. Рыночная цена – это цена, по которой торгуется облигация на рынке;
4. Дата выпуска / погашения;
5. Срок погашения – это промежуток времени между датой выпуска и датой погашения.

Из перечисленных характеристик выделим рыночную цену облигации, поскольку она потребуется для решения поставленной задачи.

Рыночная цена является одной из важнейших характеристик, которая используется при оценке облигации. Например, она нужна для расчета аналитических показателей, таких как: эффективная доходность, дюрация и модифицированная дюрация. Также с помощью нее определяют, является ли облигация переоцененной или недооцененной. Помимо этого, стоит отметить, что рыночная цена облигации зависит от спроса и предложения, предъявляемого на рынке. Следовательно, нельзя однозначно сказать, какая будет ее цена во время следующих торгов. Поэтому далее разработаем такую программу, которая будет предсказывать будущее значение цены с минимальной ошибкой прогноза.

Итак, возьмем государственную облигацию с характеристиками, представленными на рисунке ниже:

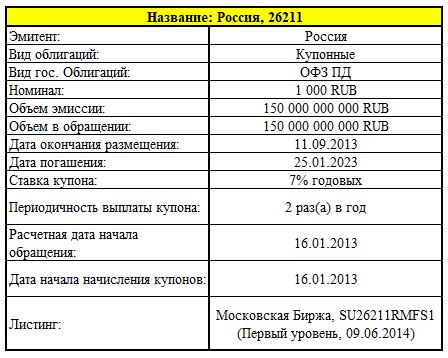


Рисунок 2.1 Информация по эмиссии облигации

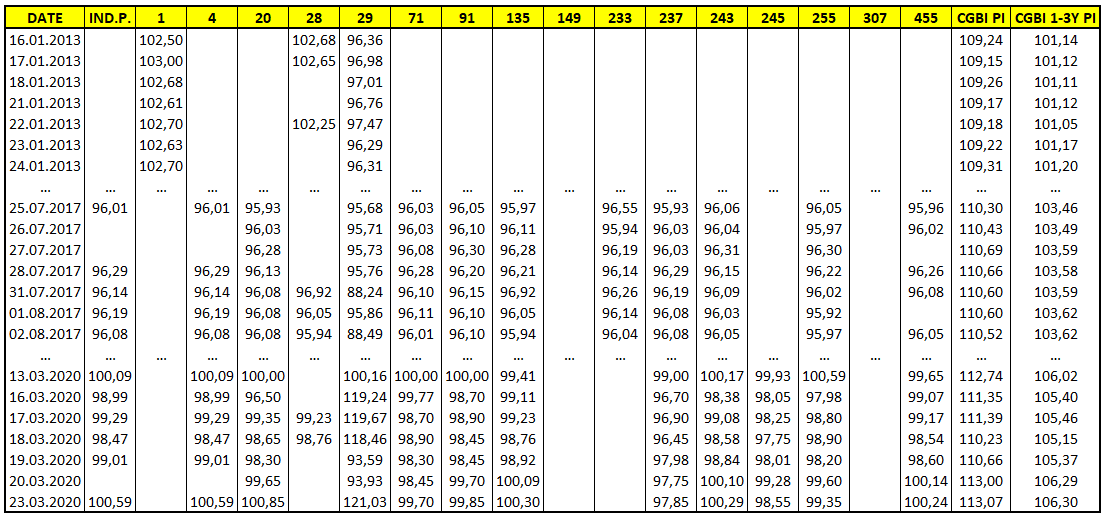
*Составлено по*: Облигации: Россия, 26211 (26211RMFS, RU000A0JTJL3, ОФЗ 26211) // URL: <http://cbonds.ru/emissions/issue/34179> (Дата обращения: 23.03.2020).

Прокомментируем полученную информацию. Сначала отметим, что рассматриваемая облигация относится к купонным облигациям, то есть по ней предусмотрены периодические процентные платежи. Ее номинальная стоимость составляет 1 000 рублей, купонная ставка равна 7% годовых, а выплаты купонов происходят каждые полгода. Также добавим, что эмитентом этой бумаги является правительство РФ, объем эмиссии равен 150 000 000 000 рублей, срок обращения составляет 10 лет.

В качестве поставщиков котировок по этой облигации выступают 15 биржевых торговых площадок. Например, сюда входит Московская биржа Т+, Франкфуртская ФБ, Берлинская биржа, Дюссельдорфская ФБ, Мюнхенская ФБ и так далее. Далее будем использовать все эти котировки для прогнозирования рыночной цены, поскольку, проанализировав их всех, нейронная сеть сможет дать более точный результат.

Помимо рыночных котировок будем также использовать индексы, которые включают в свои расчеты данную эмиссию. Исходя их этого, возьмем ценовые индексы рынка государственных облигаций России Cbonds-GBI RU PI и Cbonds-GBI RU 1-3Y PI. Первый индекс включает в себя государственные облигации со сроком до погашения более 360 дней. Второй рассчитывается для облигаций со сроком до погашения от 360 до 1080 дней.

Таким образом, нейронная сеть будет прогнозировать рыночную цену облигации на основе котировок биржевых торговых площадок и ценовых индексов. В результате получим таблицу, представленную на рисунке 2.2:



**Рисунок 2.2** Исходные данные

*Составлено по*: Облигации: Россия, 26211 (26211RMFS, RU000A0JTJL3, ОФЗ 26211) // URL: <http://cbonds.ru/emissions/issue/34179> (Дата обращения: 23.03.2020).

Во-первых, отметим, что цифры в строке с названиями столбцов представляют собой номера торговых площадок. Во-вторых, можно заметить, что в данных существуют пропуски, которые могут повлиять на работоспособность системы ИИ. Поэтому следует поступить следующим образом:

1. Определить момент, начиная с которого количество пропусков в данных станет минимальным;
2. Учитывая п.1, провести срез данных;
3. Использовать методы интерполяции для заполнения пропусков.

Помимо этого, исходный набор данных следует проверить на адекватность. Для этого необходимо найти, выделить и исключить искаженные данные, которые могут повлиять на конечный результат, сделать его некачественным и неточным. Существует достаточно много способов идентификации аномальных данных, одним из которых является применение методов интеллектуального анализа данных.

Теперь представим исходных набор данных в виде трехмерного графика, который показан на рисунке ниже, и проанализируем его.

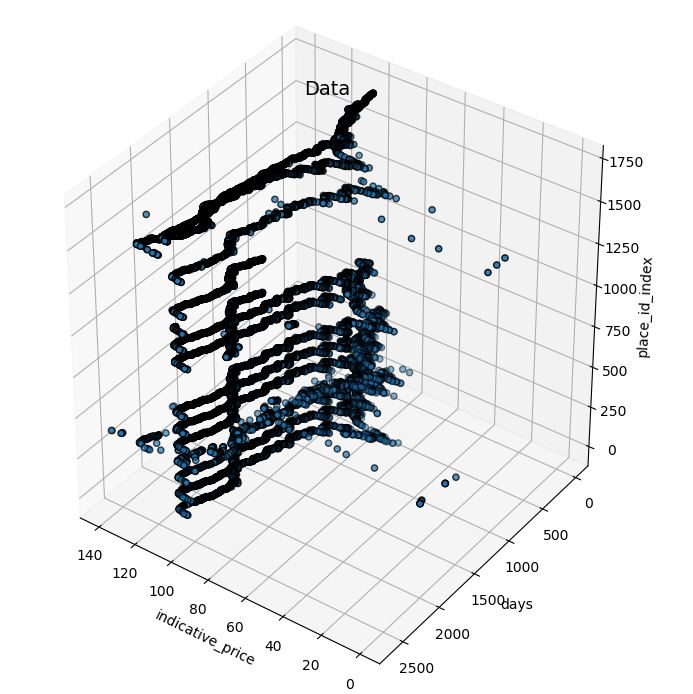


Рисунок 2.3 Графическое представление исходных данных

Оценим полученный результат. Как видно, на протяжении длительного периода времени рыночные цены оставались в некотором диапазоне. Однако можно заметить, что в определенные моменты времени цены некоторых биржевых площадок принимали слишком маленькие значения. Скорее всего, эти значения являются аномальными, поэтому их следует найти и удалить из исходного массива.

Таким образом, в этом параграфе был выбран финансовый актив, и были поставлены задачи, решением которых будет заниматься искусственная нейронная сеть. Так, были рассмотрены государственные облигации, выпускаемые правительством РФ. В частности была приведена их характеристика, выделены и проанализированы исходные данные.

## Глава 3 Интеллектуальная обработка данных

### 3.1 Общая информация об аномалиях

Аномалии (выбросы, отклонения) - это элементы из набора данных, которые явным образом выделяются на фоне остальных элементов. Они появляются по причине того, что в процессе заполнения массива данными возникают различного рода ошибки. Эти ошибки могут быть результатом неправильного расчета показателей, технических сбоев программных средств, неполадок при конвертации файла с данными. В конечном итоге, выделенный из общего массива набор данных может содержать в себе ошибки, от которых следует избавиться.

Очевидно, что аномалии отличаются друг от друга. Поэтому имеет смысл выделение некоторых характерных черт, позволяющих распределить их по различным группам. Так, обычно выделяют следующие виды аномалий:

1. Точечные аномалии
2. Контекстуальные аномалии
3. Коллективные аномалии

Разберем подробнее каждый приведенный случай.

**Точечные аномалии** являются самым распространенным видом аномалий. Они возникают в случае, когда отдельный элемент данных выделяется из общей совокупности аналогичных ему элементов. Стоит отметить, что этот вид аномалий относится к наиболее узнаваемым. Поэтому большинство методов, предназначенных для обнаружения искажений в данных, применяются для распознавания этого вида аномалий. Для примера рассмотрим рисунок 3.1, представленный ниже:

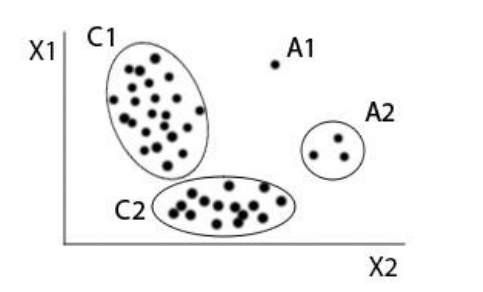


Рисунок 3.1 Точечные аномалии

*Примечания*: 1) C1, C2 – группы нормальных экземпляров;

2) A1, A2 – аномальные экземпляры.

*Источник*: Обзор методов обнаружения аномалий в потоках данных // URL:

<http://ceur-ws.org/Vol-1864/paper_33.pdf> (Дата обращения: 26.02.2020)

Проанализируем полученную ситуацию. На рисунке 3.1 выделены нормальные и аномальные экземпляры данных. Можно заметить, что нормальные данные образуют довольно “плотные” группы C1 и C2. Аномальные данные, в свою очередь, представлены в виде одного экземпляра A1 и группы A2, экземпляры которой расположены достаточно далеко друг от друга. Если сравнивать приведенные выше группы, то можно сказать, что экземпляры A1 и группы A2 являются аномальными по отношению к экземплярам группы C1 и C2.

**Контекстуальные аномалии** встречаются намного реже, чем точечные аномалии. Они наблюдаются тогда, когда экземпляр данных можно считать аномальным только при определенных условиях. Для обнаружения этого вида аномалий необходимо сначала выделить условные атрибуты, которые отражают определенную специфику данных. В качестве такого атрибута, например, можно рассматривать время во временных рядах, поскольку оно определяет особенности этого вида данных. Представим ситуацию, когда в наборе данных присутствуют контекстуальные аномалии. Для этого перейдем к рисунку 3.2:

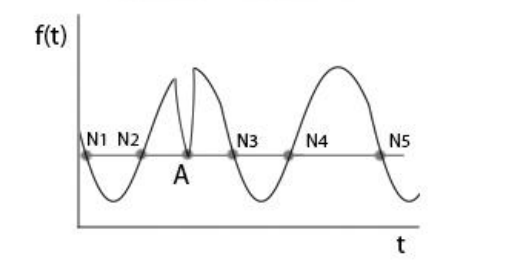


Рисунок 3.2 Контекстуальные аномалии

*Примечания*: 1) A – аномальный экземпляр;

2) N1, … , N5 – нормальные экземпляры.

