МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Брянский государственный технический университет»

Факультет информационных технологий

Кафедра «Компьютерные технологии и системы»

ДНЕВНИК

ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ (ПРЕДДИПЛОМНОЙ) СТУДЕНТА

Специальность: 10.05.04 "Информационно-аналитические системы безопасности"
Специализация: "Автоматизация информационно-аналитической деятельности"

Брянск 2022

Студент: Андронов Михаил Павлович

(ФИО)

Специальность: 10.05.04 "Информационно-аналитические системы безопасности", специализация: "Автоматизация информационно-аналитической деятельности" направляется на (в) ФГБОУ ВО «БГТУ», кафедра «КТС»

(предприятие, организация, фирма, компания)

1. Календарные сроки практики

По учебному плану начало 01.09.22 г. конец 21.12.22 г.

Дата прибытия на практику 01.09.22 г.

Дата прибытия с места практики 21.12.22 г.

2. Руководитель практики от БГТУ

Кафедра «Компьютерные технологии и системы»

Ученое звание к.т.н., доцент

Фамилия Мартыненко

Имя Алексей

Отчество Александрович

Дата	Описание работы, выполненной студентом	Отметка руководителя от базы практики
1	2	3
01.09.2022	Этап 1. Выступление руководителя практики. Инструктаж. Знакомство с программой производственной практики, режимом работы, перечнем отчетной документации.	Выполнено
7.09.2022	Этап 2. Разработка архитектуры информационной системы.	Выполнено
21.09.2022	Этап 3. Проектирование базы данных.	Выполнено
21.11.2022	Этап 4. Разработка пользовательского интерфейса для платформы Windows и Android.	Выполнено
15.11.2022	Этап 5. Тестирование программной системы.	Выполнено
02.12.2022	Этап 6. Оценка производительности системы.	Выполнено
15.12.2022	Этап 7. Подготовка материалов для отчетного семинара, оформление отчета по практике.	Выполнено
21.12.2022	Этап 8. Выступление с отчетной документацией на итоговом семинаре, промежуточная аттестация (дифференцированный зачет).	Выполнено

Руководитель практики от университета _____ _____

(Подпись) *(ФИО)*

График прохождения практики (календарно-тематический план)

№	Этапы	Выполняемая работа	Продолжительность
1	Выступление руководителя практики. Инструктаж.	Этап 1. Выступление руководителя практики. Инструктаж. Знакомство с программой производственной практики, режимом работы, перечнем отчетной документации.	1
2	Разработка архитектуры	Этап 2. Разработка архитектуры информационной системы.	1
3	Проектирование БД	Этап 3. Проектирование базы данных.	2
4	Разработка пользовательского интерфейса	Этап 4. Разработка пользовательского интерфейса для платформы Windows и Android.	8
5	Тестирование	Этап 5. Тестирование программной системы.	2
6	Оценка производительности	Этап 6. Оценка производительности системы.	2
7	Оформление отчета	Этап 7. Подготовка материалов для отчетного семинара, оформление отчета по практике.	1
8	Аудиторное представление отчета	Этап 8. Выступление с отчетной документацией на итоговом семинаре, промежуточная аттестация (дифференцированный зачет).	1

Руководитель практики от университета _____ _____

(Подпись) *(ФИО)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Брянский государственный технический университет»

Факультет информационных технологий

Кафедра «Компьютерные технологии и системы»

О Т Ч Е Т

по производственной практике (преддипломной)

Специальность: 10.05.04 "Информационно-аналитические системы безопасности"
Специализация: "Автоматизация информационно-аналитической деятельности"

Выполнил
Студент группы О-17-ИАС-аид-С
Андронов М.П.

Руководитель практики от БГТУ
к.т.н., доц. Мартыненко А.А.

Оценка отчета_____

«____» _____ 2022 г.

Брянск 2022

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Брянский государственный технический университет»

Кафедра «Компьютерные технологии и системы»
Специальность: 10.05.04 "Информационно-аналитические системы безопасности"
Специализация: "Автоматизация информационно-аналитической деятельности"

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на производственную практику (преддипломную)

для Андронова Михаила Павловича
(ФИО обучающегося полностью)
Обучающегося VI курса учебная группа № О-17-ИАС-аид-С
Место прохождения практики: БГТУ, кафедра «КТС»
адрес организации: г. Брянск
(указывается полное наименование структурного подразделения / профильной организации, а также их фактический адрес)
Срок прохождения практики с 01.09.22 по 21.12.22

Цель прохождения практики: получение теоретических и практических результатов предпроектного обследования профессиональной деятельности и информационных потребностей автоматизируемых подразделений предприятия.

Задачи практики:
- Разработать архитектуру информационной системы.
- Спроектировать API приложения.
- Спроектировать базу данных.
- Разработать интерфейс для платформы Windows.
- Разработать интерфейс для платформы Android.
- Протестировать разработанную систему.
- Оценить производительность разработанной системы.

УТВЕРЖДАЮ

_____
Руководитель практики от университета

«___»_____ 2022 г.

Задание принято к исполнению: _____ «___» _____ 2022 г.
(подпись обучающегося)

Содержание

ВВЕДЕНИЕ

В наше время фондовый рынок стремительно развивается. Инвесторам для эффективного вложения средств нужна оперативная информация о том, где купить акции, по какой стоимости, как котировки ценных бумаг (например, стоимость акций Газпрома, Сбербанка, ВТБ, Роснефти) меняются в течение торговой сессии. Также вопрос инвестиций в России становится всё более популярным для обычных граждан.

Понимание поведения инвесторов на финансовом рынке в целом и на фондовом рынке, в частности, всегда было и остается важной и актуальной проблемой для стабильности современной глобальной экономики. Существует немало примеров, вошедших в историю, когда неправильное представление или игнорирование тех или иных событий влекло к значительным убыткам не только для участников торгов, но и для компаний, акции которых обращаются на биржевом рынке, и даже для людей, которые никаким прямым образом не участвуют в данной системе. Примером может послужить «крах Уолл-стрит» 1929 года, ставший началом Великой депрессии. Другой пример: глобальный финансовый кризис 2007-2008 годов, который привел к спаду на рынке жилья, падению бизнеса и безработице.

Целью данной работы является моделирование информационно-аналитической системы анализа фондового рынка.

Для достижения данной цели необходимо решить ряд **задач**, таких как:

- анализ существующих систем анализа фондовых рынков;
- анализ источник данных, определение необходимых математических методов, наиболее подходящих для решения данной задачи;
- организация базы данных и заполнение её достаточным объемом информации;
- разработка аналитического модуля, позволяющего прогнозировать изменения на фондовых биржах;

- разработка программного модуля для актуализации имеющихся данных о котировках акций на фондовых биржах;
- разработка удобного и интуитивно понятного интерфейса для взаимодействия с информационно-аналитической системой.

Объектом исследования является изменения показателей акций различных компаний на фондовых рынках.

Предметом исследования являются подходы к автоматизации анализа изменения показателей акций компаний на фондовых рынках.

Методы и средства исследования. При реализации данной задачи используются методы системно-структурного анализа и декомпозиции предметной области, объектно-ориентированное программирование, использование методологии проектирования реляционных баз данных, формализованные методы анализа данных, работы с большими данными.

Практическую значимость работы составляет:

- разработанный программный модуль для актуализации данных о котировках акций на фондовых биржах;
- разработанный модуль анализа влияния новостей на изменения котировок акций;
- разработанный модуль для выполнения фундаментального и технического анализов фондового рынка на основе финансовой отчетности компаний;
- Выявленные в процессе анализа котировок зависимости изменения показателей акций на фондовых биржах от событий, происходящих внутри компаний, отрасли и в мире в целом.

1 Архитектура информационной системы

На этапе проектирования системы была разработана структура информационно-аналитической системы (рисунок 1).

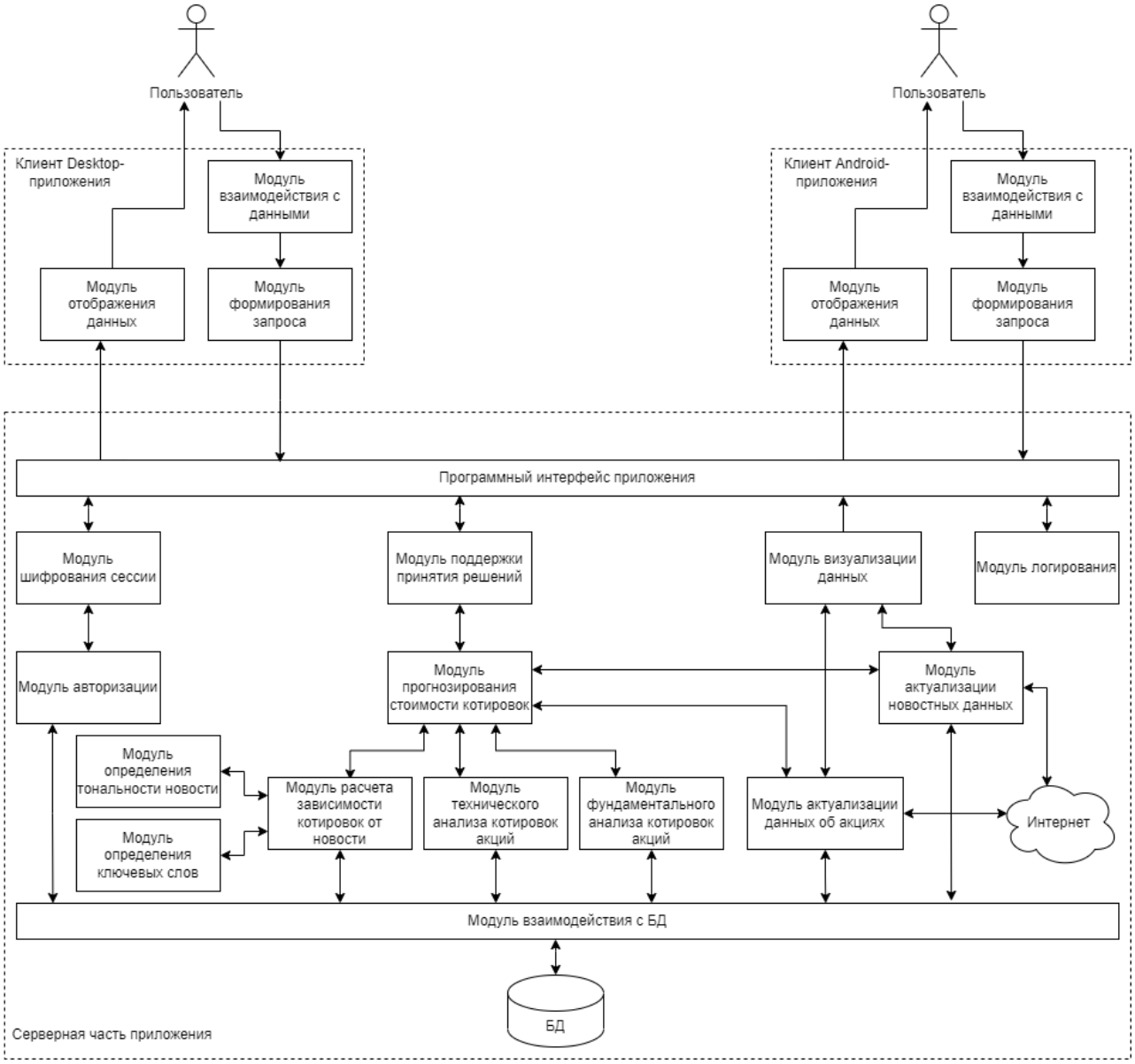

Рисунок 1 – Структура ИАС

В качестве архитектуры программного решения была выбрана архитектура «клиент-сервер». Данная архитектура подразумевает наличие серверного устройства, которое способно поддерживать одновременную работу с несколькими клиентами.

В рамках данной архитектуры на серверной части реализуется бизнес-логика приложения, модуль анализа данных, а также модуль взаимодействия с

базой данных. В свою очередь модуль взаимодействия с базой данных взаимодействует с сервером СУБД, которая производит манипуляции с данными в базе.

Описание структурных элементов ИАС

Информационная система должна иметь модульную структуру. Это позволит производить манипуляции внутри отдельных модулей, не внося изменения в другие модули. Также это обеспечит возможность расширения системы путем добавления новых модулей.

Описание серверной части ИАС

Программный интерфейс приложения – это модуль, который служит промежуточным звеном между клиентской и серверной частями приложения. В данном случае реализован универсальный интерфейс для взаимодействия с сервером. Благодаря чему, существует возможность обеспечить доступ к серверной части приложения, не зависимо от структуры и вида клиентской части.

Модуль авторизации реализует возможность идентификации пользователя, для предоставления ему персональной информации и его сохраненных данных. Так как при авторизации используется ввод пароля, то сессия пользователя должна быть зашифрована.

Модуль шифрования необходим для того, чтобы конфиденциальные данные пользователя не передавались по сети в открытом виде. Данный модуль позволяет при необходимости шифровать данные перед их отправкой и дешифровать при получении.

Модуль расчета зависимости котировок акций от новостей служит для выявления степени влияния новости на изменения стоимости акции. Функционирование данного модуля напрямую связана с модулями анализа текста новости. Данные, полученные в результате расчетов, фиксируются в базе данных.

Для анализа текста новости применяются модуль определения тональности текста и модуль определения ключевых слов. Под определением тональности текста подразумевается поиск в тексте новости обособленных сущностей и определение эмоционального отношения, с которым данные сущности

упоминаются. Модуль определения ключевых слов позволяет выделить ключевые слова из текста новости, для проведения более полноценного анализа.

Модули технического и фундаментального анализа необходимы для автоматизации соответствующих видов финансового анализа. Результаты работы данных модуль представляют их себя рассчитанные коэффициенты, значения мультипликаторов, а также значения индикаторов, влияющие на стоимость акции.

Задача модуля прогнозирования стоимости котировок акции заключается в расчете будущего значения стоимости акции исходя из данных и показателей, полученных от вышеупомянутых модулей. Кроме того, данный модуль обеспечивает взаимодействие с модулями актуализации данных, что позволяет гарантировать актуальность результатов работы данного модуля.

Модуль поддержки принятия решений реализует формирование комплекса рекомендаций, влияющих на конечный выбор пользователя. Исходя из данных прогноза модуль предлагает алгоритм действий пользователю, подкрепляя данные рекомендации результатами расчётов и анализа.