*Источник*: Обзор методов обнаружения аномалий в потоках данных // URL:

<http://ceur-ws.org/Vol-1864/paper_33.pdf> (Дата обращения: 26.02.2020)

Аналогичным образом проанализируем ситуацию. На рисунке 3.2 представлен график временного ряда. Заметим, что в точке A явным образом наблюдается аномалия, в отличие от точек N1, …, N5, которые принимают аналогичное значение. В этом случае условным атрибутом является время, и поскольку точка A принимает ненормальное значение в определенный момент, то ее можно отнести к концептуальным аномалиям.

**Коллективные аномалии** появляются достаточно редко. Они возникают тогда, когда все элементы некоторой последовательной группы данных принимают аномальные значения, отличные от исходного набора данных. Отдельные экземпляры в терминах такой группы являются нормальными. Однако если их рассматривать по отношению ко всему набору данных, то эти экземпляры станут аномалиями. В качестве примера рассмотрим рисунок 3.3:

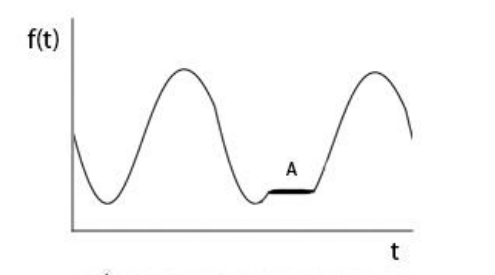


Рисунок 3.3 Коллективные аномалии

*Примечания*: 1) A – группа аномальных значений.

*Источник*: Обзор методов обнаружения аномалий в потоках данных // URL:

<http://ceur-ws.org/Vol-1864/paper_33.pdf> (Дата обращения: 26.02.2020)

Проанализируем полученный график временного ряда. Заметим, что на промежутке A временной ряд ведет себя неестественно, он принимает одинаковые значения в течение длительного времени. Скорее всего, данные, которые входят в данный промежуток, в общей совокупности представляют собой коллективную аномалию.

В дополнение ко всему вышесказанному отметим, что точечные и контекстуальные аномалии могут присутствовать в любом наборе данных, а коллективные аномалии появляются тогда, когда экземпляры данных связаны между собой.

Таким образом, в этом параграфе была представлена общая характеристика об аномалиях. Сначала было раскрыто само понятие, затем были выявлены основные причины возникновения аномалий, после чего были приведены их виды.

### 3.2 Методы обнаружения аномалий

Интеллектуальный анализ данных содержит в себе большое количество методов, которые можно использовать для обнаружения аномалий в массиве данных. К основным обычно относят следующие методы:

1. Методы классификации;
2. Методы кластеризации;
3. Статистический анализ;
4. Алгоритм ближайшего соседа;
5. Спектральные методы;
6. Гибридные методы.

Рассмотрим подробнее каждый указанный метод.

**Методы, основанные на классификации,** достаточно часто применяются для распознавания аномалий. В качестве основного предположения считается то, что набор данных, описывающий некоторую систему, может быть поделен на различные классы. Каждый класс представляет собой промаркированную выборку, сформированную по определенному, заранее заданному критерию.

Поиск аномалий с использованием методов классификации, как правило, осуществляется за несколько этапов. Сначала классификатор обучается на массиве промаркированных данных. Затем он обрабатывает немаркированные данные, распределяет их по классам. В случае, когда классификатор не может определить принадлежность экземпляра данных к одному из классов, то он помечает его как аномалию.

Перечислим наиболее известные классификаторы, которые используются для выявления аномалий в данных. К ним относятся: искусственные нейронные сети, Байесовские сети, метод опорных векторов и метод на основе правил. Подробное описание методов на основе классификации приведено в [21].

**Методы кластеризации** образуют достаточно популярный класс методов, применяемых для обнаружения отклонений в данных. Принцип их работы заключается в том, из исходного набора данных выделяются похожие друг на друга экземпляры, которые затем группируются в кластеры. Степень похожести данных определяется исходя из заранее выбранной метрики. Считается, что нормальные данные попадают в кластеры, а аномалии – нет.

Также необходимо отметить следующий факт. Возможна ситуация, когда аномальные элементы данных группируются в отдельный кластер. Примером этого может быть наличие коллективных аномалий. Для выявления таких случаев вводится дополнительное предположение: нормальные данные формируют достаточно большие и плотные кластеры, а аномальные данные, наоборот, образуют маленькие и разрозненные кластеры.

К методам кластеризации, например, можно отнести алгоритмы k-means и c-means. Подробное описание приведенных алгоритмов можно найти в [7].

**Статистический анализ** обычно применяется, когда нет точной информации о виде аномалий. Суть этого метода заключается в том, что на основе исходных данных строится модель, которая затем сопоставляется с реальной системой. Если в определенные моменты между ними наблюдаются заметные расхождения, то, как правило, делается вывод о наличии отклонений.

В качестве методов статистического анализа, например, могут рассматриваться методы, основанные на использовании нормального распределения, регрессионных моделей [22]. Также сюда можно включить методы на основе гистограмм и функций ядра, подробное описание которых представлено в [13], [15].

**Алгоритм ближайшего соседа** имеет аналогию с методами кластерного анализа. Для реализации этого алгоритма сначала необходимо задать метрику, с помощью которой данные будут сравниваться между собой. Затем осуществляется выбор подхода, на основе которого будет определяться, является ли экземпляр данных аномальным или нет. На практике обычно выделяют два основных подхода:

1. Подход на основе оценки расстояния до ближайшего соседа;
2. Подход на основе оценки плотности окрестности экземпляра данных.

В первом случае для каждого элемента рассчитывается расстояние до его ближайшего соседа. Затем полученные расстояния оцениваются, и элементы, которые наиболее отдалены от своего соседа, принимаются за аномалии.

Во втором случае оценивается окрестность каждого элемента данных. Если элемент находится в окрестности с высокой плотностью, то он считается нормальным. В противном случае, если элемент расположен в окрестности с низкой плотностью, то он отмечается как аномалия.

Можно заметить, что применение второго подхода практически идентично тому, что было описано в методах кластеризации. Но здесь есть одно важное отличие. Метод ближайших соседей является контролируемым алгоритмом, то есть ему необходимо предоставить набор помеченных данных, на основе которых он принимать решения. Методы кластеризации относятся к неконтролируемым алгоритмам, им не нужен заранее заданный помеченный набор данных. Они самостоятельно оценивают ситуацию и делают выводы по поводу данных.

**Спектральные методы** формируют группу вспомогательных средств, которые используются в дополнении с другими методами обнаружения аномалий. Их суть заключается в предобработке исходных данных, которая позволяет повысить быстродействие основного алгоритма. Так, одним из способов предобработки может быть уменьшение размерности исходных данных путем комбинирования сильно коррелирующих друг с другом атрибутов.

В качестве примера можно взять метод главных компонент (PCA), который применяется для упрощения набора данных. Подробное описание данного алгоритма представлено в [23].

**Гибридные методы** представляют собой различные комбинации методов обнаружения аномалий. Их главная особенность заключается в том, что они объединяют в себе преимущества используемых методик. Также стоит отметить, что гибридные методы подразумевают как последовательное, так и параллельное применение совмещенных алгоритмов.

Приведем некоторые возможные комбинации методов. Например, можно совмещать методы кластеризации данных и методы, основанные на нормальном распределении [16]; метод опорных векторов и искусственные нейронные сети [6]; спектральные методы и любые другие методы.

Необходимо добавить, что все вышеперечисленные методики обнаружения аномальных данных в основном используются для точечных аномалий. Если возникает потребность в обнаружении других видов аномалий, то требуется осуществить некоторые дополнительные операции над данными. В случае с контекстуальными аномалиями данные должны быть преобразованы относительно указанного условного атрибута. При работе с коллективными аномалиями предполагается, что они представляют собой единичные объекты одинаковой длины. В конечном итоге, эти виды аномалий будут сведены к точечным, и можно будет применять рассмотренные методики.

Таким образом, в этом параграфе были рассмотрены методы, которые используются для обнаружения аномалий в массиве данных. Было дано подробное описание методик, были приведены конкретные примеры.

### 3.3 Алгоритм поиска аномалий

В этом параграфе рассматриваются методы, которые непосредственно будут использованы для поиска и устранения аномальных значений. Выделим следующую комбинацию алгоритмов: метод k-средних и метод главных компонент.

**Алгоритм k-средних,** как было выявлено ранее, относится к методам кластерного анализа, суть которых заключается в разделении исходного набора данных на группы, объединяющих в себе похожие друг на друга экземпляры. Действие данного алгоритма можно разбить на два этапа: формирование кластеров, поиск аномалий. Сначала происходит выделение таких кластеров, при которых будет достигнута минимальная среднеквадратическая ошибка разбиения. Затем определяется расстояние между каждой точкой и ее ближайшим центроидом. После этого берется некоторое количество самых больших расстояний, среди них находят минимальное. В конечном итоге расстояния, которые превышают полученное минимальное значение, определяются как аномалии.

Основным преимуществом можно считать простоту реализации алгоритма на программном уровне, поскольку во многих языках программирования существуют специальные библиотеки, предназначенные для кластерного анализа данных. Также здесь можно выделить интуитивную понятность алгоритма, которая обосновывается тем, что его работа может быть представлена в виде реализации последовательных действий [35]:

1. Случайным образом выбрать точек, которые будут являться начальными “центрами масс” кластеров;
2. Отнести каждый экземпляр данных к кластеру с ближайшим “центром масс”;
3. Пересчитать “центры масс” всех кластеров;
4. Проверить выполнение критерия остановки;
5. Если система не удовлетворяет вышеназванному критерию, возвращаемся к п.2.
6. Найти расстояние между каждой точкой и ее ближайшим центроидом;
7. Определить долю аномалий;
8. Установить пороговое значение для аномалий;
9. Выделить точки, расстояния которых больше указанного порога.

К условиям остановки работы алгоритма можно отнести следующее:

1. Достигнуто минимально возможное значение среднеквадратической ошибки;
2. Нет экземпляров данных, перемещенных из одного кластера в другой (п.2).

К главным недостаткам этого метода можно отнести необходимость в самостоятельном определении количества кластеров, от которого зависит качество группировки данных. Для решения этой проблемы был придуман “метод локтя” (elbow method), который показывает, какая доля дисперсии будет объяснена тем или иным количеством кластеров [3]. Также стоит отметить, что этот алгоритм довольно чувствителен к выбору начальных центров кластеров. В зависимости от этого время его схождения и величина погрешности может быть разной.

**Метод главных компонент (PCA)** относится к спектральным методам, задача которых заключается в предварительной обработке данных с целью ускорить быстродействие основного алгоритма. Ранее было отмечено, что этот метод применяется для уменьшения размерности данных путем выделения и объединения признаков, сильно коррелирующих между собой. Представим этот процесс в виде пошаговой реализации определенных действий:

1. Отцентрировать и стандартизировать исходные данные;
2. Построить ковариационную матрицу;
3. Найти собственные вектора и собственные значения этой матрицы;
4. Построить матрицу преобразования, элементами которой являются собственные векторы, расположенные в порядке убывания собственных значений;
5. Найти главные компоненты, которые представляют собой линейные комбинации исходных признаков.
6. Определить значимость каждой главной компоненты.
7. Уменьшить размерность данных, отбросив наименее значимые из них;
8. Восстановить исходный вид данных.

К основным достоинствам можно отнести то, что этот алгоритм позволяет уменьшить размерность исходных данных, потеряв при этом малую часть информации. Помимо этого, стоит отметить, что метод главных компонент может значительно повысить производительность исходного алгоритма, так как он уменьшает количество признаков путем объединения наиболее коррелированных из них. Наконец, добавим то, что здесь существуют специальные библиотеки, которые позволяют использовать этот метод без каких-либо усилий.

Среди недостатков можно выделить обязательную стандартизацию данных перед использованием метода PCA, поскольку, в противном случае, этот алгоритм не сможет найти главные компоненты. Также сюда можно отнести проблему, связанную с потерей информации. Она заключается в том, что алгоритм может выбросить часть необходимой информации, посчитав ее малозначимой.

Далее определим план действий, по которому будет осуществляться поиск аномалий. Сначала к исходному набору данных будет применен метод главных компонент. Затем с помощью “метода локтя” будет выбрано оптимальное количество кластеров. После этого будет использован алгоритм k-средних, который должен вывести номера аномальных экземпляров данных. Для наглядности каждый метод будет сопровождаться графическим отображением результатов.

Таким образом, в этом параграфе были рассмотрены методы, которые будут составлять основу будущей программы. В результате был выделен и проанализирован алгоритм k-средних / метод главных компонент. После этого был определен план действий, по которому будет осуществляться реализация вышеуказанных методов, используемых для обнаружения аномальных данных

### 3.4 Программа для обработки данных

Итак, на текущий момент имеются исходные данные и определен план действий, по которому должен осуществляться поиск аномалий. Исходя из этого, напишем программу на языке Python и затем проверим ее работоспособность.

Необходимо отметить, что в обработке будут задействованы только цены участников рынка. Индикативная цена и ценовые индексы рынка государственных облигаций России не будут здесь использоваться, поскольку по предварительной оценке этих данных было выявлено, что они не содержат аномальных значений.

В конечном счете, была получена программа, исходный код которой представлен в Приложении 1. Теперь необходимо протестировать и проанализировать результаты ее работы.

На первом этапе осуществляется предварительная обработка данных. Для этого в программе реализован метод главных компонент, который уменьшает размерность исходных данных ценой потери части информации. Поскольку от объема потерянной информации зависит точность итоговых результатов, было решено добавить гистограмму, столбцы которой показывают, какую долю информации содержит в себе каждая главная компонента. Учитывая вышесказанное, определим, сколько компонент следует оставить для исходных данных. Результаты представлены на рисунке 3.4:

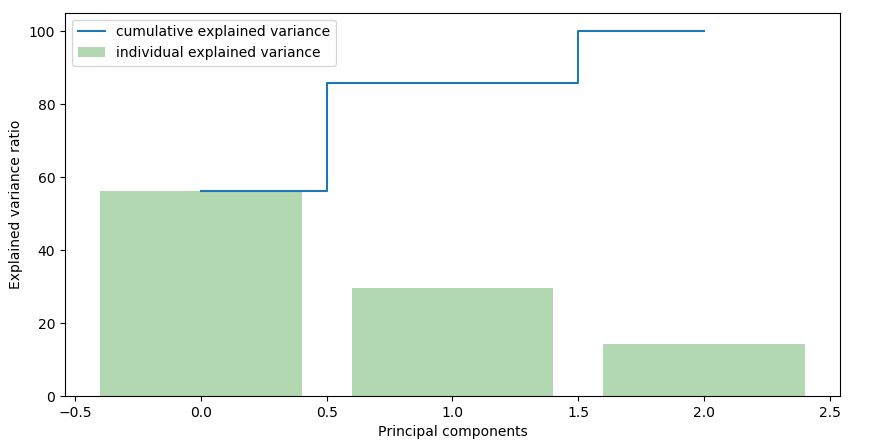


Рисунок 3.4 Метод главных компонент (PCA)

Заметим, что две главные компоненты суммарно описывают более 80% процентов информации. В большинстве случаев, этого хватает для расчетов, поэтому можно откинуть последнюю компоненту, и на основе первых двух восстановить данные.

На втором этапе необходимо выбрать оптимальное число кластеров для метода k-средних. Для решения этой задачи очень часто используют “метод локтя”. Суть данного метода заключается в построении функции, описывающей изменения вариации данных внутри кластеров при изменении количества кластеров. Считается, что оптимальному числу кластеров соответствуют моменты наибольшего изменения функции. Графическое представление этого метода показано на рисунке 3.5:

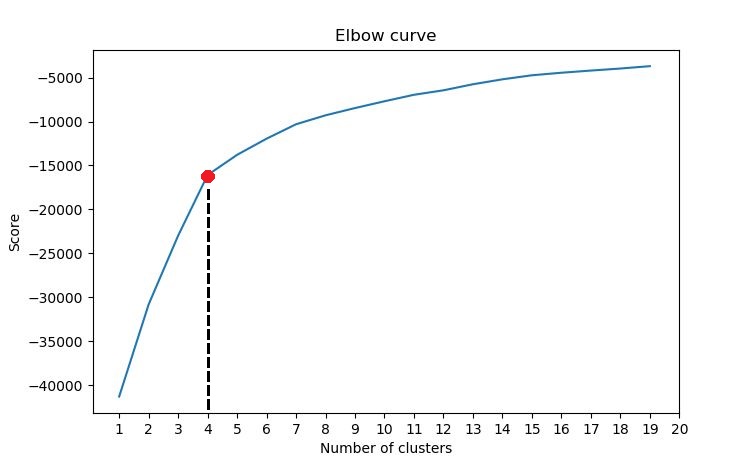


Рисунок 3.5 Метод локтя (elbow method)

На рисунке видно, что график начинает выравниваться с того момента, когда число кластеров стало равным четырем. Добавление большего количества кластеров не окажет существенного влияния на итоговый результат, поскольку изменение вариации, в этом случае, будет несущественным. Исходя из этого, можно сделать вывод, что в методе k-средних следует использовать четыре кластера.

На третьем этапе осуществляется кластеризация исходных данных и поиск аномальных значений. Для выполнения этих действий программа использует алгоритм k-средних. Первый промежуточный результат ее работы заключается в распределении данных по кластерам и построении соответствующего графика, пример которого можно увидеть на рисунке 3.6:

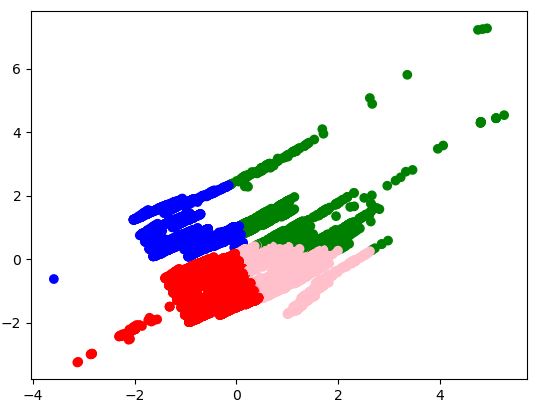


Рисунок 3.6 график кластеров (метод k-средних)

Отметим, что программа смогла отнести каждый экземпляр данных к одному из четырех кластеров. В результате были сформированы довольно большие и плотные группы. Однако здесь есть точки, которые достаточно далеко расположены от центов масс своих кластеров. Скорее всего, они представляют собой точечные аномалии.

Далее программа осуществляет поиск аномальных значений. Сначала она потребует указать долю аномалий в данных. Как правило, здесь возникают некоторые трудности, связанные с ее точным определением. Если взять слишком маленькое значение, то некоторые аномалии, вероятно, будут пропущены. В противном случае, если выбрать большое значение, то будут затронуты нормальные данные. Затем программа находит расстояния между каждой точкой и ее ближайшим центроидом. После чего она определяет точное количество выбросов в данных и на их основе устанавливает пороговое значение. Для этого она берет наибольшие расстояния в количестве, равном числу выбросов, и среди них находит минимальное. Затем она сравнивает все расстояния с этим минимальным значением. Если будет найдено расстояние, которое превышает пороговое значение, то программа отметит его как аномалию.

После выполнения основного алгоритма следует визуальное представление итоговых результатов. Для этого используется трехмерный график, на котором изображены нормальные и аномальные данные. Его можно увидеть на рисунке 3.7:

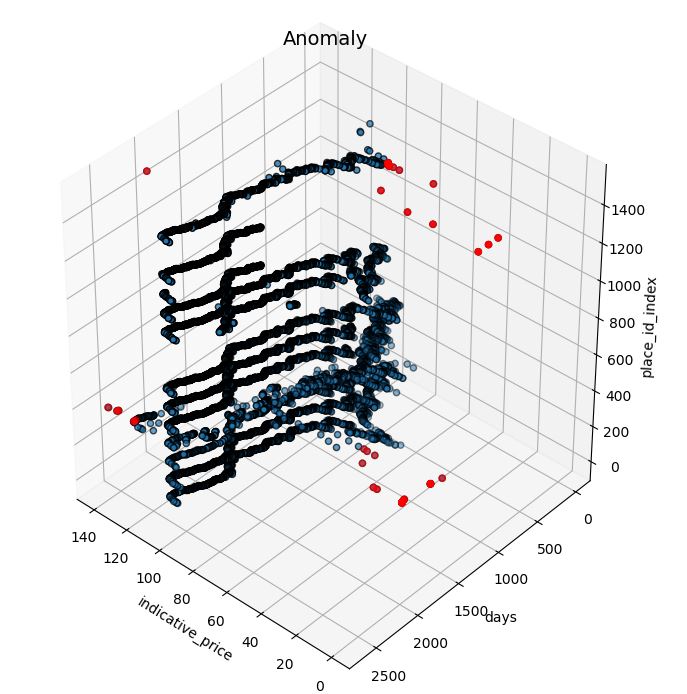


Рисунок 3.7 Визуальное представление аномальных значений

*Примечания*: 1) синие точки – нормальные данные;

2) красные точки – аномалии.

Отметим, что программа смогла определить, какие экземпляры данных являются нормальными, а какие – нет. В целом, она показала довольно неплохой результат. Основные аномалии были выявлены. Однако можно заметить, что программа все же пропустила некоторые искаженные значения и в некоторых пометила нормальные экземпляры данных. Исходя из этого, можно сделать вывод о том, что алгоритм нуждается в доработке.

Таким образом, в этом параграфе были рассмотрены и проанализированы основные этапы работы программы, цель которой заключается в поиске аномалий в данных.

## Глава 4 Разработка и практическое применение нейросетевых систем

### Разработка интеллектуальных систем

Перейдем к разработке интеллектуальных систем, которые на основе приведенной ранее информации и при помощи искусственных нейронных сетей будут прогнозировать цену государственной облигации.

Поступим следующим образом: сначала разработаем две программы, в основе которых будет многослойная нейронная сеть и рекуррентная нейронная сеть, и затем протестируем их. Далее по результатам тестирования определим, какая программа лучше справилась с поставленной задачей. Наконец, проведем оценку с помощью статистических методов анализа временных рядов, сравним результаты и сделаем выводы.

### 4.1.1 Многослойная нейронная сеть

Итак, начнем с программы, в алгоритме которой используется многослойная нейронная сеть [37]. Отметим, что разработка будет осуществляться в среде PyCharm на языке программирования Python.

Перед началом работы необходимо выделить библиотеки – наборы модулей, классов, функций и других объектов, которые используются для решения конкретного круга задач. Так, для написания программного кода потребуются следующие библиотеки, а именно:

1. Tensorflow – библиотека, предназначенная для проектирования и создания моделей глубокого обучения.
2. Numpy – библиотека, содержащая в себе средства для работы с многомерными массивами.
3. Pandas – библиотека, предназначенная для обработки и анализа данных.
4. Sklearn – библиотека, в которой реализованы алгоритмы для машинного обучения.
5. Matplotlib – библиотека, которая используется для визуализации данных.

После подключения нужных библиотек можно перейти к основному коду программы. Для начала необходимо осуществить импорт данных и затем преобразовать их таким образом, чтобы они могли быть использованы нейронной сетью. Ниже на рисунке 4.1 представлена часть программного кода, отвечающая за вышеперечисленные операции.

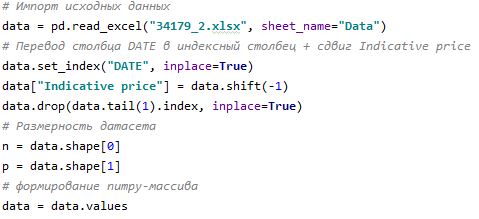


Рисунок 4.1 Импорт и подготовка данных

Сначала осуществляется импорт исходного набора данных из таблицы Excel. В результате получается таблица, называемая датафреймом. Затем происходит переход значений столбца DATE в индексный столбец и сдвиг индикативной цены на один шаг назад. Далее определяется размерность датафрейма, и на его основе формируется многомерный массив.

Итак, теперь необходимо разделить исходный набор данных на данные для обучения и тестирования нейронной сети. Поделим их следующим образом: данные для обучения составят 90 процентов от всего их объема, а тестовые данные, соответственно, будут включать в себя оставшиеся 10 процентов. Для этого используется следующая часть программного кода, представленная ниже:

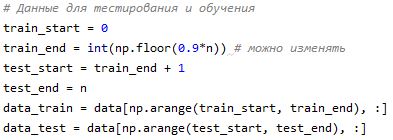


Рисунок 4.2 Разделение исходных данных

Сначала происходит выделение границ обучающей выборки. После чего определяются границы для тестовой выборки. Затем с учетом полученных границ осуществляется разделение исходного многомерного массива на две части. В результате формируются два новых массива. Первый содержит данные для обучения, а второй – для тестирования.