Модули актуализации новостных данных и данных об акциях обеспечивают поддержание актуального состояния базы данных. За счет автоматического мониторинга данных в сети Интернет. Кроме того, данные модули являются источниками данных для модуля визуализации данных. Являясь промежуточным звеном для информации из базы данных, они при необходимости дополняют данные актуальной информацией и затем сохраняют ей в базе данных.

Модуль визуализации данных необходим для формирования структуры и формата данных для последующей передачи их клиенту.

Модуль логирования осуществляет мониторинг работы системы и фиксацию в лог-файл аномальных состояний, а также состояний, при которых появляется угроза нормального функционирования программного продукта. Записи в лог-файле необходимы для определения причин перехода системы в аномальные или аварийные состояния, с целью как можно быстрее внести соответствующие корректировки в настройки система или программный код.

Описание клиентской части ИАС

Клиентская часть разрабатываемой системы подразумевает универсальную структуру вне зависимости от вида реализации клиентской части разрабатываемого продукта. Структура клиентской части включает в себя три модуля.

Модуль взаимодействия с данными определяет возможность пользователя управлять данными системы посредством понятных для пользователя действий. Тем самым данный модуль реализуют интерфейс взаимодействия пользователя с системой.

Модуль формирования запросов позволяет на основе действий пользователя сформировать запрос на получение или изменение данных. Данный модуль осуществляет связь клиентской и серверной частей программного продукта.

Для демонстрации данных используется модуль отображения данных. Функция данного модуля заключается в интерпретации данных с сервера и представлении их в удобном для пользователя виде. В частности, данный модуль обеспечивает возможность пользователя взаимодействовать с различными диаграммами, таблицами и графиками.

При создании структурной схемы необходимо использовать стандарт IDEF0 (ICAM Definition – integrated computer aided manufacturing definition).

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Данный стандарт позволяет представить программный комплекс в виде набора функциональных блоков, каждый из которых может осуществлять взаимодействие с другими функциональными блоками посредством четырех видов интерфейса (входа, управления, механизма, выхода).

Интерфейс входа (слева) описывает исходные данные или объекты для выполнения функций. Интерфейс управления (сверху) описывает правила и

ограничения. Интерфейс механизма (снизу) описывает ресурсы, используемые в процессе выполнения функции (ресурсы не должны изменяться). Интерфейс выхода (справа) описывает данные или объекты, являющиеся результатом выполнения функции.

Схема IDEF0 делиться на несколько уровней, первый уровень (рисунок 2) представляет из себя один блок «Прогноз стоимости акции». На вход поступают данные от пользователя, а именно название выбранной компании и период, на который будет производится анализ. Выходными данными является результат прогноза стоимости акции.

Рисунок 2 – Функциональная диаграмма системы (уровень 1)

На втором уровне детализации функциональной диаграммы процессы работы программы отображены более детально (рисунок 3). На данном уровне представлены блок по формированию данных для анализа, блок с расчета влияния новостей на стоимость акции, блок технического анализа котировок, блок фундаментального анализа компании и блок прогнозирования стоимости акции.

На этапе формирования данных для анализа программа получает входные данные от пользователя и необходимые данные из базы данных и из интернета.

На следующих этапах проводится комплексный анализ данных, а именно происходит расчет изменения стоимости акции в зависимости от новостей за

определенный временной период. Оценка проводится с использованием методов машинного обучения, в данном случае эти методы реализованы в виде обученной нейронной сети.

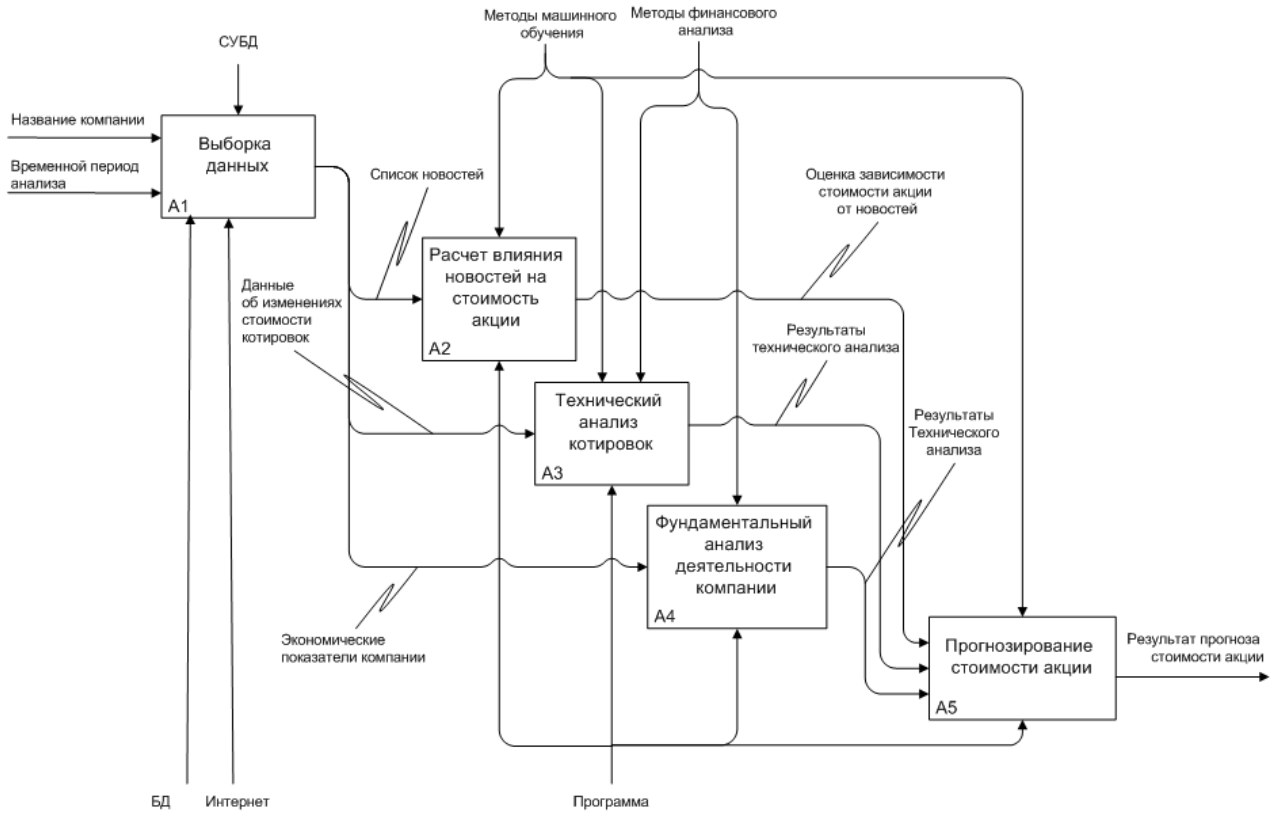

Рисунок 3 – Функциональная диаграмма системы (уровень 2)

Кроме того, данные проходят этап технического анализа. Анализ проводится на основе методов финансового анализа с применением методов машинного обучения. В качестве входных данных для анализа используются исторически данные об изменении стоимости котировок акции за определенный период времени.

В блоке фундаментального анализа деятельности компании производится оценка экономических показателей и расчет мультипликаторов для определения финансового состояния компании. В качестве входных данных используются экономические показатели компании, полученные на основе финансовой отчетности компании.

Блок прогнозирования стоимости акции агрегирует в себе результаты анализа, полученные из вышеуказанных блоков, и на основе этих данных формирует результирующую оценку будущей стоимости акции.

При анализе стоимости котировок акции одним из блоков схемы IDEF0 является блок «Расчет влияния новости на стоимость акции» (рисунок 4). На вход в данные блок поступает перечень акций для расчета влияния, а также выбранная новость, влияние которой рассчитывается. Результатом работы блока является сохранение данных о влиянии новости на стоимость котировок акции.

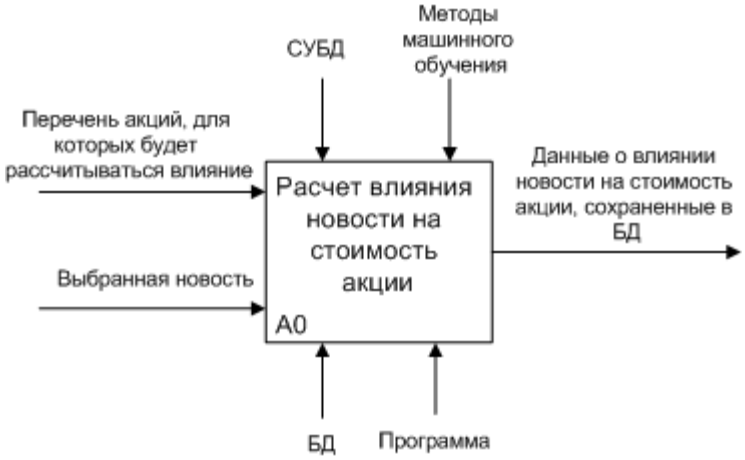

Рисунок 4 – IDEF-диаграмма расчета влияния новости на стоимость акции (уровень 1)

На втором уровне детализации блок «Расчет влияния новости на стоимость котировки акции» разделяются на 4 блока: «Выборка данных», «Определение тональности текста новости», «Определение влияния новости на стоимость акции» и «Сохранение оценки влияния в базу данных» (рисунок 5).

Блок «Выборка данных» на вход получает новость и перечень акций, для котировок которых будет производиться расчет влияния. Результатом работы данного блока является сформированный набор данных для последующего анализа.

На вход блока «Определение тональности текста новости» подается сформированный набор данных. Далее при помощи методов машинного обучения производится определение тональности текст новости, затем полученные данные о тональности передаются в следующий блок.

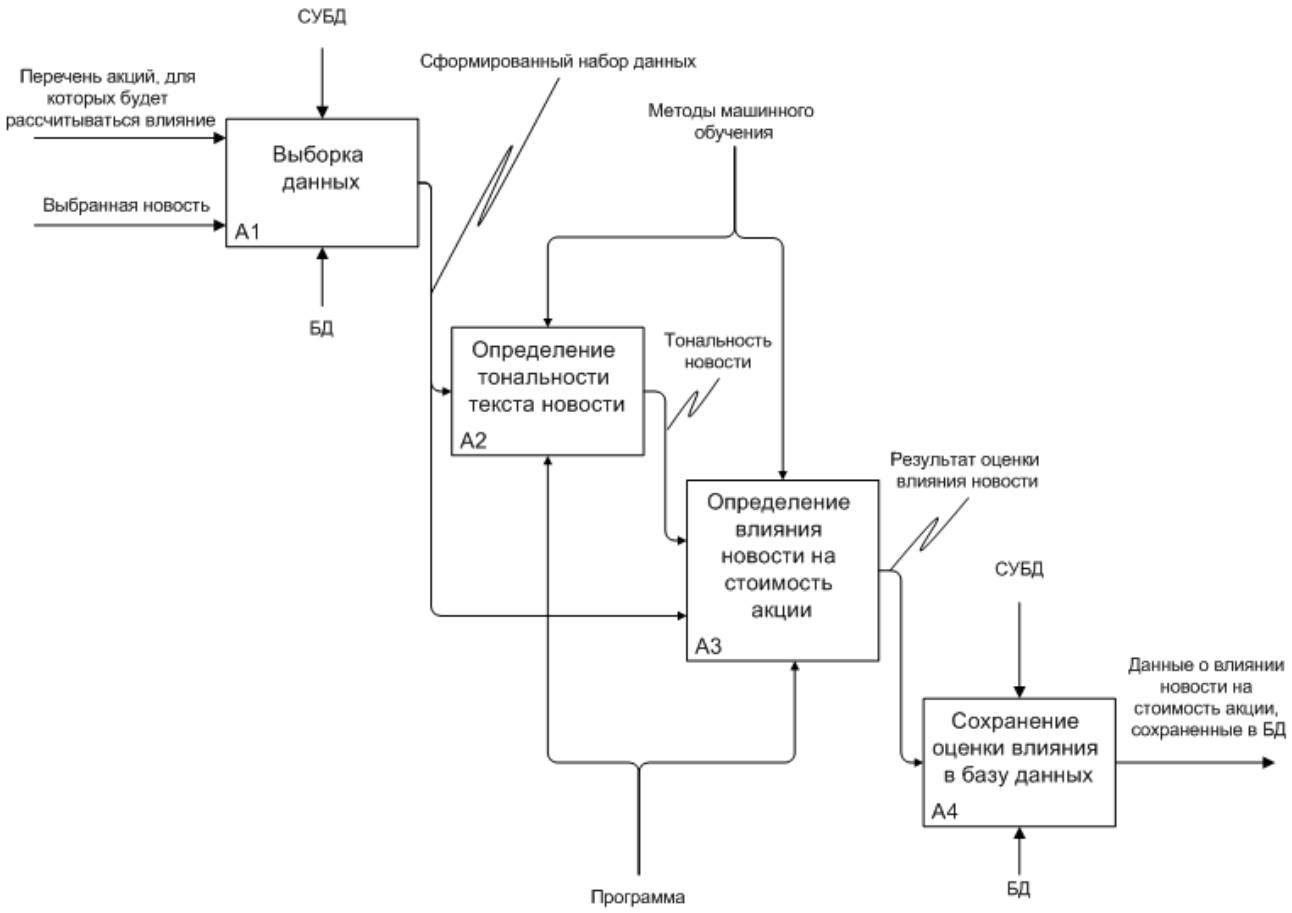

Рисунок 5 – IDEF-диаграмма расчета влияния новости на стоимость акции (уровень 2)

Блок «Определение влияния новости на стоимость котировки акции» получает на вход сформированный набор исторических данных о котировках акций, а также тональность новости, определенная в предыдущем блоке. Результатом данной области является результат оценки влияния новости.

Для расчета влияния новости на стоимость акции производится поиск и агрегация всех новостей, оказывающих влияние на акции данной компании. Далее список новостей фильтруются, т.е. из общего списка новостей исключаются те новости, которые не имеют пересечений по ключевым словам, с рассматриваемой новостью. Расчет влияния искомой новости на стоимость акций компании можно представить в виде следующей формулы:

$$impact = \sum_{n=1}^{N} \frac{\sum_{i=1}^{K}(-1^{tn} * impact_n * w_{kn})}{K}, \quad (1)$$

где N – количество новостей, влияющих на компанию, K – количество пересекающихся ключевых слов, $impact_n$ – влияние новости n на стоимость акции, w_{kn} – влияние ключевого слова k на стоимость компании в рамках новости n, tn – тональность новости по отношению к компании (1 – негативная, 2 - позитивная).