На следующем этапе проводится масштабирование данных. Причина, по которой требуется сделать это, заключается в том, что далее нейроны сети будут использовать сигмоидальные функции, которые, как правило, определены на интервале [0, 1]. Исходя из этого, ниже представлена часть программного кода, отвечающая за масштабирование:



Рисунок 4.3 Масштабирование данных

Приведение данных к единому масштабу здесь осуществляется при помощи функции MinMaxScaler из библиотеки sklearn. Сначала программа находит минимумы / максимумы переменных в наборе тренировочных данных и запоминает их. Затем, используя полученные статистики, она преобразует массив с тренировочными и тестовыми данными. После этого программа выделяет зависимые и независимые переменные, и на этой основе распределяет данные в каждом массиве.

Перейдем к описанию основных компонентов нейронной сети. Начнем с выделения плейсхолдеров – элементов сети, отвечающих за хранение исходной информации. Для работы потребуется два таких элемента: первый плейсхолдер будет содержать входные данные (рыночные цены торговых площадок и индексы государственных облигаций в момент времени t), а второй плейсхолдер будет включать выходные данные (индикативную цену в момент времени t+1). В программе они реализованы следующим образом:

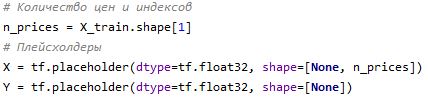


Рисунок 4.4 Плейсхолдеры нейронной сети

Входной плейсхолдер X представляет собой двумерный массив, а выходной плейсхолдер Y – одномерный массив. Можно заметить, что каждый плейсхолдер содержит аргумент None. Он означает, что на текущий момент неизвестно, какое количество элементов данных будет одновременно поступать в нейронную сеть в процессе обучения. Поэтому далее в программе будет определена переменная, через которую можно будет контролировать число элементов.

Помимо плейсхолдеров, которые содержат исходные данные, в нейронных сетях выделяют еще один важный элемент – переменные. Каждая переменная несет в себе информацию о значениях весовых коэффициентов между слоями нейронной сети. Также необходимо добавить, что во время обучения сети внутреннее состояние переменных может изменяться. Теперь выясним, какая часть программного кода отвечает за переменные. Для этого рассмотрим рисунок, представленный ниже:

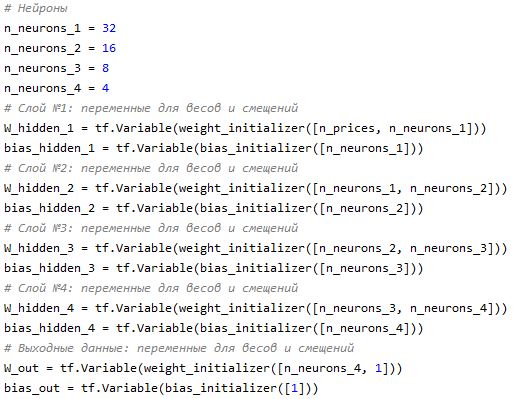


Рисунок 4.5 Переменные нейронной сети

Сначала требуется установить параметры, которые непосредственно связаны с переменными нейронной сети, а именно: количество слоев, число нейронов в каждом слое. По приведенному выше коду можно сказать, что сеть, помимо входного и выходного слоя, включает в себя 4 скрытых слоя. Отметим, что каждый скрытый слой содержит в два раза меньше нейронов, чем предыдущий слой. Такое соотношение нейронов обеспечивает сжатие информации, поступающей от предыдущего слоя. После определения параметров необходимо задать переменные. Так, для каждого слоя выделяются переменные для скрытых весов и смещений. Первые используются для хранения информации о весовых коэффициентах и представляют собой двумерные массивы. Вторые содержат значения, на величину которых смещаются функции активации, и представляют собой одномерные массивы.

Итак, на данный момент в программе заданы входные / выходные плейсхолдеры и выделены переменные сети. Теперь необходимо объединить вышеперечисленные компоненты в одну систему. Для этого рассмотрим рисунок ниже, на котором приведена схема разрабатываемой многослойной нейронной сети.

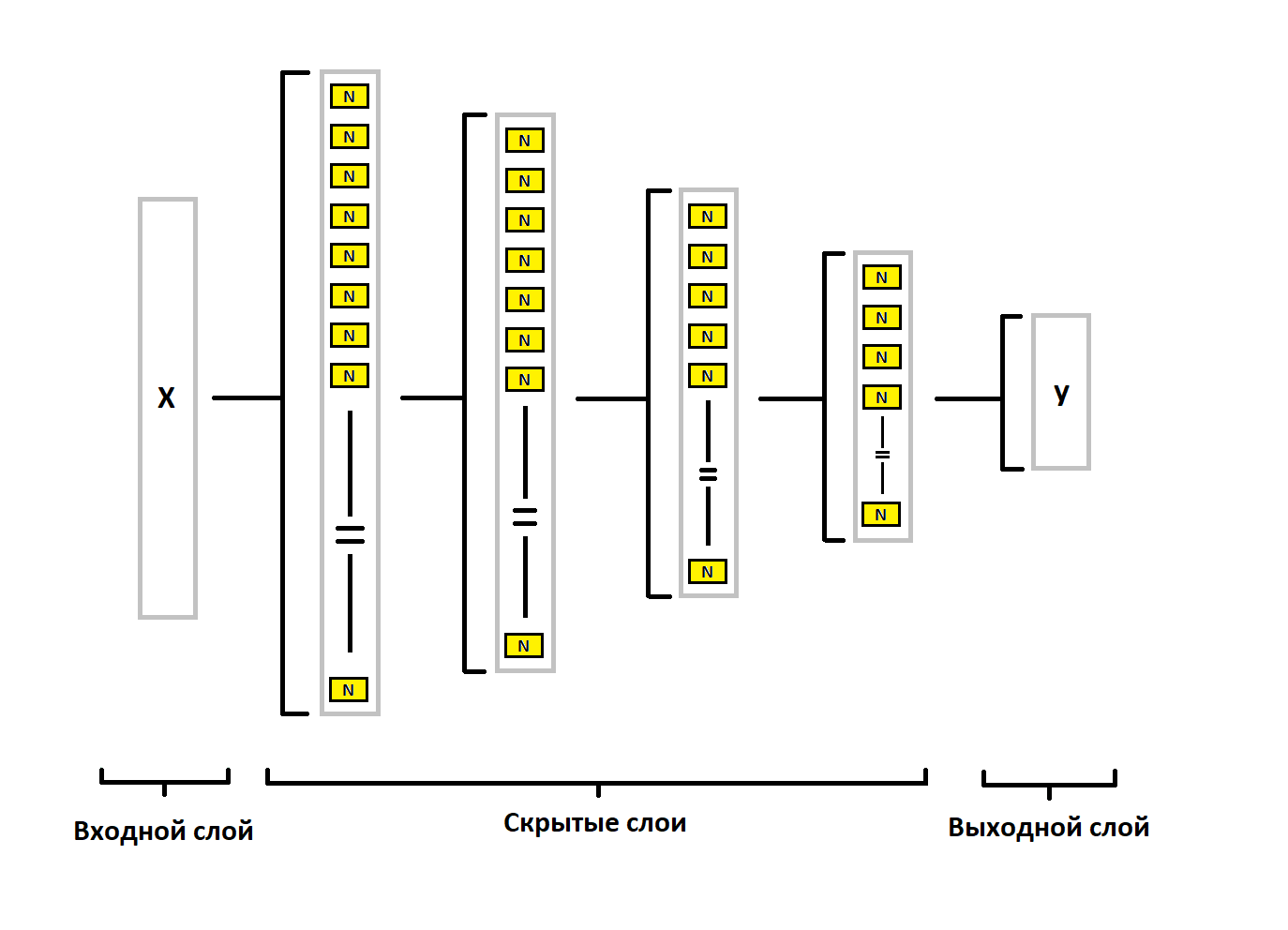


Рисунок 4.6 Структура нейронной сети

На схеме показано, что нейронная сеть состоит из трех частей. Сначала идет входной слой, затем четыре скрытых слоя и в конце выходной слой. Сигналы при заданной архитектуре сети переходят строго в одном направлении, от входного слоя к выходному.

Помимо архитектуры сети, определим, какую функцию активации будут использовать нейроны сети для генерации сигналов. Так, наиболее распространенной функцией активации является выпрямленная линейная единица (rectified linear unit, ReLU). Ее популярность связана с тем, что она менее требовательна к вычислительным ресурсам, в отличие от ступенчатых, сигмоидальных и других видов функций, и представляет собой достаточно хороший аппроксиматор [47]. Исходя из этого, далее будем использовать эту функцию.

Теперь покажем, как определяется структура нейронной сети на программном уровне. Для этого рассмотрим и проанализируем программный код, представленный на рисунке 4.7 ниже:

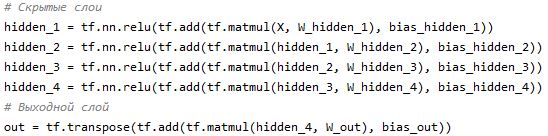


Рисунок 4.7 Слои нейронной сети

Отметим, что слои нейронной сети связаны друг с другом посредством матричных операций над плейсхолдерами и переменными. В результате таких операций на каждом слое формируется сигнальная матрица, которая затем используется в расчетах следующего слоя. Например, возьмем первый слой сети. Его сигнальная матрица получается в результате произведения входного плейсхолдера на матрицу весовых коэффициентов, добавления матрицы смещений и применения к полученному результату функции активации. Теперь рассмотрим второй слой сети. Здесь все происходит аналогичным образом: сначала осуществляется умножение матрицы сигналов первого слоя на матрицу весовых коэффициентов, добавляется смещение, применяется функция активации и так далее. Можно заметить, что функция активации не используется на выходном слое. Вместо этого необходимо транспонировать полученные результаты для дальнейших расчетов.

Таким образом, была определена архитектура нейронной сети. На следующем этапе требуется выделить компоненты, которые будут отвечать за обучение сети. К ним относится функция стоимости, оптимизатор и инициализатор. Рассмотрим их поочередно в заданной последовательности.

Функция стоимости сети является компонентой, с помощью которой происходит оценка качества прогнозной модели. Обычно в качестве функции стоимости используется функция средней квадратичной ошибки (mean squared error, MSE), которая вычисляет средний квадрат отклонения между реальными значениями и прогнозами. Реализация этой функции в программном коде выглядит следующим образом:



Рисунок 4.8 Функция стоимости сети

Следующей компонентой является оптимизатор сети. Он выполняет расчеты, связанные с обратным распространением ошибки, и с помощью него находятся оптимальные весовые коэффициенты линий связи. В качестве целевой функции оптимизатор использует функцию стоимости, которую требуется минимизировать. На текущий момент одним из наиболее распространенных оптимизационных алгоритмов является “Adaptive moment estimation”. Преимуществом этого алгоритма можно назвать то, что он выдает наилучший результат за достаточно малое число итераций. Его суть заключается в том, что он накапливает импульс, который ускоряет процесс сведения к оптимальному результату, и обновляет весовые коэффициенты, учитывая схожесть признаков (слабое обновление весов для типичных признаков). В программном коде оптимизатор имеет следующий вид:



Рисунок 4.9 Оптимизатор сети

Последней компонентой из вышеприведенного списка является инициализатор. С помощью него осуществляется инициализация весовых коэффициентов перед началом обучения нейронной сети. Поскольку в основе обучающего алгоритма заложены оптимизационные методы, то выбор первоначальных значений весовых коэффициентов оказывает существенное влияние на скорость поиска оптимального решения. В библиотеке Tensorflow реализовано достаточно много способов инициализации. Так, к ним относится инициализатор, который использует равномерное распределение для генерации чисел, и инициализатор, который задает нули в качестве первоначальных значений. Первый из них будет применен к переменным нейронной сети, а второй будет использоваться для смещений. Ниже представлен программный код, отвечающий за инициализацию сети:

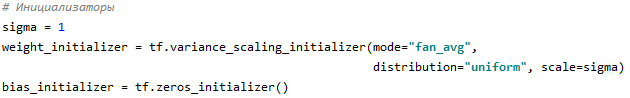


Рисунок 4.10 Инициализаторы сети

Таким образом, в программе была выделена функция стоимости, оптимизатор и инициализатор. Теперь перейдем к обучению нейронной сети. На практике довольно часто выделяют подход, основанный на том, что тренировочный набор данных должен быть поделен на части меньшего размера, которые затем поочередно загружаются в сеть для обучения. Этот подход получил название Mini-Batch [1]. Основным его преимуществом является возможность эффективной обработки больших объемов данных за счет деления их на сеты, называемые мини-партиями. Ключевым недостатком подхода считается более долгий процесс обучения, поскольку сеть должна обновить веса с учетом всех мини-партий за одну итерацию.

Обучение сети методом Mini-Batch происходит следующим образом. Сначала данные из тренировочного набора перемешиваются между собой. Затем полученный набор делится на мини-партии одинакового размера, которые затем поочередно загружаются в нейронную сеть. Так, входные данные мини-партии проходят через сеть до выходного слоя. После этого сеть сравнивает полученные результаты с реальными значениями и определяет величину ошибки. Затем осуществляется обратное распространение ошибки и корректировка весовых коэффициентов. Данная процедура повторяется до тех пор, пока все мини-партии не будут обработаны нейронной сетью. Полный цикл такой обработки называется эпохой.

Перейдем к программной реализации метода Mini-Batch. Для этого рассмотрим код, представленный на рисунке 4.11 ниже:

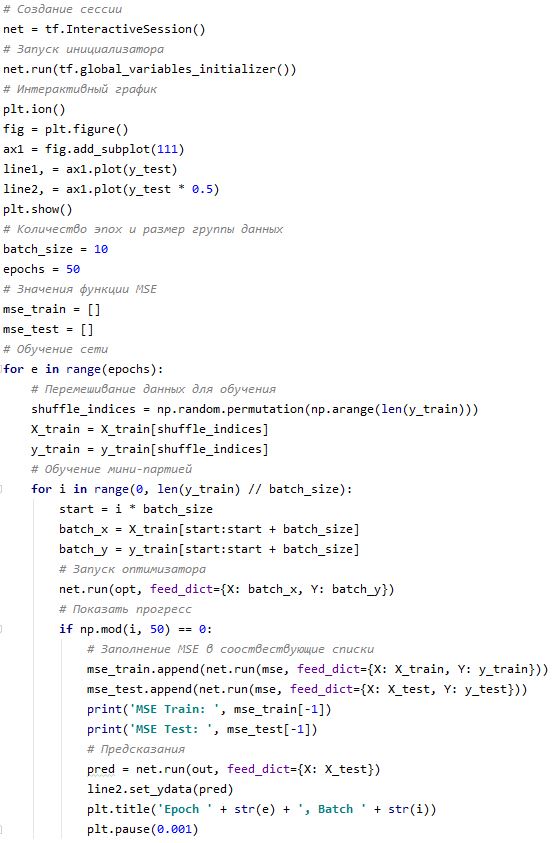


Рисунок 4.11 Обучение сети

Перед началом обучения сети необходимо определить число эпох и размер одной мини-партии. В программе установлены следующие значения этих параметров: количество эпох – 50, размер мини-партии – 10. Помимо этого в программном коде есть блок, отвечающий за визуализацию процесса обучения. Результатом его работы является график, на котором изображены реальные и прогнозные значения. Этот график будет обновляться после прохождения каждой эпохи. Теперь перейдем к обучению сети. Оно начинается с создания сессии и запуска инициализатора. После этого осуществляется реализация метода Mini-Batch. Сначала тренировочные данные перемешиваются и делятся на мини-партии. Затем нейронная сеть последовательно проводит через себя каждую мини-партию и с помощью оптимизатора обновляет весовые коэффициенты. По завершению обучающего цикла происходит вывод на график полученных результатов. Затем запускается следующий цикл и так далее.

После обучения сети необходимо привести прогнозные и тестовые значения к нормальному виду. Затем на их основе требуется рассчитать показатели, которые используются для оценки точности прогнозирования. После этого полученные результаты выгружаются в таблицу Excel. Ниже представлен код, выполняющий все вышесказанное:

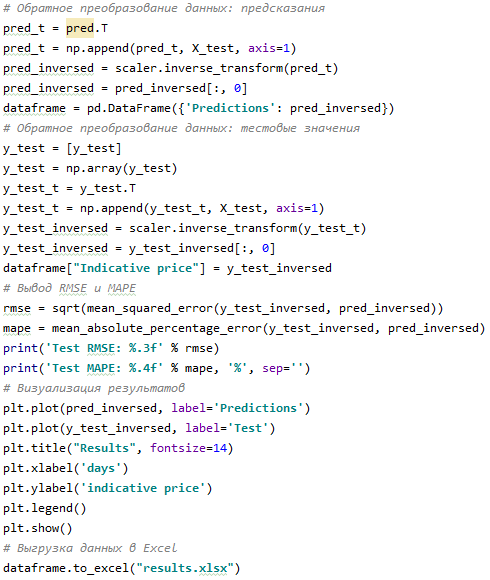


Рисунок 4.12 Вывод результатов

Сначала программа осуществляет демасштабирование прогнозных и тестовых значений, после чего добавляет их в один датафрейм. Затем она вычисляет показатель MRSE – квадратный корень из среднеквадратической ошибки, MAPE – среднюю абсолютную процентную ошибку. Оба показателя имеют схожую интерпретацию: они определяют степень расхождения между фактическими и прогнозными значениями. Разница заключается лишь в том, что MRSE – это абсолютный показатель, а MAPE – это относительный показатель. Чем меньше их значения, тем лучше будут прогнозы сети. Далее программа визуализирует полученные результаты и экспортирует их в файл Excel.

Таким образом, был полностью рассмотрен и проанализирован программный код многослойной нейронной сети. Теперь запустим программу и оценим полученные результаты. Они представлены на рисунке 4.13 ниже:

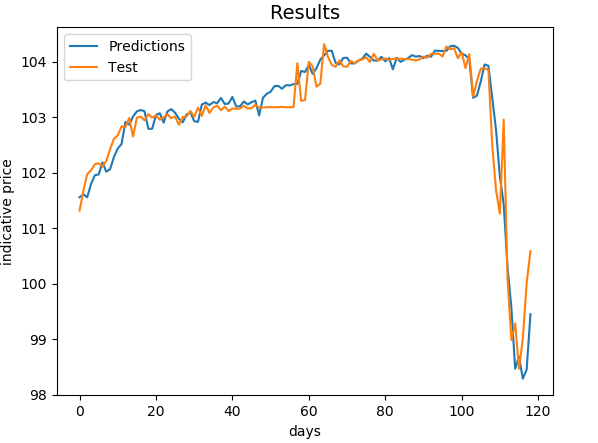


Рисунок 4.13 Результаты работы сети

Отметим, что прогнозные и тестовые значения многослойной нейронной сети практически совпадают друг с другом, хотя местами наблюдаются расхождения. Статистические показатели имеют следующие значения: MRSE – 0.344, MAPE – 0.175%. В целом, вышел неплохой результат. Можно считать, что нейронная сеть справилась с поставленной ранее задачей. Ее можно использовать для прогнозирования цены государственной облигации.

Теперь перейдем к разработке рекуррентной нейронной сети.

### 4.1.2 Рекуррентная нейронная сеть

Далее разработаем программу, в алгоритме которой используется рекуррентная нейронная сеть – разновидность нейронной сети, имеющей внутренние циклы [38]. Этот вид сетей часто применяют для оценки временных рядов, поскольку при помощи циклических связей сеть сохраняет информацию, полученную в ходе обработки прошлых элементов данных, и использует ее для текущих расчетов.

Среди рекуррентных нейронных сетей можно выделить сети с долгой краткосрочной памятью (long short-term memory, LSTM), которые достаточно хорошо решают задачи прогнозирования временных рядов [17]. Ключевой особенностью LSTM-сетей является то, что здесь каждый нейрон сохраняет информацию о своем состоянии от момента начала обработки данных до момента завершения и учитывает ее при генерации новых сигналов.

В результате получается, что когда сеть осуществляет расчеты, она учитывает информацию за предыдущий момент времени и за весь период в целом.

Перейдем к написанию программного кода. Сначала выделим библиотеки, компоненты которых будут задействованы при создании сети с долгой краткосрочной памятью. Для дальнейшей работы потребуется:

1. Keras – библиотека, предназначенная для проектирования и создания моделей глубокого обучения. В отличие от Tensorflow, она содержит готовые нейросетевые конструкции.
2. Numpy – библиотека, содержащая в себе средства для работы с многомерными массивами.
3. Pandas – библиотека, предназначенная для обработки и анализа данных.
4. Sklearn – библиотека, в которой реализованы алгоритмы машинного обучения.
5. Matplotlib – библиотека, которая используется для визуализации данных.
6. Math – библиотека, содержащая инструменты для работы с числами.

После определения библиотек можно перейти к написанию основного кода программы. Сначала требуется провести импорт и преобразование исходных данных. После этого необходимо разделить полученный набор данных на две части: первая будет включать в себя данные для обучения, а вторая – для тестирования. Затем осуществляется масштабирование тренировочных / тестовых данных и распределение их по соответствующим массивам. За все вышеперечисленные операции отвечает программный код, представленный на рисунке ниже:

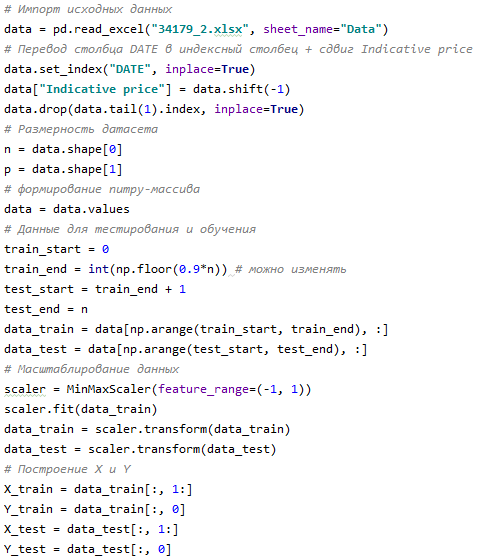


Рисунок 4.14 Преобразование исходных данных

Итак, сначала программа осуществляет выгрузку исходного набора данных из таблицы Excel и представляет его в виде датафрейма. После этого она проводит замену индексного столбца на столбец DATE и сдвигает индикативную цену на один шаг назад. Далее программа определяет размерность датафрейма и затем преобразовывает его в многомерный массив.

На следующем шаге происходит разделение массива с исходными данными на обучающий и тестовый сет. Отметим, что в первый сет входит 90 процентов данных, а остальное, соответственно, попадает во второй сет.

Далее осуществляется масштабирование обучающих и тестовых данных. Для этого используется функция MinMaxScaler. При реализации этой функции программа сначала определяет минимумы / максимумы переменных в наборе тренировочных данных и запоминает их. Затем она использует полученные статистики для преобразования обучающего и тестового сета. После этого программа разделяет данные внутри каждого приведенного сета на зависимые и независимые.

Помимо вышеприведенных преобразований для работы LSTM-сети необходимо добавить временной шаг к входным данным. Для этого в программе есть код, отвечающий за исполнение этой операции. Он представлен на рисунке 4.15 ниже:



Рисунок 4.15 Изменение массива входных данных

Отметим, что добавление шага приведет к изменению структуры массива входных данных. Первоначальная его структура имеет вид [samples, features], но после выполнения этого программного кода она изменяется на [samples, time steps, features].