Следующий блок «Сохранение оценки влияния в базу данных» получает на вход результат оценки влияния новости на изменение стоимости котировки акции. А затем сохраняет её в базу данных, для дальнейшего анализа. Поэтому результатом работы данного блока является данные о влияние новости на стоимость котировки акции, сохраненные в базу данных.

При разработки информационной системы необходимо учитывать какие данные и каким образом будут перемещаться между модулями и блоками системы. Диаграмма потоков данных (DFD – Data Flow Diagram) подразумевает визуализацию хранилищ данных, процессов, происходящих в информационной системе, соединенных потоками данных, а также внешние сущности, взаимодействующие с системой.

Ниже представлен фрагмент диаграммы потоков данных разрабатываемой ИАС, на ней продемонстрированы процессы, связанные с анализом стоимости акций и прогнозирования их стоимости в будущих периодах (рисунок 6).

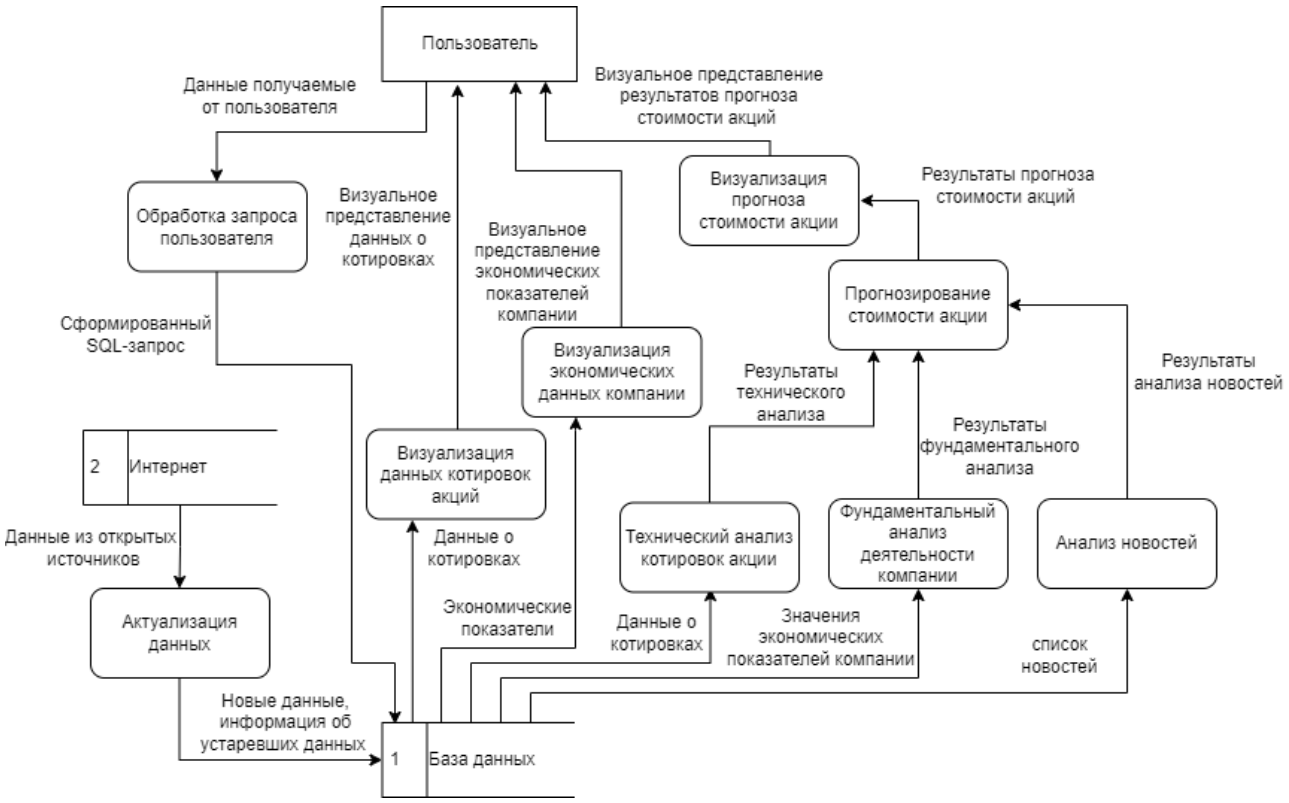

Рисунок 6 – Диаграмма потоков данных (фрагмент)

В качестве хранилищ данных в системе используются база данных и сеть Интернет. При этом основным хранилищем является база данных, получение новой информации происходит из сети Интернет в процессе актуализации данных. Важно отметить, что актуализация данных подразумевает под собой не только добавление новой информации, но и редактирование уже существующих данных.

Внешней сущностью системы является пользователь. Он является инициатором отправки данных на сервер, и он же является конечной точкой для отправляемых данных.

Данные от пользователя проходя процесс обработки, при котором происходит их преобразование в набор необходимых SQL-запросов. Полученные запросы обрабатываются базой данных. В результате обработки на выходе формируется набор данных, который в зависимости от цели перенаправляется в модули визуализации информации или в модули анализа информации. К визуализации информации можно отнести процессы визуализации данных о котировках акций и визуализации экономических данных компании.

В случае выполнения анализа данных, сформированные данные из базы данных используются в процессах технического и фундаментального анализов, а также в процессе анализа новостей. Полученные на выходе данные используются для составления прогноза стоимости акции. Данные полученные в процессе прогнозирования отправляются в модуль визуализации для представления пользователю в удобном для восприятия виде.

2 Описание основных методов серверной части ПО

Так как разрабатываемая информационная система построена по принципам клиент-серверной архитектуры, то необходимо реализовать возможность обмена информацией между клиентской и серверной частями.

Одним из способов такого обмена является разработка программного интерфейса приложения или API (Application Programming Interface). Данный подход подразумевает набор URL-адресов (эндпоинтов), которые при обращении возвращают необходимые данные в универсальном формате или запускают определенный код обработки данных на сервере.

Главным плюсом использования API является его универсальность. Благодаря тому, что передача данных происходит посредством протокола HTTP, обращение к API можно производить независимо от конфигурации клиентского приложения. Это может быть как web-сайт или мобильное приложение, так и внешний сервис, которые взаимодействует с разрабатываемой системой.

В разрабатываемой ИАС программный интерфейс можно разделить на несколько блоков в зависимости от обрабатываемой информации (рисунок 7).

Блок «Auth» представляет собой набор эндпоинтов для осуществления регистрации пользователя, авторизации и процесса разлогирования. Также в данном блоке содержатся эндпоинты, необходимые для работы личного кабинета пользователя, например получения списка избранных акций.

Эндпоинты из блока «World news» используются для получения списка мировых новостей, а также информации о конкретной новости.

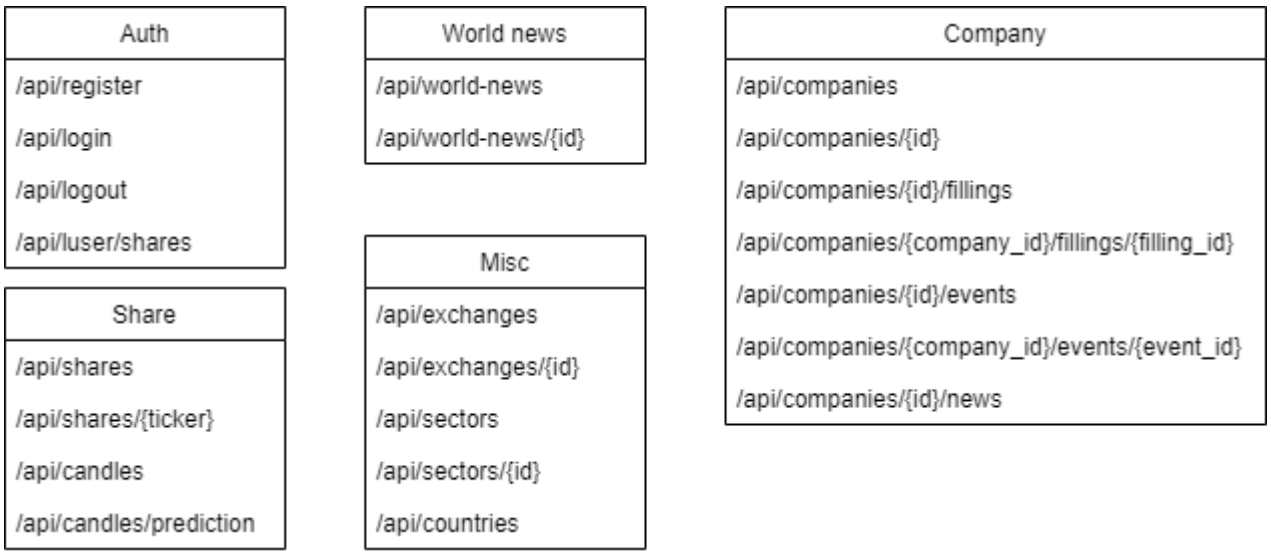

Рисунок 7 – Структура программного интерфейса приложения

Блок «Share» содержит в себе набор эндпоинтов для работы с акциями и их котировками. Например, получения списка акций, получение детальной информации об акции, получения котировок акций или получения результатов анализа и прогноза стоимости котировок акций.

В блоке «Company» объединены эндпоинты для получения данных о компании. В частности, это эндпоинты для получения списка компаний, детальной информации о компании, включающую экономические показатели, получение финансовой отчетности компании, получение списка новостей и событий, происходящих в компании.

Различные сервисные эндпоинты сгруппированы в блоке «Misc». В данном блоке перечислены эндпоинты для получения списка фондовых бирж и перечня акций, представленных на конкретной бирже; получения списка секторов экономики и списка компаний, входящих в их состав; а также списка стран, которые используются для фильтрации и группировки компаний и акций.

Реализуемые эндпоинты доступны по протоколу HTTP и возвращают ответ от сервера в формате JSON. Так как данный формат данных является одним из часто используемых форматов, используемых при построении API. Это обеспечит возможность интеграции разрабатываемой системы с различными сервисами.

3 Проектирование базы данных

На основе проведенного анализа предметной области была спроектирована база данных, в которой будет храниться вся необходимая информация для разрабатываемой системы. Данная структура представлена в виде отдельных взаимосвязанных таблиц (рисунок 8).

Таблица «companies» содержит в себе информацию о компании, эмитенте акций. Данная информация включает в себя общую информацию о компании, такую как название компании, описание её деятельности, контакты, численность работников и т.д. Также в данной таблице содержатся экономические характеристики компании, например доходы и расходы компании, её стоимость, выручку и прочие экономические параметры. Таблица «companies» связана с таблицами «company_filings» и «company_ events» связями один-ко-многим, так как внутри одной компании публикуется множество документов и происходит множество событий.

Таблица «company_filings» содержит данные об экономических отчетах компании. В таблице содержится информация о различных бухгалтерских отчетах, например бухгалтерские балансы, отчеты об изменении капитала, движении денежных средств и т.д. Для этого таблица содержит такие поля как дата публикации, тип отчета, ссылка на сам документ и т.д.

В таблице «company_events» хранится информация о событиях, происходящих в компании, например выплаты дивидендов и даты их проведения. Учитываются такие свойства события как его типа, дата наступления события, название и ссылку для переходя к дополнительной информации.

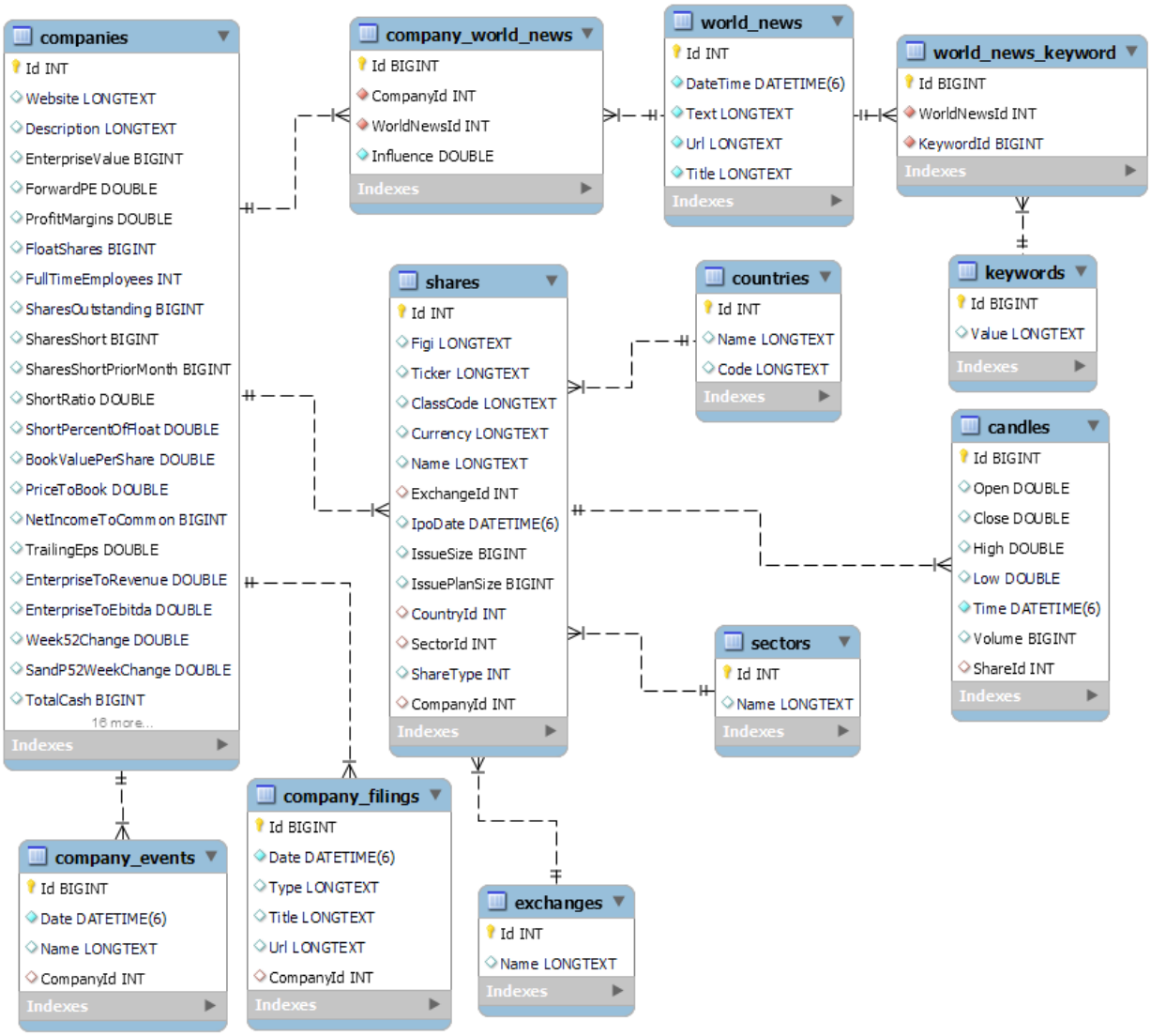

Рисунок 8 – ER-диаграмма (фрагмент)

Для хранения данных об акциях была использована таблица «shares». Данная таблица является промежуточной между компаниями и биржами. Она необходима, так как акции одной и той же компании могут размещаться на различных биржах. Таблица содержит информацию о дате IPO, количестве выпущенных акций и т.д. Кроме того, таблица содержит внешние ключи на таблицы «countries», «exchanges» и «sectors», которые хранят информацию о стране, фондовая биржа и отрасли, к которым относится конкретная акция. Данные таблицы необходимы для фильтрации и группировки акций компаний.