После осуществления всех необходимых преобразований над данными перейдем к разработке LSTM-сети. Для начала требуется определить ее структуру. В общем виде ее можно представить следующим образом:

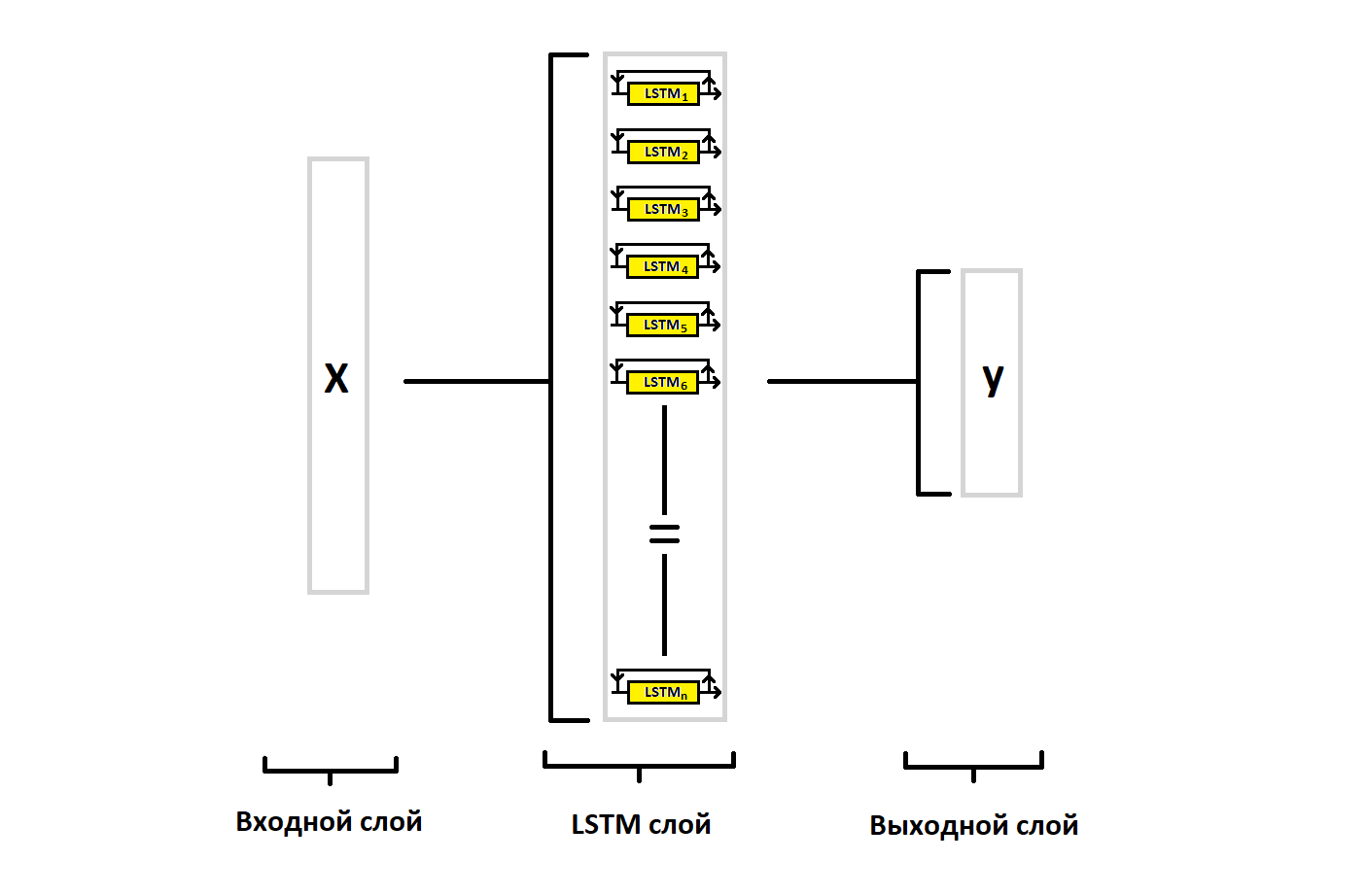


Рисунок 4.16 Структура LSTM-сети

По рисунку 4.16 видно, что LSTM-сеть состоит из трех частей. Сначала идет входной слой, затем один скрытый слой и в конце выходной слой. Сигналы при заданной структуре сети сначала передаются от входного слоя к скрытому слою. После этого нейроны LSTM-сети генерируют новые сигналы, передают их назад через обратные линии связи и вперед к выходному слою.

Теперь представим структуру нейронной сети на программном уровне. Для этого выделим слои, зададим функцию стоимости / оптимизатор и добавим код, отвечающий за обучение сети. Все вышеназванные компоненты программы приведены на рисунке 4.17:

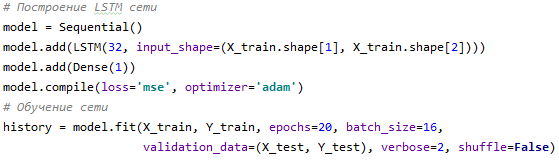


Рисунок 4.17 Построение LSTM-сети

Для начала необходимо определить способ организации слоев нейронной сети. Так, в программе используется метод Sequential, который позволяет задавать слои последовательно друг за другом. Первым слоем LSTM-сети является скрытый слой, затем идет выходной слой. Входной слой не имеет отдельной строчки программного кода, поскольку он задается в качестве параметра следующего слоя. Отметим, что скрытый слой содержит 32 нейрона, а выходной слой состоит из одного нейрона.

Далее требуется выделить компоненты, которые отвечают за обучение сети. По приведенному выше коду можно сказать, что потери сети будут оцениваться с помощью функции средней квадратичной ошибки (mean squared error, MSE), а оптимизация весовых коэффициентов будет осуществляться через алгоритм “Adaptive moment estimation”. Заметим, что в программном коде нет инициализаторов переменных и смещений сети. Это связано с тем, что по умолчанию здесь берется инициализатор, который использует равномерное распределение для генерации первоначальных весовых коэффициентов переменных, и инциализатор, который присваивает нулевые веса смещениям.

После добавления вышеперечисленных компонентов осуществляется обучение нейронной сети. Для этого используется метод MiniBatch со следующими параметрами: число эпох – 20, размер мини-партии – 16. Помимо этого, здесь установлен запрет на перемешивание обучающих данных. Это необходимо для того, чтобы сеть могла обучаться на последовательностях с исходной временной структурой. В результате постоянного использования одних и тех же мини-партий с каждой эпохой можно добиться того, что весовые коэффициенты сети практически перестанут изменяться.

Следующая часть программного кода отвечает за построение графика, отображающего зависимость потери сети от числа пройденных эпох. Помимо этого, здесь также представлена функция, через которую осуществляется прогнозирование сети. Все вышеназванные компоненты программы показаны на рисунке 4.18 ниже:

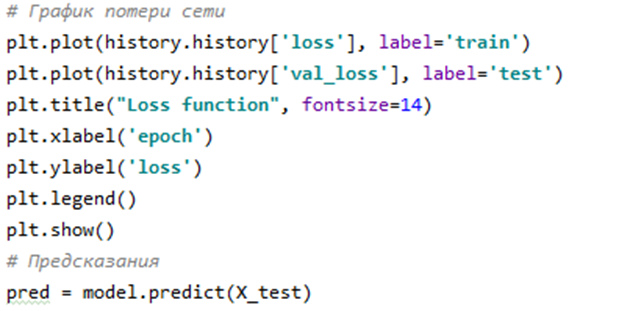


Рисунок 4.18 График потерь сети

Для начала отметим, что график потери сети используется для оценки качества обучения после прохождения полного цикла обработки. На нем, как правило, отображаются функции потерь, получаемые в ходе обучения и тестирования сети. Если на некоторой эпохе между функциями потерь наблюдается сильное расхождение, то это значит, что сеть на тот момент еще недостаточно обучилась. В противном случае, если функции потерь лежат достаточно близко друг к другу, то это говорит о том, что сеть смогла обучиться.

Помимо программного кода, отвечающего за визуализацию потерь, на рисунке отмечена строка, через которую осуществляется генерация прогнозов сети. Результатом выполнения этой строки является массив, который состоит из прогнозных значений, полученных после обработки тестового сета.

На следующем этапе необходимо привести прогнозные и тестовые значения к исходному виду. Для этого требуется выполнить следующие обратные преобразования, а именно:

1. Изменить структуру тестового и прогнозного массива на [samples, features];
2. Провести демасштабирование данных;
3. Объединить полученные значения в один датафрейм.

После обратного преобразования данных требуется вычислить квадратный корень из среднеквадратической ошибки (RMSE) и определить величину средней абсолютной процентной ошибки (MAPE). Далее необходимо визуализировать полученные результаты и выгрузить их в таблицу Excel. За все вышеперечисленные операции отвечает код программы, который показан на рисунке 4.19 ниже:



Рисунок 4.19 Обратное преобразование данных

Таким образом, была разработана программа, в которой используется сеть с долгой краткосрочной памятью. Теперь запустим ее и оценим полученные результаты. Они представлены на рисунке 4.20 ниже:

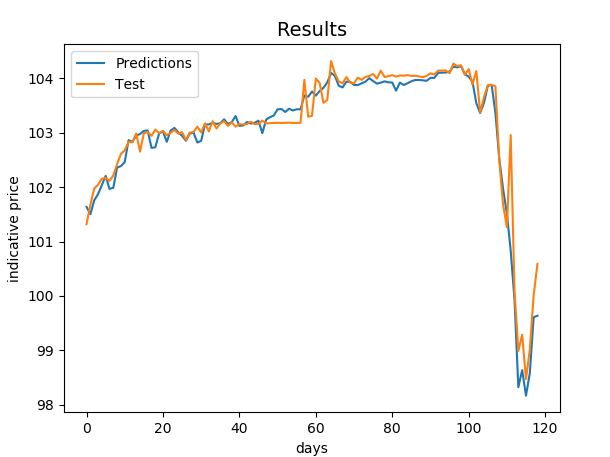


Рисунок 4.20 Результаты работы LSTM-сети

Отметим, что прогнозные и тестовые значения LSTM-сети расположены достаточно близко друг к другу, хотя в некоторых местах наблюдаются заметные расхождения. Статистические показатели принимают следующие значения: RMSE – 0.29, MAPE – 0.161%. Примерно такие же результаты были получены при использовании программы с многослойной нейронной сетью. В общем и целом, LSTM-сеть показала хороший результат. Ее можно применять для прогнозирования цены облигации.

Таким образом, в этом параграфе были представлены нейросетевые системы ИИ. Сначала была разработана и протестирована программа, в которой использовалась многослойная нейронная сеть. Затем был написан и проанализирован программный код сети с долгой краткосрочной памятью.

### Статистический анализ временных рядов

Итак, в предыдущем параграфе были представлены программы с нейросетевой структурой, которые на основе прошлой информации могли спрогнозировать цену государственной облигации. Теперь проверим, какие результаты можно получить при использовании стандартных методов анализа временных рядов.

Далее в работе будет рассмотрена интегрированная модель авторегрессии-скользящего среднего (ARIMA), с помощью которой будут получены прогнозные значения индикативной цены. Отметим, что, помимо нее, были построены и другие модели, предназначенные для оценки временных рядов (например: VAR, ARCH-модели). Однако после проверки моделей на адекватность оказалось, что только ARIMA смогла удовлетворить почти всем основным критериям (стационарность и обратимость, значимость коэффициентов модели, отсутствие автокорреляции в остатках).

Также отметим, что для работы с временными рядами используется программный пакет для эконометрического моделирования Gretl.

Теперь перейдем к построению модели ARIMA. На первом шаге требуется определить порядок интегрируемости временного ряда Indicative\_price. Для этого воспользуемся расширенным тестом Дики-Фуллера (Augmented Dickey-Fuller test, ADF), с помощью которого осуществляется проверка временных рядов на наличие единичных корней. Этот тест можно представить в виде последовательной реализации следующих действий. Сначала при помощи МНК оценивается модель, которую можно представить в виде формулы 3:

Где:

– значение временного ряда в момент времени t;

– оператор первой разности;

, – параметры модели;

– линейный тренд (может отсутствовать);

– белый шум.

*Источник*: Подкорытова О.А, Соколов М.В. Анализ временных рядов. Стр. 84.

После оценки приведенного выше уравнения осуществляется проверка следующей гипотезы:

1. (есть единичный корень, процесс не является стационарным);
2. (нет единичного корня, процесс стационарен).

Затем определяется справедливость нулевой гипотезы. Для этого берется t-статистика, соответствующая коэффициенту , и сравнивается с критическим значением, получаемым с помощью асимптотической формулы МакКиннона. Если нулевая гипотеза отвергается, то ряд будет считаться стационарным. В противном случае, требуется взять вторую разность, провести ADF-тест еще раз и так далее. В конечном итоге, порядок интегрируемости будет равен порядку оператора разности.

Итак, воспользуемся встроенным ADF-тестом и определим порядок интегрируемости исходного ряда. Ниже представлены результаты тестирования для текущего уровня переменной.

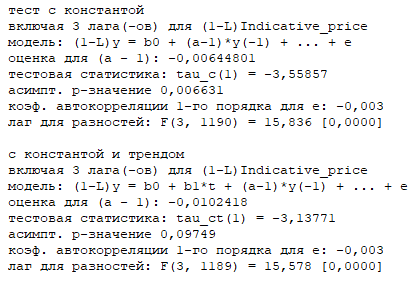


Рисунок 4.21 ADF-тест

Отметим, что на рисунке 4.21 выделены две спецификации ADF-теста: тест с константой, тест с константой и трендом. В первом случае предполагается, что ряд имеет ненулевое среднее. Во втором случае считается, что в нем есть детерминированный тренд. Можно заметить, что асимптотическое p-значение первого теста меньше 5% уровня значимости, а p-значение второго теста, наоборот, больше заданного уровня значимости. Поскольку один из тестов не отвергает нулевую гипотезу, то повторим ADF-тест для первых разностей. Далее будем использовать только спецификацию теста с константой, так как при дифференцировании исчезнет детерминированный тренд. Результаты тестирования представлены на рисунке 4.22 ниже:

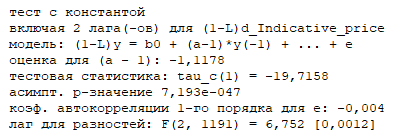


Рисунок 4.22 ADF-тест для первых разностей

В конечном счете, получилось, что асимптотическое p-значение для первых разностей меньше 5% уровня значимости. Следовательно, исходный ряд не стационарен, стационарна его первая разность. Другими словами, ряд Indicative\_price имеет первый порядок интегрируемости.