Таблица «candles» хранит в себе исторические данные о котировках. Данные включают в себя количество проданных акций, стоимость акции на момент открытия, закрытия, а также наибольшая и наименьшая стоимости акции, за

конкретную дату. Также в данной таблице хранится ссылка на таблицу «shares» реализуя связь один-ко-многим.

Таблица «company_world_news» является связующей таблицей между компаниями и мировыми новостями, которые влияют на показатели изменения котировок акций. Она содержит в себя ссылки на таблицы «companies» и «world_news», также данная таблица хранит в себе оценку влияния мировых новостей на изменения котировок акций.

4 Выбор формализованных методов анализа

Одной из основных задач, которые должна решать разрабатываемая ИАС, является задача прогнозирования стоимости акции. При декомпозиции задачи прогнозирования были выделены три под задачи: прогнозирование на базе технического анализа, фундаментального анализа и анализа новостей.

Технический анализ акций компании представляет собой систему прогнозирования цен, основанную на информации, полученной в результате рыночных торгов. Иными словами, в основе технического анализа лежит выделение и изучение определенных закономерностей в движении графика котировок. То есть принятое решение основывается только на графическом изображении линии тренда.

Для формализации данного метода анализа было принято решение использовать методы машинного обучения, а именно обученную рекуррентную нейронную сеть. Особенность рекуррентных нейронных сетей заключается в возможности передавать информацию между итерациями обработки данных (рисунок 9).

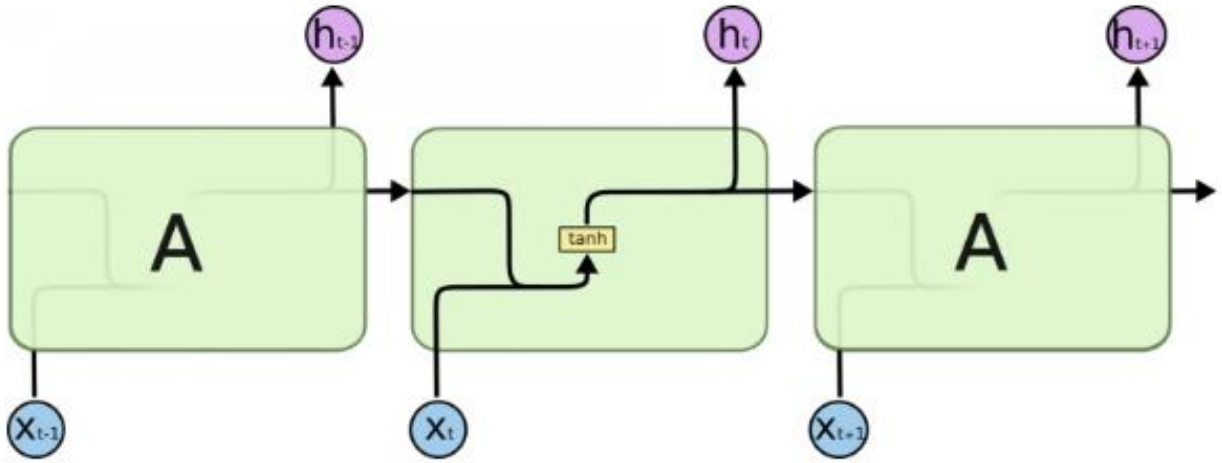

Рисунок 9 – Архитектура рекуррентной нейронной сети

Данные, полученные на предыдущей итерации нейронной сети, принято обозначать как h_{t-1}, входные данные на текущем шаге – x_t, функция зависимости результирующего показателя от входного параметра обозначается как σ. В качестве расчетной функции используется гиперболический тангенс tanh(). Таким образом значение текущей итерации (h_t) нейронной сети рассчитывается как:

$$h_t = \tanh\left(\delta(h_{t-1}, x_t)\right) \quad (2)$$

Также было проведено тестирование различных типов нейронных сетей, для определения наиболее подходящей к данной задаче структуры. В качестве тестового набора данных были взяты данные о стоимости акций компании Лукойл.

Полносвязная нейронная сеть

Наилучшие результаты были получены при использовании четырех полносвязных слоев из 100 нейронов с линейной функцией активации (рисунок 10).

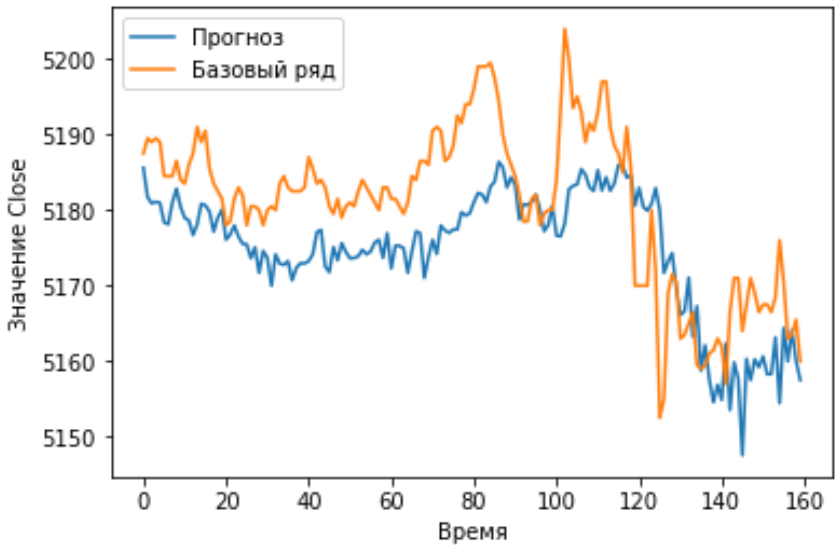

Рисунок 10 – Результат прогнозирования полносвязной нейронной сети

Исследование проводилось для прогнозируемого периода от 10 до 60 часов. В зависимости от длительности прогнозируемого периода точность изменялась от 0.997 до 0.982 (Рисунок 11).

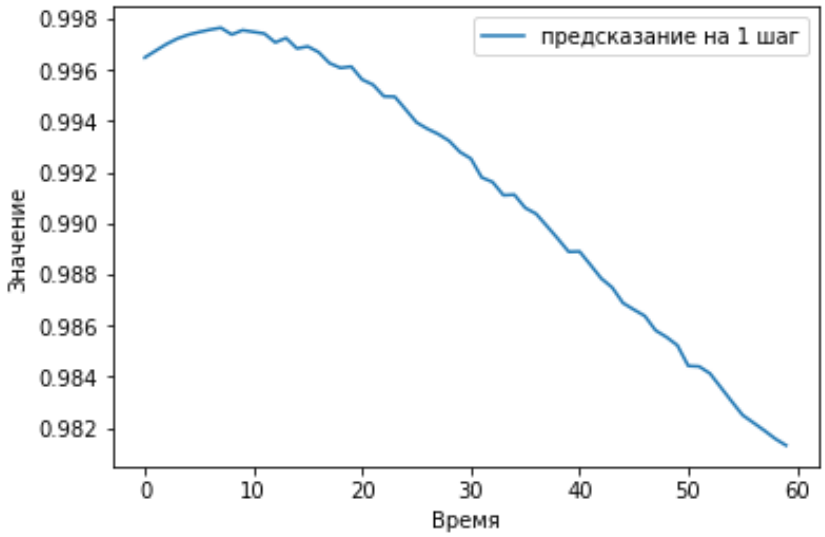

Рисунок 11 – Изменение точности прогнозирования для полносвязной сети

Сверточная нейронная сеть

Наилучшие результаты были получены при использовании трех сверточных слоев первый слой из 50 нейронов, остальные из 100 нейронов с линейной функцией активации (рисунок 12).

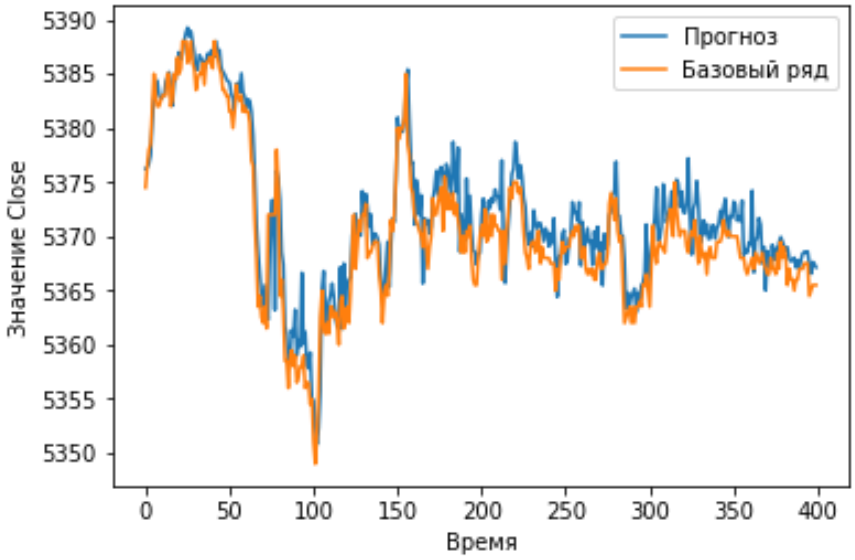

Рисунок 12 – Результат прогнозирования сверточной нейронной сети

Исследование проводилось для прогнозируемого периода от 0 до 10 часов, т.к. точность работы данной сети на порядок отличается от точности полносвязной сети и не имеет смысла производить расчеты для большего периода. В зависимости от длительности прогнозируемого периода точность изменялась от 0.9997 до 0.997 (рисунок 13).

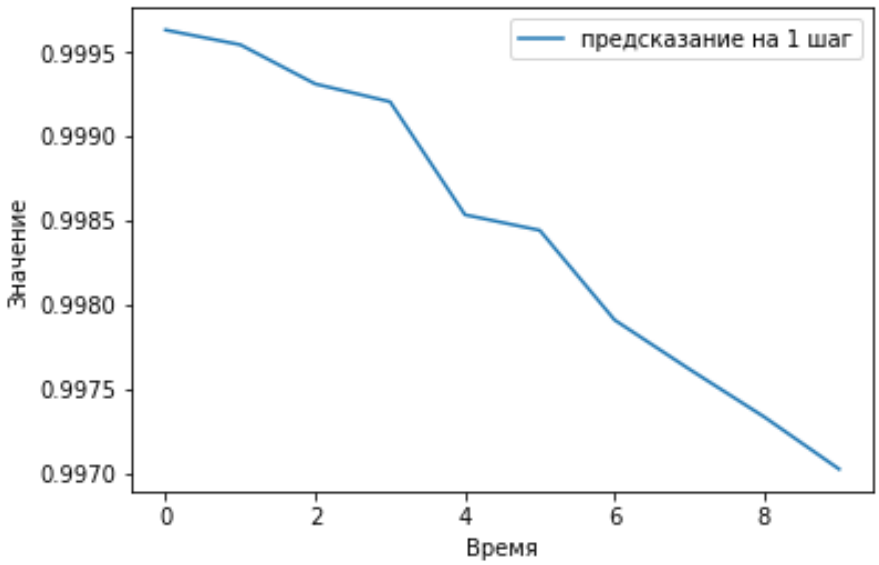

Рисунок 13 – Изменение точности прогнозирования для сверточной сети

Рекуррентная нейронная сеть

Наилучшие результаты были получены при использовании трех слоев из 100 нейронов, в качестве функции активации был использован гиперболический тангенс (рисунок 14).

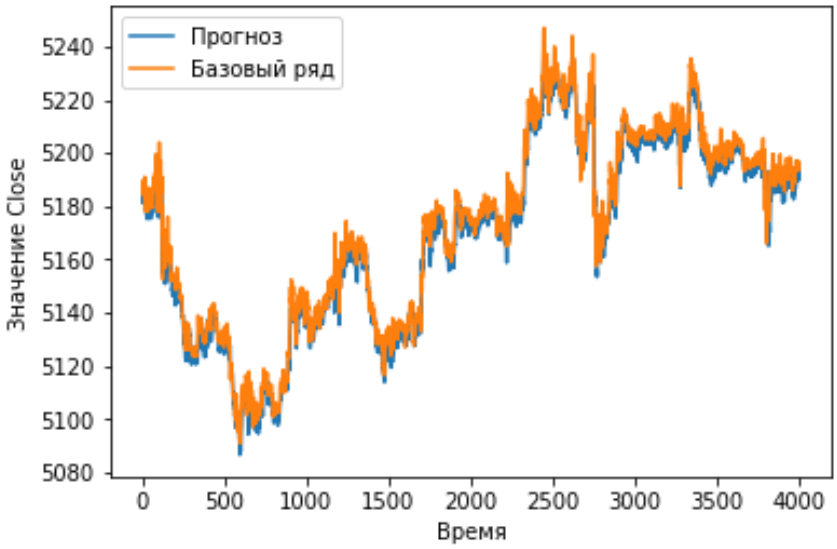

Рисунок 14 – Результат прогнозирования рекуррентной сети

Исследование проводилось для прогнозируемого периода от 0 до 10 часов. В зависимости от длительности прогнозируемого периода точность изменялась от 0.9996 до 0.998 (рисунок 15).