На следующем этапе требуется определить порядок процесса авторегрессии (AR) и порядок процесса скользящего среднего (MA). Для этого рассмотрим коррелограмму первой разности d\_Indicative\_price. Она показана на рисунке 4.23 ниже:

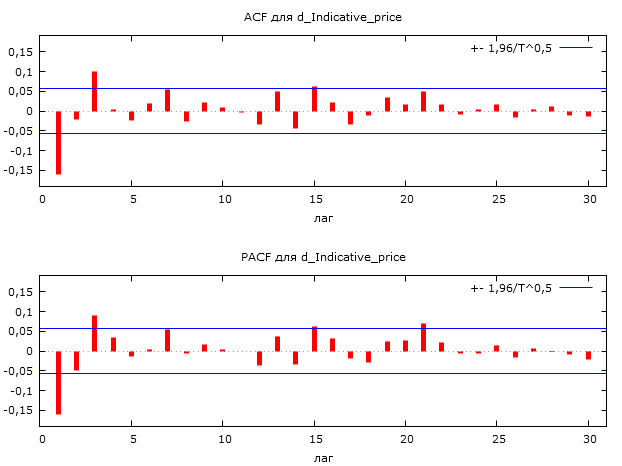


Рисунок 4.23 Коррелограмма d\_Indicative\_price

На этом рисунке представлены значения автокорреляционной функции (ACF) и частной автокорреляционной функции (PACF) в зависимости от лага. Помимо этого здесь можно увидеть границу, которая отделяет нулевые значения приведенных функций от ненулевых. Для определения порядка вышеназванных процессов требуется выделить лаги, при которых значения ACF и PACF оказываются ненулевыми. Также необходимо отметить, что номер лага ACF соответствует порядку процесса скользящего среднего, а номер лага PACF совпадает с порядком процесса авторегрессии. Исходя из этого, можно выделить следующие порядки: AR = [1, 3, 15, 21], MA = [1, 3, 5].

Далее определяется порядок модели ARIMA. Для этого требуется рассмотреть различные комбинации процессов AR / MA и среди них выбрать такую, при которой объединенная модель будет удовлетворять основным критериям адекватности. Если в результате перебора получится несколько хороших моделей, то выбор будет происходить с учетом информационных критериев. Таким образом, была выбрана модель ARIMA (1, 1, 3), общий вид которой представлен ниже (формула 4):

Где:

– значение временного ряда в момент времени t;

– оператор первой разности;

, , – параметры модели;

– белый шум.

*Источник:* Подкорытова О.А, Соколов М.В. Анализ временных рядов. Стр. 60.

Теперь необходимо оценить полученную модель при помощи эконометрического пакета Gretl. После этого необходимо показать, что она удовлетворяет следующим критериям адекватности, а именно:

1. Стационарность и обратимость;
2. Значимость коэффициентов модели;
3. Отсутствие автокорреляции в остатках;
4. Нормальность ошибок.

В конечном итоге, была построена и оценена модель ARIMA (1, 1, 3). Основные результаты расчетов представлены на рисунке 4.24 ниже:

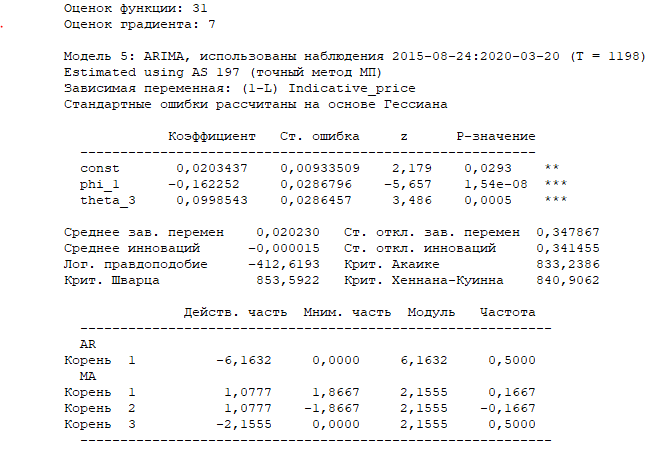


Рисунок 4.24 Модель ARIMA (1, 1, 3)

Итак, сначала проверим выполнение условий стационарности и обратимости. Для этого необходимо выяснить, лежат ли корни характеристических многочленов AR- и MA- компонент модели вне единичного круга [44]. По результатам оценки можно сказать, что модель является стационарной и обратимой, поскольку все корни по модулю больше единицы.

Далее требуется определить значимость коэффициентов модели. Обычно для этого используются асимптотические тесты, такие как: тест множителей Лагранжа, тест отношения правдоподобия, тест Вальда. Отметим, что при выполнении нулевой гипотезы любого вышеприведенного теста отвергается значимость коэффициентов модели. Теперь посмотрим, какие результаты представлены на рисунке 18. Можно заметить, что каждый коэффициент исходной модели является значимым, поскольку соответствующие им p-значения меньше 5% уровня значимости.

На следующем шаге требуется проверить отсутствие автокорреляции в остатках модели. Для этого воспользуемся тестом Люнга-Бокса о незначимости первых значений автокорреляционной функции. Здесь проверяется следующая гипотеза:

1. (автокорреляция отсутствует);
2. .

Для проверки достоверности нулевой гипотезы рассчитывается статистика, которая затем сравнивается с критическим значением распределения хи-квадрат. В случае, если полученное значение попадает в критическую область этого распределения, то делается вывод о наличии автокорреляции в остатках. В противном случае, автокорреляция будет отсутствовать, и остатки модели будут представлять собой белый шум.

Теперь посмотрим, какие результаты выдает статистический пакет Gretl. Они приведены на рисунке 4.25 ниже:

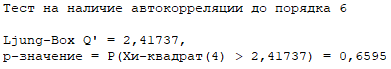


Рисунок 4.25 Тест Люнга-Бокса

По рисунку видно, что статистика теста Люнга-Бокса меньше значения хи-квадрат (4) с вероятностью 65,95%. Из этого следует, что нулевая гипотеза не отвергается, и остатки модели являются белым шумом.

Наконец, требуется проверить ошибки модели на нормальность. Для этого воспользуемся тестом Жака-Бера, который сопоставляет коэффициенты асимметрии (S) и эксцесса (K), рассчитанные по остаткам модели, с аналогичными коэффициентами из нормального распределения. Здесь проверяется следующая гипотеза:

1. (ошибки распределены по нормальному закону);
2. .

Отметим, что справедливость нулевой гипотезы определяется путем расчета статистики теста Жака-Бера и ее сравнения со значением распределения хи-квадрат (2). Если полученная статистика будет меньше критического значения, то делается вывод о нормальном распределении остатков модели. В противном случае, ошибки будут распределены ненормально.

Для определения нормальности ошибок в исходной модели воспользуемся встроенным тестом Жака-Бера. Результаты тестирования представлены на рисунке 4.26 ниже:

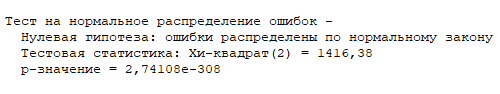


Рисунок 4.26 Тест на нормальное распределение

Заметим, что нулевая гипотеза данного теста отвергается, поскольку p-значение меньше 5% уровня значимости. Соответственно, можно сделать вывод о том, что ошибки модели ARIMA (1, 1, 3) распределены не по нормальному закону. Здесь необходимо отметить, что этот критерий не слишком сильно влияет на качество модели и точность получаемых прогнозов. Поэтому его выполнение не является столь существенным, в отличие от ранее приведенных критериев.

Таким образом, были проверены гипотезы, связанные с оценкой адекватности построенной модели. В результате было выявлено следующее:

1. Модель является стационарной и обратимой;
2. Коэффициенты модели – значимые;
3. Остатки представляют собой белый шум;
4. Ошибки не распределены нормально.

Теперь оценим качество приведенной модели. Для этого выделим наблюдаемые и расчетные значения, затем на их основе рассчитаем статистические показатели RMSE и MAPE. Начнем с выполнения первого пункта. Основные результаты можно увидеть на рисунке 4.27 ниже:

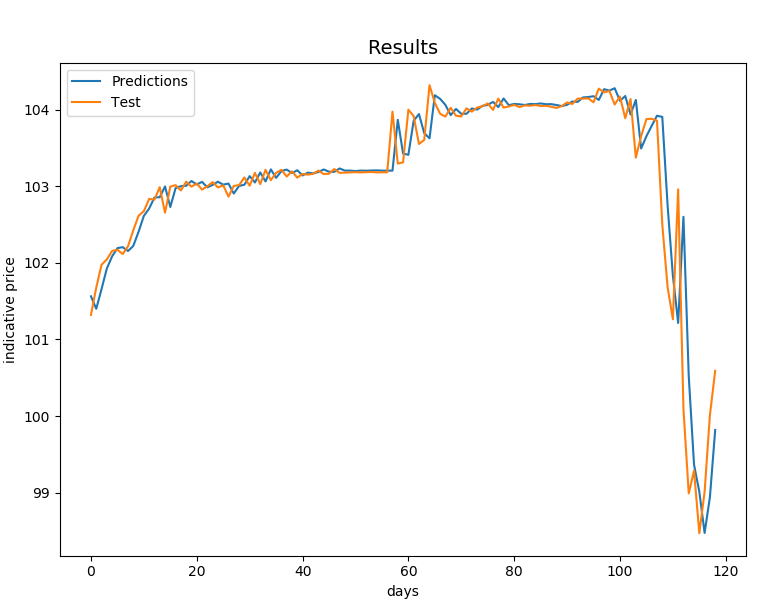


Рисунок 4.27 Значения модели ARIMA

Во-первых, необходимо отметить, что наблюдаемые и расчетные значения модели лежат достаточно близко друг к другу, между ними практически нет явных расхождений. Статистические показатели имеют следующие значения: RMSE – 0.422, MAPE – 0.196%. В целом, получились довольно неплохие результаты. Данную модель можно использовать для оценки индикативной цены облигации.

Перейдем к сравнительному анализу полученных результатов. Для этого воспользуемся статистическими показателями, которые были рассчитаны для многослойной нейронной сети, LSTM-сети и модели ARIMA. Они представлены на рисунке 4.28 ниже:

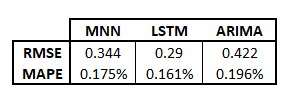


Рисунок 4.28 Статистические показатели

Итак, по результатам можно сказать, что сеть с долгой краткосрочной памятью лучше справилась с задачей прогнозирования, нежели чем многослойная нейронная сеть и интегрированная модель авторегрессии – скользящего среднего, поскольку ее показатели принимают наименьшие значения. Так, например, показатель RMSE для LSTM-сети на 15,59% меньше, чем RMSE для многослойной сети, и на 31,27% меньше, чем RMSE для модели ARIMA. Если рассматривать показатель MAPE для LSTM-сети, то он на 8% меньше, чем MAPE для многослойной сети, и на 17,85% меньше, чем MAPE для модели ARIMA. Из этого следует, что прогнозы LSTM-сети, в отличие от многослойной сети и модели ARIMA, в среднем отклоняются от индикативной цены на меньшую величину.

Таким образом, в этом параграфе были применены методы анализа временных рядов. Сначала была построена и оценена интегрированная модель авторегрессии-скользящего среднего. После чего был проведен сравнительный анализ между этой моделью и ранее разработанными нейросетевыми программами.

## Заключение

Подведем итоги. На основании проведенного научного исследования удалось решить следующие задачи:

1. Выделить ключевые особенности систем ИИ, построенных на базе ИНС;
2. Составить поэтапный план разработки нейронной сети;
3. Определить методы использования нейронных сетей на фондовом рынке;
4. Выбрать и проанализировать финансовые активы;
5. Выполнить предобработку исходных данных;
6. Разработать и протестировать нейросетевые программы;
7. Провести оценку методами статистического анализа данных;
8. Сравнить полученные результаты.