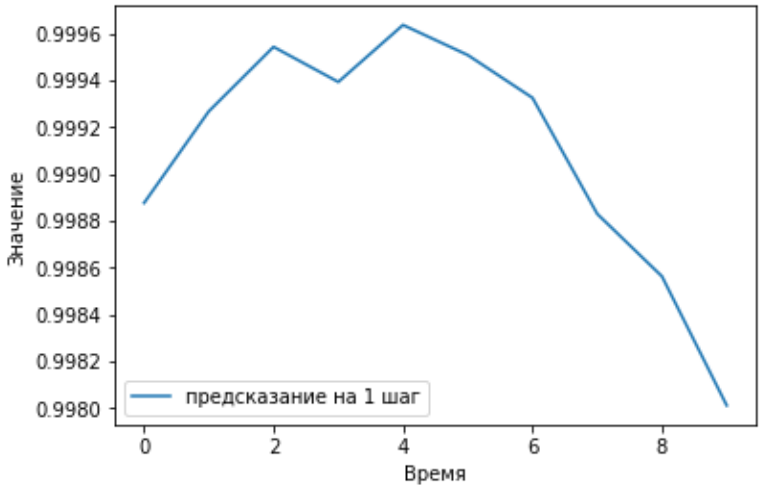

Рисунок 15 – Изменение точности прогнозирования для рекуррентной сети

При сравнении результатов работы нейронных сетей, было замечено, что рекуррентная НС обеспечивает большую точность при увеличении периода прогноза, чем сверточная.

Для осуществления анализа новостей и оценки их влияния на стоимость акции необходимо произвести необходимо решить задачи, связанные с анализом естественного языка. В частности, необходимо в тексте определить перечень компаний, упоминаемых в новости, а также оценить тональность текста новости относительно каждой из обнаруженных компаний. Кроме того, необходимо выделить из текста новости ключевые слова, для формирования более полной оценки отношения встречающихся слов в тексте к упоминаемым компаниям.

В качестве средства для анализа текста наилучшим образом подойдет модель LSTM (Long short-term memory), так как для данной модели в свободном доступе имеется множество наборов данных, как на английском, так и на русском языках. Для достижения поставленной цели это использование метода замены сущности при формировании набора данных для обучения.

Например, анализируя текст новости «… компания Apple фиксирует увеличение прибыли на фоне пожара, произошедшего на заводе Samsung …», лингвистической модели будет сложно определить тональность данной новости. Поэтому при обучении сущности «Apple» и «Samsung» необходимо последовательно выделять как ключевую сущность (таблица 1).

Таблица 1 – Пример набора данных для обучения лингвистической модели

Ключевая сущность	Текст новости	Тональность текста
Apple	компания **X** фиксирует увеличение прибыли на фоне пожара, произошедшего на заводе Samsung	позитивно
Samsung	компания Apple фиксирует увеличение прибыли на фоне пожара, произошедшего на заводе **X**	негативно

Для определения тональности новости используется следующий алгоритм, изображенный на IDEF-диаграмме (рисунок 16).

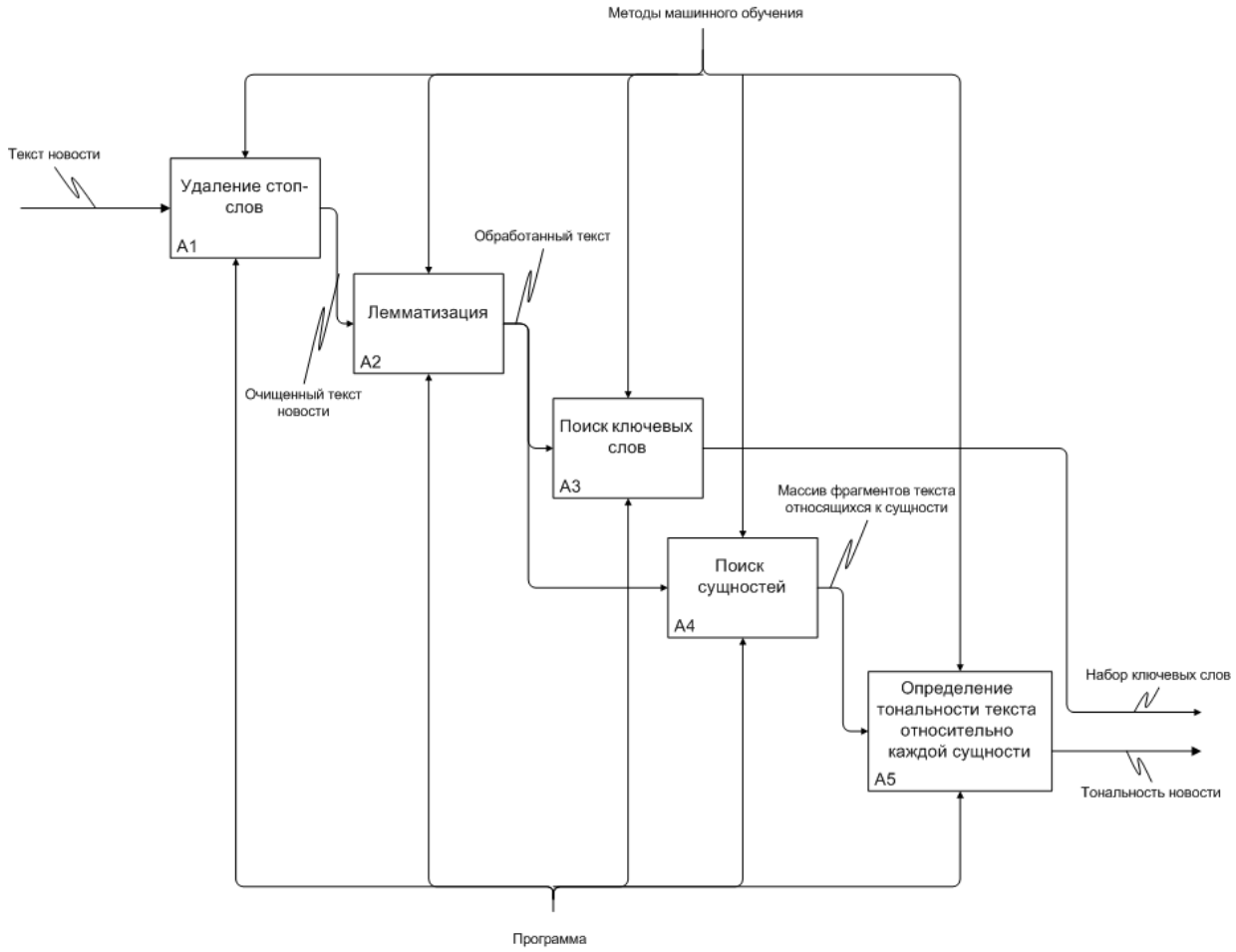

Рисунок 16 – Алгоритм для определения тональности текста

Влияние новости на изменение стоимости акций оценивается исходя из набора исторических данных о котировках компаний (рисунок 17). Изменение стоимости акции компании за определенный период рассчитывается как разница между стоимостью в начале периода и в конце.

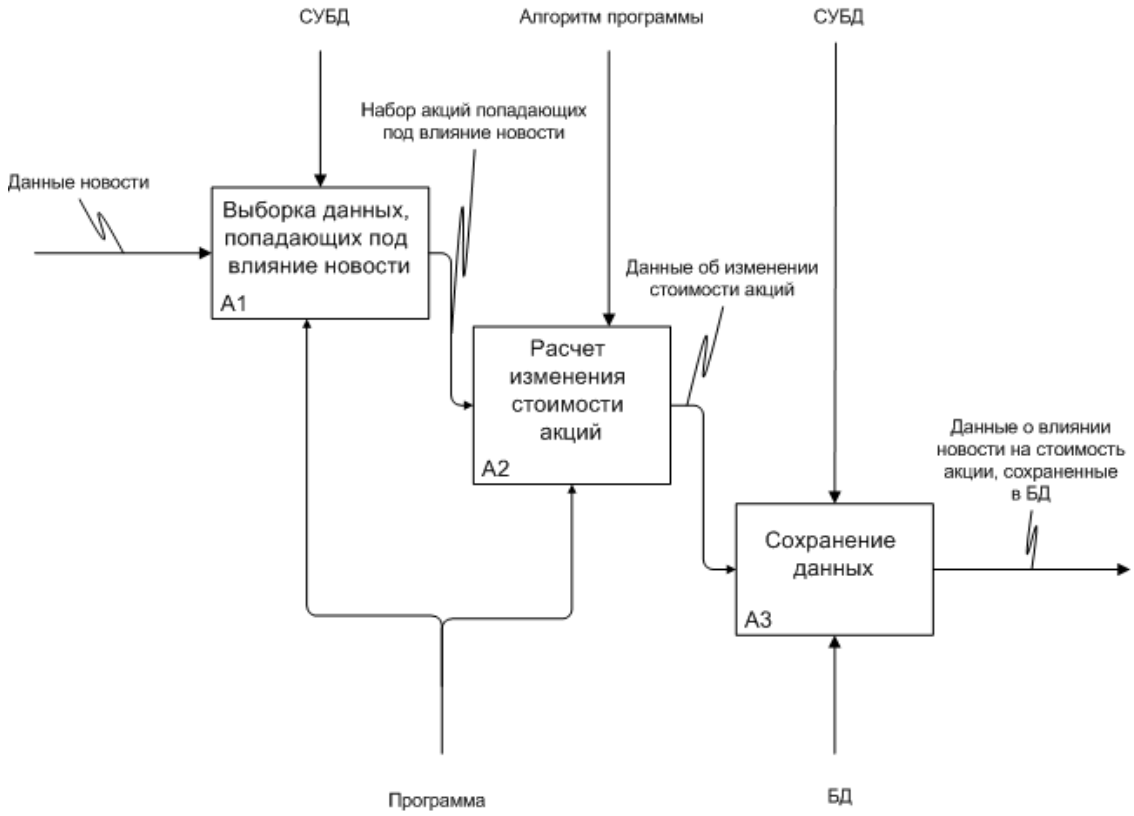

Рисунок 17 – Алгоритм для расчета влияние новости на стоимость акции

Для получения наиболее достоверных и полных результатов анализа необходимо агрегировать данные полученные при анализе влияния новостей на стоимость акции компаний и техническом анализе котировок.

Агрегирование результатов анализа является необходимым этапом прогнозирования стоимости акции компании. В виду того, что анализ осуществляется с применением нескольких независимых видов оценки, результаты, полученные при использовании различных методов, могут отличатся друг от друга. В связи с этим необходимо обеспечить согласованность результатов и разработать формулу для получения прогноза на основе полного многофакторного анализа.

Для получения совокупного результата можно применять формулу среднего арифметического:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{1}{3}\sum_{i=1}^{3} x_i, \quad (3)$$

где n – количество методов анализа, x_i – результат анализа i-го метода анализа.

Важным этапом при расчете совокупного результата анализа является проверка полученных данных на согласованность. Данный этап необходим, потому что получение совокупного прогноза может дать неверные результаты в случае, если результаты методов анализа противоречат друг другу.

Для определения согласованности результата необходимо рассчитать отклонение результатов анализа определенного метода от среднего совокупного результата. Расчет может быть произведен по формуле среднего квадратичного отклонения.

Кроме того, необходимо произвести расчет коэффициента вариации, для выражения согласованности в процентах:

$$V = \frac{\sigma}{\bar{x}} * 100\%, \quad (4)$$

где σ - результат среднего квадратичного отклонения.

Результаты можно считать согласованными в случае, если коэффициент больше 75%, в данном случае результат агрегирования является достоверным. В случае, если коэффициент меньше 75%, то это говорит о невозможности сделать однозначный прогноз. Лучшим решением в данном случае является предоставление аналитику результатов анализа по каждому из методов, для самостоятельного принятия решения

5 Разработка пользовательского интерфейса

При разработке информационно-аналитической системы необходимо реализовать интерфейс взаимодействия пользователя с системой. При этом пользовательский интерфейс должен удовлетворять определенным критериям:

- Естественность интерфейса – свойство интерфейса, которое означает, что сообщения и результаты, выдаваемые приложением, не должны требовать дополнительных пояснений.
- Согласованность интерфейса позволяет пользователям переносить имеющиеся знания на новые задания, осваивать новые аспекты быстрее, и

благодаря этому фокусировать внимание на решаемой задаче, а не тратить время на уяснение различий в использовании тех или иных элементов управления, команд и т. д. Обеспечивая преемственность полученных ранее знаний и навыков, согласованность делает интерфейс узнаваемым и предсказуемым.

- Принцип «обратной связи» – принцип, который означает, что каждое действие пользователя должно получать визуальное, а иногда и звуковое подтверждение того, что программное обеспечение восприняло введенную команду; при этом вид реакции, по возможности, должен учитывать природу выполненного действия.

- Простота интерфейса – представление на экране информации, минимально необходимой для выполнения пользователем очередного шага задания.

Базовым элементом, вокруг которого строится анализ и прогнозирование стоимости акции, является график изменения её стоимости во времени. Однако кроме на стоимость могут также влиять экономические показатели компании, мировые новости, а также изменения стоимости других акций.

С целью расширить возможности анализа было принято решение использовать построение диаграммы с возможность добавления на нее графиков других компаний. При этом для удобства реализована функция, которая при клике на определенную дату графика выводит список новостей за этот день (рисунок 18), что позволяет пользователю одним нажатием перейти от технического анализа к анализу новостей.

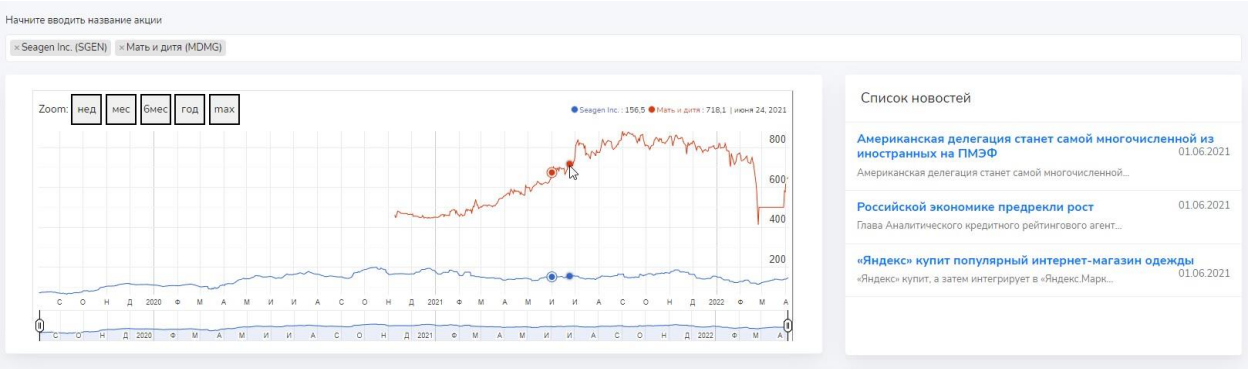

Рисунок 18 – Интерфейс программного продукта (фрагмент)

При переходе на страницу компании, пользователь может получить информацию о её деятельности, экономическом состоянии, а также оценить тренд изменения стоимости акции на основе графика.

Также при переходе на страницу новости, пользователь может увидеть не только текст новости, но и влияние, которое данная новость оказала на стоимость котировок тех или иных компаний (рисунок 19).

Рисунок 19 – Интерфейс страницы новости (фрагмент)

Интерфейс мобильного приложения был спроектирован таким образом, чтобы пользователь мог быстро совершить необходимые действия. Кроме того, важно отметить, что функционал мобильного приложения является неполным.

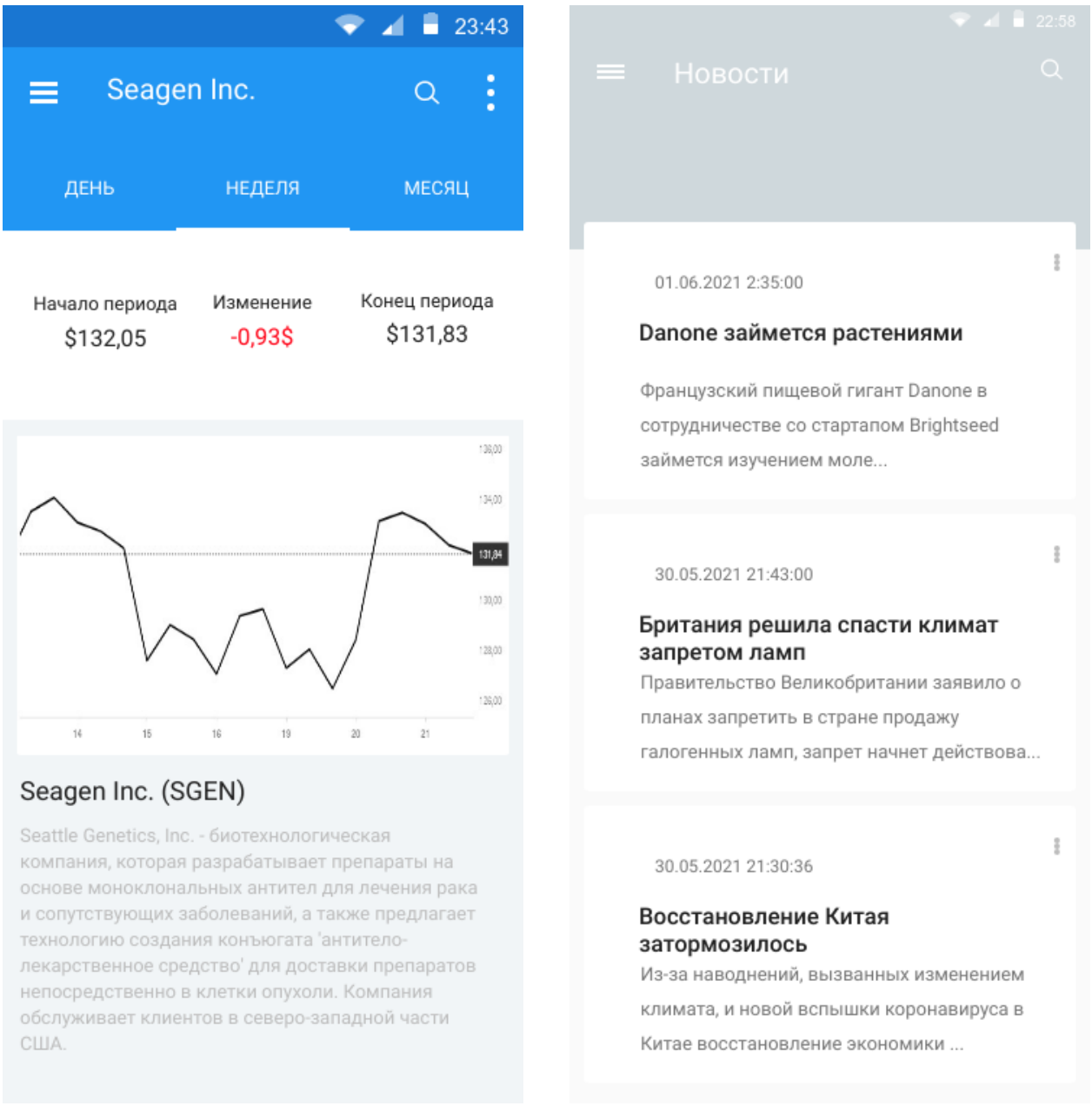

Рисунок 20 – Интерфейс мобильного приложения
(страницы «Акция» и «Новости»)

6 Тестирование программной системы

Задача тестирования ПО нуждается в больших трудозатратах и тщательном применении выбранных подходов. Для успешного внедрения программы в рамках рабочего процесса организации-эксплуатанта необходимо обеспечение полной работоспособности системы в рамках предусмотренных задач, включающее отсутствие критических уязвимостей в процессах, обработку потенциальных ошибок, конфигурацию системы в соответствии с требованиями среды её эксплуатации.

В тестировании системы применимы следующие виды контроля:

Модульное тестирование (module testing, unit testing) – контроль исполнения функций отдельного модуля программной системы.

Тестирование интеграции (integration testing) – контроль обеспечения работоспособности взаимодействия между отдельными функциональными компонентами системы.

Тестирование внешних функций (external functions testing) – контроль взаимодействия системы с применяемыми при эксплуатации функциями внешнего программного обеспечения.

Приемочное тестирование (acceptance testing) – проверка системы на соответствие техническому заданию и требованиям конечного эксплуатанта (заказчика).

Комплексное тестирование (system testing) – контроль и / или испытание полного содержания процесса эксплуатации системы по отношению к исходным целям её создания. При выполнении в искусственной (моделируемой) среде данный вид тестирования следует относить к процессам контроля, тогда как выполнение данного вида тестирования при внедрении на предприятие относится к процессам испытания.

Тестирование настройки (installation testing) – проверка корректности процесса установки системы для эксплуатации.

Для проверки системы могут использоваться статическое и динамическое тестирование. Статическое тестирование предполагает проверку исходного кода программы на наличие структурных дефектов, потенциально приводящих к неисправностям в процессе эксплуатации (корректность применяемой логики, взаимодействия методов и классов). Динамическое тестирование предполагает использование написанного, скомпилированного и запущенного программного кода. Анализируется поведение системы во время выполнения рабочих задач.

К основным принципам организации тестирования относятся:

- Необходимой частью каждого теста должно являться определение предполагаемых результатов работы системы для быстрого устранения выявленных недостатков;
- Следует по возможности избегать тестирования программы ее автором, т.к. процесс реальной эксплуатации системы может значительно отличаться от предполагаемого автором (отладка программы при этом входит в задачи автора программного кода).
- Должны являться правилом доскональное изучение результатов каждого теста, чтобы не пропустить малозаметную на поверхностный взгляд ошибку в программе.
- Необходимо проводить проверку как корректных (предусмотренных) входных данных, так и некорректных (непредусмотренных).
- При анализе результатов каждого теста необходимо проверять, не выполняет ли программа не предусмотренных процессом эксплуатации функций.

7 Оценка производительности системы

Для оценки программного обеспечения была произведена оценка использования ресурсов и быстродействия персонального компьютера разработанной системы.

Для проведения тестирования были выбраны замеры скорости загрузки модуля работы с БД, время загрузки модуля формирования web-страниц, скорость загрузки модуля серверов для разных платформ, а также полное время работы алгоритма анализа данных. Кроме выше озвученных тестов, был проведен замер потребления оперативной памяти и загрузки процессора в заданных конфигурациях.

В соответствии с тестированием нагрузки, были сформулированы минимальные системные требования к серверной части программного обеспечения: процессор не менее 4 ядер/8 потоков и 2.70 ГГц, не менее 8 ГБ ОЗУ.

Таблица 2 – Производительность системы

№	Конфигурация	Загрузка модуля работы с БД, мс	Загрузка модуля формирования web-страницы, мс	Загрузка модуля серверов для разных платформ, мс	Скорость работы алгоритма анализа
1	CPU 3.6 ГГц на 12 потоков, 16 ГБ RAM, Windows 10	4 578	21 367	1 964	3 мин 19 с
2	CPU 2.5 ГГц на 4 потока, 4 ГБ ОЗУ, Windows 10	10 256	34 546	2 566	5 мин 15 с
3	CPU 2.7 ГГц на 8 потоков, 8 ГБ ОЗУ, Windows 10	7 436	27 843	2 344	4 мин 58 с

В результате поведенного тестирования, можно сделать вывод о том, что наибольшее время в работе ИАС занимает модуль формирования web-страниц, предназначенный для визуализации исторических данных и данных проведенного анализа. В соответствии с конфигурацией (табл. 1) был проведен замер потребления ОЗУ и нагрузки на процессор (табл. 2).

Таблица 3 - Нагрузка системы

№	Конфигурация	Потребление ОЗУ	Нагрузка на процессор
1	CPU 3.6 ГГц на 12 потоков, 16 ГБ RAM, Windows 10	693МБ	25%
2	CPU 2.5 ГГц на 4 потока, 4 ГБ ОЗУ, Windows 10	747МБ	79%
3	CPU 2.7 ГГц на 8 потоков, 8 ГБ ОЗУ, Windows 10	645МБ	56%

ЗАКЛЮЧЕНИЕ

В результате прохождения производственной (преддипломной практики) была разработана информационно-аналитическая система для анализа фондового рынка. В работе была обоснована актуальность данной темы, исследована предметная область и проанализированы существующие решения.

Автоматизация рассмотренных методов анализа также позволит бухгалтерам и экономистам различных компаний в автоматическом режиме производить оценку финансовой длительности предприятия. Анализ может быть произведен на основе бухгалтерской отчетности компании.

Также, благодаря оценки влияния новостей на стоимость акции компании, реализуется возможность более глубокого исследования результатов маркетинговой политики компании, а также возможность анализа нетривиальных факторов, влияющих на результаты финансовой деятельности компании.

Кроме того, автоматизация анализа позволит уменьшить время, необходимое для принятия решения по действиям инвестора на фондовом рынке, также автоматизация процесса анализа позволит проводить наиболее полное исследование, с учетом множества факторов и не очевидных зависимостей.

СПИСОК ЛИТЕРАТУРЫ

1. Джек Швагер Технический анализ : полный курс / Джек Швагер. — Москва : Альпина Паблишер, 2020. — 808 с. — ISBN 978-5-9614-6342-2. — Текст : электронный // IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/93060.html — Режим доступа: для авторизир. пользователей

2. Кондратьева Т.Н. Прогнозирование тенденции финансовых временных рядов с помощью нейронной сети LSTM // Интернет-журнал «Науковедение» Том 9, №4 (2019).

3. Кузнецова Н.В. Фундаментальный и технический анализ фондового рынка // Baikal Research Journal, 2019. № 5.

4. Егоров С.Р. Использование мультипликаторов при оценке стоимости компаний различных отраслей // Экономика и бизнес: теория и практика, 2020. № 5-2.

5. Е. А. Федорова, И. С. Демин, О. Ю. Рогов Применение словарей тональности для текстового анализа. - Litres, 2022.

6. Асирян А.К. Оценка тональности новостных сообщений // Научный взгляд в будущее, 2019. № 9.

7. Джексон П. Введение в экспертные системы. - М.: Вильямс, 2021. - 624 с

8. Данелян Т.Я. Формальные методы экспертных оценок // Научно-практический рецензируемый журнал «Статистика и Экономика», 2019. №3.

9. Васюткина, И. А. Разработка клиент-серверных приложений на языке C# : учебное пособие / И. А. Васюткина. — Новосибирск : Новосибирский государственный технический университет, 2016. — 112 с. — ISBN 978-5-7782-2932-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/91508.html (дата обращения: 25.12.2021). — Режим доступа: для авторизир. пользователей

10. Васюткина И.А. Технология разработки объектно-ориентированных программ на JAVA : учебно-методическое пособие / Васюткина И.А.. — Новосибирск : Новосибирский государственный технический университет, 2012. — 152 c. — ISBN 978-5-7782-1973-1. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/45047.html (дата обращения: 25.12.2021). — Режим доступа: для авторизир. пользователей

11. Пирская Л.В. Разработка мобильных приложений в среде Android Studio : учебное пособие / Пирская Л.В.. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2019. — 123 c. — ISBN 978-5-9275-3346-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/100196.html (дата обращения: 25.12.2021). — Режим доступа: для авторизир. пользователей

12. Шацков В.В. Программирование приложений баз данных с использованием СУБД MS SQL Server : учебное пособие / Шацков В.В.. — Санкт-Петербург : Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2015. — 80 c. — ISBN 978-5-9227-0607-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/63638.html (дата обращения: 25.12.2021). — Режим доступа: для авторизир. пользователей

13. Введение в СУБД MySQL : учебное пособие / . — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 228 c. — ISBN 978-5-4497-0912-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/102004.html (дата обращения: 25.12.2021). — Режим доступа: для авторизир. пользователей

ПРИЛОЖЕНИЕ

Приложение 1. Листинг

```
public class EntityFrequency
    {
        public Referent Entity { get; set; }
        public List<int> SentenseIds { get; set; }
        public int MentionsCount { get; set; }
    }

    public class SentimentPredictionModel
    {
        public static string NEGATIVE = "negative";
        public static string POSITIVE = "positive";
        public static string NEUTRAL = "neutral";

        public static string DatasetPath = Path.Combine(Environment.CurrentDirectory, "Models\\SentimentPrediction\\Model", "train.txt");
        public string Text { get; set; }
        public string[] Sentenses { get; set; }
        public List<EntityFrequency> EntitiesFrequency { get; set; }
        protected ITrainerBase trainer;
        public SentimentPredictionModel()
        {
            this.trainer = new OneVersusAllTrainer();
        }

        public void Train()
        {
            Console.WriteLine("*********************************");
            Console.WriteLine($"{trainer.Name}");
            Console.WriteLine("*********************************");

            trainer.Load();
            trainer.Fit(SentimentPredictionModel.DatasetPath);
```

```
            var modelMetrics = trainer.Evaluate();

            Console.WriteLine($"Macro Accuracy: {modelMetrics.MacroAccuracy:#.##}{Environment.NewLine}" +
                              $"Micro Accuracy: {modelMetrics.MicroAccuracy:#.##}{Environment.NewLine}" +
                              $"Log Loss: {modelMetrics.LogLoss:#.##}{Environment.NewLine}" +
                              $"Log Loss Reduction: {modelMetrics.LogLossReduction:#.##}{Environment.NewLine}");
            trainer.Save();
        }

        public List<EntitySentimentPrediction> Predict(string text)
        {
            Console.WriteLine("*********************************");
            Console.WriteLine($"{trainer.Name}");
            Console.WriteLine("*********************************");

            List<SentimentData> sentimentData = new List<SentimentData>();
            string[] sentenses = text.Split(". ", StringSplitOptions.RemoveEmptyEntries);
            List<EntityFrequency> entities = GetTextEntities(sentenses);

            trainer.Load();
            var predictor = new Predictor();
            List<EntitySentimentPrediction> predictions = new List<EntitySentimentPrediction>();

            foreach (EntityFrequency entity in entities)
            {
                SentimentData data = new SentimentData(GetTextWithReplaceEntity(entity, sentenses));

                var prediction = predictor.Predict(data);
                //if (prediction.PredictedLabel != SentimentPredictionModel.NEUTRAL)
                //{
```

```
                    EntitySentimentPrediction entityPrediction = new
EntitySentimentPrediction(entity, prediction);
                    predictions.Add(entityPrediction);
                //}
                Console.WriteLine("-----------------------------");
                Console.WriteLine($"Text: {data.SentimentText}");
                Console.WriteLine($"Prediction: {prediction.PredictedLabel:#.##}");
                Console.WriteLine("-----------------------------");
            }

            return predictions;
        }

        protected List<EntityFrequency> GetTextEntities(string[] sentenses)
        {
            List<EntityFrequency> entitiesFrequency = new List<EntityFrequency>();

            Pullenti.Sdk.InitializeAll();
            for (int i = 0; i < sentenses.Length; i++)
            {
                AnalysisResult result = null;
                try
                {
                    // создаём экземпляр процессора со стандартными анализаторами
                    Processor processor = ProcessorService.CreateProcessor();
                    // запускаем на тексте text
                    result = processor.Process(new SourceOfAnalysis(sentenses[i]));
                }
                catch (Exception e)
                {
                    continue;
                }
                // получили выделенные сущности
                foreach (Referent entity in result.Entities)
                {
```

```
                    if (entity.InstanceOf.Name != "ORGANIZATION" && !(entity.InstanceOf.Name
== "GEO" && ((GeoReferent)entity).IsState))
                        continue;

                    bool needToCreateNewEntityFrequency = true;
                    foreach (EntityFrequency entityFrequency in entitiesFrequency)
                    {
                        if (entity.CanBeEquals(entityFrequency.Entity))
                        {
                            entityFrequency.MentionsCount++;
                            if (!entityFrequency.SentenseIds.Contains(i))
                            {
                                entityFrequency.SentenseIds.Add(i);
                            }
                            if (i > 0 && !entityFrequency.SentenseIds.Contains(i - 1))
                            {
                                entityFrequency.SentenseIds.Add(i - 1);
                            }
                            if (i < sentenses.Length - 1 && !entityFrequency.SentenseIds.Contains(i + 1))
                            {
                                entityFrequency.SentenseIds.Add(i + 1);
                            }
                            needToCreateNewEntityFrequency = false;
                            continue;
                        }
                    }
                    if (needToCreateNewEntityFrequency)
                    {
                        entitiesFrequency.Add(new EntityFrequency
                        {
                            Entity = entity,
                            MentionsCount = 1,
                            SentenseIds = new List<int>() { i }
                        });
                        if (i > 0 && !entitiesFrequency.Last().SentenseIds.Contains(i - 1))
```

```
                    {
                        entitiesFrequency.Last().SentenseIds.Add(i - 1);
                    }
                    if (i < sentenses.Length - 1 &&
!entitiesFrequency.Last().SentenseIds.Contains(i + 1))
                    {
                        entitiesFrequency.Last().SentenseIds.Add(i + 1);
                    }
                }

                Console.WriteLine(entity.ToString());
            }
        }
        Console.WriteLine(entitiesFrequency);
        return entitiesFrequency;
    }

    protected string GetTextWithReplaceEntity(EntityFrequency entityFrequency, string[]
sentenses, string replacer = "XxX")
    {
        string text = String.Empty;
        foreach (int sentenseIndex in entityFrequency.SentenseIds)
        {
            text += sentenses[sentenseIndex];
        }
        foreach (TextAnnotation annotation in entityFrequency.Entity.Occurrence)
        {
            text = text.Replace(annotation.ToString(), replacer);
        }
        return text;
    }

    public void TrainEvaluatePredict(ITrainerBase trainer, SentimentData newSample, bool
needToFit = false)
    {
```

```
            Console.WriteLine("*********************************");
            Console.WriteLine($"{trainer.Name}");
            Console.WriteLine("*********************************");

            trainer.Load();

            if (needToFit)
            {
                trainer.Fit(SentimentPredictionModel.DatasetPath);
                var modelMetrics = trainer.Evaluate();

                Console.WriteLine($"Macro Accuracy: {modelMetrics.MacroAccuracy:#.##}{Environment.NewLine}" +
                                  $"Micro Accuracy: {modelMetrics.MicroAccuracy:#.##}{Environment.NewLine}" +
                                  $"Log Loss: {modelMetrics.LogLoss:#.##}{Environment.NewLine}" +
                                  $"Log Loss Reduction: {modelMetrics.LogLossReduction:#.##}{Environment.NewLine}");
                trainer.Save();
            }

            var predictor = new Predictor();
            var prediction = predictor.Predict(newSample);
            Console.WriteLine("-----------------------------");
            Console.WriteLine($"Prediction: {prediction.PredictedLabel:#.##}");
            Console.WriteLine("-----------------------------");
        }
    }

    public class SentimentData
    {
        [LoadColumn(0)]
        public string SentimentText;

        [LoadColumn(1)]
```

```
    public string Sentiment;

    public SentimentData(string text)
    {
        SentimentText = text;
    }
}

public class SentimentPrediction
{
    [ColumnName("PredictedLabel")]
    public string PredictedLabel { get; set; }
}

public class EntitySentimentPrediction
{
    public SentimentPrediction Prediction;

    public EntityFrequency Entity;

    public EntitySentimentPrediction(EntityFrequency entity, SentimentPrediction
prediction)
    {
        Prediction = prediction;
        Entity = entity;
    }
}

public interface ITrainerBase
{
    string Name { get; }
    void Fit(string trainingFileName);
    MulticlassClassificationMetrics Evaluate();
    void Save();
    void Load();
```

```
        }

        /// <summary>
        /// Base class for Trainers.
        /// This class exposes methods for training, evaluating and saving ML Models.
        /// Classes that inherit this class need to assing concrete model and name; and to implement data pre-processing.
        /// </summary>
        public abstract class TrainerBase<TParameters> : ITrainerBase
            where TParameters : class
        {
            public string Name { get; protected set; }

            protected static string ModelPath => Path.Combine(AppContext.BaseDirectory, "classification.mdl");

            protected readonly MLContext MlContext;

            protected DataOperationsCatalog.TrainTestData _dataSplit;
            protected ITrainerEstimator<MulticlassPredictionTransformer<TParameters>, TParameters> _model;
            protected ITransformer _trainedModel;

            protected TrainerBase()
            {
                MlContext = new MLContext(111);
            }

            /// <summary>
            /// Train model on defined data.
            /// </summary>
            /// <param name="trainingFileName"></param>
            public void Fit(string trainingFileName)
            {
                if (!File.Exists(trainingFileName))
```

```
            {
                throw new FileNotFoundException($"File {trainingFileName} doesn't exist.");
            }

            _dataSplit = LoadAndPrepareData(trainingFileName);
            var dataProcessPipeline = BuildDataProcessingPipeline();
            var trainingPipeline = dataProcessPipeline
                                .Append(_model)

.Append(MlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel"));

            _trainedModel = trainingPipeline.Fit(_dataSplit.TrainSet);
        }

        /// <summary>
        /// Evaluate trained model.
        /// </summary>
        /// <returns>RegressionMetrics object which contain information about model performance.</returns>
        public MulticlassClassificationMetrics Evaluate()
        {
            var testSetTransform = _trainedModel.Transform(_dataSplit.TestSet);

            return MlContext.MulticlassClassification.Evaluate(testSetTransform);
        }

        /// <summary>
        /// Save Model in the file.
        /// </summary>
        public void Save()
        {
            MlContext.Model.Save(_trainedModel, _dataSplit.TrainSet.Schema, ModelPath);
        }

        /// <summary>
```

```
        /// Load Model from the file.
        /// </summary>
        public void Load()
        {
            DataViewSchema schema;
            _trainedModel = MlContext.Model.Load(ModelPath, out schema);
        }

        /// <summary>
        /// Feature engeneering and data pre-processing.
        /// </summary>
        /// <returns>Data Processing Pipeline.</returns>
        private EstimatorChain<NormalizingTransformer> BuildDataProcessingPipeline()
        {
            var dataProcessPipeline = MlContext.Transforms.Conversion.MapValueToKey(inputColumnName: nameof(SentimentData.Sentiment), outputColumnName: "Label")
                .Append(MlContext.Transforms.Text.FeaturizeText("Features", nameof(SentimentData.SentimentText)))
                .Append(MlContext.Transforms.NormalizeMinMax("Features", "Features"))
                .AppendCacheCheckpoint(MlContext);

            return dataProcessPipeline;
        }

        private DataOperationsCatalog.TrainTestData LoadAndPrepareData(string trainingFileName)
        {
            var trainingDataView = MlContext.Data.LoadFromTextFile<SentimentData>(trainingFileName, hasHeader: false, separatorChar: '\t');
            return MlContext.Data.TrainTestSplit(trainingDataView, testFraction: 0.3);
        }
    }
```

```
public class LbfgsMaximumEntropyTrainer : TrainerBase<MaximumEntropyModelParameters>
{
   public LbfgsMaximumEntropyTrainer() : base()
   {
      Name = "LBFGS Maximum Entropy";
      _model = MlContext.MulticlassClassification.Trainers
        .LbfgsMaximumEntropy(labelColumnName: "Label", featureColumnName: "Features");
   }
}

public class NaiveBayesTrainer : TrainerBase<NaiveBayesMulticlassModelParameters>
{
   public NaiveBayesTrainer() : base()
   {
      Name = "Naive Bayes";
      _model = MlContext.MulticlassClassification.Trainers
            .NaiveBayes(labelColumnName: "Label", featureColumnName: "Features");
   }
}

public class OneVersusAllTrainer : TrainerBase<OneVersusAllModelParameters>
{
   public OneVersusAllTrainer() : base()
   {
      Name = "One Versus All";
      _model = MlContext.MulticlassClassification.Trainers
     .OneVersusAll(binaryEstimator: MlContext.BinaryClassification.Trainers.SgdCalibrated());
   }
}

public class SdcaMaximumEntropyTrainer : TrainerBase<MaximumEntropyModelParameters>
```

```
        {
            public SdcaMaximumEntropyTrainer() : base()
            {
                Name = "Sdca Maximum Entropy";
                _model = MlContext.MulticlassClassification.Trainers
                  .SdcaMaximumEntropy(labelColumnName: "Label", featureColumnName: "Features");
            }
        }

        public class SdcaNonCalibratedTrainer : TrainerBase<LinearMulticlassModelParameters>
        {
            public SdcaNonCalibratedTrainer() : base()
            {
                Name = "Sdca NonCalibrated";
                _model = MlContext.MulticlassClassification.Trainers
                  .SdcaNonCalibrated(labelColumnName: "Label", featureColumnName: "Features");
            }
        }
        public class Predictor
        {
            protected static string ModelPath => Path.Combine(AppContext.BaseDirectory, "classification.mdl");
            private readonly MLContext _mlContext;

            private ITransformer _model;

            public Predictor()
            {
                _mlContext = new MLContext(111);
            }

            /// <summary>
            /// Runs prediction on new data.
            /// </summary>
```

```
        /// <param name="newSample">New data sample.</param>
        /// <returns>PalmerPenguinsData object, which contains predictions made by model.</returns>
        public SentimentPrediction Predict(SentimentData newSample)
        {
            LoadModel();

            var predictionEngine = _mlContext.Model.CreatePredictionEngine<SentimentData, SentimentPrediction>(_model);

            return predictionEngine.Predict(newSample);
        }

        private void LoadModel()
        {
            if (!File.Exists(ModelPath))
            {
                throw new FileNotFoundException($"File {ModelPath} doesn't exist.");
            }

            using (var stream = new FileStream(ModelPath, FileMode.Open, FileAccess.Read, FileShare.Read))
            {
                _model = _mlContext.Model.Load(stream, out _);
            }

            if (_model == null)
            {
                throw new Exception($"Failed to load Model");
            }
        }
    }

    public class ModelInput
    {
```

```
        public float Close { get; set; }
        public DateTime Time { get; set; }
    }

    public class ModelOutput
    {
        public float[] ForecastedClose { get; set; }
        public float[] LowerBoundClose { get; set; }
        public float[] UpperBoundClose { get; set; }
    }

    public class ForecastingModel
    {
        const string connectionString = "server=localhost;database=test;user=root;password=1234";

        private string getModelPath()
        {
            string rootDir = Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "../../../"));
            return Path.Combine(rootDir, "MLModel.zip");
        }

        private SsaForecastingEstimator GetForecastingPipline(MLContext mlContext)
        {
            SsaForecastingEstimator forecastingPipeline = mlContext.Forecasting.ForecastBySsa(
                outputColumnName: "ForecastedClose",
                inputColumnName: "Close",
                windowSize: 24,
                seriesLength: 30 * 24,
                trainSize: 6917,
                horizon: 24,
                confidenceLevel: 0.95f,
                confidenceLowerBoundColumn: "LowerBoundClose",
                confidenceUpperBoundColumn: "UpperBoundClose"
```

```
            );

            return forecastingPipeline;
        }

        private SsaForecastingTransformer getForecaster(out MLContext mlContext, out IDataView data, out DatabaseLoader loader)
        {
            string modelPath = this.getModelPath();
            long shareId = 10;
            string query = $@"SELECT close, time FROM candles where shareId = {shareId};";
            SsaForecastingTransformer forecaster = null;

            DatabaseSource source = new DatabaseSource(MySqlConnectorFactory.Instance, ForecastingModel.connectionString, query);
            mlContext = new MLContext();
            MLContext ctx = new MLContext();

            loader = mlContext.Data.CreateDatabaseLoader<ModelInput>();
            DatabaseLoader.Options options = new DatabaseLoader.Options();
            options.Columns = new[] { new DatabaseLoader.Column("Close", DbType.Single, 0), new DatabaseLoader.Column("Time", DbType.DateTime, 1) };
            loader = ctx.Data.CreateDatabaseLoader(options);

            data = loader.Load(source);

            if (File.Exists(modelPath))
            {
                using (var file = File.OpenRead(modelPath))
                {
                    forecaster = (SsaForecastingTransformer)mlContext.Model.Load(file, out DataViewSchema schema);
                }
            }
            else
```

```
            {
                forecaster = this.GetForecastingPipline(mlContext).Fit(data);
                mlContext.Model.Save(forecaster, loader, "MLModel.zip");
            }

            return forecaster;
        }

        public void GetForecast(IConfiguration configuration, diplomContext context, int shareId)
        {
            string rootDir = Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "../../../"));
            string modelPath = Path.Combine(rootDir, "MLModel.zip");
            string connectionString = "server=localhost;database=test;user=root;password=1234";

            MLContext mlContext = new MLContext();

            IDataView data = null, evaluateDate = null;
            //SsaForecastingTransformer forecaster = null;
            ITransformer forecaster = null;
            TimeSeriesPredictionEngine<ModelInput, ModelOutput> forecastEngine = null;
            foreach (Share share in context.Shares.Where(share => share.Id == shareId).ToList())
            {
                DatabaseLoader loader = mlContext.Data.CreateDatabaseLoader<ModelInput>();

                long candlesCount = context.Candles.Where(candle => candle.Share.Id == share.Id).Count();
                if (candlesCount < 30)
                    continue;

                string query = @$"select closeByDay.day as Time, closeByDay.close from (
                    select
                      DATE_FORMAT(Time, '%Y-%m-%d') as day,
```

```
                    CAST(substring_index(group_concat(cast(close as CHAR) order by Time
desc), ',', 1 ) AS REAL) as close
                    from
                     candles
                     where shareId = {share.Id}
                    group by
                     DATE_FORMAT(day, '%Y-%m-%d'), ShareId
                    order by
                     day
                    ) as closeByDay;";
                query = $@"SELECT close, time FROM candles where shareId = {share.Id};";
                DatabaseSource source = new DatabaseSource(MySqlConnectorFactory.Instance,
connectionString, query);
                MLContext ctx = new MLContext();
                DatabaseLoader.Options options = new DatabaseLoader.Options();
                options.Columns = new[] { new DatabaseLoader.Column("Close", DbType.Single,
0), new DatabaseLoader.Column("Time", DbType.DateTime, 1) };
                loader = ctx.Data.CreateDatabaseLoader(options);

                data = loader.Load(source);

                using (var file = File.OpenRead(modelPath))
                {
                   forecaster = mlContext.Model.Load(file, out DataViewSchema schema);
                   //forecastEngine = forecaster.CreateTimeSeriesEngine<ModelInput,
ModelOutput>(mlContext);
                }

                SsaForecastingEstimator forecastingPipeline =
mlContext.Forecasting.ForecastBySsa(
                   outputColumnName: "ForecastedClose",
                   inputColumnName: "Close",
                   windowSize: 120*24,
                   seriesLength: 180*24,
                   trainSize: 6917,
```

```
                horizon: 7*24,
                confidenceLevel: 0.95f,
                confidenceLowerBoundColumn: "LowerBoundClose",
                confidenceUpperBoundColumn: "UpperBoundClose");

            forecaster = forecastingPipeline.Fit(data);

            Evaluate(data, forecaster, mlContext);
            forecastEngine = forecaster.CreateTimeSeriesEngine<ModelInput, ModelOutput>(mlContext);

            mlContext.Model.Save(forecaster, loader, "MLModel.zip");
        }

        if (data != null && forecastEngine != null && forecaster != null)
        {
            forecastEngine.CheckPoint(mlContext, modelPath);
            Forecast(data, 7*24, forecastEngine, mlContext);
        }
    }

    public void Fit()
    {
        SsaForecastingTransformer forecaster = this.getForecaster(out MLContext mlContext, out IDataView data, out DatabaseLoader loader);

        forecaster = this.GetForecastingPipline(mlContext).Fit(data);
        Evaluate(data, forecaster, mlContext);
        mlContext.Model.Save(forecaster, loader, "MLModel.zip");
    }

    public void GetForecast()
    {
        SsaForecastingTransformer forecaster = this.getForecaster(out MLContext mlContext, out IDataView data, out DatabaseLoader loader);
```

```
            TimeSeriesPredictionEngine<ModelInput, ModelOutput> forecastEngine =
forecaster.CreateTimeSeriesEngine<ModelInput, ModelOutput>(mlContext);
            if (data != null && forecastEngine != null && forecaster != null)
            {
                forecastEngine.CheckPoint(mlContext, this.getModelPath());
                Forecast(data, 7, forecastEngine, mlContext);
            }
        }

        private void Evaluate(IDataView testData, ITransformer model, MLContext mlContext)
        {
            IDataView predictions = model.Transform(testData);

            IEnumerable<float> actual =
                mlContext.Data.CreateEnumerable<ModelInput>(testData, true)
                    .Select(observed => observed.Close);

            IEnumerable<float> forecast =
                mlContext.Data.CreateEnumerable<ModelOutput>(predictions, true)
                    .Select(prediction => prediction.ForecastedClose[0]);

            var metrics = actual.Zip(forecast, (actualValue, forecastValue) => actualValue -
forecastValue);

            var MAE = metrics.Average(error => Math.Abs(error));
            var RMSE = Math.Sqrt(metrics.Average(error => Math.Pow(error, 2)));
            Console.WriteLine("Evaluation Metrics");
            Console.WriteLine("---------------------");
            Console.WriteLine($"Mean Absolute Error: {MAE:F3}");
            Console.WriteLine($"Root Mean Squared Error: {RMSE:F3}\n");
        }

        private float[] Forecast(IDataView testData, int horizon,
TimeSeriesPredictionEngine<ModelInput, ModelOutput> forecaster, MLContext mlContext)
```

```
        {
            ModelOutput forecast = forecaster.Predict();

            IEnumerable<string> forecastOutput =
                mlContext.Data.CreateEnumerable<ModelInput>(testData, reuseRowObject: false)
                    .TakeLast(horizon)
                    .Select((ModelInput input, int index) =>
                    {
                        float actualClose = input.Close;
                        float lowerEstimate = Math.Max(0, forecast.LowerBoundClose[index]);
                        float estimate = forecast.ForecastedClose[index];
                        float upperEstimate = forecast.UpperBoundClose[index];
                        return $"Date: {input.Time}\n" +
                            $"Actual Rentals: {actualClose}\n" +
                            $"Lower Estimate: {lowerEstimate}\n" +
                            $"Forecast: {estimate}\n" +
                            $"Upper Estimate: {upperEstimate}\n";
                    });

            Console.WriteLine("Rental Forecast");
            Console.WriteLine("----------------------");
            foreach (var prediction in forecastOutput)
            {
                Console.WriteLine(prediction);
            }
            return forecast.ForecastedClose;
        }
    }


        public void GetWorldnewsImpact(int newsId)
        {
            //WorldNews news = _context.WorldNews.Where(news => news.Id ==
newsId).ToList().First();
            //if (news == null)
```

```
            // return;
            List<WorldNews> allNews = _context.WorldNews.Where(news => news.Id > 29985).ToList();
            List<EntitySentimentPrediction> predictions = new List<EntitySentimentPrediction>();
            foreach (var news in allNews)
            {
               if (news.Text.Contains("articleBody"))
               {
                  string newsTextPattern = "\"articleBody\":\"(.*?)\"";
                  foreach (Match textMatch in Regex.Matches(news.Text, newsTextPattern))
                  {
                     news.Text = textMatch.Groups[1].Value;
                     _context.WorldNews.Update(news);
                  }
               }

               SentimentPredictionModel model = new SentimentPredictionModel();
               predictions = model.Predict(news.Text);
               if (predictions.Count > 0)
               {
                  MorphAnalyzer morph;
                  List<MorphInfo> results = new List<MorphInfo>();
                  try
                  {
                     morph = new MorphAnalyzer(withLemmatization: true);
                     results = morph.Parse(news.Text.Split(' ')).ToList();
                  }
                  catch (Exception)
                  {
                     continue;
                  }
                  var keywords = new List<string>();
                  foreach (var morphInfo in results)
                  {
```

```
                    if (keywords.Count >= 5)
                        continue;
                    var tag = morphInfo.BestTag;
                    if (tag.HasLemma && tag.Power > 0.98)
                    {
                        Console.WriteLine($"{morphInfo.Text}:");
                        Console.WriteLine($"    {tag} : {tag.Power}");
                        keywords.Add(tag.Lemma);
                    }
                }
                List<NewsQuotesImpact> impacts = new List<NewsQuotesImpact>();
                foreach (var prediction in predictions)
                {
                    List<Share> shares = _context.Shares.ToList();
                    List<Candle> candles = new List<Candle>();// _context.Candles.Where(candle
=> candle.Time > news.DateTime.AddDays(-1).Date && candle.Time <
news.DateTime.AddDays(5).Date).ToList();
                    foreach (var share in shares)
                    {
                        if (share.Company == null || share.Company.BrandInfo == null)
                            continue;

                        candles = share.Candles.Where(candle => candle.Time >
news.DateTime.AddDays(-1).Date && candle.Time < news.DateTime.AddDays(5).Date).ToList();
                        if (candles.Count > 0)
                        {
                            Candle firstCandle = candles.First();
                            Candle lastCandle = candles.Last();
                            double? influence = ((firstCandle.Close - lastCandle.Close) * 100) /
(firstCandle.Close);
                            if (influence != null && Math.Abs((double)influence) > 5)
                            {
                                var companyMorphInfo =
morph.Parse(share.Company.BrandInfo.Split(' ')).ToList();
                                var companyKeywords = new List<string>();
```

```
                        foreach (var morphInfo in companyMorphInfo)
                        {
                            if (companyKeywords.Count >= 5)
                                continue;
                            var tag = morphInfo.BestTag;
                            if (tag.HasLemma && tag.Power > 0.5 && tag.Power < 0.7)
                            {
                                Console.WriteLine($"{morphInfo.Text}:");
                                Console.WriteLine($"    {tag} : {tag.Power}");
                                companyKeywords.Add(tag.Lemma);
                            }
                        }
                        impacts.Add(new NewsQuotesImpact(share.Company, news, (double)influence));
                    }
                }
            }
            if (impacts.Count > 0)
            {
                foreach (var keyword in keywords)
                {
                    Keyword dbKeyword = null;
                    if (_context.Keywords.Where(k => k.Value == keyword).Count() > 0)
                    {
                        dbKeyword = _context.Keywords.Where(k => k.Value == keyword).First();
                    }
                    else
                    {
                        dbKeyword = new Keyword(keyword);
                        _context.Keywords.Add(dbKeyword);
                        _context.SaveChanges();
                    }
```

```
                        _context.WorldNewsKeywords.Add(new WorldNewsKeyword(news,
dbKeyword));
                        _context.SaveChanges();
                    }
                    _context.NewsQuotesImpacts.AddRange(impacts);
                    _context.SaveChanges();
                }
            }
        }
    }

    public enum StockRecommendation
    {
        Buy,
        Sell,
        Hold,
        Undefined
    }

    public static class FundamentalAnalysis
    {
        private const double EnterpriseToEbitdaTurningPoint = 3;

        private const double DeptToEBITDAMaxTurningPoint = 2.5;
        private const double DeptToEBITDAMinTurningPoint = 2;

        private const double ForwardPETurningPoint = 5;

        private const double PriceToBookTurningPoint = 1;

        public static StockRecommendation EnterpriseValueToEBITDARatio(Share share)
        {
            if (share.Company == null || share.Company.EnterpriseToEbitda == null)
                return StockRecommendation.Undefined;
```

```
            if (share.Company.EnterpriseToEbitda <= EnterpriseToEbitdaTurningPoint)
                return StockRecommendation.Buy;
            else
                return StockRecommendation.Sell;
        }

        public static double DeptToEBITDARatioValue(Share share)
        {
            if (share.Company == null || share.Company.TotalDebt == null || share.Company.Ebitda
== null)
                return double.NaN;

            return Math.Round((double)(double.Parse(share.Company.TotalDebt.ToString()) /
share.Company.Ebitda));
        }

        public static StockRecommendation DeptToEBITDARatio(Share share)
        {
            if (share.Company == null || share.Company.TotalDebt == null || share.Company.Ebitda
== null)
                return StockRecommendation.Undefined;

            double deptToEBITDARatio = DeptToEBITDARatioValue(share);
            if (deptToEBITDARatio <= 0 || deptToEBITDARatio >=
DeptToEBITDAMaxTurningPoint)
                return StockRecommendation.Sell;
            if (deptToEBITDARatio < DeptToEBITDAMinTurningPoint)
                return StockRecommendation.Hold;
            else
                return StockRecommendation.Buy;
        }

        public static StockRecommendation ForwardPERatio(Share share)
        {
            if (share.Company == null || share.Company.ForwardPE == null)
```

```
            return StockRecommendation.Undefined;
        if (share.Company.ForwardPE <= 0 || share.Company.ForwardPE >= ForwardPETurningPoint)
            return StockRecommendation.Sell;
        else
            return StockRecommendation.Buy;
    }

    public static StockRecommendation GetSummaryRecommendation(Share share)
    {
        int result = 0;
        result += ParseRecommendationToInt(EnterpriseValueToEBITDARatio(share));
        result += ParseRecommendationToInt(DeptToEBITDARatio(share));
        result += ParseRecommendationToInt(ForwardPERatio(share));
        if (result > 0)
            return StockRecommendation.Buy;
        if (result < 0)
            return StockRecommendation.Sell;
        else
            return StockRecommendation.Hold;
    }

    private static int ParseRecommendationToInt(StockRecommendation recommendation)
    {
        switch (recommendation)
        {
            case StockRecommendation.Buy:
                return 1;
            case StockRecommendation.Sell:
                return -1;
            default:
                return 0;
        }
    }
}
```