Начнем с того, что первым пунктом научно-исследовательской работы было определение и раскрытие общей информации о системах ИИ, в которых используются искусственные нейронные сети. Например, были выделены главные особенности таких систем, указаны их основные преимущества и недостатки.

Затем было отмечено, что каждая нейронная сеть по-своему уникальна. Она может принимать различные структуры и использовать разные способы обработки информации. Исходя из этого, были приведены классификации и архитектуры нейронной сети.

Далее был представлен и разобран процесс разработки искусственной нейронной сети. Было отмечено, что данный процесс может быть сведен к последовательной реализации следующих этапов:

1. Сбор, подготовка и нормализация данных;
2. Выбор топологии сети;
3. Подбор характеристик сети и параметров обучения;
4. Обучение сети;
5. Проверка адекватности;
6. Регулировка параметров.

Отметим, что каждый вышеназванный этап был подробно описан и проанализирован. Также были обозначены проблемы, которые могут возникать на некоторых стадиях разработки, и представлены варианты их решения.

Затем была приведена информация о том, как используются искусственные нейронные сети на фондовом рынке. Так, была выявлена их область применения, определены положительные и отрицательные стороны.

После этого был выбран и проанализирован финансовый актив. В результате были рассмотрены государственные облигации, выпуском которых занимается правительство РФ, были выделены исходные данные, и была определена задача для нейронной сети.

Затем было отмечено, что некоторые значения из набора данных являются аномальными. Поэтому далее пришлось осуществить предобработку данных. Для этого сначала была приведена общая характеристика об аномалиях. Так, было раскрыто само понятие, затем были выявлены основные причины возникновения аномалий, после чего были определены и проанализированы их виды.

Дальше были рассмотрены методы, которые используются для обнаружения аномалий в массиве данных. Было дано подробное описание методик, были приведены конкретные примеры.

Далее на основе приведенных методик был разработан алгоритм, по которому должен осуществляться поиск аномальных экземпляров данных. В его основе лежит метод главных компонент, который отвечает за предобработку исходных данных, и метод кластерного анализа k-средних, с помощью которого определяются аномальные значения.

Наконец, была получена программа, предназначенная для решения поставленной задачи. Были рассмотрены основные этапы ее работы, проанализированы результаты ее вычислений.

На следующем этапе были представлены системы ИИ с нейросетевой структурой. Сначала была разработана и протестирована программа, в которой использовалась многослойная нейронная сеть. Затем был написан и проанализирован программный код сети с долгой краткосрочной памятью.

Далее была проведена оценка с помощью методов анализа временных рядов. Сначала была построена и оценена интегрированная модель авторегрессии-скользящего среднего. После чего был проведен сравнительный анализ между этой моделью и ранее разработанными нейросетевыми программами.

Таким образом, на основе данных, полученных в ходе написания научно-исследовательской работы, можно сказать, что необходимо постоянно совершенствовать алгоритмы машинного обучения из-за ряда причин:

Во-первых, на системы ИИ довольно часто перекладывают задачи, связанные с обработкой большого массива данных. Основная проблема заключается в том, что необходимо выделить только те данные, которые непосредственно нужны для решения. Так, одним из способов устранения данной проблемы является улучшение обучающих алгоритмов. Благодаря этому система ИИ начинает лучше понимать, какую информацию следует использовать для принятия решений.

Во-вторых, довольно часто возникают трудности, связанные с применением систем ИИ в динамичной среде. В этом случае, основная проблема заключается в постоянном переобучении системы, поскольку оно может занимать продолжительное время. Но стоит отметить, что это время можно сократить, если правильно оптимизировать исходный алгоритм обучения.

В-третьих, на поддержание работоспособности ИИ может уходить большое количество технических, энергетических, финансовых и трудовых ресурсов. Для этого необходимо создавать алгоритмы, способные поддерживать эффективную работу ИИ с оптимальными затратами.

## Список использованных источников

1. Abadi M. et al. Tensorflow: A system for large-scale machine learning // 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). – 2016. – P. 265-283.
2. Adebiyi A. A., Adewumi A. O., Ayo C. K. Comparison of ARIMA and artificial neural networks models for stock price prediction // Journal of Applied Mathematics. – 2014. – Vol. 2014.
3. Bholowalia P., Kumar A. EBK-means: A clustering technique based on elbow method and k-means in WSN // International Journal of Computer Applications. – 2014. – Vol. 105. – №. 9.
4. Brownlee J. Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras. – Machine Learning Mastery, 2016. – 256 p.
5. Chen K., Zhou Y., Dai F. A LSTM-based method for stock returns prediction: A case study of China stock market // 2015 IEEE international conference on big data (big data). – IEEE, 2015. – P. 2823-2824.
6. Erfani S. M. et al. R1SVM: a randomised nonlinear approach to large-scale anomaly detection // Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. – 2015. – P. 432-438.
7. Ghosh S., Dubey S. K. Comparative analysis of k-means and fuzzy c-means algorithms // International Journal of Advanced Computer Science and Applications. – 2013. – Vol. 4. – №. 4.
8. Gulli A., Pal S. Deep learning with Keras. – Packt Publishing Ltd, 2017.
9. Hoffer E., Hubara I., Soudry D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks // Advances in Neural Information Processing Systems. – 2017. – P. 1731-1741.
10. Hope T., Resheff Y. S., Lieder I. Learning tensorflow: A guide to building deep learning systems. – "O'Reilly Media, Inc.", 2017.
11. Ketkar N. Introduction to keras // Deep learning with Python. – Apress, Berkeley, CA, 2017. – P. 97-111.
12. Ketkar N. Introduction to Tensorflow // Deep Learning with Python. – Apress, Berkeley, CA, 2017. – P. 159-194.
13. Kind A., Stoecklin M. P., Dimitropoulos X. Histogram-based traffic anomaly detection // IEEE Transactions on Network and Service Management. – 2009. – Vol. 6. – №. 2. – P. 110-121.
14. Lai G. et al. Modeling long-and short-term temporal patterns with deep neural networks // The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. – 2018. – P. 95-104.
15. Laxhammar R., Falkman G., Sviestins E. Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator // 2009 12th International Conference on Information Fusion. – IEEE, 2009. – P. 756-763.
16. Leung K., Leckie C. Unsupervised anomaly detection in network intrusion detection using clusters // Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38. – 2005. – P. 333-342.
17. Nelson D. M. Q., Pereira A. C. M., de Oliveira R. A. Stock market's price movement prediction with LSTM neural networks // 2017 International joint conference on neural networks (IJCNN). – IEEE, 2017. – P. 1419-1426.
18. Pattanayak S., Pattanayak, John S. Pro Deep Learning with TensorFlow. – New York, NY, USA:: Apress, 2017.
19. Ramasubramanian K., Singh A. Deep learning using keras and tensorflow // Machine Learning Using R. – Apress, Berkeley, CA, 2019. – P. 667-688.
20. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. – Packt Publishing Ltd, 2019.
21. Shkodyrev V. et al. The Overview Of Anomaly Detection Methods in Data Streams // Proceedings of the Second Conference on Software Engineering and Information Management, Saint Petersburg, Russia. – 2017. – Vol. 1864.
22. Thatte G., Mitra U., Heidemann J. Parametric methods for anomaly detection in aggregate traffic // IEEE/ACM Transactions On Networking. – 2010. – Vol. 19. – №. 2. – P. 512-525.
23. Wold S., Esbensen K., Geladi P. Principal component analysis // Chemometrics and intelligent laboratory systems. – 1987. – Vol. 2. – №. 1-3. – P. 37-52.
24. Zhou L. et al. Machine learning on big data: Opportunities and challenges // Neurocomputing. – 2017. – Vol. 237. – P. 350-361.
25. Аведьян Э. Д. Алгоритмы настройки многослойных нейронных сетей // Автоматика и телемеханика. – 1995. – №. 4. – С. 106-118.
26. Антипов С. Г., Фомина М. В. Проблема обнаружения аномалий в наборах временных рядов // Программные продукты и системы. – 2012. – №. 2.
27. Бринк Х. Машинное обучение / пер. с англ. – Санкт-Петербург: Питер, 2017 – 336 с.
28. Буренин А.Н. Рынок ценных бумаг и производных финансовых инструментов: Учебное пособие — М.: 1 Федеративная Книготорговая Компания, 1998. — 352 с.
29. Васенков Д. В. Методы обучения искусственных нейронных сетей // Компьютерные инструменты в образовании. – 2007. – №. 1.
30. Галушкин А. И. Нейронные сети: основы теории. — М: Горячая линия — Телеком, 2012. – 496 с.
31. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.
32. Горбачевская Е. Н. Классификация нейронных сетей // Вестник Волжского университета им. ВН Татищева. – 2012. – №. 2. – С. 128-134.
33. Грэхем Б., Додд Д. Анализ ценных бумаг / Пер. с англ. – Москва: ЗАО “Олимп-Бизнес”, 2000. – 704 с.
34. Дмитриев Е. А. Нейронные сети долгосрочной памяти // Научные исследования и разработки студентов. – 2018. – С. 70-72.
35. Ершов К. С., Романова Т. Н. Анализ и классификация алгоритмов кластеризации // Новые информационные технологии в автоматизированных системах. – 2016. – №. 19.
36. Коэльо Л. П., Ричарт В. Построение систем машинного обучения на языке Python. 2-е издание / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2016. – 302 с.
37. Круг П. Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры». – М.: Издательство МЭИ, 2002. – 176 с.
38. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия - Телеком, 2002. – 382 с.
39. Мелихова О. А. и др. Режимы обучения в искусственных нейронных сетях // Инновации в науке. – 2016. – №. 50-1.
40. Мельникова В. А., Медведев Д. А. Анализ больших данных с использованием Python // Труды Братского государственного университета. Серия: Естественные и инженерные науки. – 2019. – Т. 1. – С. 46-49.
41. Мицель А. А., Ефремова Е. А. Прогнозирование динамики цен на фондовом рынке // Известия Томского политехнического университета. – 2006. – Т. 309. – №. 8. – С. 197-200.
42. Мюллер А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / пер. с англ. – Москва: Вильямс, 2017 – 480 с.
43. Николаева Ю. В. Критерии выбора архитектуры нейронной сети для прогнозирования финансовых рынков // Вестник ИжГТУ им. МТ Калашникова. – 2015. – Т. 18. – №. 1. – С. 96-97.
44. Подкорытова О. А. Анализ временных рядов: учебное пособие для вузов / О. А. Подкорытова, М. В. Соколов. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2020. – 267 с.
45. Рашид Т. Создаем нейронную сеть / пер. с англ. – Санкт-Петербург: ООО “Альфа-книга”, 2017. — 272 с.
46. Романов Д. Е. Нейронные сети обратного распространения ошибки // Инженерный вестник дона. – 2009. – Т. 9. – №. 3.
47. Рудой Г. И. Выбор функции активации при прогнозировании нейронными сетями // Машинное обучение и анализ данных. – 2011. – Т. 1. – №. 1. – С. 16-39.
48. Рындин А. А., Ульев В. П. Исследование скорости обучения нейронных сетей // Вестник Воронежского государственного технического университета. – 2012. – Т. 8. – №. 5.
49. Сайт информационного агентства Cbonds // URL: <http://cbonds.ru/> (Дата обращения: 23.03.2020). – Режим доступа: для авторизир. пользователей.
50. Созыкин А. В. Обзор методов обучения глубоких нейронных сетей // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2017. – Т. 6. – №. 3.
51. Теплова Т.В. Исследовательские поля облигационных рынков: монография / Т.В. Теплова, Т.В. Соколова. — Москва: ИНФРА-М, 2018. — 455 с.
52. Фабоцци Ф. Рынок облигаций: Анализ и стратегии / пер. с англ. – Москва: Альпина Бизнес Букс, 2007.
53. Федосин С. А., Ладяев Д. А., Марьина О. А. Анализ и сравнение методов обучения нейронных сетей // Инженерные технологии и системы. – 2010. – №. 4.
54. Шкодырев В. П. и др. Обзор методов обнаружения аномалий в потоках данных // Second Conference on Software Engineering and Information Management (SEIM-2017). – 2017. – С. 50.
55. Шолле Ф. Глубокое обучение на Python – Санкт-Петербург: Питер, 2018. – 400с.
56. Яхъяева Г. Э. Нечеткие множества и нейронные сети. — М.: Бином. – 2006. — 315 с.

## Приложение 1

Код программы для поиска аномалий:

