

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
ИМЕНИ ПАТРИСА ЛУМУМБЫ»**

Факультет физико-математических и естественных наук

Кафедра информационных технологий

«Допустить к защите»

Заведующий кафедрой
информационных технологий

д.ф.-м.н.
Ю.Н. Орлов

«____» _____ 2023Г.

**Выпускная квалификационная работа
магистра**

Направление 02.04.02 «Фундаментальная информатика и информационные технологии»

Магистерская программа «Управление инфокоммуникациями и интеллектуальные системы»

ТЕМА «Применение глубокого обучения для оценки финансовых инструментов»

Выполнил студент **Леонова Алина Дмитриевна**

(Фамилия, имя, отчество)

Группа НФИмд-01-21

Студ. билет № 1032212306

Руководитель выпускной
квалификационной работы

Шорохов Сергей Геннадьевич,
к.ф.-м.н, доцент кафедры ИТ ____
(Ф.И.О., степень, звание, должность)

(Подпись)

Автор _____
(Подпись)

г. Москва

2023 г.

**Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский университет дружбы народов имени Патриса Лумумбы»**

**АННОТАЦИЯ
выпускной квалификационной работы**

Леонова Алина Дмитриевна

(фамилия, имя, отчество)

на тему: Применение глубокого обучения для оценки финансовых инструментов

В выпускной квалификационной работе магистра были рассмотрены вопросы использования методов и средств глубокого обучения для оценки финансовых инструментов. Была разработана архитектура системы финансового информирования и был создан прототип системы, пользовательский интерфейс которого реализован через Telegram-бот. В рамках работы рассматривается вся технологическая цепочка от сбора данных, подготовки и обучения модели до её использования в популярном мессенджере.

Используемый язык программирования: Python.

Автор ВКР

(Подпись)

Леонова А.Д.

(ФИО)

Оглавление

Введение	3
1. Проблемы анализа показателей финансовых инструментов	5
1.1. Основные определения и классификация финансовых инструментов	5
1.2. Характеристики финансовых инструментов	10
1.3. Методы анализа финансовых инструментов	11
1.4. Задача прогнозирования временных рядов	12
1.5. Система финансового информирования	15
1.6. Постановка задачи	17
2. Использование методов машинного обучения	19
2.1. Задачи машинного обучения и методы их решения	19
2.2. Глубокое обучение и нейронные сети	21
2.3. Виды нейронных сетей	27
2.4. Реализация методов машинного обучения	33
2.5. Показатели точности	37
2.6. Предполагаемые процессы в системе финансового информирования	38
3. Разработка системы финансового информирования	42
3.1. Архитектура системы	42
3.2. Подсистема актуализации информации	43
3.3. Подсистема прогнозирования	51
3.4. Подсистема взаимодействия с пользователем	73
3.5. Оценка результатов и перспективы развития	77
Заключение	78
Список используемых источников	79
Приложение А — kurs_table.py	81
Приложение Б — parse_news.py	82
Приложение В — predict_kurs.py	87
Приложение Г — FinAnalysisBot.py	93

Введение

Целью работы является исследование применимости средств глубокого обучения к решению задачи оценки финансовых инструментов, а также разработка прототипа системы финансового информирования, обеспечивающего использование реализованных средств.

Можно выделить множество разных финансовых инструментов, это всевозможные документы, продажа или передача которых обеспечивает получение денежных средств.

Предсказание изменения цены любого финансового инструмента требует индивидуального подхода. Подбор подходящего метода и его характеристик для конкретной задачи на конкретном наборе данных может позволить существенно сэкономить время и дать более точные результаты. Таким образом, отдельные практические задачи требуют дополнительного анализа для определения наиболее эффективных подходов к решению.

Однако, для некоторых задач даже наилучший метод не будет давать хорошей точности предсказания, поскольку на нее влияет слишком много факторов, учесть которые слишком трудозатратно или вовсе невозможно.

Актуальность работы обусловлена наличием огромных массивов данных и желанием наиболее эффективно действовать с учётом экономической ситуации. Данная задача требует анализа огромных массивов накопленных данных, что всё чаще решается с использованием глубокого обучения.

Глубокое обучение (Deep Learning) — это метод машинного обучения с помощью искусственной нейронной сети, математической модели, строящейся на принципах функционирования биологических нейросетей нервных клеток живых организмов, и реализованной в программном и/или аппаратном виде.

Также в работе рассматриваются вопросы разработки кроссплатформенных систем, обеспечивающих быстрый доступ к накопленной информации и результатам прогнозирования.

Структура выпускной работы

Выпускная работа состоит из трёх разделов. В первом разделе рассмотрены проблемы анализа показателей финансовых инструментов: основные определения, общепринятая классификация финансовых инструментов, их ключевые характеристики и методы их анализа.

Второй раздел работы посвящён глубокому обучению: рассмотрены виды нейронных сетей, выделены методы, подходящие для работы с финансовыми инструментами.

В заключительном разделе описана практическая часть работы: разработка структуры системы финансового информирования, создание прототипа системы, реализация автоматизированного сбора данных, создание моделей глубокого обучения, разработка средств совершенствования моделей с использованием технологии обработки естественного языка, разработка Telegram-бота в качестве пользовательского интерфейса, а также дана оценка результата и перспективы развития.

Апробация работы

Часть результатов работы была представлена на конференции «Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems» ИТТММ 2023 (19 апреля 2023 г.) и опубликована в сборнике материалов конференции [1].

Кроме того, была подана заявка на получение свидетельства о государственной регистрации программы для ЭВМ. Название программы «Система финансового информирования FinAnalysisBot».

1. Проблемы анализа показателей финансовых инструментов

Правильная оценка финансовой ситуации, своевременное предсказание значительных изменений на рынке может помочь заработать или не потерять деньги. Данная задача актуальна не только для человека из банковской сферы или человека, работающего с финансами своей компании, но также это актуально для индивидуального инвестора, кем может стать каждый.

1.1. Основные определения и классификация финансовых инструментов

Финансовый инструмент — это финансовый документ (ценная бумага, облигация, дериватив, фьючерс, опцион и другие), продажа или передача которого обеспечивает получение денежных средств. Другими словами, это договоры, в результате заключения которых у одной компании возникают финансовые активы, а у другой финансовые обязательства или долевыми инструментами.

Производный финансовый инструмент или дериватив — договор (контракт) реализации для его сторон прав и/или исполнение обязательств, связанных с изменением цены базового актива, лежащего в основе данного финансового инструмента, и ведущих к положительному или отрицательному финансовому результату для каждой стороны.

Согласно российскому законодательству, к производным финансовым инструментам относятся фьючерсы, форварды, опционы и свопы. Различают биржевые и внебиржевые деривативы.

К биржевым деривативам относят:

- *фьючерс* — биржевой контракт, обязывающий его владельца осуществить (или принять) поставку товара определенного вида, качества и количества по определенной цене в оговоренный момент времени в будущем;

- *форвард* — это контракт на совершение сделки купли-продажи базового актива в будущем по заранее оговорённой цене, обращается на внебиржевом рынке, составляется на договорной основе, базовый актив не стандартизирован [2];

- *опцион* — это контракт, который даёт право (но не налагает обязательства) его покупателю на покупку (в случае, если это опцион-колл) либо на продажу (в случае,

если это опцион-пут) определённого базового актива у продавца (надписана) опциона в течение оговоренного срока исполнения контракта (до даты экспирации) по заранее оговоренной цене (цене-страйк) с уплатой за это право продавцу премии (цены контракта);

- *своп* — это соглашение между сторонами, заключающееся в обмене платежами; по сути, это несколько форвардных контрактов, обязательства по которым возникают с определённой периодичностью.

1.1.1. Классификация финансовых инструментов

Структура и формат классификации финансовых инструментов определены стандартом ISO 10962, утвержденным Международной организацией по стандартизации (ИСО) [3].

Для определения и описания финансовых инструментов для всех участников рынка по единому принципу используется код классификации финансовых инструментов (CFI). Это 6-и символьный код, отражающий тип и основные характеристики финансового инструмента (Таблица 1). Эта договорённость позволяет избежать путаницы, возникающей между разными странами в силу лингвистических различий, а также она упрощает процесс сравнения инструментов между рынками.

Таблица 1— Коды классификации финансового инструмента

1	2	3	4	5	6
Категория	Группа	Атрибут 1	Атрибут 2	Атрибут 3	Атрибут 4

Первый символ отображает категорию, у большинства финансовых инструментов определён первой буквой их английского названия. Второй символ указывает на группу, атрибуты, общие у некоторых финансовых инструментов, так, например, F значит, что финансовый инструмент основан на валюте. Символы с третьего по шестой указывают на атрибуты, различающиеся от группы к группе.

В качестве примера можно привести фрагмент таблицы по опционам из стандарта CFI ISO 10962 версии от 2015 года [4] (Таблица 2), как одной из наименее объёмных категорий [5]:

Таблица 2 — Фрагмент стандарта по опционам

CFI Category (1st Char)	CFI Group (2nd Char)	Attribute - 1 (3rd Char)	Attribute - 2 (4th Char)	Attribute - 3 (5th Char)	Attribute - 4 (6th Char)
O = Listed Options	C = Call Options	Exercise option style 1. A = American 2. E = European 3. B = Bermudan	Underlying asset 1. B = Baskets 2. S = Stock-Equities 3. D = Debt Instruments 4. T = Commodities 5. C = Currencies 6. I = Indices 7. O = Options 8. F = Futures 9. W = Swaps 10. N = Interest Rates 11. M = Others (Misc.)	Delivery 1. P = Physical 2. C = Cash 3. N = Non-Deliverable 4. E = Elect at Exercise	Standard 1. S = Standardized 2. N = Non-Standardized
	P = Put Options				
	M = Others (Misc.)	X = Not Appl./Undefined	X = Not Appl./Undefined	X = Not Appl./Undefined	X = Not Appl./Undefined

Таким образом, у стандартизированного европейского опциона на продажу фьючерсов будет код: OPEFXS.

1.1.2. Список всех категорий с группами

- **Долевые инструменты (Equity, E-**-**-***)**
 - ES — Обыкновенные акции
 - EP — Привилегированные акции (первоочередность выплат)
 - EC — Обыкновенные конвертируемые акции
 - EF — Привилегированные конвертируемые акции
 - EL — Сертификат ограниченного партнерства
 - ED — Депозитарные расписки по акциям
 - EY — Структурированные инструменты
 - EM — Иное
- **Инструменты коллективных инвестиций (Collective investment vehicles, C-**-**-***)**
 - CI — Паевые инвестиционные фонды
 - CH — Хеджированные фонды
 - CV — Инвестиционный фонды (инвестиции в недвижимость)

- CE — Биржевые инвестиционные фонды
- CS — Пенсионные фонды
- CF — Фонды фондов
- CP — Фонды прямых инвестиций
- CM — Иное
- **Долговые инструменты (Debt instruments, D-*-*-*-**)**
 - DB — Облигации
 - DC — Конвертируемые облигации
 - DW — Облигации с прилагаемыми варрантами
 - DT — Среднесрочные ноты
 - DS — Структурированные продукты (с защитой капитала)
 - DE — Структурированные продукты (без защиты капитала)
 - DG — Ипотечные ценные бумаги
 - DA — Ценные бумаги, обеспеченные активами
 - DN — Муниципальные облигации
 - DD — Депозитарные расписки по долговым инструментам
 - DM — Иное
 - DY — Инструменты денежного рынка
- **Права (Rights, R-*-*-*-**)**
 - RA — Бонусные права
 - RS — Право подписки
 - RP — Права на покупку
 - RW — Варранты
 - RF — Сертификаты постоянного кредитного плеча
 - RD — Депозитарные расписки о правах
 - RM — Иное
- **Опционы (Listed options, O-*-*-*-**)**
 - OC — Опцион на покупку
 - OP — Опцион на продажу
 - OM — Иное
- **Фьючерсы (Futures, F-*-*-*-*X)**

- FF — Финансовые фьючерсы
- FC — Товарные фьючерсы
- **Свопы (Swaps, S-*-**-*-*)**

- SR — Своп ставок
- ST — Своп товаров
- SE — Своп акций
- SC — Кредитные свопы
- SF — Валютные свопы
- SM — Иное

● **Не перечисленные и сложные опционы (Non-listed and complex listed options, H-*-**-*-*)**

- HR — На основе ставок
- HT — На основе товаров
- HE — На основе акций
- HC — На основе кредита
- HF — На основе валюты
- HM — Иное

● **Споты (Spot, I-*-*-X-X-*)**

- IF — На основе валюты
- IT — На основе товара

● **Форварды (Forwards, J-*-*-X-*-*)**

- JE — На основе акций
- JF — На основе валюты
- JC — На основе кредита
- JR — На основе ставок
- JT — На основе товаров

● **Стратегии (Strategies, K-*-X-X-X-X)**

- KR — На основе ставок
- KT — На основе товаров
- KE — На основе акций

- КС — На основе кредита
- KF — На основе валюты
- KY — На основе смешанных активов
- KM — Иное
- **Финансирование (Financing, L-**-*-X-*)**
 - LL — Кредит - аренда
 - LR — Сделка РЕПО
 - LS — Кредитование ценными бумагами
- **Референсные инструменты (Reference instruments, T-**-*-*-X)**
 - TC — Валюты
 - TT — Товары
 - TR — Процентные ставки
 - TI — Индексы
 - TB — Корзины
 - TD — Дивиденды по акциям
 - TM — Иное
- **Иное (Others (miscellaneous), M-**-*-M-*)**
 - MC — Комбинированные инструменты
 - MM — Иное

1.2. Характеристики финансовых инструментов

Ключевой характеристикой любого финансового инструмента является цена, доход, скрывающийся за ней. Обычно она и является предметом интереса при прогнозировании. Однако это не единственный показатель, который можно использовать. Для составления общей картины рассмотрим основные характеристики финансовых инструментов:

- *Срок обращения* — отрезок времени до окончательного платежа или требования ликвидации (погашения) финансового инструмента.
- *Ликвидность* — возможность быстрого обналаживания без значительных потерь. Понятие ликвидности прежде всего связывают с фактом обращения актива на рынке независимо от того, это облигация или акция.

- *Доход по инструментам* определяется ожидаемыми процентными, дивидендными выплатами, а также суммами, полученными от погашения или перепродажи финансового актива другим участникам рынка.

- *Номинальная ставка дохода* отражает в денежном выражении доход, полученный от инвестирования средств в денежный актив, абсолютную плату за внедрение средств. Настоящая ставка дохода равна номинальной ставке дохода за вычетом темпов инфляции.

- *Риск финансового инструмента* отражает неопределенность, связанную с величиной и сроком получения дохода в будущем.

- *Делимость* характеризуется минимальным его объемом, который можно купить или продать на рынке.

- *Конвертируемость* — это возможность обмена финансовым инструментом на другие финансовые активы.

- *Механизм налогообложения* определяет, как и по каким ставкам облагаются доходы от владения и перепродажи финансового инструмента.

- *Валюта платежа* — это валюта, в которой производится выплата по финансовому инструменту [6].

1.3. Методы анализа финансовых инструментов

В общем случае можно разделить существующие методы прогнозирования цен актива на три категории:

- *Фундаментальный анализ* — на основе анализа экономики в целом, учитывает рыночных показателей компании, состояние отрасли и в стране, применим в долгосрочной перспективе (на интервалах: недели, месяцы, годы).
- *Технический анализ* — на основе изучения графика изменения цены, выделение фигур технического анализа для предсказания моментов разворота трендов (вымпел, голова и плечи, ромб и другие).

В качестве примера рассмотрим график из свечей с ромбом, появление этой фигуры вероятно приведет к развороту тренда, растущая цена будет падать, а падающая — расти (Рисунок 1).



Рисунок 1— Пример фигуры технического анализа “Ромб”

- *Технологические методы* — с помощью построения математических моделей, моделей машинного обучения, глубокого обучения.

Часто большую ценность несёт не значения конкретных показателей, а их совокупность, позволяющая показать взаимосвязи и изменения. Но далеко не всегда очевидно какие показатели стоит изучать. Предсказание изменения одного показателя по сути является задачей прогнозирования временных рядов.

1.4. Задача прогнозирования временных рядов

Временной ряд — это упорядоченная в хронологическом порядке последовательность значений некоторого показателя, например изменение цены акций, курса валют или драгметаллов. Прогнозирование будущих значений является основной целью анализа временных рядов поскольку результат прогнозирования напрямую связан с планированием дальнейших действий.

1.4.1. Основные компоненты и определения статистического анализа временных рядов:

Тренд — компонента, описывающая долгосрочное изменение уровня ряда.

Сезонность — компонента, описывающая циклические изменения уровня ряда.

Ошибка (random noise) — случайная компонента, описывающая нерегулярные изменения данных.

В качестве примера на рисунке показан временной ряд и выделенные из него: тренд, сезонность и ошибки (Рисунок 2).

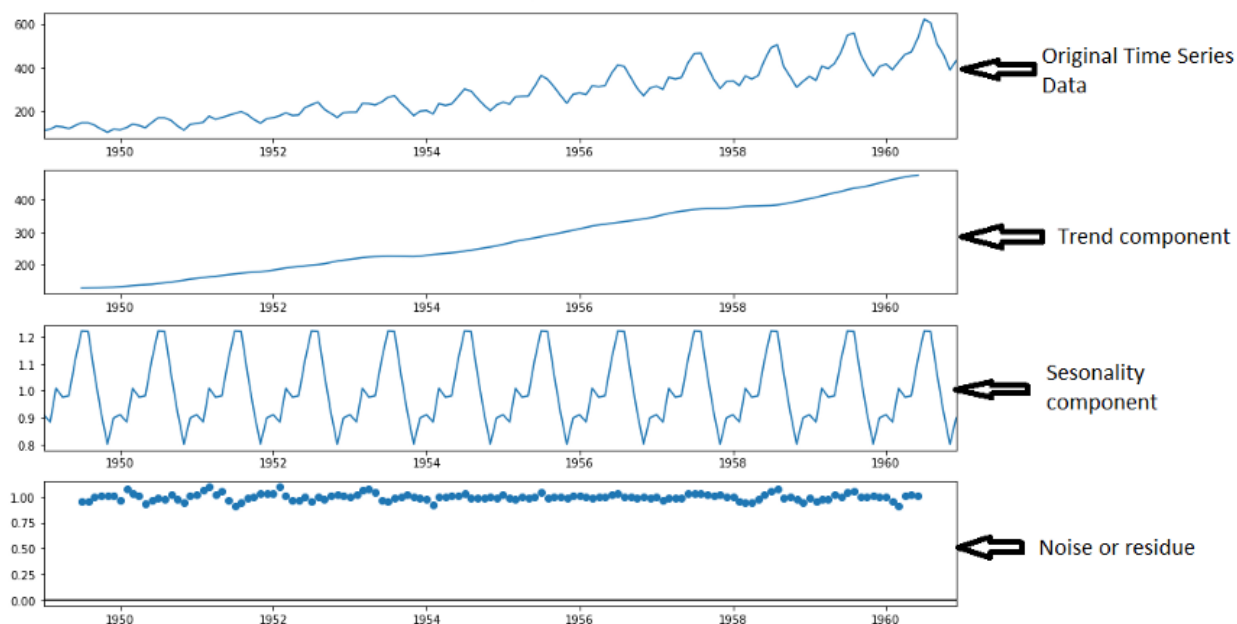


Рисунок 2 — Компоненты временного ряда

Автокорреляция — статистическая взаимосвязь между последовательностями величин одного ряда. Для подсчёта автокорреляции используется корреляция между временным рядом и её сдвинутой копией от величины временного сдвига. Сдвиг ряда называется лагом.

Автокорреляционная функция — график автокорреляции при разных лагах.

Стационарный ряд — ряд, в котором свойства не зависят от времени. С наличием у ряда компонентов: тренд или сезонность, ряд не является стационарным. Выделяют два вида стационарности: строгая и слабая. Слабая стационарность говорит о постоянной дисперсии и о постоянности среднего значения ряда.

Гетероскедастичность — неравномерная дисперсия, неоднородность наблюдений [7].

1.4.2. Основные статистические методы анализа временных рядов

- Вычисление показателей временного ряда, например, *показателя Хёрста* — фрактальной размерности временного ряда, значение этой меры уменьшается, когда задержка между двумя одинаковыми парами значений во временном ряду увеличивается. С помощью него оценивают наличие и устойчивость тренда [8].

- *Модель скользящего среднего (Moving average (MA))* — это вычисление для анализа точек данных путем создания ряда средних значений различных подмножеств полного набора данных, выражается формулой (1).

$$SMA_k = \frac{p_{n-k+1} + p_{n-k+2} + \dots + p_n}{k} = \frac{1}{k} \sum_{i=n-k+1}^n p_i \quad (1)$$

- *Авторегрессионная модель (Auto regression (AR))* — это линейная модель, в которой спрогнозированная величина является суммой прошлых значений, умноженных на числовой множитель (2).

$$X_t = C + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \epsilon_t \quad (2)$$

- *Модель скользящего среднего с авторегрессией (ARMA)* — объединения двух предыдущих моделей, выражается формулой (3). Хорош для описания регулярных стационарных временных рядов [9].

$$X_t = c + \epsilon_t + \sum_{i=1}^p \alpha_i X_{t-i} + \sum_{j=1}^q \beta_j \epsilon_{t-j}, \quad (3)$$

где c — константа, ϵ_t — белый шум, $\alpha_i, \dots, \alpha_p$ и β_j, \dots, β_q — действительные числа, авторегрессионные коэффициенты и коэффициенты скользящего среднего, соответственно.

- *Модель скользящего среднего с авторегрессией и интегрированием (ARIMA)* — это расширение модели ARMA для нестационарных временных рядов. Работает на принципе, что нестационарные ряды можно сделать стационарными взятием разностей некоторого порядка от исходного временного ряда, также их называют интегрированными или разностно-стационарными временными рядами, выражается формулой (4), в которой происходит последовательное взятие d раз разностей (первого порядка, второго и т.д.) сначала от временного ряда, затем от полученных разностей.

$$\Delta^d X_t = c + \epsilon_t + \sum_{i=1}^p \alpha_i \Delta^d X_{t-i} + \sum_{j=1}^q \beta_j \epsilon_{t-j}, \quad (4)$$

где c, α_i, β_j — параметры модели, как в ARMA, Δ^d — оператор разности временного ряда порядка d .

Однако финансовые инструменты зачастую сильно подвержены случайной компоненте и не являются стационарными, потому одни только статистические методы используются редко.

1.5. Система финансового информирования

Существует множество отдельных открытых источников, предлагающих какую-то ограниченную информацию, но не хватает одного места, системы, предоставляющей все интересующие показатели.

Сайты банков обычно предоставляют только актуальные показатели, связанные с предоставляемыми ими услугами, но не динамику их изменений, которую можно было бы изучить. Что-то большее было найдено только на сайте Центрального банка Российской Федерации (cbr.ru), но предлагаемая ими статистика не связана с динамикой финансовых инструментов. Сайты отдельных крупных компаний делятся историей изменения своих активов, можно собирать информацию с сайтов всех интересующих компаний по-отдельности.

Существуют сайты, ориентированные на инвесторов и аналитиков (например, Investing.com), постоянно обновляющиеся и предоставляющие в одном месте полную информацию изменения множества важных показателей, например: цен на драгметаллы и ценные природные ресурсы, курсов валют, криптовалют, индексов, акций компаний, фьючерсов и облигаций, как Российские, так и зарубежные. Однако такие сайты не делятся своим функционалом бесплатно, неавторизованный пользователь может только увидеть котировки и графики динамики изменений, но не может получить список всех значений.

Таким образом, системы финансового информирования облегчают людям жизнь и пользуются спросом, а значит создание подобной доступной системы является актуальной задачей. С учётом сложившейся обстановки в области информационных систем (рост популярности мобильных устройств и альтернативных операционных систем в том числе из-за проблем с лицензированием и вынужденной миграцией корпоративных пользователей на отечественные операционные систему) возросла актуальность кроссплатформенных решений.

1.5.1. Автономные приложения («толстые» клиенты)

Традиционный подход, теряющий популярность в последнее время из-за трудоёмкости применения в разработке кроссплатформенных и мобильных решений. Толстый клиент — это клиент, обеспечивающий полную функциональность и независимость приложения от центрального сервера. Преимущества: высокая функциональность и скорость обработки данных, независимость от подключения к сети и от удаленных серверов. Недостатки: сложно синхронизировать данные, проблемы с обеспечением безопасности данных, большой размер дистрибутива, необходимость в постоянном техническом обслуживании и, поддержке клиентов. Популярные кроссплатформенные средства разработки — Qt, .NET, Electron и т.п.

1.5.2. Тонкие клиенты

Использование приложений, работающих в браузерах. Хорошая кроссплатформенность, распространенные инструменты (Angular и т.п.), недостатки — усложнение работы с извещениями, необходимость разработки как серверной, так и клиентской стороны.

Тонкие клиенты получили широкое распространение с ростом популярности Internet, это программные решения, визуализирующие информацию, обработка которой преимущественно происходит на удалённом сервере. Преимущества: низкие технические требования к оборудованию и минимизирование риска возникновения неисправностей. Недостатки: необходимо подключение к сети, происходящее с одним пользователем может повлиять на работу остальных пользователей на сервере, снижение производительности при нагруженном сервере, перегрузка и остановка работы.

1.5.3. Telegram-бот

Telegram — это популярный мессенджер, кроссплатформенная система мгновенного обмена сообщениями, он ориентирован на безопасность и скорость работы. Первая версия клиента появилась в 2013 году, в 2022 году Telegram вошёл в топ-5 самых загружаемых приложений в мире, а количество его активных пользователей превысило 700 миллионов человек в месяц [10].

Бот — программный модуль, запрограммированный на автономное выполнение определенных задач. Telegram-боты с пользовательской точки зрения являются специальными аккаунтами, которые служат интерфейсом для кода, работающего на сервере. Они являются инструментами, добавляющими функционал в мессенджер, например, они могут информировать о чем-то, к примеру отправлять новости или состояние изменения курса, в заданное пользователем время, при появлении обновления или по запросу пользователя, поддерживать диалог или даже являться текстовой игрой. Один пользователь может создать до 20 ботов.

В своей работе Telegram использует собственный протокол шифрования MTProto (известный также как Telegram API) и предлагает два основных подхода к разработке приложений, взаимодействующих с мессенджером.

TDLib (Telegram Database Library) — кроссплатформенная библиотека базы данных Telegram, позволяет разрабатывать полноценные альтернативные клиенты Telegram, не задумываясь о низкоуровневых деталях реализации, интегрируется с любым языком программирования, поддерживающим выполнение C-функций.

Кроме того, для создания ботов существует интерфейс Telegram Bot API, представляющий собой надстройку над MTProto, работающую на основе HTTP и переводящую http-запросы в вызовы MTProto [11].

Помимо стандартных средств, существует ряд вспомогательных библиотек, упрощающих написание приложений, интегрирующихся с Telegram. Например, *Aioogram* — это асинхронная библиотека для Telegram Bot API, написанная на Python. Асинхронные операции ввода/вывода позволяют программе продолжать выполнение, не дожидаясь результата их исполнения. Как только такая операция завершается, вызывается исполнение функции обратного вызова (callback). Таким образом, можно ускорить работу бота, используя асинхронный код.

1.6. Постановка задачи

С учётом изложенного, цель работы может быть сформулирована как разработка системы финансового информирования, предоставляющей лёгкий доступ к использованию технологии глубокого обучения.

Для этого необходимо решить следующие задачи:

- рассмотреть инструментальные средства, реализующие методы глубокого

обучения;

- выбрать конкретные финансовые инструменты и найти источники с обновляемыми актуальными данными про них;
- подобрать параметры и обучить нейронные сети оценивать выбранные финансовые инструменты, предсказывать их развитие;
- разработать архитектуру системы финансового информирования;
- обосновать выбор средств разработки пользовательского интерфейса системы финансового информирования;
- разработать прототип системы финансового информирования.

2. Использование методов машинного обучения

Большие объёмы данных, необходимых для анализа экономической ситуации, и их многомерная природа всё чаще приводят исследователей к использованию машинного обучения и искусственных нейронных сетей, поскольку это сильно экономит время и даёт более точные результаты, если подобрать подходящий метод для конкретной задачи.

2.1. Задачи машинного обучения и методы их решения

Машинное обучение — обширный подраздел искусственного интеллекта, изучающий методы построения моделей, способных обучаться. Машинное обучение активно использует методы математической статистики и теории вероятностей, методов оптимизации и других классических математических дисциплин, но также является и инженерной дисциплиной, активно использующей различные эвристики, эксперименты на модельных и реальных данных.

Формально можно выделить два типа обучения. Первое из них, дедуктивное, подразумевает перенос в машинную форму знаний экспертов и традиционно относится к области экспертных систем. Второе, индуктивное, к которому фактически и сводится машинное обучение, занимается анализом собранных эмпирических данных (прецедентов) в целях выявления некоторых общих закономерностей [12].

2.1.1. Классификация задач машинного обучения

Наиболее распространённым классом задач является **обучение с учителем (контролируемое)**, которое можно считать обобщением классической задачи аппроксимации функций, и в рамках которого прецеденты образуют пары «объект/ответ», между которыми существует некоторая неизвестная зависимость. Задачей обучения в данном случае является восстановление этой зависимости путём построения алгоритма, обеспечивающего получение достаточно точного классифицирующего ответа для любого возможного входного объекта. В качестве функционала качества обычно берётся средняя ошибка полученных ответов по всем объектам выборки.

В рамках обучения с учителем выделяются следующие задачи (Рисунок 3):

- **классификация:** конечное множество допустимых ответов образует совокупность классов, каждому из которых присвоена своя метка;
- **регрессия:** допустимый ответ представляет собой действительное число либо вектор;
- **ранжирование:** ответы получаются сразу на множестве объектов и далее сортируются; обычно сводится к задачам классификации или регрессии;
- **прогнозирование:** ответы образуют временные ряды, требующие продолжения, т.е. прогноза на будущее; также часто сводится к классификации или регрессии.



Рисунок 3 — Классификация задач машинного обучения

Второй крупный класс задач — **обучение без учителя (неконтролируемое)**, при котором ответы не заданы, т.е. требуется найти зависимости между объектами. Наиболее популярные задачи этой группы:

- **кластеризация:** группировка объектов в кластеры на основе информации об их попарном сходстве; функционал качества задаётся, например, как отношение средних внутрикластерных и межкластерных расстояний;
- **уменьшение размерности:** позволяет свести большое число признаков к меньшему (как правило, 2-3), например, для удобства их последующей

визуализации без потери существенной информации об объектах выборки;

- **выявление аномалий (фильтрация выбросов):** обеспечивает обнаружение небольшого числа нетипичных объектов (например, отсечение шумов или выявление случаев мошенничества).

Частичное обучение объединяет в себе черты обучения с учителем и без учителя. Как и в обучении с учителем, каждому прецеденту соответствует пара «объект/ответ», но ответы известны не для всех прецедентов.

Трансдуктивное обучение подразумевает необходимость предсказания только для прецедентов из тестовой выборки; как правило, сводится к задаче частичного обучения.

При **обучении с подкреплением** для каждого прецедента имеется пара «ситуация/принятое решение», играющая роль объекта. Ответы определяются как значения функционала качества, который оценивает эффективность принятого решения. Таким образом, целью является поиск стратегии действий в некотором окружении для максимизации долговременного выигрыша [13].

2.2. Глубокое обучение и нейронные сети

Глубокое обучение — это метод машинного обучения с помощью искусственной нейронной сети, имитирующей человеческий мозг, состоящий из нейронов, организованных в сеть (Рисунок 4).

Нейронная сеть или просто нейросеть — это математическая модель, строящаяся на принципах функционирования биологических нейросетей нервных клеток живых организмов, и реализованная в программном и/или аппаратном виде.

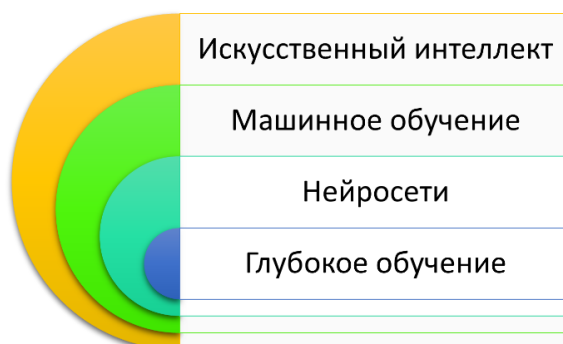


Рисунок 4 — Глубокое обучение и нейронные сети

Они образуются множеством соединений между простыми процессорами (нейронами сети), а обучение сводится к построению оптимальной структуры связей и настройке параметров этих связей. Некоторые нейроны сети представляют собой входы, а некоторые — выходы сети. Задавая значения на входе сети, на выходе получается отклик, таким образом входной вектор преобразуется в выходной (Рисунок 5). Управление процессом преобразования осуществляется путём задания весов связей сети, которые формируются в процессе обучения. Таким образом, обучение позволяет выявить зависимости между входными и выходными векторами, формируя способность сети к прогнозированию.

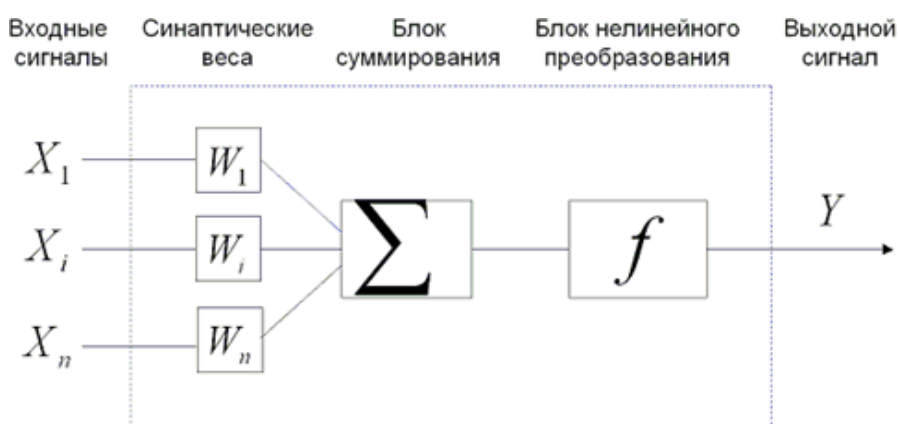


Рисунок 5 — Модель искусственного нейрона

Каждый входной сигнал (inputs) умножается на соответствующий ему вес W_i (weight). Уровень активации нейрона определяется как сумма всех полученных произведений, к которому также может добавляться смещение (bias), можно выразить формулой (5).

$$Y = \sum (weight * input) + bias \quad (5)$$

Далее полученный сигнал поступает на вход функции активации и преобразуется, после чего передаётся на вход нейронов следующего слоя.

2.2.1. Функции активации нейросети

Функция активации определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения. В зависимости от результата функции активации искусственный нейрон считает следует ли это значение исключать (игнорировать) или использовать дальше (активизировать).

Рассмотрим классические функции активации (Рисунок 6):

- *Сигмоида* — нормализует поступающие в неё значения, преобразует в вещественный диапазон $[0, 1]$. Присутствует проблема исчезающего градиента в случае слишком больших положительных либо отрицательных значений. Другая проблема, выходные значения не центрированы нулем, это приводит к тому, что все веса при обновлении либо увеличиваются, либо уменьшаются и градиентный поток становится зигзагообразным. Поэтому исходные данные необходимо предварительно подготавливать так, чтобы получить нулевое среднее значение. Медленная в сравнении с остальными функциями активации. Хорошо подходит для задач классификации благодаря свойству “прижимания” к асимптотам.
- *Гиперболический тангенс* — является скорректированной сигмоидной функцией, она сохраняет те же преимущества и недостатки, но для диапазона значений $(-1;1)$, устраняя тем самым проблему нулевого центрирования. Также производная гиперболического тангенса значительно выше вблизи нуля, давая большую амплитуду градиентному спуску, и, следовательно, более быструю сходимость.
- *ReLU (Rectified Linear Unit)* — возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число. Быстрая в вычислении, количество включаемых нейронов в сетях с большим количеством нейронов будет меньше, чем при использовании сигмоиды или гиперболического тангенса, что сделает сеть легче. Но сохраняются проблемы нулевого центрирования и проблема «умирающего» ReLU, при отрицательных входных значениях веса не будут изменяться во время спуска, из-за чего половина нейронов не будет обновляться.
- *Leaky ReLU (ReLU с “утечкой”)* — борется с проблемой умирающего ReLU, график функции активации на отрицательных значениях образует наклонную, с маленьким угловым коэффициентом (порядка 0,01), что помогает добиться ненулевого градиента. Считать её сложнее, уступает в скорости обычной ReLU.
- *Maxout* — выбирает максимальную сумму из двух наборов весов, умноженных на исходные данные с учётом смещения. Обобщает ReLU и leaky ReLU, не обнуляя градиент, но требует удвоения параметров и нейронов.

- ELU (Exponential Linear Unit) — обладает всеми преимуществами Leaky ReLU, но включает отрицательные значения. Медленно сглаживается до тех пор, пока его выходное значение не достигает $-\alpha$, тогда как ReLU резко сглаживается. Включает вычисление экспоненты, потому работает медленнее ReLU [14].

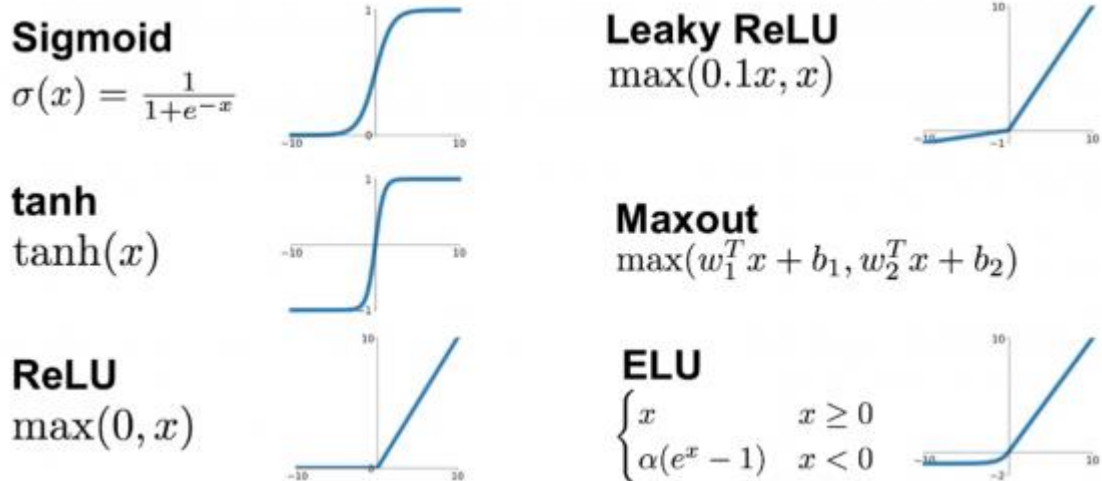


Рисунок 6 — Классические функции активаций

2.2.2. Классификации нейронных сетей

Существует множество признаков, по которым можно выделить классификации:

- *Классификация по способу обучения* — классическая классификация для машинного обучения в целом:
 - ♦ С учителем — контролируемое.
 - ♦ Без учителя — неконтролируемое.
 - ♦ Комбинированные методы.
 - ♦ Обучении с подкреплением.
- *Классификация по количеству слоев*. У любой нейросети есть первый слой — входной слой, получающий входные сигналы и распределяющий их по остальным нейронам, а дальнейшая структура может отличаться, например, по количеству слоев:
 - ♦ Однослойная нейросеть — сигналы со входного слоя направляются сразу на выходной, он преобразует сигналы и выдает ответ.
 - ♦ Многослойная нейросеть — помимо входного и выходного есть промежуточные слои, на каждом из которых может осуществляться особая обработка и распределение сигналов.

- *Классификация по топологии* — по количеству и структуре соединений между нейронами:
 - ♦ Полносвязные (Рисунок 7 а) — все нейроны передают свои выходные сигналы всем нейронам, включая себя. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.
 - ♦ Многослойные (Рисунок 7 б) — помимо входного и выходного есть промежуточные слои
 - ♦ Слабосвязные (Рисунок 7 в) — нейроны располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с четырьмя в окрестности фон Неймана (слева на рисунке), шестью в окрестности Голя или восемью в окрестности Мура (справа на рисунке) своими ближайшими соседями [15].

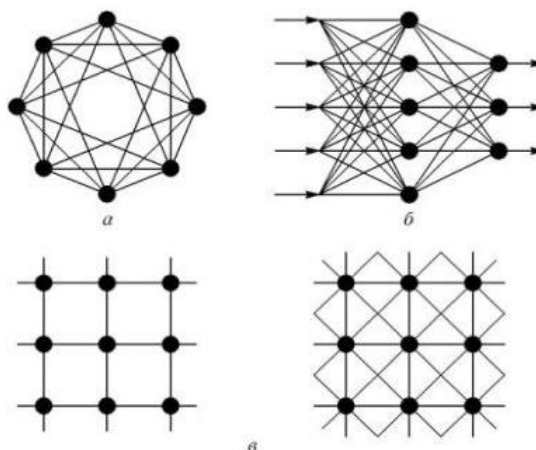


Рисунок 7 — Классификация нейросетей по топологии

- *Классификация по направлению распределения информации по синапсам между нейронами:*
 - ♦ Однонаправленная нейросеть / Нейросеть прямого распространения — сигнал перемещается строго по направлению от входного слоя к выходному, в обратном направлении движение невозможно.
 - ♦ Нейросеть с обратными связями / Рекуррентная нейросеть — сигнал двигается и в прямом, и в обратном направлении. В итоге результат выхода способен возвращаться на вход. Выход нейрона определяется весовыми

характеристиками и входными сигналами, а также дополняется предыдущими выходами, снова вернувшись на вход.

- *Классификация по способу настройки весовых коэффициентов:*
 - ◆ Фиксированные коэффициенты — весовые коэффициенты выбираются сразу исходя из условий задачи.
 - ◆ Динамические коэффициенты — меняются в процессе обучения.
- *Классификация по видам задач:*
 - ◆ Регрессия — для прогнозирования числовых значений по помеченным данным. Используется, например, для составления прогнозов изменения курсов на бирже, для определения возраста по фотографии, для оценки стоимости недвижимости и многие другие задачи.
 - ◆ Прогнозирование временных рядов — для составления долгосрочных прогнозов на основе динамического временного ряда значений. Например, нейросети применяются для предсказания динамики изменения цен акций, для прогнозирования изменения физических явлений и множества других показателей.
 - ◆ Классификация — для распознавания классов объектов, выбора конкретного объекта из четко заданного множества объектов, например, распознавание лиц, эмоций, образов. Применяется только для размеченных данных.
 - ◆ Кластеризация — для изучения и сортировки большого объема неразмеченных данных в условиях, когда неизвестны классы и их количество. Кластеризация применяется для объединения данных по признакам и выявления классов объектов, сегментации данных по выделенным признакам.
 - ◆ Генерация — для автоматизированного создания или преобразования данных. Генерация с помощью нейросетей применяется для создания уникальных текстов, аудиофайлов, видео, для улучшения качества фотографий и раскрашивания черно-белых фильмов.

2.3. Виды нейронных сетей

2.3.1. Персептрон

Одной из первых моделей нейронных сетей, заложившей основ всей этой теории, стал персептрон, созданный Фрэнком Розенблаттом с целью построения модели мозга [16]. Передающая сеть, образующая персептрон, включает генераторы сигнала трёх типов: сенсорных (S-элементы, вырабатывающие сигналы на основе внешнего воздействия), ассоциативных (A-элементы, выдающие сигнал, когда сумма входов превышает пороговое значение) и реагирующие (R-элементы, выдающие +1, если сумма входов положительная, и -1, если сумма входов отрицательна). Схема элементарного персептрона представлена на рисунке 8.

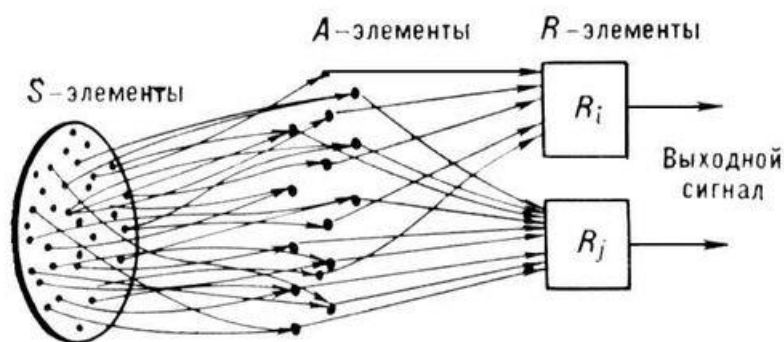


Рисунок 8 — Схема элементарного персептрона

2.3.2. Рекуррентные нейронные сети

Рекуррентные нейронные сети – это вид нейронных сетей, которые позволяют обрабатывать последовательности данных, в том числе временные ряды. Основная особенность такого рода сетей заключается в том, что связи между элементами сети представляют собой направленную последовательность.

Рекуррентные нейросети неограниченны за счет постоянного перезаписывания самих себя, потому они подходят для обработки последовательности произвольной длины, в то время как многослойный перцептрон ограничен размером входного слоя, увеличение которого приводит к падению производительности вычислений.

Среди рекуррентных сетей наиболее часто применяется архитектуры LSTM и GRU.

Обработка связной последовательности данных подразумевает необходимость учитывать связи и порядок элементов последовательности данных. То есть сеть должна «помнить» элементы последовательности данных, которые были переданы на нее ранее. Для решения данной задачи сеть помимо вектора входных данных должна также иметь вектор внутреннего состояния, который отражает «память» о ранее переданных в сеть элементах.

2.3.3. Нейронные сети долгой краткосрочной памяти (LSTM)

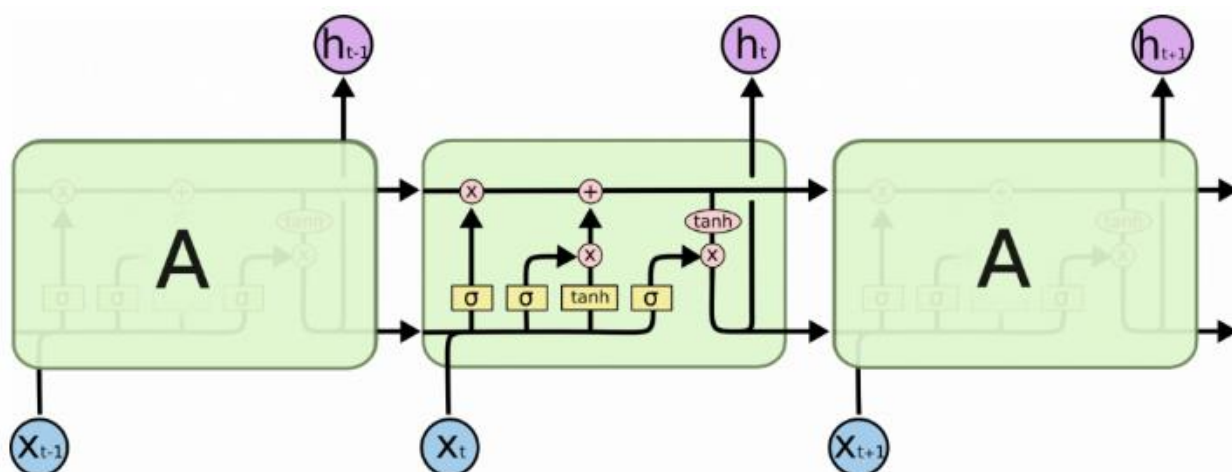
При обучении традиционных рекуррентных нейронных сетей информация в памяти смешивается с новой, и вскоре ранняя информация полностью перезаписывается. LSTM-модули разработаны специально, чтобы избежать проблемы долговременной зависимости. Простыми словами, она сохраняет часть данных, которые сочла важными.

LSTM (Long short-term memory) реализуется за счёт цепной структуры и содержит 4 слоя нейронной сети, желтые прямоугольники на схеме (Рисунок 9), их также называют воротами. Внутри каждого слоя происходит: умножение входного вектора на матрицу весов слоя W , прибавление к этому сдвига b , и применение к результату функции активации нейронов (сигмоида σ (формула (6)) или гиперболический тангенс \tanh (7)):

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f), \quad (6)$$

$$C_t = \tanh(W_c * [h_{t-1}, x_t] + b_c). \quad (7)$$

Рассмотрим схему, X — вектор входных данных, h — предсказание, t — номер слоя LSTM, состояние памяти модели визуализировано горизонтальной линией в верхней части схемы (также память порой называют ячейкой).



Сначала данные проходят через первый слой с сигмоидной функцией активации, она решает, какие элементы нужно забыть, какие использовать дальше, формируется оценочный вектор (со значениями от 0 до 1). Оставшиеся данные идут дальше, следующий слой сигмоидной функцией активации формирует новый оценочный вектор, а слой с гиперболическим тангенсом формирует вектор новых значений. Далее эти два вектора перемножаются, таким образом, избавляются от части новых значений, отфильтровывают лишнее. После происходит изменение вектора памяти, забываются лишние значения путем перемножения вектора памяти на первый оценочный вектор, а новые взвешенные значения добавляются к вектору памяти (умножение и плюс в красных кружках на верхней стрелке). Измененный вектор памяти поэлементно проходит через функцию гиперболического тангенса. Последний слой с сигмоидной активацией строит оценочный вектор, определяя какие части вектора памяти важны. И наконец, преобразованный вектор памяти умножается на оценочный вектор, результат и будет предсказанием ht . А дальше всё будет рекуррентно повторяться [17].

2.3.4. Свёрточные нейронные сети

Свёрточные нейросети (Convolutional neural network, CNN) произвели революцию в распознавании образов и компьютерном зрении, кроме того, они успешно используются и для других сложных для компьютера человеческих задач, таких как распознавание речи и анализ текстов. Нередко свёрточные нейронные сети называют наиболее успешной моделью глубокого обучения.

Они появились в результате изучения особенностей зрительной коры головного мозга. Эксперименты Х. Дэвида и В. Торстена на кошках и обезьянах показали, что многие нейроны в зрительной коре имеют небольшое локальное рецепторное поле, а потому реагируют только на зрительные раздражители, находящиеся в ограниченной области поля зрения. Также одни нейроны реагируют исключительно на изображения вертикальных линий, а другие на линии с различными направлениями.

Таким образом, рецепторные поля разных нейронов в совокупности охватывают всё поле зрения. Кроме того, рецепторные поля нейронов могут быть

разных размеров, нейроны с большим рецепторным полем реагируют на более сложные образы. Это привело к мысли, что существуют разные уровни нейронов и что нейроны более высокого уровня получают и учитывают выводы нейронов более низкого уровня (Рисунок 10).

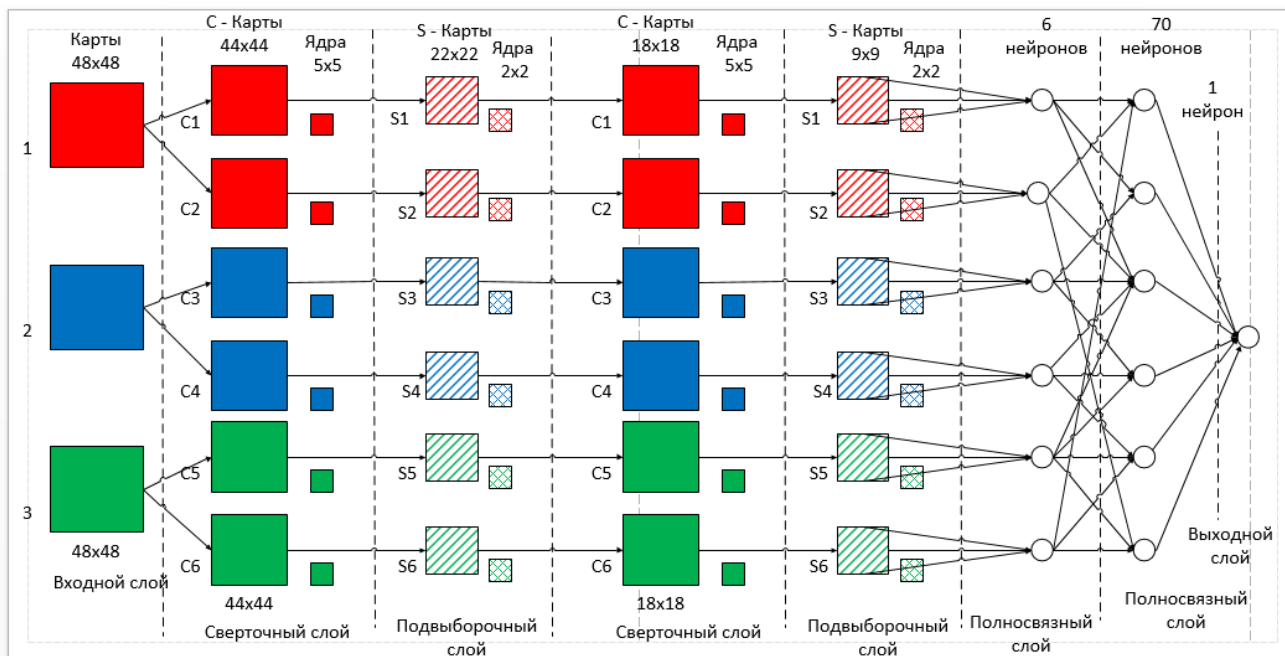


Рисунок 10 — Пример топологии сверточной нейронной сети

Особенность свёрточной нейронной сети заключается в её принципиальной многослойности: нейроны в ней, как в зрительной коре, сгруппированы в слои разных типов, при этом нейроны каждого слоя в свою очередь обладают своими особенностями и только некоторые из них связаны некоторыми другими из соседних слоёв, реагируя на различные особенности изображений [13].

В **свёрточных** слоях активация нейронов следующего уровня формируется из линейной комбинации активаций нейронов предыдущего уровня за счёт умножения фрагмента на матрицу свёртки; результат каждой свёртки подаётся на вход некоторой нелинейной функции активации, фактически представляющей собой слой активации; обычно слой активации явно не выделяется, а логически объединяется со свёрточным слоем. Ядро свертки представляет собой матрицу весов, которая применяется для различных нейронов предыдущего слоя, и которую можно интерпретировать как кодирование признака, например, наличие линии под определенным углом. Следующий слой, получивший сигналы от матрицы свёртки, представляет наличие признака в обрабатываемом слое, а также его координаты,

таким образом формируется карта признаков (Рисунок 11). Ядра свёртки формируются сетью самостоятельно в процессе обучения методом обратного распространения ошибки. Каждое ядро свёртки формирует свою независимую карту признаков, поэтому такая сеть является многоканальной.

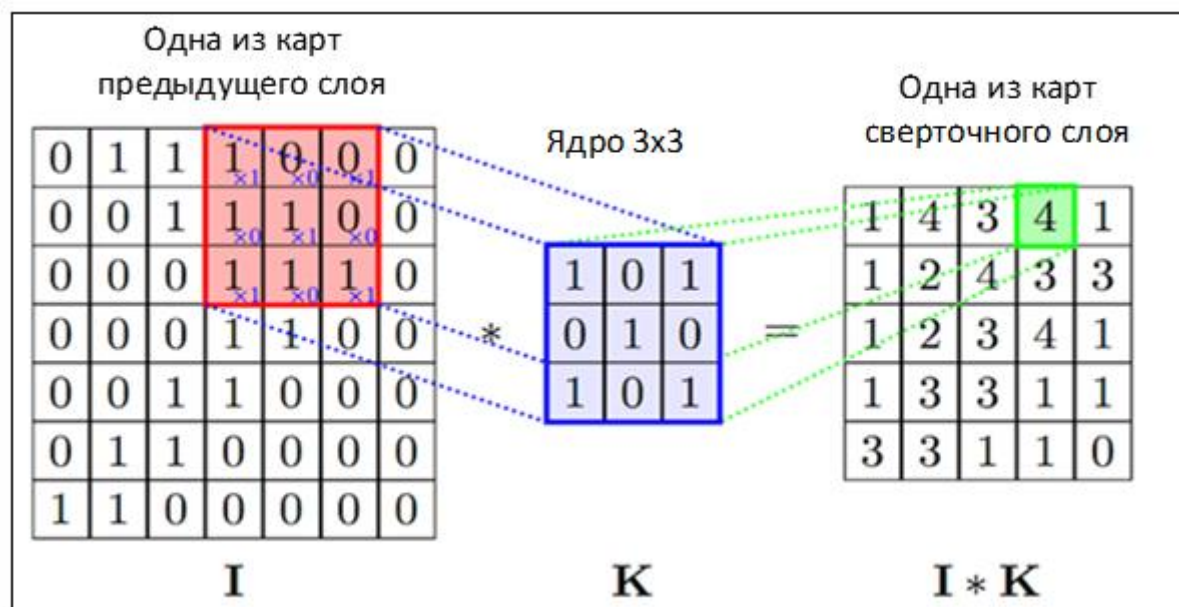


Рисунок 11 — Операция свертки и получение значений карты признаков

В **субдискретизирующих** слоях активация нейронов следующего уровня воспроизводит активацию нейронов предыдущего уровня, но размеры изображения уменьшаются за счёт процедуры пулинга (подвыборки или субдискретизации), заключающейся в замене активации рядом расположенных нейронов на их максимум или их среднее. Данная операция призвана уменьшить размер карт признаков в целях ускорения дальнейших вычислений и позволяет добиться необходимой производительности при больших значениях первого (входного) слоя. Принято считать, что для данной архитектуры информация о наличии признака более важна, чем точность его координат в исходном изображении. Поэтому среди некоторой зоны соседних нейронов карты признаков определяется путём нелинейного преобразования новое результирующее значение (например, максимальный или среднее арифметическое) и переносится на новую уменьшенную карту признаков, это позволяет уменьшить размерность карты признаков (Рисунок 12).

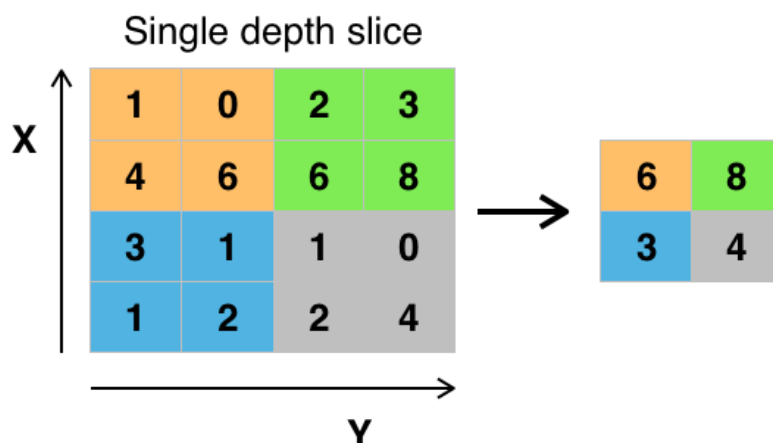


Рисунок 12 — Пример субдискретизации с функцией максимума

Данные, выходящие из свёрточной сети, передаются полносвязному слою (или полносвязной сети, которая может в свою очередь состоять из нескольких слоёв), где осуществляется окончательная классификация самых абстрактных понятий, выявленных свёрточной сетью из исходного изображения.

Структура сети состоит из большого количества слоев (Рисунок 13). Из первого (входного) слоя сигнал активации проходит набор сверточных слоев, в которых происходит чередование свертки и субдискретизации (пулинг). Это позволяет уменьшать размерность карт признаков, увеличивая количество каналов, то есть возникает возможность распознавания сложных иерархий признаков с разным уровнем абстракции.

После сверточных слоев дополнительно устанавливается многослойный перцептрон, на вход которого передаются модифицированные карты признаков небольших размерностей.

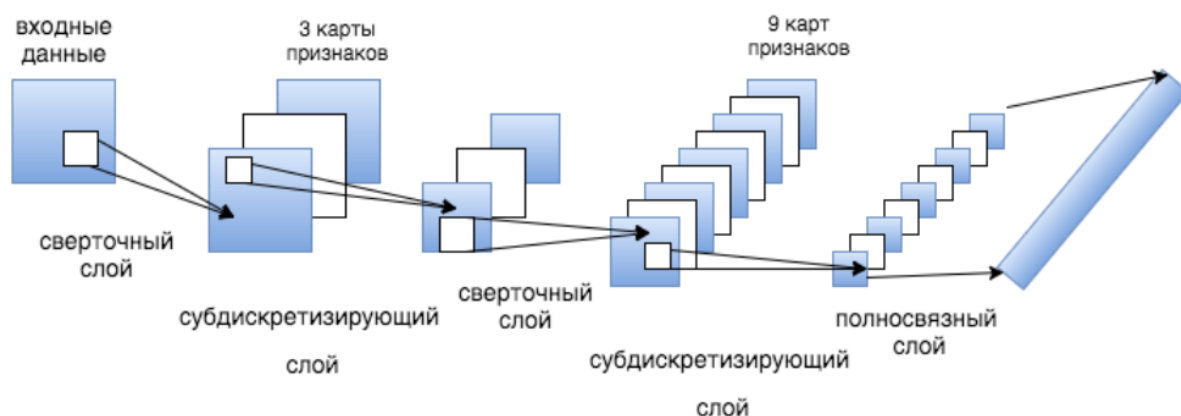


Рисунок 13 — Архитектур свёрточной сети

Можно описать суть работы сверточной нейронной сети как переход от конкретных признаков к признакам более высокого уровня абстракции. Сеть в процессе обучения формирует иерархию признаков и производит фильтрацию признаков по значимости. Данная интерпретация носит иллюстративный характер, так как на практике признаки, выделяемые сетью, тяжело поддаются интерпретации человеком.

При обучении свёрточных сетей используются традиционные методы обучения нейронных сетей, как правило, метод обратного распространения ошибки. Для уменьшения эффекта переобучения сети популярно применение метода дропаута (исключения), заключающийся в случайном исключении определённой доли нейронов на разных эпохах обучения. Это приводит к приданию обученным нейронам большего веса и значительному увеличению скорости и качества обучения.

2.4. Реализация методов машинного обучения

При выборе языка программирования для реализации методов машинного обучения были рассмотрены современные тенденции и исследования, сравнивающие преимущества использования того или иного языка для актуальных задач. В результате выяснилось, что во всех рейтингах языков программирования последних лет одну из лидирующих позиций всегда занимает Python. Например, IEEE Spectrum в своих рейтингах уже который год ставит его на первую строчку [18].

Python — это высокоуровневый, динамически типизированный и платформенно независимый язык программирования с открытым исходным кодом (open-source). Он был создан в 1991 году и широко используется для анализа данных. Благодаря активному сообществу для Python разрабатывается множество разнообразных библиотек и инструментов, в том числе для машинного обучения. В целях ускорения работы многие из данных библиотек используют средства аппаратного ускорения, в первую очередь ориентирующиеся на графические ускорители GPU (Graphics Processing Unit) и технологию CUDA (Compute Unified Device Architecture — архитектура параллельных вычислений), используемую в GPU-ускорителях компании Nvidia.

Далее рассмотрены наиболее популярные средства машинного обучения, их особенности и функционал, предположительно подходящий мне для дальнейшей работы.

2.4.1. Инструменты машинного обучения Python

Разница между API, фреймворком и библиотекой

API (application programming interface) — это интерфейс взаимодействия с программной системой извне. В API описаны методы взаимодействия с внешними программами. Использование API даёт приложению возможность обратиться к коду вне этого приложения.

Библиотека — это набор готовых функций для решения некоторых типичных задач. При подключении библиотеки она становится частью приложения.

Фреймворк — это каркас для будущего приложения. Это заготовка, включающая в себя: начальный код, структуру, библиотеки, полезные для некоторой задачи. Фреймворки используются для облегчения разработки и объединения разных компонентов большого программного проекта.

Популярные библиотеки

Shogun — открытая (open-source) библиотека машинного обучения с фокусировкой на SVM (Support Vector Machines). Написана на C++. Поддерживаемые языки: Python, Octave, R, Java/Scala, Lua, C#, Ruby, etc.

PyTorch — фреймворк для глубокого машинного обучения с открытым исходным кодом. Написана на Python, C++, CUDA. Используется для решения задач компьютерного зрения и обработки естественного языка. Опиерирует тензорами, под которыми в данном контексте понимаются массивы, которые могут обрабатываться на GPU.

Theano — библиотека с открытым исходным кодом для глубокого обучения и численного вычисления. Написана на Python и CUDA. Основные возможности: интеграция с NumPy, параллельные вычисления, прозрачное использование ресурсов CPU (Central Processing Unit — центральный процессор) и GPU [19].

TensorFlow — библиотека с открытым исходным кодом, предоставляющая возможности по построению и тренировке нейронных сетей, ориентированных на автоматическое нахождение и классификацию образов. Написана на: Python, C++,

CUDA. Особенность в возможности использования аппаратного ускорителя собственной разработки — тензорного процессора (TPU) на основе специализированной интегральной схемы, что даёт на порядок лучшие показатели работы. Используется по умолчанию в основе работы рассмотренного далее Keras (заменила Theano).

Keras — это высокоуровневый API глубокого обучения, написанный на Python, работающий поверх фреймворка машинного обучения TensorFlow, Theano или другого.

Caffe (Convolution Architecture For Feature Extraction, Свёрточная архитектура для извлечения признаков) — это фреймворк для глубокого обучения, созданный в первую очередь для решения задач классификации и сегментации изображений. Написан на C++.

LIBSVM (A Library for Support Vector Machines) — библиотека машинного обучения с открытым исходным кодом, реализующая в первую очередь методы опорных векторов. Написана на Java и C++. Есть интерфейсы для Python, R, MATLAB, Perl, Ruby и других языков.

XGBoost (A Scalable Tree Boosting System, Масштабируемая система повышения качества дерева) — это оптимизированная распределенная система повышения градиента, разработанная для обеспечения повышения эффективности, гибкости и портативности. Она реализует алгоритмы машинного обучения в рамках платформы Gradient Boosting. XGBoost обеспечивает усиление параллельного дерева (GBDT, GBM).

ONNX — это кроссплатформенный ускоритель логического вывода и обучения, совместимый с популярными фреймворками ML/DNN, включая PyTorch, TensorFlow/Keras, scikit-learn и другие. Обучение происходит на Python, но использоваться может в приложениях на C#, C++ и Java.

Scikit-Learn (sklearn) — библиотека с открытым исходным кодом, широко используемая в задачах анализа данных и машинного обучения. Она основана на NumPy и SciPy [20].

Таблица 3 — Сравнение фреймворков машинного обучения

Фреймворк	Основное предназначение	Написан на языках	Поддерживаемые языки	Специальный конвертер в CoreML	Скорость работы	Возможность ускорения	Поддержка нейронных сетей
CreateML	упрощает разработку моделей	Swift	Swift	CreateML	++	GPU (BNNS, Metal CNN)	+
Turi Create	упрощает разработку моделей	Python, C++	Python, C++	Turi Create	+	GPU	-
Keras	работа поверх других фреймворков МО	Python, C++	Python, C++	Keras	+	GPU	+
TensorFlow	классификация образов	Python, C++	Python, C++	TF-coreml	+	GPU, TPU	+
ONNX	глубокое обучение и ускорение логического вывода	Python, C++	Python, C#, C++, Java	—	+	GPU	+
Scikit-learn	для взаимодействия с числовыми и научными библиотеками NumPy и SciPy	Python, Cython, C, C++	Python, C++	—	-	—	+
XGBoost	повышение качества дерева	Python, C++	Python, C++	—	+	GPU	-
PyTorch	глубокое обучение	Python, C++	Python, C++	—	++	GPU	+
LibSVM	методы опорных векторов	Java, C++	Python, R, MATLAB, Perl, Ruby и др.	—	-	—	-
Caffe	классификация и сегментация изображений	C++	Python, C++, MATLAB	Caffe	+	GPU	+
Torch	для глубинного обучения и научных расчётов	C, C++, Lua	C++, Lua	Torch2CoreML	+	GPU	+

Основные возможности Tensorflow и Keras

Keras — предназначенная для построения нейронных сетей библиотека с открытым исходным кодом, реализованная на языке Python, первая версия которой создавалась как надстройка над фреймворками DeepLearning4j, TensorFlow и Theano. Она содержит реализации всех необходимых компонентов нейросетей: слоёв, целевых и передаточных функций, оптимизаторов. Также Keras предоставляет высокоуровневый набор абстракций, упрощающий процесс формирования нейронных сетей [21].

TensorFlow является системой машинного обучения, разработанной Google Brain, она была открыта для свободного доступа в 2015 года. TensorFlow поддерживает распараллеленную работу как на CPU, так и на GPU (используя архитектуру CUDA) [22].

Кроме того, поддерживается специализированный тензорный процессор (TPU) — нейронный процессор со специализированной интегральной схемой, разработанной Google и обеспечивающей по сравнению с графическими процессорами более высокую производительность для больших объёмов вычислений со сниженной точностью.

2.5. Показатели точности

2.5.1. Метрики для задачи предсказания временных рядов

Показатели точности для оценки результатов прогнозов моделей (n — количество значений, a — реальные данные, p — предсказание):

- explained variance score — объясненную дисперсию (вариацию), чем она выше, тем лучше модель способна объяснить вариацию данных, выражается формулой (8);

$$EV = 1 - \frac{Var(a - p)}{Var(a)} \quad (8)$$

- mean squared log error — среднее логарифмическое квадратичное отклонение, выражается формулой (9);

$$EMSLE = \frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2 \quad (9)$$

- mean squared error — средний квадрат отклонения (10);

$$MSE = \frac{1}{n} \sum_{i=1}^n (p_i - a_i)^2 \quad (10)$$

- root mean squared error — корень среднего квадрата отклонения;
- mean absolute error — среднюю абсолютную ошибку;
- median absolute error — медианную абсолютную ошибку;
- max error — максимальную ошибку [23].

2.5.2. Метрики в задачах классификации

Матрица ошибок, для двух классов 0 и 1, где a — реальные значения, p — предсказание (Таблица 4).

Таблица 4 — Матрица ошибок

	$a = 1$	$a = 0$
$p = 1$	True Positive (TP)	False Positive (FP)
$p = 0$	False Negative (FN)	True Negative (TN)

- Accuracy — доля правильных ответов алгоритма, имеет смысл для задач с равным распределением классов, формула (11).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

- Precision — точность, доля объектов, положительно определённых положительными классификатором из всех объектов (12).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

- Recall — полнота, доля объектов, положительно определённых положительными классификатором из всех объектов положительного класса (13).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (13)$$

- F-мера — способ объединения precision и recall (14), при $\beta = 1$ это среднее гармоническое двух метрик.

$$F_{\beta} = \frac{\text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}} \quad (14)$$

2.6. Предполагаемые процессы в системе финансового информирования

С учётом рассмотренных методов и средств машинного обучения можно сформулировать следующее предварительное описание ожидаемого поведения предполагаемой системы финансового информирования и взаимодействия её ключевых процессов. Общая диаграмма системы отображает подсистемы и их связи с роботом (автоматизированный процесс, который может управляться администратором) и пользователями (Рисунок 14).

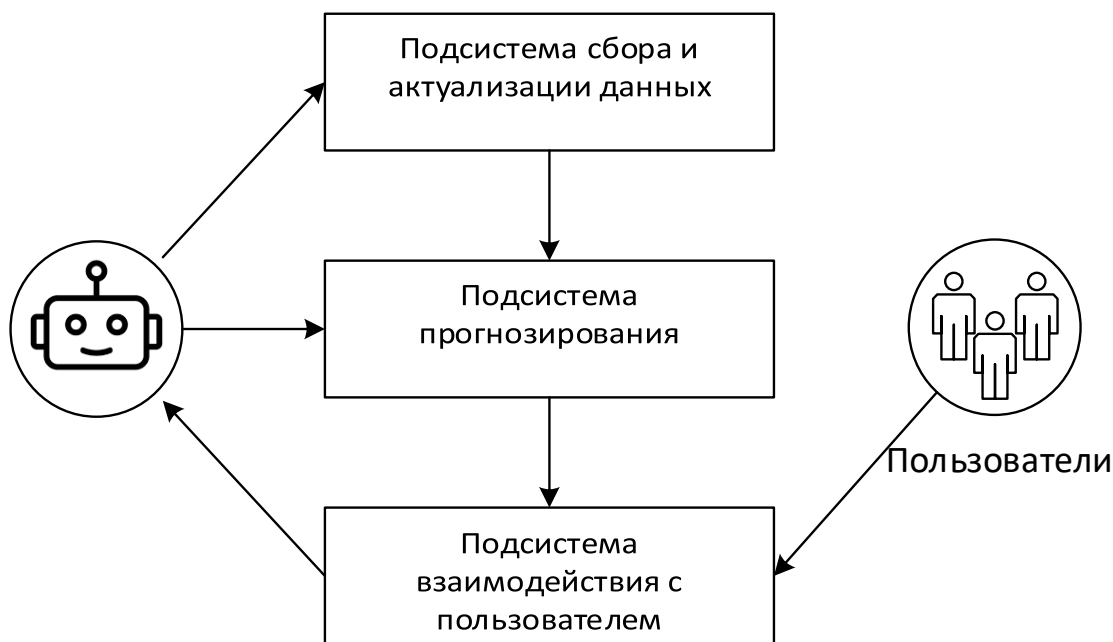


Рисунок 14 — Общая диаграмма системы финансового информирования

Далее рассмотрим предполагаемое поведение всех подсистем.

В подсистеме сбора и актуализации данных робот инициирует процесс с некоторой периодичностью, например, раз в день, или по запросу пользователя (Рисунок 15).

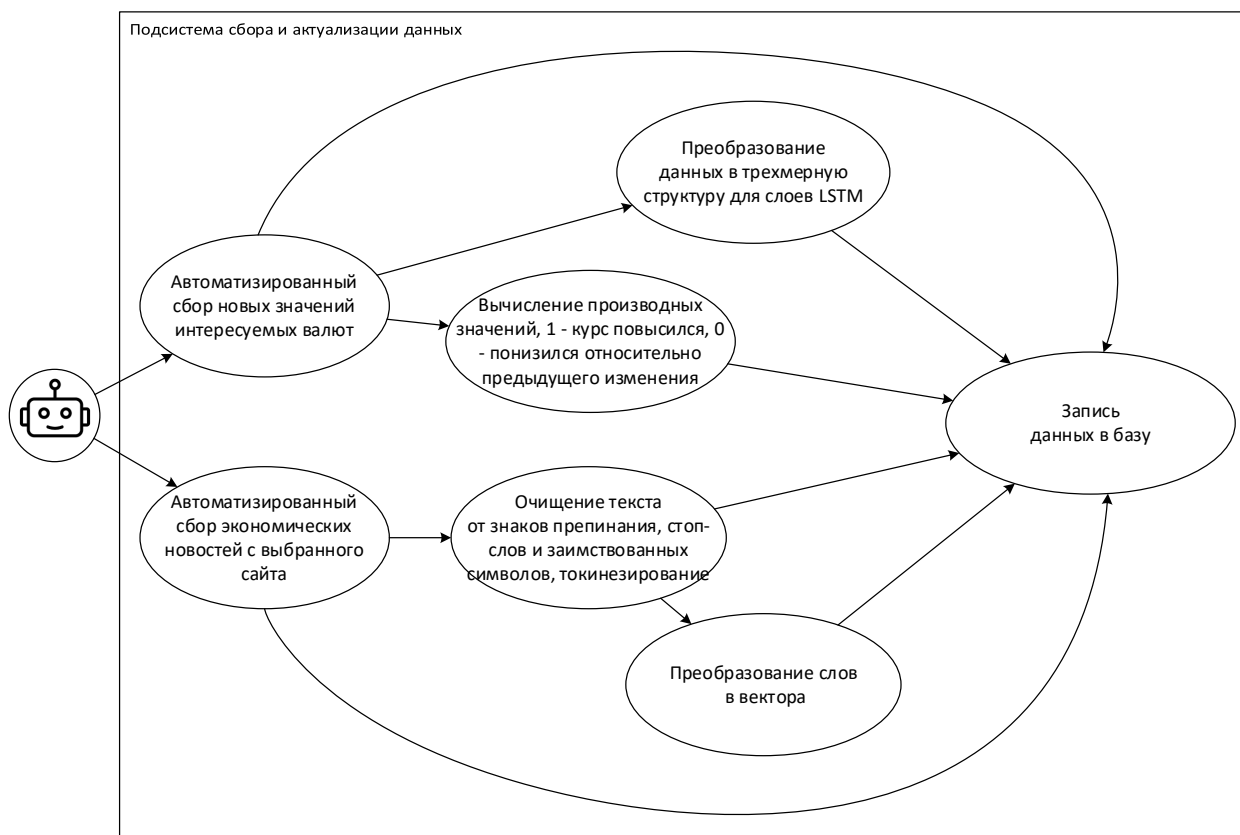


Рисунок 15 — Диаграмма подсистемы сбора и актуализации данных

Работу подсистемы прогнозирования можно разделить на две части, обучение моделей и использование их для предсказаний (Рисунок 16).

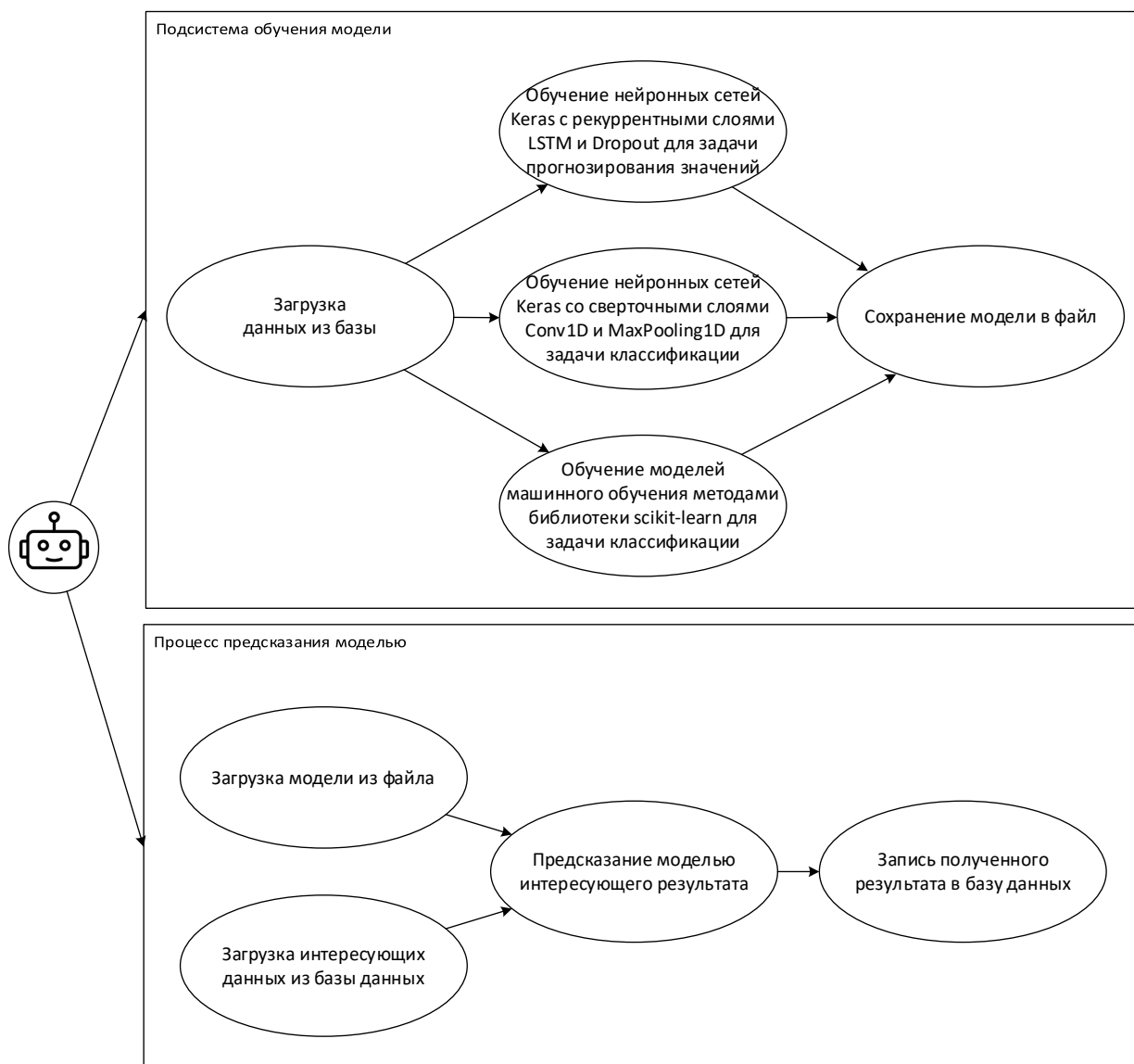


Рисунок 16 — Диаграмма подсистемы прогнозирования

На рисунке 17 представлено описание логики работы подсистемы взаимодействия с пользователем через Telegram-бот, предусматривающей отображение актуального и прогнозируемого значений характеристик выбранных финансовых инструментов.

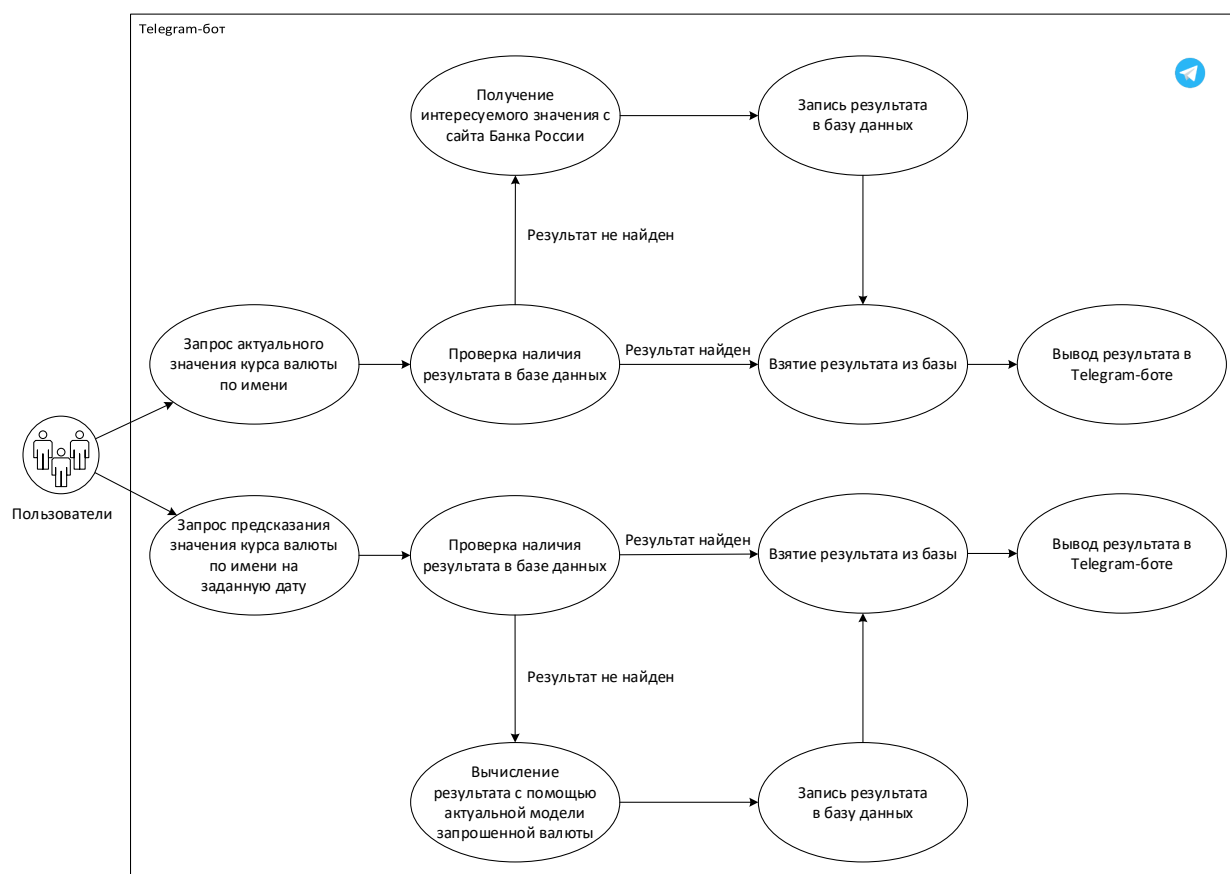


Рисунок 17 — Диаграмма подсистемы взаимодействия с пользователем через Telegram-бот

3. Разработка системы финансового информирования

В бакалаврском дипломе одной из моих задач была разработка мобильного приложения. Рассматривалась проблема переносимости программного обеспечения, были найдены способы для использования модели машинного обучения, написанной на Python, в мобильном iOS приложении на языке Swift.

На этот раз была спроектирована целая система финансового информирования, в качестве пользовательского интерфейса которой было выбрано создание Telegram-бота, также на Python. Данное решение обеспечивает кроссплатформенность, что повышает доступность для использования. Любой пользователь мессенджера может найти его и протестировать, и он будет работать с любого устройства, будь то смартфон или десктопное приложение.

3.1. Архитектура системы

Разработанная архитектура системы показана ниже (Рисунок 18). Она включает подсистему актуализации данных, где свежие данные собираются из открытых источников и пополняют базу данных. Подсистема прогнозирования получает информацию из базы данных, обрабатывает их и использует для обучения моделей машинного и глубокого обучения. Подсистема взаимодействия с пользователем включает Telegram-бот в качестве пользовательского интерфейса и сервер бота, которым в реализованном прототипе является моя консоль.

В реализованном в рамках данной ВКР прототипе системы средствами SQLite была создана база данных для передачи результатов прогнозирования моделей в Telegram-бот, остальные данные хранятся и используются как отдельные csv файлы. Использование базы данных позволяет повысить эффективность работы бота, устраняя необходимость повторного проведения расчетов при однотипных запросах пользователей.

SQLite — это библиотека языка C, которая предоставляет легковесную дисковую базу данных. Для работы с SQLite в Python есть стандартная библиотека `sqlite3`.

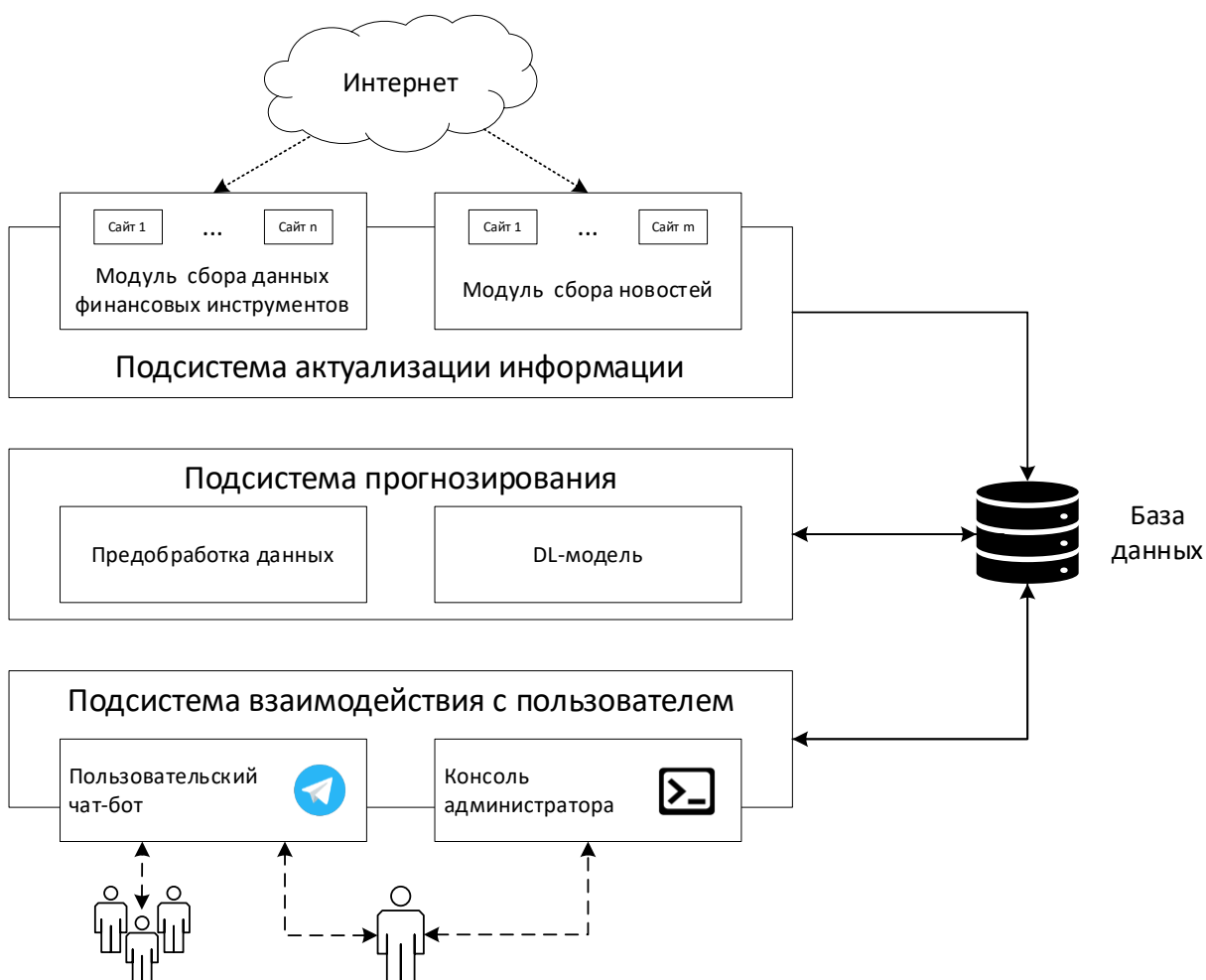


Рисунок 18 — Архитектура системы

3.2. Подсистема актуализации информации

Подсистема актуализации информации занимается сбором актуальных данных из выбранных источников и добавлением их в общую базу данных.

3.2.1. Сбор данных

Некоторые сайты с финансовыми данными предоставляют возможность сохранения интересующих данных в файл (как например mfd.ru, ...), однако далеко не все. Кроме того, следует иметь возможность получения актуальных данных в любой день, для этого существует парсинг данных.

Парсинг — автоматизированный сбор данных с последующей обработкой.

Существует много инструментов для этой задачи, но рассмотрим два простых в освоении и эффективных в использовании, они и использовались в создании прототипа системы.

Beautiful Soup — Python библиотека для извлечения данных из файлов HTML (HyperText Markup Language — язык гипертекстовой разметки) и XML (eXtensible Markup Language — расширяемый язык разметки). Она упрощает поиск и извлечение искомого содержимого, поскольку предоставляет много возможностей для навигации по структурированному дереву данных. Подходит, если нужно извлечь данные со статичного веб-сайта.

Selenium — это инструмент автоматизации тестирования веб-приложений, который часто используется для парсинга веб-страниц. Другими словами, он позволяет имитировать действия пользователя на сайте. Поддерживаемые языки программирования: C#, Java, Python, Ruby и другие. Поддерживаемые для автоматизации браузеры: Firefox, Edge, Safari и собственный QtWebKit. Подходит, если нужно работать с динамической веб-страницей, если требуется вход в систему или для сайтов, которые борются с парсингом. Недостаток, скорость работы напрямую зависит от объёма данных. Может занять значительное время, если все действия будут выполняться с задержками, имитируя работу человека.

3.2.2. Модуль сбора данных финансовых инструментов

В рамках прототипа была поставлена цель продемонстрировать логику работы запланированной системы на примере реализации всех этапов системы для одного финансового инструмента. Однако поскольку цель в создании актуализированной системы, требовалось найти постоянно обновляемый источник данных. При поиске подходящего источника данных было замечено, что ценность множества финансовых инструментов напрямую зависит от изменения курса валют, это видно и из международной классификации, рассмотренной в первой главе. Также данная информация актуальна не только людям, имеющим дело с финансовыми инструментами, но и простым смертным, и потому данные общедоступны.

Помимо курса валют в работе были исследованы данные цен акций за 05.08.2017-05.08.2022 годы 10-ти крупных мировых компаний. Это статичные данные, сразу собранные в один датасет структурированных данных, найденный на сайте Kaggle, про них будет в подсистеме прогнозирования.

Модуль сбора значений курса валют

В рамках прототипа были рассмотрены курсы к Российскому рублю (RUB) следующих валют: доллара США (USD), Евро (EUR) и Китайский юань (CNY). Данные о изменении курса валют на момент открытия биржи были взяты с сайта mfd.ru [24] в диапазоне 1.01.2022-28.04.2023, сайт позволяет задать интересующий диапазон дат и сохранить все в csv файл.

Важно отметить, что по выходным биржа не работает и курс не меняется, в таблице указаны все изменения курса в заданном диапазоне. Дата рядом с курсом значит, что этот курс установлен в последний день работы биржи до этого дня.

Далее данные сразу были минимально обработаны и объединены в один файл (Приложение А) (Рисунок 19). Также были добавлены значения, производные от курса:

- `currency_ch` — изменение значения курса относительно предыдущего дня
- `currency_ch_bi` — производное от предыдущего, где 1 - значение увеличилось, 0 – уменьшилось.

	Date	USD	USD_ch	USD_ch_bi	EUR	EUR_ch	EUR_ch_bi	CNY	CNY_ch	CNY_ch_bi
0	11.01.2022	75.1315	NaN	NaN	85.1315	NaN	NaN	11.7907	NaN	NaN
1	12.01.2022	74.8355	-0.2960	0.0	84.8784	-0.2531	0.0	11.7446	-0.0461	0.0
2	13.01.2022	74.5277	-0.3078	0.0	84.6709	-0.2075	0.0	11.7088	-0.0358	0.0
3	14.01.2022	74.5686	0.0409	1.0	85.4556	0.7847	1.0	11.7246	0.0158	1.0
4	15.01.2022	75.7668	1.1982	1.0	86.8894	1.4338	1.0	11.9457	0.2211	1.0
...
318	22.04.2023	81.4863	-0.1325	0.0	89.3495	-0.1143	0.0	11.8111	-0.0359	0.0
319	25.04.2023	81.2745	-0.2118	0.0	89.4589	0.1094	1.0	11.7700	-0.0411	0.0
320	26.04.2023	81.5499	0.2754	1.0	90.0332	0.5743	1.0	11.7664	-0.0036	0.0
321	27.04.2023	81.6274	0.0775	1.0	90.1436	0.1104	1.0	11.7626	-0.0038	0.0
322	28.04.2023	81.5601	-0.0673	0.0	90.2023	0.0587	1.0	11.7609	-0.0017	0.0

323 rows x 10 columns

Рисунок 19 — История изменения курса открытия USD, EUR и CNY

Модуль сбора новостей

В попытке учесть в предсказании множество факторов, формальный учет которых невозможен, появилась идея применить текстовый анализ к заголовкам новостей за интересующие даты. Смысл в том, что заголовки по-идее должны

выделять основную идею новости, какое-то изменение в мире. В прототипе для прогнозирования будут использоваться для анализа заголовки 40 самых популярных новостей за каждый день из диапазона.

Для сбора новостей был написан код, реализующий парсинг новостей с сайта РИА Новости по тегу Экономика [25] (Приложение Б). В нем используются ранее упомянутые инструменты Selenium и BeautifulSoup.

Рассмотрим ключевые фрагменты программы. Для парсинга любого сайта будет примерно тоже самое, с изменением интересующих элементов и, возможно, каких-то особенности поведения сайта.

Всегда первым делом идет открытие интересующего сайта, функция `start_driver` (Листинг 1).

Листинг 1 — Функция инициализации и запуска веб драйвера по ссылке

```
from selenium import webdriver
def start_driver(link):
    # опции для игнорирования некоторых ошибок и предупреждений
    options = webdriver.ChromeOptions()
    options.add_experimental_option('excludeSwitches', ['enable-logging'])

    # Запускаем драйвер, открываем веб-страницу
    driver = webdriver.Chrome(options=options)
    driver.set_window_size(1500,1000)
    driver.implicitly_wait(randint(5, 10)) # случайная задержка, в секундах
    driver.get(link)
    return driver
```

Также часто требуется листание страницы, позволяющее прогрузить всю возможную информацию и только после забирать её, функция для этого `scroll_page` (Листинг 2).

Листинг 2 — Листание страницы вниз

```
def scroll_page(driver, pause_time):
    # Получение высоты прокрутки
    last_height = driver.execute_script("return document.body.scrollHeight")

    while True:
        # Листание вниз до конца страницы
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

        # Ожидание загрузки страницы
        time.sleep(pause_time)

        # Расчет новой высоты прокрутки и сравнение ее с предыдущей
```

```

new_height = driver.execute_script(
    "return document.body.scrollHeight")
if new_height == last_height:
    break
last_height = new_height

```

Дальше рассмотрим функцию `parse_by_date` для парсинга новостей за одну дату (Листинг 3). На сайте РИА Новости есть логика создания ссылок, потому можно легко открыть архив новостей за конкретный день просто по ссылке. Ещё одна особенность сайта, при открытии подобной страницы архива он сразу загружает не больше 20 новостей. Потому для получения всех новостей за день из html-кода по классу достаётся количество новостей за день, и, если оно превышает 20, драйвер Selenium листает страницу, нажимает на кнопку внизу для загрузки дополнительных материалов, ждёт пять секунд, чтобы всё загрузилось, и опять листает вниз, на этот раз загружая все новости за день. Результатом выполнения функции является получение полноценного html-кода архива новостей за указанную дату в переменной библиотеки Beautiful Soup.

Листинг 3 — Парсинг новостей за одну дату, создание супа данных

```

def parse_by_date(date):
    #print('Дата:', date)
    # создание ссылки новостного архива заданного дня
    link = 'https://ria.ru/economy/' + date + '/'

    driver = start_driver(link)

    # Суп из HTML-кода страницы
    soup = BeautifulSoup(driver.page_source, 'lxml')

    # Выделение количества новостей за день, оно влияет на количество страниц
    number_of_news = int(html_stripper(soup.find('div', class_ =
        'rubric-count m-active').span))

    # 20 - максимальное количество новостей на странице ria.ru
    if number_of_news > 20:
        # Листание страницы вниз
        scroll_page(driver, 1)

        # Нажатие на кнопку внизу для загрузки дополнительных материалов,
        # если она есть
        driver.find_element(By.XPATH,
            '//*[@id="content"]/div/div[1]/div/div[3]').click()
        time.sleep(5)

    # Листание страницы вниз, дальше новые данные будут загружаться

```



```

scroll_page(driver, 1)

# Обновление супа, появились новые новости
soup = BeautifulSoup(driver.page_source, 'lxml')

#time.sleep(randint(2, 4))

# Закрытие драйвера, больше он не нужен
driver.close()
# Передача супа данных
return soup

```

Поднимаемся ещё на уровень выше. Функция `parse_all` для парсинга данных в диапазоне дат (Листинг 4). В ней, первым делом, по начальной и конечной датам генерируется список всех интересующих дат для ссылок на архивы новостей. Далее в цикле на каждую дату из списка создается суп из html-кода с помощью ранее описанной функции. На сайте РИА Новости все блоки новостей находятся в классах "list-item", далее из каждого такого блока выделяются: заголовок новости, ссылка на полную новость, дата публикации новости, количество просмотров новости. Полученные элементы добавляются одной строкой в конец датасета, и наконец финальный датасет сохраняется в csv-файл. Перебор дат происходит с индикатором прогресса, `alive_bar`, чтобы в любой момент можно было отслеживать на каком этапе находится парсинг данных.

Листинг 4 — Функция для парсинга данных в диапазоне дат и её вызов

```

# Функция для парсинга данных в диапазоне дат [start, end] и сохранение табли
цы в csv файл
def parse_all(start, end, file_name):
    # Генерирование списка всех интересующих дат для ссылок
    dates_list = pd.date_range(start, end).strftime('%Y%m%d').tolist()
    total = len(dates_list)
    print('\nКоличество дат = ', total, '\n')

    # Создание пустого датафрейма для дальнейшего накопления информации
    # Столбцы: дата публикации, количество просмотров, заголовок статьи,
                                                    ссылка на статью

    index_list = ['date', 'views', 'name', 'url']
    news_table = pd.DataFrame(columns=index_list)

    with alive_bar(total) as bar:    # индикатор прогресса
        # Вытаскивание данных для каждой даты
        for date in dates_list:
            # Получение супа данных для даты
            soup = parse_by_date(date)

```

```

# Выделение списка из кусков кода, относящихся к новостям
news = soup.find_all(class_="list-item")

# Добавление всех новостей дня в общую таблицу
for i in news:
    name = html_stripper(i.find(class_="list-item__title color-
                                font-hover-only"))

    name = check_for_broken(name)
    url = i.find(class_="list-item__title color-font-
                        hover-only")['href']

    # Получение даты из ссылки
    url_date = re.search(r'\d{8}', url)[0]
    date = url_date[:4]+'-'+url_date[4:6]+'-'+url_date[6:]
    views = html_stripper(i.find(class_="list-item__views-text"))

    # Добавление новой строки в конец
    news_table.loc[len(news_table.index)] = [date, views,
                                              name, url]

bar()    # Изменение индикатора прогресса

#print('\nКоличество новостей = ', news_table.shape[0], '\n')

# Сохранение таблицы в файл
news_table.to_csv(file_name, index=False, sep=";", encoding='utf-8-sig')

#-----
parse_all('2022-01-01', '2023-03-28', 'news_table.csv')

```

Теперь функция `data_actualization` для актуализации данных (Листинг 5), она вызывает описанную ранее функцию парсинга данных на диапазоне, потом склеивает полученный датафрейм с указанным и сохраняет результат под указанным именем.

Листинг 5 — Функция для актуализации данных

```

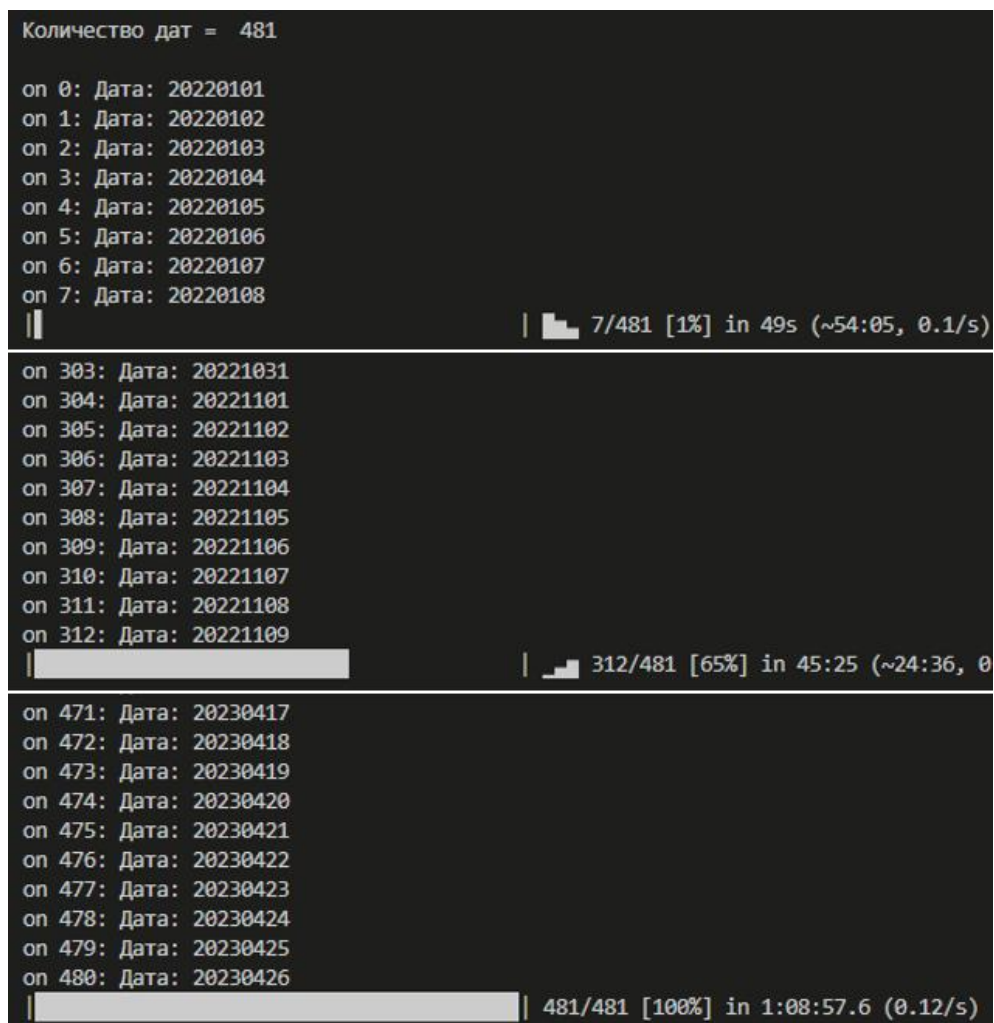
# Дополнение таблицы, актуализация данных
def data_actualization(new_start, new_end, df_csv, res_name_csv):
    df_old = pd.read_csv(df_csv, sep=";", encoding='utf-8-sig')
    # Получение новых данных
    parse_all(new_start, new_end, 'news_last_actualization_add.csv')
    df_new = pd.read_csv('news_last_actualization_add.csv', sep=";",
                        encoding='utf-8-sig')

    # Соединение всех данных
    news_result = pd.concat([df_old, df_new])
    # Сохранение обновленной таблицы
    news_result.to_csv(res_name_csv, index=False, sep=";",
                        encoding='utf-8-sig')

```

```
#-----  
data_actualization('2023-03-29', '2023-04-28', 'news_table.csv',  
                  'economy_news_.csv')
```

Выводы в терминале во время парсинга новостей вместе с индикатором прогресса (Рисунок 20)



```
Количество дат = 481  
on 0: Дата: 20220101  
on 1: Дата: 20220102  
on 2: Дата: 20220103  
on 3: Дата: 20220104  
on 4: Дата: 20220105  
on 5: Дата: 20220106  
on 6: Дата: 20220107  
on 7: Дата: 20220108  
|| 7/481 [1%] in 49s (~54:05, 0.1/s)  
on 303: Дата: 20221031  
on 304: Дата: 20221101  
on 305: Дата: 20221102  
on 306: Дата: 20221103  
on 307: Дата: 20221104  
on 308: Дата: 20221105  
on 309: Дата: 20221106  
on 310: Дата: 20221107  
on 311: Дата: 20221108  
on 312: Дата: 20221109  
| 312/481 [65%] in 45:25 (~24:36, 0  
on 471: Дата: 20230417  
on 472: Дата: 20230418  
on 473: Дата: 20230419  
on 474: Дата: 20230420  
on 475: Дата: 20230421  
on 476: Дата: 20230422  
on 477: Дата: 20230423  
on 478: Дата: 20230424  
on 479: Дата: 20230425  
on 480: Дата: 20230426  
| 481/481 [100%] in 1:08:57.6 (0.12/s)
```

Рисунок 20 — Терминал во время парсинга новостей

Теперь посмотрим на финальный результат (Рисунок 21). В итоге было собрано 25081 новость за 481 день.

	date	views	name	url
0	01.01.2022	18238	На Украине ввели предельную надбавку к цене на...	https://ria.ru/20220101/gaz-1766394128.html
1	01.01.2022	2085	Запасы газа на Украине за год сократились почт...	https://radiosputnik.ria.ru/20220101/gaz-17663...
2	01.01.2022	15134	СМИ: часть крупного металлургического завода У...	https://ria.ru/20220101/zavod-1766379413.html
3	01.01.2022	32239	В Британии заявили о катастрофичном снижении у...	https://ria.ru/20220101/padenie-1766374961.html
4	01.01.2022	22886	Брюссель намерен присвоить "зеленый" статус ат...	https://ria.ru/20220101/energetika-1766368303....
...
25076	27.04.2023	45090	Россия организовала Западу жесткое приземление	https://ria.ru/20230427/prizemlenie-1867983358...
25077	27.04.2023	52676	Сработал эффект домино. В США поплыл еще один ...	https://ria.ru/20230427/bank-1867967185.html
25078	27.04.2023	1130	Аналитики рассказали, какую карту разыграет кр...	https://ria.ru/20230427/dividendy-1868007554.html
25079	27.04.2023	4177	В России почти полностью прекратились госзакуп...	https://ria.ru/20230427/windows-1868019655.html
25080	27.04.2023	9520	Российский экспорт в Индию впервые превысил 40...	https://ria.ru/20230427/eksport-1868017979.html

25081 rows x 4 columns

Рисунок 21 — Датафрейм всех собранных новостей, файл *economy_news.csv*

3.3. Подсистема прогнозирования

Подсистема прогнозирования занимается подготовкой данных, обучением моделей и непосредственно прогнозированием. Здесь рассматриваются:

1. Предсказание цен акций 10-ти мировых компаний — несколько игрушечный пример, предсказание цен акций нейросетью, обученной и проверенной на готовом датасете.
2. Предсказание курса валют (USD, EUR, CNY) — как временных рядов с помощью LSTM моделей.
3. Предсказание курса валют по заголовкам новостей — использование истории изменения курса валюты и заголовков главных новостей за день.

3.3.1. Предсказание цен акций 10-ти мировых компаний

Данные

Использовался статичный датасет цен акций за 05.08.2017-05.08.2022 годы 10-ти крупнейших мировых компаний: Apple, Amazon, Google, JP-Morgan, Microsoft, Netflix, Nvidia, Tesla, Visa, Walmart [26].

Файл *ZAll_Combine_Stock_Histry.csv* включает по 1258 строк данных для каждой компании, 12580 всего (Рисунок 22).

	Date	Open	High	Low	Close	Adj Close	Volume	Company
0	07-08-2017	39.264999	39.730000	39.167500	39.702499	37.585022	87481200	AAPL
1	08-08-2017	39.650002	40.457500	39.567501	40.020000	37.885582	144823600	AAPL
2	09-08-2017	39.814999	40.317501	39.777500	40.264999	38.117512	104526000	AAPL
3	10-08-2017	39.974998	40.000000	38.657501	38.830002	36.903397	163217200	AAPL
4	11-08-2017	39.150002	39.642502	39.017502	39.369999	37.416603	105028400	AAPL
...
12575	29-07-2022	128.320007	132.259995	128.009995	132.050003	132.050003	10045500	WLMART
12576	01-08-2022	131.059998	134.229996	131.000000	132.539993	132.539993	8332100	WLMART
12577	02-08-2022	133.149994	133.710007	131.399994	132.679993	132.679993	6565000	WLMART
12578	03-08-2022	132.160004	132.940002	129.860001	130.500000	130.500000	10667700	WLMART
12579	04-08-2022	130.669998	130.800003	125.330002	125.570000	125.570000	17702000	WLMART

12580 rows x 8 columns

Рисунок 22 — Данные ZAll_Combine_Stock_Histry.csv

Каждая строка содержит информацию про каждую компанию за день для: цену акций компании на момент открытия биржи в этот день, максимальную цену акций, минимальную цену, цену закрытия (цену на момент закрытия биржи), скорректированную цену закрытия и количество акций в обороте за день.

Первичный анализ данных

Было проверено, что данные не содержат пропусков, интересующие столбцы цен акций в формате float64. Проверяю какие есть уникальные компании и количество их строк, также получаю список устойчивых сокращений компаний (Рисунок 23).

```
unique_c = df_all.Company.unique()
unique_c

array(['AAPL', 'AMZN', 'GOOGL', 'JP-MRGN', 'M-SOFT', 'NTFLX', 'NVDA',
      'TSLA', 'VISA', 'WLMART'], dtype=object)

for u in unique_c:
    print(u, ': ', len(df_all[df_all['Company'] == u]))

AAPL : 1258
AMZN : 1258
GOOGL : 1258
JP-MRGN : 1258
M-SOFT : 1258
NTFLX : 1258
NVDA : 1258
TSLA : 1258
VISA : 1258
WLMART : 1258
```

Рисунок 23 — Уникальные компании

Посмотрим на данных отдельной компании, например, Tesla (Рисунок 24).

```
df_tesla = df_all[df_all['Company'] == 'TSLA']
df_tesla
```

	Open	High	Low	Close	Adj Close	Volume	Company
Date							
2017-07-08	71.470001	71.896004	70.550003	71.033997	71.033997	31622500	TSLA
2017-08-08	71.505997	73.716003	71.480003	73.043999	73.043999	37249000	TSLA
2017-09-08	72.199997	74.000000	71.790001	72.706001	72.706001	34460500	TSLA
2017-10-08	72.320000	73.330002	70.931999	71.080002	71.080002	35464500	TSLA
2017-11-08	71.393997	72.251999	70.723999	71.573997	71.573997	21829000	TSLA
...
2022-07-29	842.099976	894.960022	837.299988	891.450012	891.450012	31771000	TSLA
2022-01-08	903.830017	935.630005	885.000000	891.830017	891.830017	39014300	TSLA
2022-02-08	882.010010	923.500000	878.000000	901.760010	901.760010	31859200	TSLA
2022-03-08	915.000000	928.650024	903.450012	922.190002	922.190002	26697000	TSLA
2022-04-08	933.000000	940.820007	915.000000	925.900024	925.900024	24085400	TSLA

1258 rows × 7 columns

Рисунок 24 — Данные Tesla

Визуализируем данные (Рисунок 25). Динамики изменения первых 5 столбцов идентичны, так и должно быть, поскольку все они являются значениями цены и на таком масштабе слабо отличаются друг от друга.

```
def ShowCompanyPlots2(company):
    company.plot(subplots=True, figsize=(16,12), title=company.Company.unique()[0])
    plt.show()

ShowCompanyPlots2(df_tesla)
```

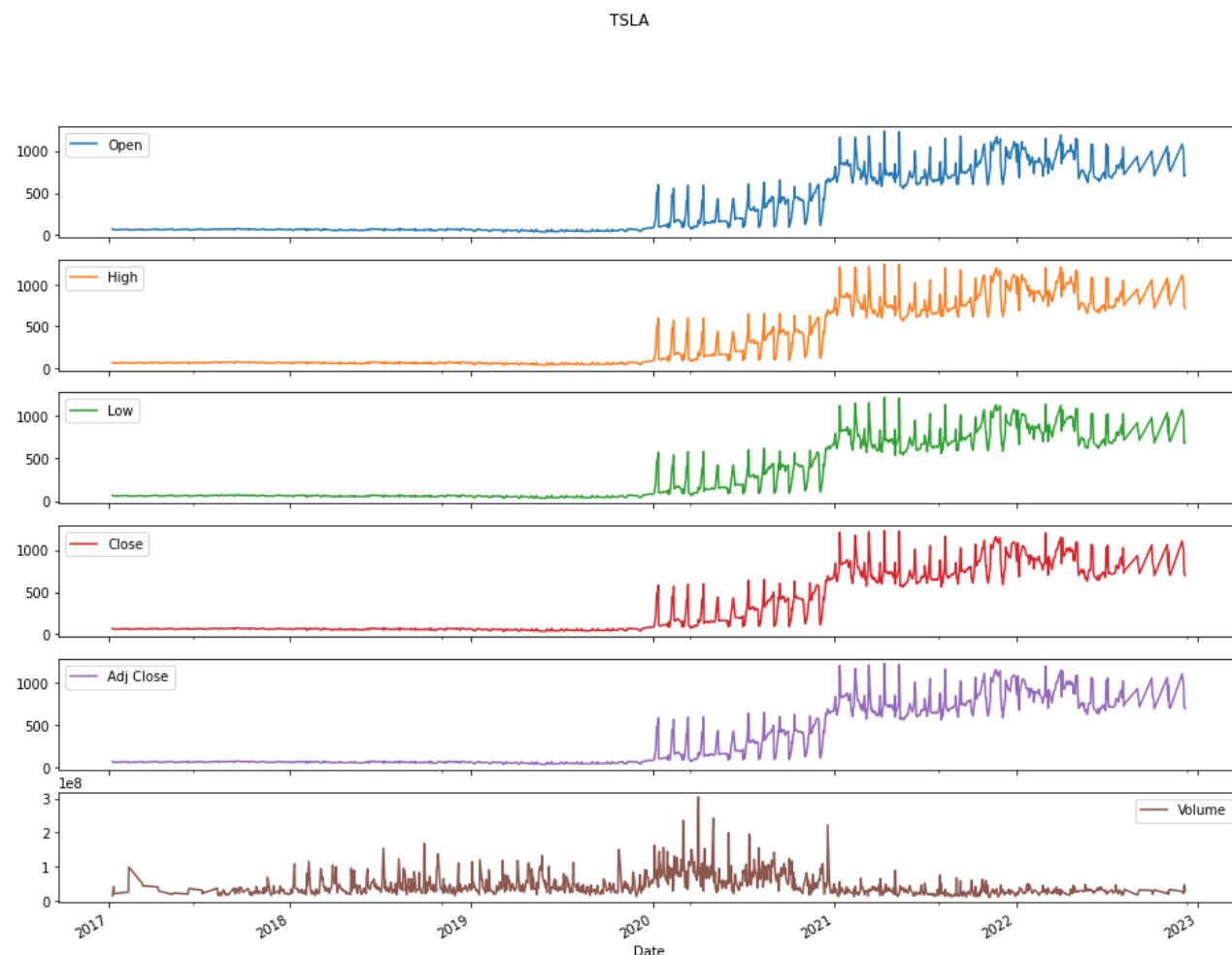


Рисунок 25 — Визуализация данных Tesla

Наиболее важным параметром здесь является цена закрытия (Close), с ее помощью отслеживают динамику изменения актива. Потому этот параметр и будет предсказывать модель. Визуализируем данные, историю изменения цен закрытия для всех 10-ти компаний (Рисунок 26). Две компании значительно выделяются в сравнении с остальными.

Сравнение цен акций компаний (Close)

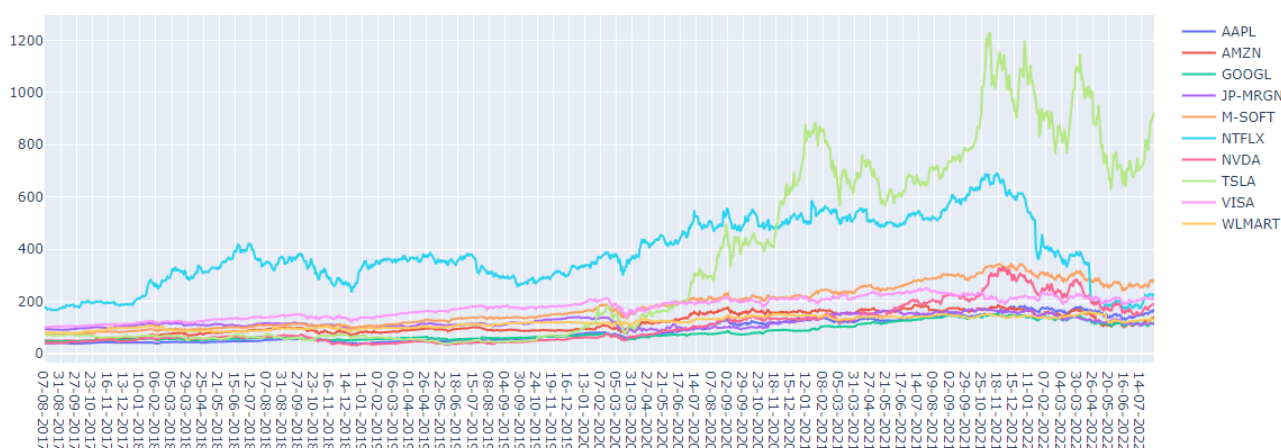


Рисунок 26 — Динамика изменения цен закрытия акций (Close) 10-ти компаний

Модель глубокого обучения

Модель создана для прогнозирования цены закрытия на основании 60 предыдущих дней. Перед обучением и тестированием цены закрытия были нормализованы в диапазоне от [0,1].

Для этого используем последовательную модель Keras (Sequential) [27], добавляем в нее рекуррентные слои LSTM для обучения долгосрочным зависимостям и вспомогательные слои Dropout для минимизирования переобучения, получаем следующую модель (Рисунок 27).

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 300)	362400
dropout (Dropout)	(None, 60, 300)	0
lstm_1 (LSTM)	(None, 60, 100)	160400
dropout_1 (Dropout)	(None, 60, 100)	0
lstm_2 (LSTM)	(None, 60, 100)	80400
dropout_2 (Dropout)	(None, 60, 100)	0
lstm_3 (LSTM)	(None, 100)	80400
dropout_3 (Dropout)	(None, 100)	0
dense (Dense)	(None, 1)	101

=====

Total params: 683,701
 Trainable params: 683,701
 Non-trainable params: 0

Рисунок 27 — Сводное представление модели с LSTM слоями

Результаты прогнозирования

Для наглядности рассмотрим результаты работы только двух моделей, одной, обученной на динамике изменения цен акций компании Netflix, и другой, обученной на динамике Apple. У этих компаний значительно отличаются цены акций и динамика их изменения, поэтому различия точности результатов заметней.

Визуализируем результаты (Рисунок 28). Сравнивая реальные цены акций Netflix с предсказанными моделью, обученной по данным этой компании, может показаться, что она работает хорошо, ведь ломанные близки друг другу. Однако можно заметить, что предсказания зачастую находятся ближе к предыдущему значению, чем к реальному.

Но нельзя и сказать, что модель совсем не работает. Направление изменения цены зачастую правильное. А также она прогнозирует ближе к реальному, чем модель, обученная на динамике другой компании.



Рисунок 28 — Визуализация реальных и предсказанных цен акций Netflix (Close)

Полученные значения метрик подтверждают визуальный анализ, проделанный ранее, модель, обученная на данных компании работает лучше, показатели отклонений у нее меньше, а объясненная дисперсия больше (Рисунок 29).

```

Сравнение реальных цен акций Netflix с предсказанными моделью model_LSTM_ntflx
explained_variance_score = 0.8283577305406103
mean_squared_log_error = 0.0014417284006341536
mean_squared_error = 51.65850456682597
root_mean_squared_error = 7.187385099382526
mean_absolute_error = 5.745468885907454
median_absolute_error = 4.229095576171886
max_error = 17.515243971679695

```

```

Сравнение реальных цен акций Netflix с предсказанными моделью model_LSTM_aapl
explained_variance_score = 0.7334152138627299
mean_squared_log_error = 0.00286076893809836
mean_squared_error = 96.75199855525523
root_mean_squared_error = 9.836259378201412
mean_absolute_error = 7.733496624368991
median_absolute_error = 6.085647245117201
max_error = 30.242141333984364

```

Рисунок 29 — Сравнение реальных цен акций Netflix с предсказанными моделями

3.3.2. Предсказание курса валют как временного ряда

Здесь будут обрабатываться данные изменения курсов доллара США (USD), Евро (EUR) и Китайский юань (CNY) к Российскому рублю (RUB), собранные и описанные ранее, в пункте 3.2.1.

Анализ данных будет аналогичен проделанному в прошлом пункте для предсказания акций компаний. Но была улучшена реализация. Для валют выделен класс `Currency` с атрибутами: имя `name`, данные `df` и объект для масштабирования данных `scaler` (Листинг 6).

Листинг 6 — Класс `Currency`

```

class Currency:
    def __init__(self, name, courses):
        self.name = name
        self.df = courses[['Date', name]]
        self.scaler = MinMaxScaler(feature_range = (0, 1))

```

Предобработка данных

Модели прогнозируют новые значения курса (`lable`) на основании 30 предыдущих дней (`features`). LSTM слои Keras работают с трехмерными данными структуры (`nb_sequence`, `nb_timestep`, `nb_feature`), где `nb_sequence` соответствует общему количеству последовательностей в наборе данных, `nb_timestep` соответствует размеру последовательностей и `nb_feature` соответствует количеству признаков, описывающих каждый временной шаг. Посмотрим, как была реализована предобработка данные (Листинг 7).

```

def make_features_labels(self, data):
    set_features = []
    set_labels = []
    len_df_train = len(data)
    for i in range(30, len_df_train):
        set_features.append(data[i-30:i, 0])
        set_labels.append(data[i, 0])
    return set_features, set_labels

def prepare_features_labels(self, data):
    data_scaled = self.scaler.fit_transform(data.reshape(-1, 1))
    set_features, set_labels = self.make_features_labels(data_scaled)
    set_features, set_labels = np.array(set_features), np.array(set_labels)

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

    return set_features, set_labels

def prepare_currency(self, currency):
    split_point = len(currency) - len(currency) // 10

    currency_train = currency[:split_point]
    currency_test = currency[split_point-30:]

    train_features, train_labels =
        self.prepare_features_labels(currency_train)
    test_features, test_labels = self.prepare_features_labels(currency_test)

    return train_features, train_labels, test_features, test_labels

# - # - # - # - #

def do(self, flag_train=False, flag_print_values=False, flag_plot=False):
    train_f, train_l, test_f, test_l =
        self.prepare_currency(self.df.iloc[:, 1].values)

```

Модель глубокого обучения и результаты прогнозирования

Вновь используем последовательную модель Keras с рекуррентными слоями LSTM, почти как для предсказания цен акций в прошлом пункте, только теперь размер последовательностей уменьшен до 30, а значение units установим равным 100, а потом 300 (Приложение В).

Визуализируем полученные результаты, предсказания моделей для USD (Рисунок 30) и EUR (Рисунок 31) находятся немного ближе к их реальным значениям, чем предсказания CNY к своим реальным значениям (Рисунок 32).

Возможно, это связано с тем, что значения USD и EUR находятся недалеко друг от друга, а значения CNY значительно меньше, и возможно требуют выставления других параметров модели для обучения.

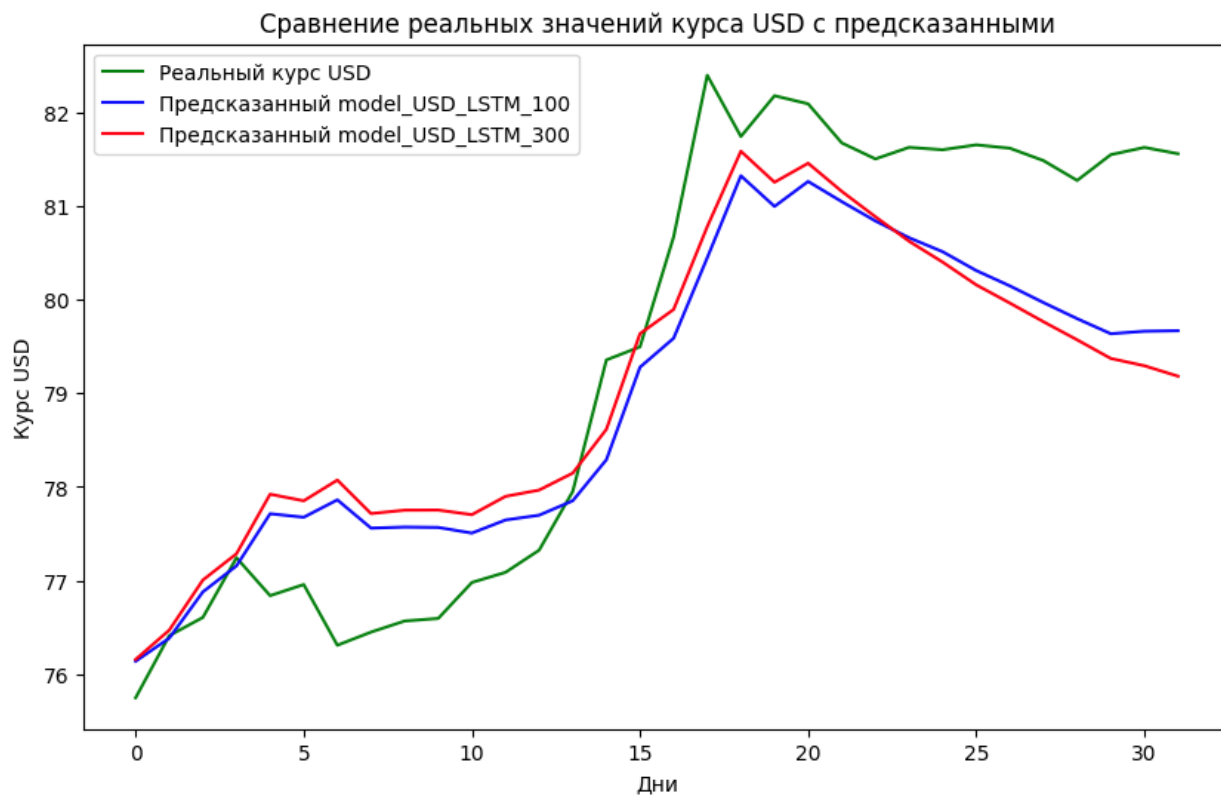


Рисунок 30 — Сравнение реальных значений курса USD с предсказанными

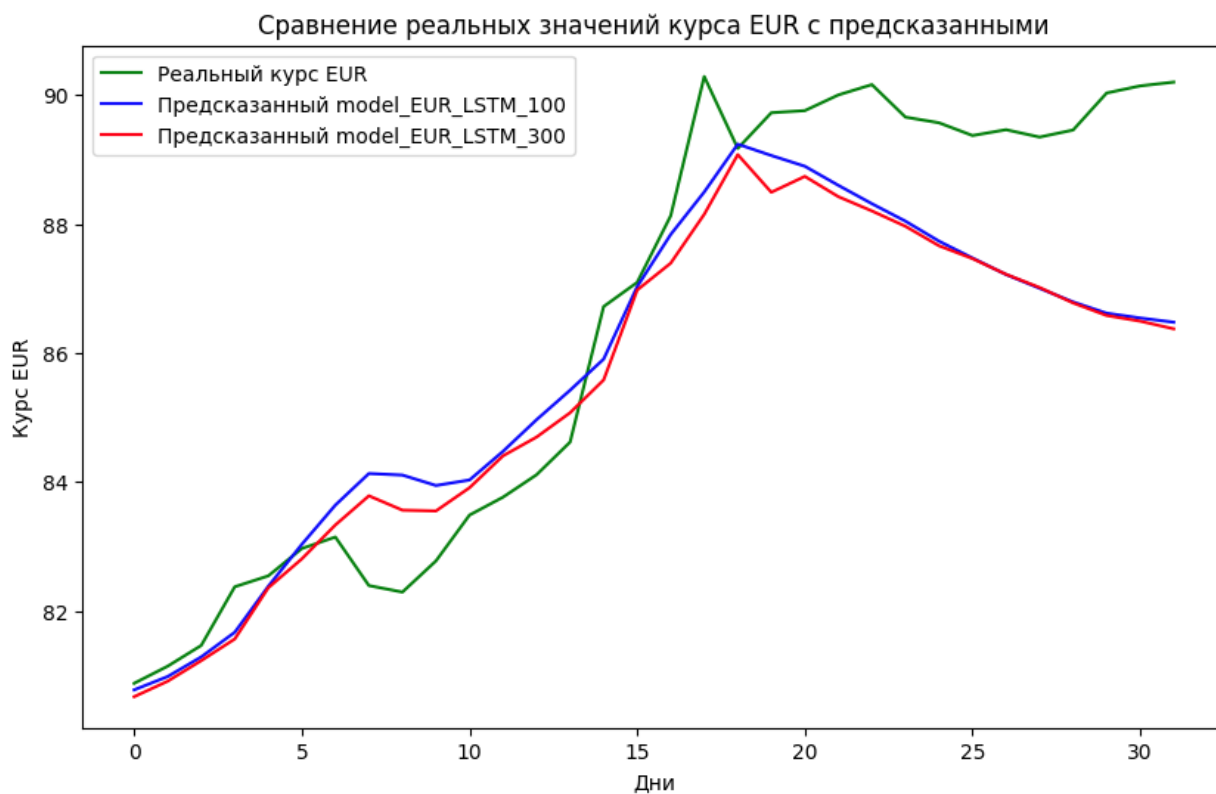


Рисунок 31 — Сравнение реальных значений курса EUR с предсказанными

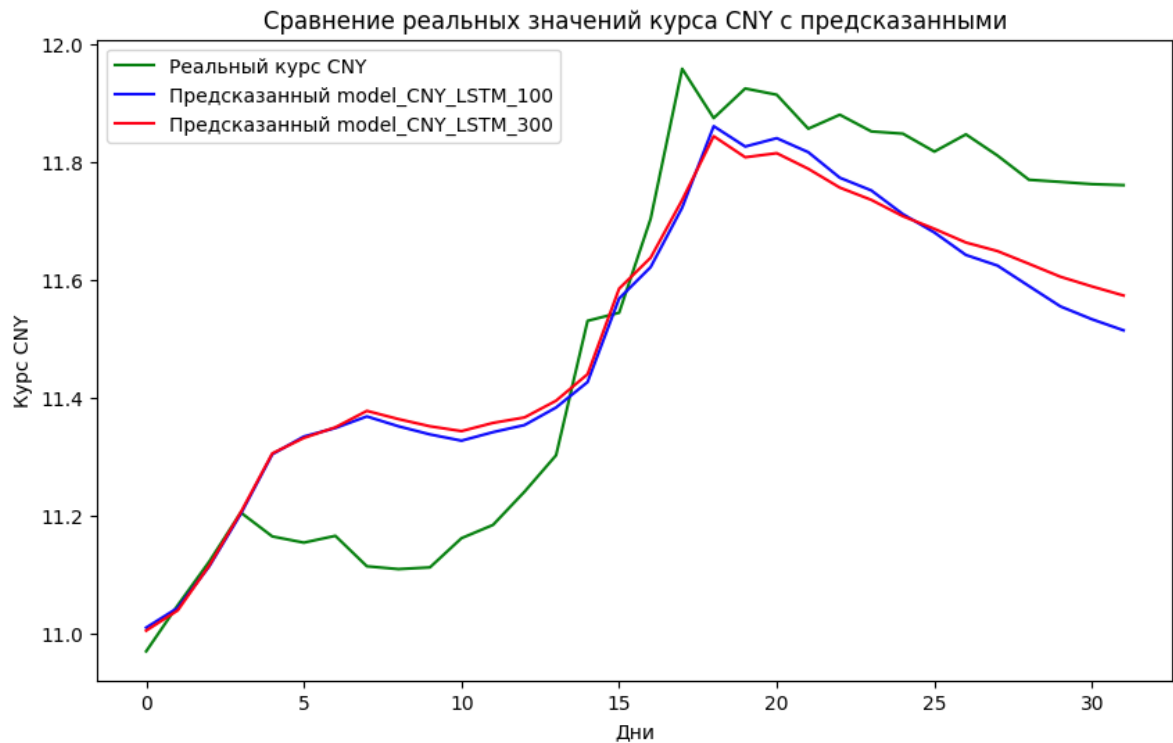


Рисунок 32 — Сравнение реальных значений курса CNY с предсказанными

Полученные значения метрик (Рисунок 33) подтверждают визуальный анализ, модели для предсказания значений USD и EUR показали большую точность, чем модель для CNY. Кроме того, модели, обучаемые 300 эпох в среднем показали немного большую точность по сравнению с моделями, обучаемые 100 эпох.

	USD_LSTM_100	USD_LSTM_300
explained_variance_score	0.818028	0.765815
mean_squared_log_error	0.000185	0.000222
mean_squared_error	1.212222	1.445675
root_mean_squared_error	1.101010	1.202362
mean_absolute_error	0.945215	1.013626
median_absolute_error	0.968950	0.909652
max_error	1.964436	2.377704
	EUR_LSTM_100	EUR_LSTM_300
explained_variance_score	0.819169	0.838483
mean_squared_log_error	0.000345	0.000352
mean_squared_error	2.718329	2.788356
root_mean_squared_error	1.648735	1.669837
mean_absolute_error	1.270622	1.289954
median_absolute_error	0.858379	1.079251
max_error	3.724608	3.827331
	CNY_LSTM_100	CNY_LSTM_300
explained_variance_score	0.799693	0.812195
mean_squared_log_error	0.000150	0.000139
mean_squared_error	0.023393	0.021708
root_mean_squared_error	0.152947	0.147337
mean_absolute_error	0.131409	0.129307
median_absolute_error	0.136820	0.135501
max_error	0.254086	0.263512

3.3.3. Предсказание направления изменения курса валют по новостям

Изменение курса валют зависит от множества факторов, множества изменений в мире, точно учесть которые невозможно. А важные изменения в мире как правило придают огласке. Новостные заголовки, в свою очередь, должны отображать основной смысл какого-то изменения.

Гипотеза заключается в том, что новости влияют на изменение курса валют, и что модель сможет найти закономерности и использовать их для улучшения качества прогнозирования.

Здесь будут обрабатываться данные новостей, собранные ранее в пункте 3.2.1. На каждой строке файла содержится: дата публикации, заголовок новости, количество просмотров и ссылка на полную новость.

Предобработка данных

Для обучения модели нам потребуются в качестве лейблов значения изменения курса валют или производные от него. Но поскольку биржа не работает по выходным, существует множество дней, в которые курс не менялся, но мир менялся и новости появлялись. Поэтому, первым делом, преобразуем даты написания новостей так, чтобы каждой новости соответствовала дата, на которую она теоретически влияет (Листинг 8).

Листинг 8 — Изменение дат в таблице новостей

```
# Изменение дат в спаршенной таблице новостей и сохранение в файл
if flag_change_dates:
    # Открытие спаршенной таблицы новостей с 01.01.2022 по 27.04.2023
    news_all = pd.read_csv('economy_news.csv', sep=";", encoding='utf-8-sig')
    print(news_all)
    print('\n')
    print(news_all.info())
    print('\nОсновные характеристики столбца количества просмотров новости')
    print(news_all.describe())

    # Список всех дней в заданном диапазоне
    dates_list = pd.date_range('2022-01-01', '2023-04-27')
                                     .strftime('%d.%m.%Y').tolist()

    # Список дат, на которые менялся курс
    kurs_change_dates = list(kurses['Date'])
```

```

# Выделение дней, на которые курс не менялся
just_days = []
for day in dates_list:
    if day not in kurs_change_dates:
        just_days.append(day)

#-----

# Изменение дат в таблице и сохранение в файл
# Изменение, чтобы новости соответствовала дата,
#                               на которую она теоретически влияет

# Сдвиг всех новости на день вперед (идя с конца)
for i in reversed(range(news_all.shape[0])):
    d = news_all.loc[i, 'date']
    today = datetime.date(int(d[6:10]), int(d[3:5]), int(d[:2])) # '22.02.2022'
    tomorrow = today + datetime.timedelta(days=1)
    news_all.loc[i, 'date'] = tomorrow.strftime('%d.%m.%Y')
    #news_all.loc[i, 'date1'] = tomorrow.strftime('%d.%m.%Y')

# Сдвиг всех новостей из списка just_days на день вперед (идя с начала)
for day in just_days:
    for i in range(news_all.shape[0]):
        d = news_all.loc[i, 'date']
        if d == day:
            today = datetime.date(int(d[6:10]), int(d[3:5]), int(d[:2]))
            tomorrow = today + datetime.timedelta(days=1)
            news_all.loc[i, 'date'] = tomorrow.strftime('%d.%m.%Y')

# Сохранение результата в файл
news_all.to_csv('economy_news_changed_dates.csv', index=False, sep=";",
               encoding='utf-8-sig')

```

Теперь, когда новости за выходные дни сгруппированы вместе и соответствуют дням изменений курса, посмотрим на гистограмму распределения количества новостей за день (Рисунок 34) и на гистограмму распределения количества дней для количества новостей в день (Рисунок 35). Это нужно для определения оптимального количества новостей, которые будут учитываться моделью. Если взять слишком много новостей в день, может оказаться слишком много пропусков в некоторые дни и это ухудшит качество обучения и предсказания модели. Если слишком мало, у модели будет недостаточно информации для нахождения зависимостей.

На первой гистограмме заметно, что многие столбцы снизу не доходят до отметки в 50 новостей в день. На второй гистограмме видно, что 42 дня количество новостей находится в интервале 40-49, и в подавляющее большинство дней (все

справа на гистограмме) новостей было ещё больше, так что одна из моделей будет обучаться по 40 самым просматриваемым новостям за день.

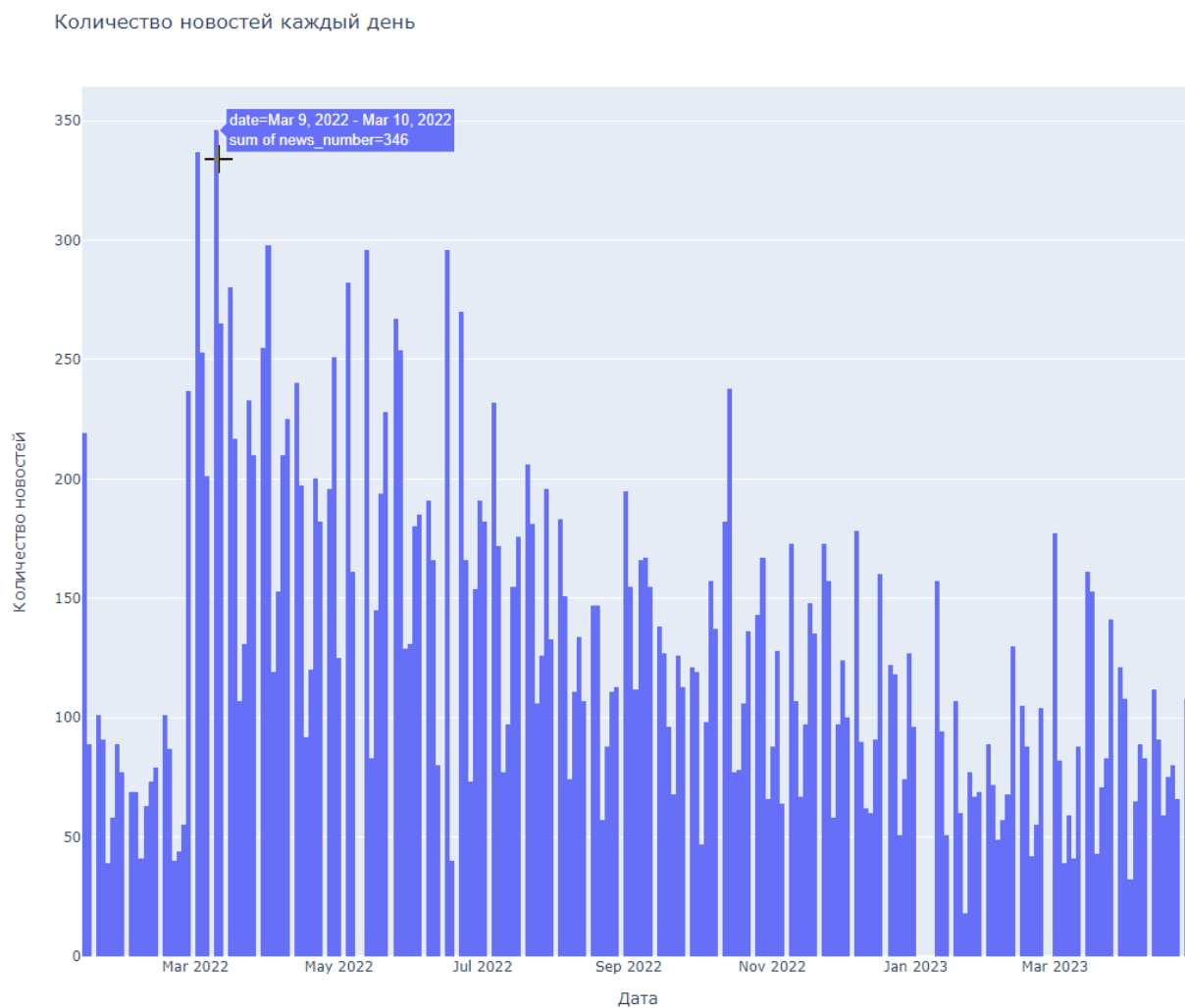


Рисунок 34 — Гистограмма распределения количества новостей по дням изменения курса

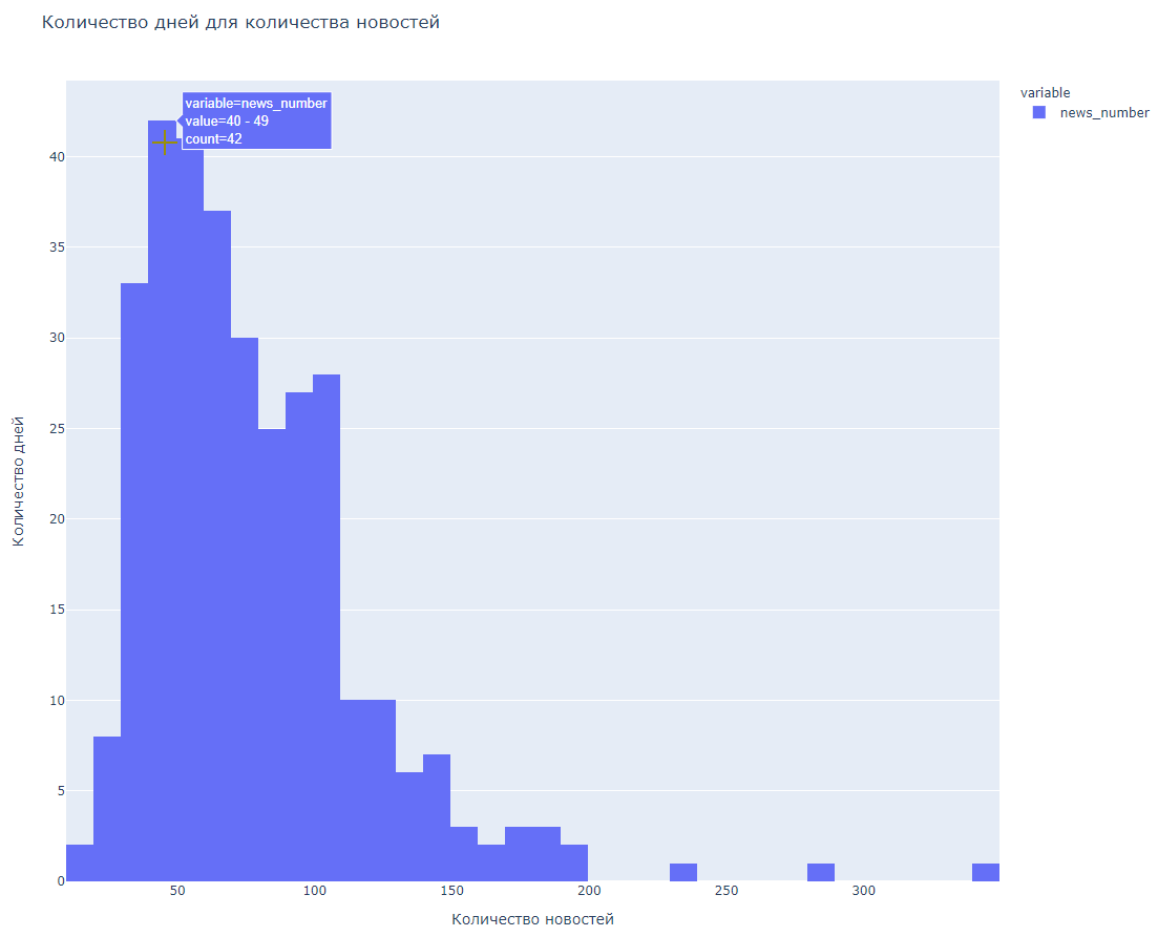


Рисунок 35 — Гистограмма распределения количества дней для количества новостей в день

Проанализируем базовые статистические показатели распределения количества новостей в день (Рисунок 36). Минимальное количество новостей за день 18, вроде не мало, так что вторая модель будет обучаться без пропусков, по 18 самым просматриваемым новостям за день. Ещё возможно стоит сделать модель для медианного значения, с 69 новостями за день.

	news_number
Количество значений	322.000000
Количество уникальных значений	122.000000
Сумма всех	25081.000000
Среднее значение	77.891304
Медианное значение	69.000000
Максимальное значение	346.000000
Минимальное значение	18.000000

Рисунок 36 — Статистические показатели распределения количества новостей в день

Теперь рассмотрим, как происходит объединение самых просматриваемых новостей по дням (Листинг 9). В начале генерируется список названий будущих колонок для новостей, добавляю пустые столбцы с ними к датафрейму с датами и количеством новостей в дату. После, для каждого дня достаю все новости, сортирую их по количеству просмотров, и добавляю нужное или возможное количество наиболее просматриваемых новостей в соответствующие ячейки подготовленного ранее датафрейма.

Листинг 9 — Объединение самых просматриваемых новостей по дням

```
# Задание интересующего количества самых популярных новостей за день
number_of_top = 40
#number_of_top = 18
file_name_ending = str(number_of_top)+'.csv'

# Создание списка названия колонок
list_tops = []
for i in range(number_of_top):
    list_tops.append('top'+str(i+1))

# Объединение самых просматриваемых новостей по дням
if flag_make_news_top_n:
    # В большинство групп дней было опубликовано 40 и более новостей

    empty_cols = pd.DataFrame(columns=list_tops)
    news = pd.concat([news, empty_cols])

# Добавление самых просматриваемых новостей за каждый день в news
for d in list_of_dates:
    # Достаю записи за день
    news_one_date = news_all[news_all['date'] == d]
    # Сортирую новости по количеству просмотров и обновляю индексы
    news_one_date = news_one_date.sort_values(by='views',
                                              ascending=False).reset_index(drop=True)

    #print(news_one_date)

# Количество топ новостей за день
number_of_top_in = number_of_top
nn = len(news_one_date)
if nn < number_of_top_in:
    number_of_top_in = nn

# Добавление самых просматриваемых новостей в news
for i in range(number_of_top_in):
    top = 'top'+str(i+1)
    news.loc[news['date'] == d, top] = news_one_date.name.loc[i]

# Сохранение результата в файл
```

```
news.to_csv('news_top'+file_name_ending, index=False, sep=";",
            encoding='utf-8-sig')
```

На рисунке ниже представлена часть полученного файла news_top40.csv (Рисунок 37).

	date	news_number	top1	top2	top3	top4	top5	top6	top7	top8	...	top31
0	12.01.2022	189	ДНР начала поставлять в Россию колбасу	Пушков назвал условие для транзита газа	Daily Express: Путин переиграл Европу в ситуац...	Не "Северным потоком" единым. Что еще с размах...	Россия не обязана поставлять весь газ в Европу...	На Украине допустили "воровство" российского газа...	Запад встревожен из-за сделки России и Китая п...	Байден дал возможность России зарабатывать три...	...	Fitch расскзало, как "Сила Сибири – 2" может ...
1	13.01.2022	30	Путин предложил проиндексировать пенсии выше и...	Покупать или подождать: что будет с ценами на ...	Допрасходы на индексацию пенсий в 2022 году со...	Аналитик назвал срок, на который имеет смысл п...	Премьер Молдавии рассказала о переговорах с Т...	"Молдовагаз" предупредил о рисках прекращения ...	Молдавия попросит "Газпром" об отсрочке оплаты...	Продажи обновленной Lada Vesta начнутся к сере...	...	NaN
2	14.01.2022	42	В ФРГ назвали истинную причину задержки с запу...	В Госдуму внесен законопроект о закрытии дел д...	Путин увел Россию с "тупого" пути развития, за...	Греф заявил о готовности к возможным санкциям ...	Курс рубля снизился после заявлений Рябкова	В Литву прибыл новый танкер со сжиженным приро...	Шторм на рынке труда: у кого самые высокие зарп...	Боррель высказался о сертификации "Северного п...	...	Бюджет по итогам 2021 года исполнен с профицит...
3	15.01.2022	47	В Польше признали потери от "капризной" борьбы...	WSJ: Европа откажется от жестких санкций проти...	"Газпром" подал в суд на польскую компанию PGNiG	Мясо, овощи, хлеб: какие еще продукты вскоре п...	Иллюзии об экономическом сотрудничестве с Запа...	Первый зампред ЦБ Швецов покинет пост	"Газпром" сообщил о ремонте газопровода "Валда...	Цены на недвижимость в Ливане упали на 70%	...	Правительство готовит план поддержки системы з...
4	18.01.2022	58	Китай украл у Америки самое ценное	Поворот на Север: Россия отказалась выбирать м...	"Пора спуститься с небес на землю": поляки под...	Захарова оценила запрос Германии к Нидерландам...	Из списка поручений Путина уберут пункты по за...	Холодильник с телевизором схлестнулись в смерт...	"Газпром" не забронировал мощности для транзит...	По тормозам: когда ждать падения цен на машины	...	Додон предложил меры по выходу Молдавии из газ...
...
317	22.04.2023	35	Forbes опубликовал рейтинг самых обедневших ми...	США решили снова подключить Россию к SWIFT. Эк...	Выходят из-под контроля. Украина снова подстав...	СМИ: запрет на экспорт в Россию авто отразится...	Сбербанк по итогам 2022 года выплатит рекордные...	Посол Украины в Турции призвал страну искать а...	Банк ТВС предупредил об изменениях в связи с а...	В Ленинградской области рассказали о работе за...	...	Иранские бизнесмены предложили создать в Астра...
318	25.04.2023	61	Медведев прокомментировал идею G7 о запрете эк...	СМИ: ЦБ призвал коллекторов и банки скорординир...	Силуанов объяснил, почему Россия перешла на ра...	СМИ: Минфин и Центробанк завершили проект о до...	В Южной Осетии назвали размер долга перед Росс...	Песков считает, что Россия будет развиваться т...	Министры сельского хозяйства G7 призвали к див...	Henkel рассчитывает на обратный выкуп активов	Эксперт оценил запасы оборудования в попавших ...

Рисунок 37 — Фрагмент файла news_top40

Далее ко всем заголовкам были применены базовые методы очистки текстов от лишнего: избавление от знаков препинания, цифр, переносов строк, замена 'ё' на 'е', токенизирование (разделение текста на слова), удаление стоп слов (слова, не несущие смысловой нагрузки, местоимения, частицы) и была учтена статистическая мера TF-IDF оценки важности слов на основе частот их появления.

Кроме того, были проведены попытки улучшения результатов работы модели методом лемматизации текста (приведение слов к леммам, начальным формам, к именительному падежу единственному числу) и методом выделения n-грамм (последовательностей из n слов). Но эти методы не улучшили точность предсказания моделей на тестовой выборке.

Изобразим наиболее часто встречаемые слова в заголовках новостей в облаке слов (Рисунок 38).


```

model_names = ["Logistic Regression", "Support Vector Classifier",
               "Random Forest", "Gradient Boosting",
               "K-Nearest Neighbors", "Decision Tree"]

# Обучение каждой модели на train и получение предсказаний для test
for model, model_name in zip(models, model_names):
    model.fit(train_data, train_labels)
    predictions = model.predict(test_data)

    # Calculate evaluation metrics for each model
    accuracy = accuracy_score(test_labels, predictions)
    precision = precision_score(test_labels, predictions)
    recall = recall_score(test_labels, predictions)
    f1 = f1_score(test_labels, predictions)

    # Добавление результатов в общую таблицу
    i = len(prediction_results.index)
    prediction_results.loc[i] = [model_name, accuracy*100, precision*100,
                                recall*100, f1*100]

print(prediction_results)

# Сохранение таблицы в файл
prediction_results.to_csv(res_file_name, index=False, sep=";",
                          encoding='utf-8-sig')

# Выделение только заголовков новостей
news_text = news[list_tops]
# Объединение всех текстов в один string для каждой строки
text_data = news_text.apply(lambda x: " ".join(str(x) for x in x), axis=1)
# Конвертирование строки к численным данным
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text_data)

# обучение и проверка работы
predict_sklearn(X, 'USD_ch_bi', 0.2, 'pred_res_USD_bi_top'+file_name_ending)
predict_sklearn(X, 'EUR_ch_bi', 0.2, 'pred_res_EUR_bi_top'+file_name_ending)
predict_sklearn(X, 'CNY_ch_bi', 0.2, 'pred_res_CNY_bi_top'+file_name_ending)

```

Результаты предсказания всех курсов валют средствами scikit-learn на примере курса USD (Рисунок 39). В моделях, реализованных для USD и EUR точности Accuracy (доля правильных ответов) варьируются от 50 до 70 процентов и значение F-меры (F1 Score — среднее гармоническое precision и recall) в среднем немного больше от 55 до 75. А результаты предсказания курса CNY вновь оказались менее точными (Рисунок 40), хотя все три модели обучались аналогично друг другу. В среднем лучше всех себя показал градиентный бустинг.

pred_res_USD_bi_top18					
	Models - USD_ch_bi - top18	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	61.538462	62.500000	60.606061	61.538462
1	Support Vector Classifier	63.076923	59.574468	84.848485	70.000000
2	Random Forest	49.230769	50.000000	63.636364	56.000000
3	Gradient Boosting	49.230769	50.000000	60.606061	54.794521
4	K-Nearest Neighbors	63.076923	62.162162	69.696970	65.714286
5	Decision Tree	61.538462	62.500000	60.606061	61.538462

pred_res_USD_bi_top40					
	Models - USD_ch_bi - top40	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	64.615385	64.705882	66.666667	65.671642
1	Support Vector Classifier	70.769231	65.909091	87.878788	75.324675
2	Random Forest	66.153846	67.741935	63.636364	65.625000
3	Gradient Boosting	66.153846	66.666667	66.666667	66.666667
4	K-Nearest Neighbors	49.230769	50.000000	33.333333	40.000000
5	Decision Tree	63.076923	66.666667	54.545455	60.000000

Рисунок 39 — Результаты предсказания курса USD средствами scikit-learn

pred_res_CNY_bi_top18					
	Models - CNY_ch_bi - top18	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	60.000000	60.869565	45.161290	51.851852
1	Support Vector Classifier	50.769231	42.857143	9.677419	15.789474
2	Random Forest	47.692308	28.571429	6.451613	10.526316
3	Gradient Boosting	61.538462	66.666667	38.709677	48.979592
4	K-Nearest Neighbors	52.307692	50.000000	45.161290	47.457627
5	Decision Tree	46.153846	43.750000	45.161290	44.444444

pred_res_CNY_bi_top40					
	Models - CNY_ch_bi - top40	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	58.461538	56.666667	54.838710	55.737705
1	Support Vector Classifier	63.076923	76.923077	32.258065	45.454545
2	Random Forest	61.538462	80.000000	25.806452	39.024390
3	Gradient Boosting	69.230769	72.000000	58.064516	64.285714
4	K-Nearest Neighbors	53.846154	51.428571	58.064516	54.545455
5	Decision Tree	60.000000	60.869565	45.161290	51.851852

Рисунок 40 — Результаты предсказания курса CNY средствами scikit-learn

Применение глубокого обучения

Задача предсказания повышения или понижения курса валют по новостным заголовкам является задачей классификации. Упростим данные, преобразовав все слова в целые числа в зависимости от их частоты вхождения, чтобы в дальнейшем один из слоев нейросети (Embedding) преобразовал их в векторные представления (Листинг 11).

Листинг 11 — Подготовка текстовых данных для слоя Embedding

```
max_features = 10000
#max_features = total_words
#embedding_dim = 16
embedding_dim = total_words

# Выделение наиболее часто появляющихся слов
results = Counter()
df['name_cleaned'].str.split().apply(results.update)
vocabulary = [key[0] for key in results.most_common(max_features)]

# Создание токенайзера на основе частот вхождения
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(vocabulary)

# Конвертирование всех слов в числовые int значения
X = tokenizer.texts_to_sequences(df['name_cleaned'].values)
X = pad_sequences(X)

max_input_length = X.shape[1]

return X, y, num_classes, embedding_dim, max_input_length
```

В данной последовательной Sequential модели ключевыми являются сверточные слои Conv1D, одномерные слои для обработки последовательностей. Они используются в стеке со слоями MaxPooling1D, для уменьшения размера ввода и извлечения информации, получаем следующую модель (Рисунок 41). Объект history сохраняет в себе колбеки со всех эпох обучения.

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 139, 2076)	20760000
dropout (Dropout)	(None, 139, 2076)	0
conv1d (Conv1D)	(None, 139, 64)	664384
max_pooling1d (MaxPooling1D)	(None, 69, 64)	0
dropout_1 (Dropout)	(None, 69, 64)	0
conv1d_1 (Conv1D)	(None, 69, 64)	28736
max_pooling1d_1 (MaxPooling1D)	(None, 34, 64)	0
dropout_2 (Dropout)	(None, 34, 64)	0
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)	0
dropout_3 (Dropout)	(None, 64)	0
dense (Dense)	(None, 2)	130

=====
Total params: 21,453,250
Trainable params: 21,453,250
Non-trainable params: 0

Рисунок 41 — Сводное представление модели со сверточными (Conv1D) слоями

Результаты прогнозирования

Для визуализации результатов разделим значения, предсказанные успешно от ошибочных (Листинг 12).

Листинг 12 — Визуализация предсказаний модели относительно реальных значений

```
if flag_plot_preds:
    y = y_test.reset_index(drop=True)
    true_preds = pd.DataFrame(columns=['x', 'y'])
    false_preds = pd.DataFrame(columns=['x', 'y'])
    for i in range(len(X_test)):
        if y[i] == pred[i]:
            true_preds.loc[len(true_preds.index)] = [i, pred[i]]
        else:
            false_preds.loc[len(false_preds.index)] = [i, pred[i]]

    plt.plot(true_preds.x[0], true_preds.y[0], '.g', label='Успешные предсказания')
    plt.plot(false_preds.x[0], false_preds.y[0], '.r', label='Ошибочные предсказания')

    plt.plot(true_preds.x, true_preds.y, '.g')
    plt.plot(false_preds.x, false_preds.y, '.r')
```



```
plt.title(f'Предсказания модели {model_name}')
plt.xlabel('День')
plt.ylabel('Понижилась (0) или повысилась (1) цена')
plt.legend()

plt.show()
```

В результате для модели предсказания повысится или понизится значение курса доллара (USD) по новостям, обученной на 30 эпохах, получена следующая картина (Рисунок 42). Визуально нельзя сказать, что один из классов существенно преобладает, аналогичная картина наблюдалась и с прочими моделями.

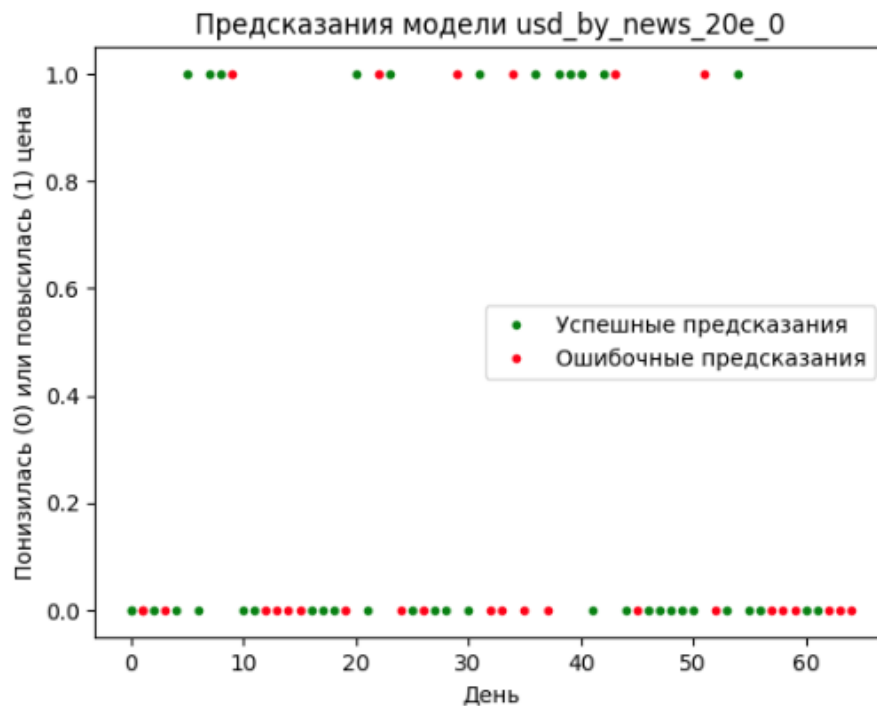


Рисунок 42 — Визуализация предсказаний модели

Для оценки результатов также вычислим основные метрики с помощью `accuracy_score` и `classification_report` из библиотеки `sklearn`. В результате мы видим, что успешные предсказания преобладают, Аккуратность = 58%, F-мера — среднее гармоническое между `precision` и `recall` — для обоих классов находится около 60%. (Рисунок 43).

```

Epoch 29/30
26/26 [=====] - 10s 382ms/step - loss: 0.0022
Epoch 30/30
26/26 [=====] - 10s 381ms/step - loss: 0.0023
3/3 [=====] - 0s 65ms/step

```

```

Results
Accuracy 0: 0.5846

```

	precision	recall	f1-score	support
0	0.57	0.62	0.60	32
1	0.60	0.55	0.57	33
accuracy			0.58	65
macro avg	0.59	0.59	0.58	65
weighted avg	0.59	0.58	0.58	65

Рисунок 43 — Оценка результатов работы модели со сверточными слоями

Кроме того, в качестве проверки были построены графики изменения потерь (loss) и точности (accuracy) в процессе обучения (Рисунок 44). На полученных графиках наблюдается эффект переобучения. Точность на обучающих графиках линейно растет и достигает 100 процентов, потери достигают минимума примерно к 15 эпохе, при этом, точность на тестовых данных находится вокруг 60%.

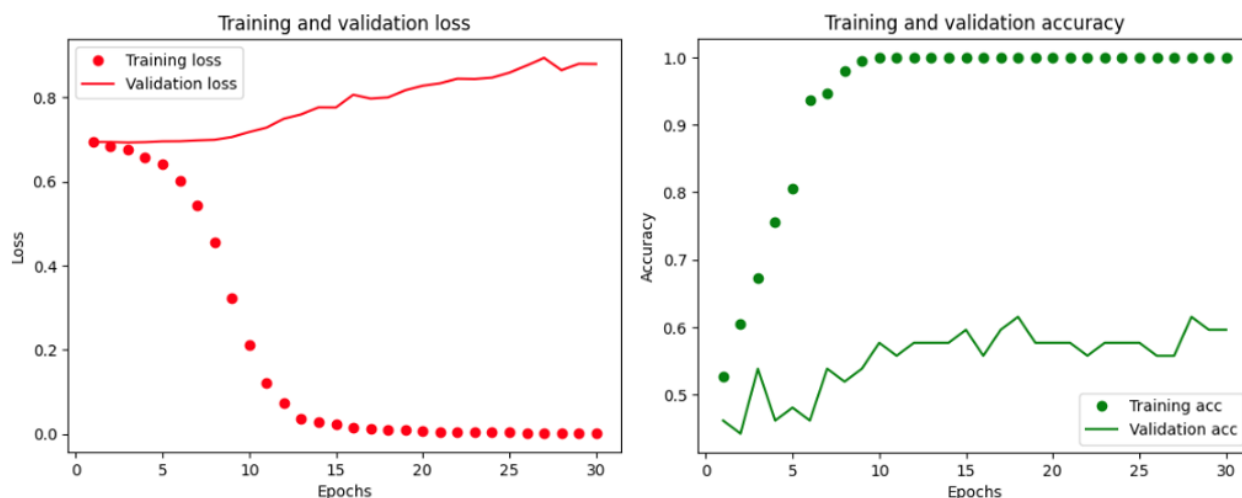


Рисунок 44 — Графики изменения потерь (loss) и точности (accuracy) в процессе обучения

3.4. Подсистема взаимодействия с пользователем

Полную программу можно найти в Приложении Г.

3.4.1. Запуск Telegram-бота

После создания бота в Телеграме, BotFather выдает уникальный API токен, это артефакт, позволяющий пройти аутентификацию и контролировать бот программно.

Запуск сервера бота средствами Aiogram (Листинг 13).

Листинг 13 — Запуск сервера бота

```
API_TOKEN = 'API_TOKEN_API_TOKEN_API_TOKEN_API_TOKEN_API_TOKEN'

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

class Form(StatesGroup):
    currencyState = State()
```

3.4.2. Описание базовых возможностей

Пример описания простой команды help, возвращающей текст, а также это список доступных возможностей бота (Листинг 14).

Листинг 14 — Описание команды help

```
# команда help
help_message = '''
    Доступные команды:
    /start - приветствие
    /help - список команд

    /links - полезные ссылки
    /kurs_now [интересующая валюта]- нынешний курс интересующей валюты,
        вызов без параметра выдаст инлайн кнопки для популярных валют
    /kurs [интересующая валюта интересующая дата] -
        предсказание курса валют на указанную дату,
        вызов без параметра выдаст предсказание на завтра
'''

@dp.message_handler(commands=['help'])
async def process_help_command(message: types.Message):
    await message.reply(help_message)
```

И сразу прилагаю как это выглядит в боте (Рисунок 45).

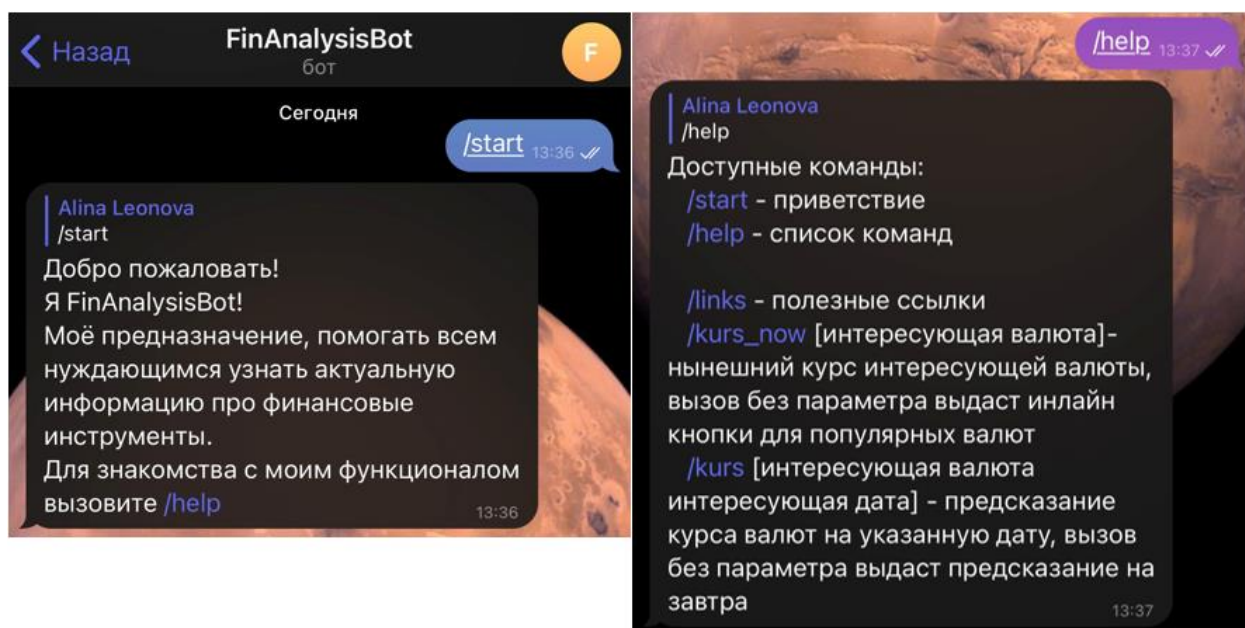


Рисунок 45 — Вызов команд start и help

3.4.3. Команда kurs_now

Команда kurs_now для нахождения и вывода нынешнего курса интересующей валюты (Листинг 15). Функция currentCurrency достаёт по ссылке html-код страницы с сегодняшними значениями валют, находит там упоминание интересующей валюты, достаёт и возвращает актуальное значение курса. При вызове команды без параметров, бот ответит кнопками внутри области диалога InlineKeyboardButton для трёх наиболее актуальных валют: USD, EUR, CNY. Нажатие на эти кнопки передает в возвращаемую (CallbackData) переменную соответствующее название валюты, и запускает для него функцию currentCurrency и бот отправляет сообщение с результатом. Вызов команды с параметром сразу запускает для него функцию currentCurrency и бот отправляет результат.

Листинг 15 — Описание команды kurs_now

```
# находит нынешний курс запрошенной валюты
def currentCurrency(currency):
    url = 'https://www.cbr.ru/currency_base/daily/'

    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')

    # убирает все скобочки, тэги и прочую ерунду
    cleanedSoup = html_stripper(soup.findAll('table', class_='data'))
    listCleanedSoup = cleanedSoup.split('\n\n')

    if __name__ == '__main__':
        print('Курс ' + currency)
```

```

f = False
all_res = ''

for el in listCleanedSoup:
    if re.findall(currency.lower(), el.lower()) != []:
        res = el.split('\n')
        all_res += res[2] + ' ' + res[4] + ': ' + res[5] + ' к ' + res[3] + '\n'
        f = True

if f:
    return all_res
else:
    return 'Такая валюта не найдена'

kurs_cb = CallbackData("currency_prefix", 'currency')

def get_kb():
    kurs_now = InlineKeyboardMarkup(row_width=1)
    knButton1 = InlineKeyboardButton(text=emoji.emojize("us") + ' USD',
                                     callback_data=kurs_cb.new(currency = 'USD'))
    knButton2 = InlineKeyboardButton(text=emoji.emojize("eu") + ' EUR',
                                     callback_data=kurs_cb.new(currency = 'EUR'))
    knButton3 = InlineKeyboardButton(text=emoji.emojize("cn") + ' CNY',
                                     callback_data=kurs_cb.new(currency = 'CNY'))
    kurs_now.add(knButton1, knButton2, knButton3)
    return kurs_now;

# команда kurs_now
@dp.message_handler(commands='kurs_now')
async def url_command(message: types.Message):
    if int(len(message.text)) < 10:
        await message.answer('Узнать нынешний курс', reply_markup=get_kb())
    else:
        print(message.text)
        await message.answer(text = currentCurrency(message.text[10:]))

# обработка колбэков
@dp.callback_query_handler(kurs_cb.filter())
async def kbhandler(query: types.CallbackQuery, callback_data: dict):
    await bot.send_message(text = currentCurrency(
        callback_data.get("currency")), chat_id=query.from_user.id)

```

Команда `kurs` для возвращения предсказания значения запрошенного курса валют на указанную дату, вызов без параметра выдаст предсказание на завтра.

Для повышения эффективности работы Telegram-бота, он связывается с базой данных. И только если в базе не находится результата для запрашиваемой команды,

вызывается предсказание, результат которого отправляется в базу и печатается пользователю. Тестирование возможностей работы команды в боте (Рисунок 46).

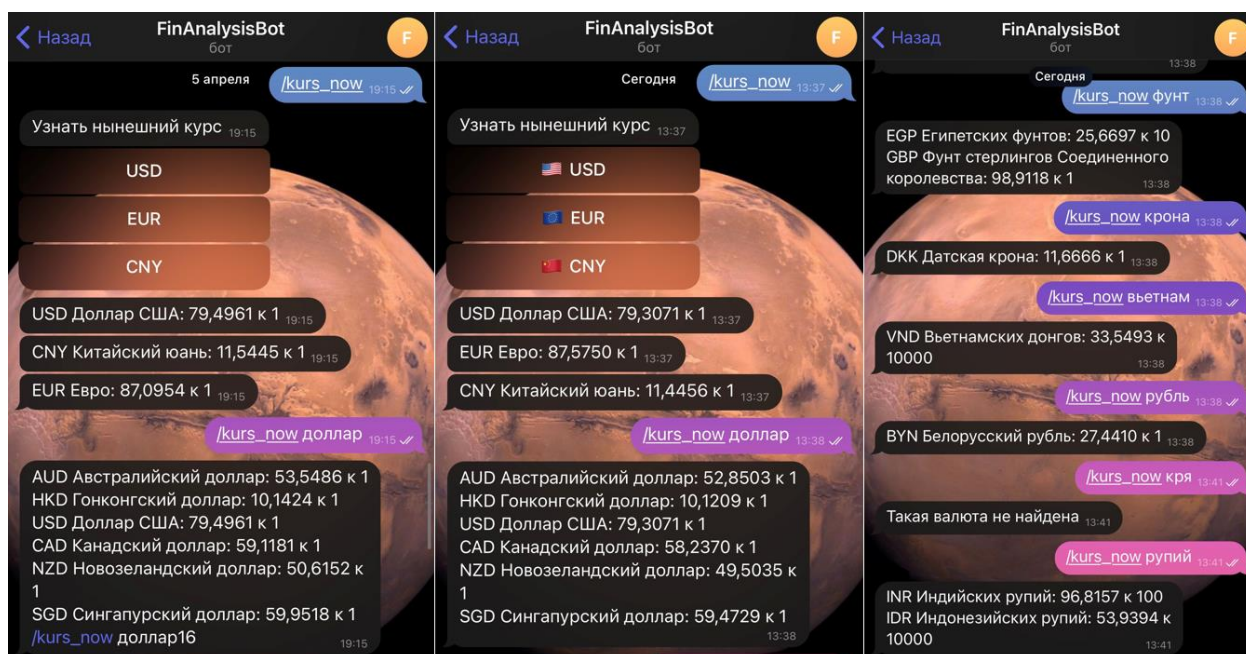


Рисунок 46 — Работа команды *kurs_now*

3.5. Оценка результатов и перспективы развития

В результате работы были выполнены все поставленные задачи.

Была разработана архитектура системы финансового информирования и был создан её прототип, включающий в себя все запланированные модули для предсказания изменения курса валют, от сбора данных, создания и обучения моделей до использования результатов в популярном мессенджере.

Кроме того, были созданы разные виды моделей машинного обучения для предсказания изменения акций десятка компаний и была произведена попытка учесть хаотичное множество факторов с помощью применения технологии обработки естественного языка к новостным заголовкам.

Направления возможного развития:

- Провести дополнительные эксперименты с гиперпараметрами модели: с числом слоёв, числом нейронов в слое, с функциями активации и т.д.;
- Добавление обработки новых финансовых инструментов во все подсистемы.
- Добавление новых возможностей в систему финансового информирования.

Заключение

В работе были рассмотрены вопросы использования методов и средств глубокого обучения для оценки финансовых инструментов.

Проведено исследование финансовых инструментов: рассмотрена общепринятая классификация финансовых инструментов, их ключевые характеристики и методы их анализа. Подготовлен обзор видов нейронных сетей, выделены методы, подходящие для работы с финансовыми инструментами.

Практические результаты работы включают разработку архитектуры системы финансового информирования, создание прототипа системы, реализацию автоматизированного сбора интересующих данных, создание моделей машинного обучения и нейросетевых моделей для предсказания курса акций нескольких компаний и предсказания курса наиболее актуальных валют к российскому рублю, разработку средств совершенствования моделей с использованием технологии обработки естественного языка, а также создание Telegram-бота в качестве пользовательского интерфейса.

Список используемых источников

1. Леонова А.Д. Применение глубокого обучения для предсказания цен акций // Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем: материалы Всероссийской конференции с международным участием. Москва, РУДН, 17–21 апреля 2023 г. – Москва : РУДН, 2023.
2. Макшанова А. Производные финансовые инструменты: понятие, виды и основные стратегии использования. – 2014.
3. Кравченко В., Абрамов А. О международном стандарте финансовых инструментов. – 1999.
4. Securities and related financial instruments — Classification of Financial Instruments (CFI code).
5. ISO 10962 [Электронный ресурс]. – URL: https://en.wikipedia.org/wiki/ISO_10962.
6. Зайцев А.В. Основы финансового инструментария. – г. Сумы, 2019.
7. Thabet A. Анализ временных рядов, применение нейросетей [Электронный ресурс]. – URL: <https://habr.com/ru/articles/693562/>.
8. Piskarev D. Вычисление коэффициента Херста [Электронный ресурс]. – URL: <https://www.mql5.com/ru/articles/2930>.
9. Анализ и модели временных рядов [Электронный ресурс]. – URL: <https://www.statmethods.ru/statistics-metody/modeli-vremennykh-ryadov/>.
10. 700 миллионов пользователей и Telegram Premium [Электронный ресурс]. – URL: <https://telegram.org/blog/700-million-and-premium/ru>.
11. Telegram APIs [Электронный ресурс]. – URL: <https://core.telegram.org/api#bot-api>.
12. Машинное обучение [Электронный ресурс]. – URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение.
13. Леонова А.Д. Выпускная квалификационная работа бакалавра «Разработка мобильного приложения под iOS с применением технологий машинного обучения». – г. Москва: РУДН, 2021.

14. Функция активации [Электронный ресурс]. – URL: <https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-6-obuchenie-nejrosetej-chast-1-2/>.
15. Волхонский А.Н. Алгоритмы нечеткого и нейронного управления в системах реального времени [Электронный ресурс]. – URL: <https://eduherald.ru/ru/article/view?id=20348>.
16. Розенблатт Ф. Принципы нейродинамики: перцептроны и теория механизмов мозга. – 1965.
17. Understanding LSTM Networks [Электронный ресурс]. – URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
18. IEEE. Top Programming Languages 2022 [Электронный ресурс]. – URL: <https://spectrum.ieee.org/top-programming-languages-2022>.
19. Топ-10 инструментов Python для машинного обучения и data-science [Электронный ресурс]. – URL: <https://habr.com/ru/company/skillbox/blog/420819/>.
20. Что такое Scikit Learn - гайд по популярной библиотеке Python для начинающих [Электронный ресурс]. – URL: <https://datastart.ru/blog/read/chto-takoe-scikit-learn-gayd-po-populyarnoy-biblioteke-python-dlya-nachinayuschih>.
21. Francois C. Deep Learning with Python. – 2023.
22. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. – 2018.
23. Metrics and scoring: quantifying the quality of predictions [Электронный ресурс]. – URL: https://scikit-learn.org/stable/modules/model_evaluation.html.
24. ЦБ РФ – Курсы валют [Электронный ресурс]. – URL: <https://mfd.ru/currency/?currency=USD> (дата обращения: 27.04.2023).
25. РИА Новости. Экономика [Электронный ресурс]. – URL: <https://ria.ru/economy> (дата обращения: 27.04.2023).
26. Azam W. Stock price history TOP 10 companies [Электронный ресурс]. – URL: <https://www.kaggle.com/datasets/mdwaquarazam/stock-price-history-top-10-companies?resource=download>.
27. Sequential model: руководство [Электронный ресурс]. – URL: <https://ru-keras.com/guide-sequential/>.

Приложение А — kurs_table.py

```
import numpy as np

# Объединение данных всех интересующих валют
currency_list = ['USD', 'EUR', 'CNY']

# Формирование имен столбцов сводной таблицы
cols = ['Date']
for k in currency_list:
    cols.append(k)
    cols.append(k+'_ch')
    cols.append(k+'_ch_bi')

kurses = pd.DataFrame(columns=cols)

# currency - значение курса валюты
# currency_ch - изменение значения курса относительно предыдущего дня
# currency_ch_bi - 1 - значение увеличилось, 0 - уменьшилось
flag_is_first = True
for k in currency_list:
    file = 'mfdexport_'+k.lower()+'.csv'
    df_kurs = pd.read_csv(file, ';').iloc[:, -1].reset_index(drop=True)

    # Оставляю только даты, избавляюсь от времени (0:00:00)
    df_kurs['Date'] = df_kurs['Date'].str.split().str[0]
    # Меняю ',' на '.' и формат значений
    for i in range(len(df_kurs)):
        df_kurs.iloc[i, 1] = df_kurs.iloc[i, 1].replace(',', '.', '.')
    df_kurs.iloc[:, 1] = df_kurs.iloc[:, 1].astype('float64')

    # Добавление дат
    if flag_is_first:
        curses['Date'] = df_kurs['Date']
        flag_is_first = False

    curses[k] = df_kurs[k]

# Добавление столбцов изменения значения относительно предыдущего дня
# и производного от него, где 1 - значение увеличилось, 0 - уменьшилось
for i in range(1, curses.shape[0]):
    curses[k+'_ch'][i] = curses[k][i] - curses[k][i-1]

    if curses[k+'_ch'][i] > 0:
        curses[k+'_ch_bi'][i] = 1
    else:
        curses[k+'_ch_bi'][i] = 0

curses.to_csv('curses.csv', index=False, sep=";")
```

Приложение Б — parse_news.py

```
import os
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from random import randint
from alive_progress import alive_bar

from bs4 import BeautifulSoup
import requests
import re

[импорт функций библиотеки selenium: By, WebDriverWait, Select,
ActionChains, expected_conditions, lxml)

#-----

# Убирает все скобочки, тэги и прочую ерунду
def html_stripper(text):
    return re.sub('<[<]+?>', '', str(text))

# Изменение символов, неизвестных выбранной кодировке
def check_for_broken(text):
    text = re.sub(r'ü',r'ue', text)
    text = re.sub(r'ä',r'ae', text)
    text = re.sub(r'ö',r'oe', text)
    text = re.sub(r'ß',r'ss', text)
    text = re.sub(r'ó',r'o', text)
    text = re.sub(r'ø',r'o', text)
    text = re.sub(r'&',r'&', text)
    text = re.sub(r'◆',r'', text)
    text = re.sub(r"^[^w\s]", "", text)
    return text

# Листание страницы вниз
def scroll_page(driver, pause_time):
    # Получение высоты прокрутки
    last_height = driver.execute_script(
        "return document.body.scrollHeight")

    while True:
        # Листание вниз до конца страницы
        driver.execute_script("window.scrollTo(0,
            document.body.scrollHeight);")

        # Ожидание загрузки страницы
        time.sleep(pause_time)
```

```

# Расчет новой высоты прокрутки и сравнение ее с предыдущей
new_height = driver.execute_script(
    "return document.body.scrollHeight")
if new_height == last_height:
    break
last_height = new_height

# Нажатие на кнопку внизу для загрузки дополнительных материалов, если
она есть
def more_news(driver, pause_time):
    driver.find_element(By.XPATH,
        '//*[@id="content"]/div/div[1]/div/div[3]').click()
    time.sleep(pause_time)

# Инициализация и запуск веб драйвера по ссылке
def start_driver(link):
    # опции для игнорирования некоторых ошибок и предупреждений
    options = webdriver.ChromeOptions()
    options.add_experimental_option('excludeSwitches',
        ['enable-logging'])

    # Запускаем драйвер, открываем веб-страницу
    driver = webdriver.Chrome(options=options)
    driver.set_window_size(1500,1000)
    driver.implicitly_wait(randint(5, 10)) # случайная задержка,
                                            в секундах

    driver.get(link)
    return driver

# Выбор периода даты "За год", в итоге не используется
def date_range(driver):
    # открытие выбора периода даты
    driver.find_element(By.XPATH,
        '//*[@id="content"]/div/div[1]/div/div[1]/div[3]').click()
    time.sleep(3)
    # выбора периода даты "За год"
    driver.find_element(By.XPATH,
        '/html/body/div[20]/div[2]/ul/li[3]').click()

# Дополнение таблицы, актуализация данных
def data_actualization(new_start, new_end, df_csv, res_name_csv):
    df_old = pd.read_csv(df_csv, sep=";", encoding='utf-8-sig')
    # Получение новых данных
    parse_all(new_start, new_end,
        'news_last_actualization_add.csv')
    df_new = pd.read_csv('news_last_actualization_add.csv',
        sep=";", encoding='utf-8-sig')

    # Соединение всех данных
    news_result = pd.concat([df_old, df_new])
    # Сохранение обновленной таблицы

```

```

news_result.to_csv(res_name_csv, index=False, sep=";",
                  encoding='utf-8-sig')

#-----

# Парсинг новостей за одну дату, создание супа данных
def parse_by_date(date):
    #print('Дата:', date)
    # создание ссылки новостного архива заданного дня
    link = 'https://ria.ru/economy/' + date + '/'

    driver = start_driver(link)

    # Суп из HTML-кода страницы
    soup = BeautifulSoup(driver.page_source, 'lxml')

    # Выделение количества новостей за день,
    # оно влияет на количество страниц
    number_of_news = int(html_stripper(soup.find('div', class_ =
        'rubric-count m-active').span))

    #print('!!!!!!!!!!!!!!')
    #print('Количество новостей: ', number_of_news)
    #print('!!!!!!!!!!!!!!\n')

    # 20 - максимальное количество новостей на странице ria.ru
    if number_of_news > 20:
        # Листание страницы вниз
        scroll_page(driver, 1)

        # Нажатие на кнопку внизу для загрузки дополнительных
        # материалов, если она есть
        more_news(driver, 5)

        # Листание страницы вниз, дальше новые данные будут
        # загружаться автоматически
        scroll_page(driver, 1)

        # Обновление супа, появились новые новости
        soup = BeautifulSoup(driver.page_source, 'lxml')

    #time.sleep(randint(2, 4))

    # Закрытие драйвера, больше он не нужен
    driver.close()
    # Передача супа данных
    return soup

# Функция для парсинга данных в диапазоне дат [start, end] и
# сохранение таблицы в csv файл
def parse_all(start, end, file_name):

```

```

# Генерирование списка всех интересующих дат для ссылок
dates_list = pd.date_range(start,
                             end).strftime('%Y%m%d').tolist()

total = len(dates_list)
print('\nКоличество дат = ', total, '\n')

# Создание пустого датафрейма для дальнейшего накопления информации
# Столбцы: дата публикации, количество просмотров,
#          заголовок статьи, ссылка на статью
index_list = ['date', 'views', 'name', 'url']
news_table = pd.DataFrame(columns=index_list)

with alive_bar(total) as bar:  # индикатор прогресса
    # Вытаскивание данных для каждой даты
    for date in dates_list:
        # Получение супа данных для даты
        soup = parse_by_date(date)

        # Выделение списка из кусков кода,
        #          относящихся к новостям
        news = soup.find_all(class_="list-item")

        # Добавление всех новостей дня в общую таблицу
        for i in news:
            name = html_stripper(i.find(class_=
                                         "list-item__title color-font-hover-only"))
            name = check_for_broken(name)
            url = i.find(class_="list-item__title
                               color-font-hover-only")['href']
            #date_time = html_stripper(i.find(class_=
                                               "list-item__date"))

            # Получение даты из ссылки
            url_date = re.search(r'\d{8}', url)[0]
            date = url_date[:4]+'-'+url_date[4:6]+'-'
            #          +url_date[6:]

            views = html_stripper(i.find(class_=
                                         "list-item__views-text"))
            #tags = html_stripper(i.find_all(class_=
            "list-tag m-active color-border color-font color-svg m-add"))

            # Добавление новой строки в конец
            news_table.loc[len(news_table.index)] =
                [date, views, name, url]

        bar()  # Изменение индикатора прогресса

print('\nКоличество новостей = ', news_table.shape[0], '\n')

# Сохранение таблицы в файл

```

```

# encoding='cp1251'
news_table.to_csv(file_name, index=False, sep=";",
                  encoding='utf-8-sig')

#-----

# Вызов парсинга данных в диапазоне дат [start, end] и сохранение
# таблицы в csv файл
flag_parse_all = False
if flag_parse_all:
    #file_name = 'economy_news.csv'
    file_name = 'news_table_.csv'
    parse_all('2022-01-01', '2023-03-28', file_name)

# Дополнение старой таблицы, актуализация данных
flag_actualize = False
if flag_actualize:
    old_table_name = 'news_table_all.csv'
    res_file_name = 'economy_news_.csv'
    data_actualization('2023-03-29', '2023-04-28', old_table_name,
                      res_file_name)

# Открытие актуализированной таблицы новостей
economy_news = pd.read_csv('economy_news.csv', sep=";",
                          encoding='utf-8-sig')

print(economy_news)
print('!!!!!!!!!!!!!!!!!!!!')
print(economy_news.info())
print('!!!!!!!!!!!!!!!!!!!!')
print(economy_news.describe())

```

Приложение В — predict_kurs.py

[Импорт стандартных библиотек, моделей и слоев keras, метрик sklearn]

```
#-----

# Открытие истории изменения курса открытия USD, EUR и CNY к рублю
kurses = pd.read_csv('kurses.csv', ';')
print(kurses)
print('\n')

#-----

class Currency:
    def __init__(self, name, courses):
        self.name = name
        self.df = courses[['Date', name]]
        self.scaler = MinMaxScaler(feature_range = (0, 1))

    def make_features_labels(self, data):
        set_features = []
        set_labels = []
        len_df_train = len(data)
        for i in range(30, len_df_train):
            set_features.append(data[i-30:i, 0])
            set_labels.append(data[i, 0])
        return set_features, set_labels

    def prepare_features_labels(self, data):
        data_scaled = self.scaler.fit_transform(data.reshape(-1, 1))
        set_features, set_labels = self.make_features_labels(data_scaled)
        set_features, set_labels = np.array(set_features),
                                   np.array(set_labels)

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

        return set_features, set_labels

    def prepare_currency(self, currency):

        split_point = len(currency) - len(currency) // 10
        #print(len(currency))
        #print(split_point)

        #-----

        currency_train = currency[:split_point]
        currency_test = currency[split_point-30:]
```



```

train_features, train_labels =
    self.prepare_features_labels(currency_train)
test_features, test_labels =
    self.prepare_features_labels(currency_test)

#print('!!! Shapes !!! ', train_features.shape,
      train_labels.shape, test_features.shape, test_labels.shape)

return train_features, train_labels, test_features, test_labels

def train_model_1(self, train_features, train_labels, model_name,
                  units_number):
    # создание
    model = Sequential()

    model.add(LSTM(units=units_number, return_sequences=True,
                  input_shape=(train_features.shape[1], 1)))
    model.add(Dropout(0.2))
    model.add(LSTM(units=100, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(units=100, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(units=100))
    model.add(Dropout(0.2))

    model.add(Dense(units = 1))

    model.compile(optimizer = 'adam', loss = 'mean_squared_error')

    model.summary()

    # - # - # - # - #

    # обучение
    model.fit(train_features, train_labels, epochs = 300,
              batch_size = 32)

    # сохранение в файл для дальнейшего использования
    model.save(model_name)

def predict(self, test_features, test_labels, model):
    predictions = model.predict(test_features)

    predictions = self.scaler.inverse_transform(predictions)
    real_labels = self.scaler.inverse_transform(
        test_labels.reshape(-1, 1))

    return real_labels, predictions

# - # - # - # - #

```

```

# Сравнение реальных значений с предсказанными по графику
# Для одной модели
def plot_preds(self, real_labels, predictions, model_name):
    currency = model_name[6:9].upper()
    plt.figure(figsize=(10,6))
    plt.plot(real_labels, color='blue', label='Реальный курс ' +
                                                    currency)
    plt.plot(predictions , color='red', label='Предсказанный ' +
                                                    model_name)
    plt.title('Сравнение реальных значений курса '+currency+' с
                                                    предсказанными ' + model_name)
    plt.xlabel('Дни')
    plt.ylabel('Курс ' + currency)
    plt.legend()
    plt.show()

```

```

# Сравнение реальных значений с предсказанными по графику
# Для двух моделей
def compare_plot_preds(self, real_labels, predictions_1,
                                                    predictions_2, m_1, m_2):
    currency = m_1[6:9].upper()
    plt.figure(figsize=(10,6))
    plt.plot(real_labels, color='green', label='Реальный курс ' +
                                                    currency)
    plt.plot(predictions_1 , color='blue', label='Предсказанный ' +
                                                    m_1)
    plt.plot(predictions_2 , color='red', label='Предсказанный ' +
                                                    m_2)
    plt.title('Сравнение реальных значений курса '+currency+' с
                                                    предсказанными')
    plt.xlabel('Дни')
    plt.ylabel('Курс ' + currency)
    plt.legend()
    plt.show()

```

```

# - # - # - # - #

```

```

# Сравнение реальных значений с предсказанными
def compare_values(self, real_labels, predictions, flag_print):
    p = predictions[:,0]
    r = real_labels[:,0]

    res = np.array([explained_variance_score(r, p)])
    res = np.append(res, mean_squared_log_error(r, p))
    res = np.append(res, mean_squared_error(r, p))
    res = np.append(res, np.sqrt(mean_squared_error(r, p)))
    res = np.append(res, mean_absolute_error(r, p))
    res = np.append(res, median_absolute_error(r, p))

```

```

res = np.append(res, max_error(r, p))

#print('!!! res модели !!! ', res)

if flag_print:
    print('    explained_variance_score = ', res[0])
    print('    mean_squared_log_error = ', res[1])
    print('    mean_squared_error = ', res[2])
    print('    root_mean_squared_error = ', res[3])
    print('    mean_absolute_error = ', res[4])
    print('    median_absolute_error = ', res[5])
    print('    max_error = ', res[6])

return res

# - # - # - # - #

def predict_check_model(self, test_features, test_labels, model,
                        flag_print):
    real_labels, predictions = self.predict(test_features,
                                           test_labels, model)
    return self.compare_values(real_labels, predictions, flag_print)

# для одной модели
def predict_and_plot(self, test_features, test_labels, model, name):
    real_labels, predictions = self.predict(test_features,
                                           test_labels, model)
    self.plot_preds(real_labels, predictions, name)

# для двух моделей
def predict_and_plot_2(self, test_features, test_labels, model_1,
                      model_2, m_1, m_2):
    real_labels, predictions_1 = self.predict(test_features,
                                           test_labels, model_1)
    real_labels, predictions_2 = self.predict(test_features,
                                           test_labels, model_2)

    self.compare_plot_preds(real_labels, predictions_1,
                           predictions_2, m_1, m_2)

# - # - # - # - #

def do(self, flag_train=False, flag_print_values=False,
      flag_plot=False):
    train_f, train_l, test_f, test_l =
        self.prepare_currency(self.df.iloc[:,1].values)

```

```

#----- LSTM -----

# Функция для создания и обучения моделей по одному принципц

# - # - # - # - #

# Создание и обучение моделей по одному принципу, если надо
if flag_train:
    self.train_model_1(train_f, train_l,
                        f"model_{self.name}_LSTM_100.h5", 100)
    self.train_model_1(train_f, train_l,
                        f"model_{self.name}_LSTM_300.h5", 300)

    print('!!!!!!!!!!!!!! Models got trained !!!!!!!!!!!!!!!')

# Загрузка моделей из файлов
model_LSTM_100 = load_model(f"model_{self.name}_LSTM_100.h5")
model_LSTM_300 = load_model(f"model_{self.name}_LSTM_300.h5")

print('!!!!!!!!!!!!!! Models got loaded !!!!!!!!!!!!!!!')

#-----

# Функция для предсказания значений выбранной моделью по выбранным
                                         тестовым данным

flag_print = False

# Общий вывод всех значений для сравнения работы всех моделей
                                         для валюты
if flag_print_values:
    index_list = ['explained_variance_score',
                  'mean_squared_log_error', 'mean_squared_error',
                  'root_mean_squared_error', 'mean_absolute_error',
                  'median_absolute_error', 'max_error']
    compare_values_dict = { f"{self.name}_LSTM_100":
self.predict_check_model(test_f, test_l, model_LSTM_100, flag_print),
                           f"{self.name}_LSTM_300":
self.predict_check_model(test_f, test_l, model_LSTM_300, flag_print)}
    df_compare_values = pd.DataFrame(compare_values_dict,
                                     index=index_list)

    print('Models')
    print(df_compare_values)

# - # - # - # - #

```

```

    if flag_plot:
        self.predict_and_plot_2(test_f, test_l, model_LSTM_100,
                                model_LSTM_300, f'model_{self.name}_LSTM_100',
                                                f'model_{self.name}_LSTM_300')

#-----

cur_usd = Currency('USD', courses)
cur_eur = Currency('EUR', courses)
cur_cny = Currency('CNY', courses)

# Флаги вызовов do
f_train = True
f_print_values = True
f_plot = True

cur_usd.do(f_train, f_print_values, f_plot)
cur_eur.do(f_train, f_print_values, f_plot)
cur_cny.do(f_train, f_print_values, f_plot)

```

Приложение Г — FinAnalysisBot.py

```
[Импорт стандартных функций библиотеки aiogram]
import emoji
```

```
import requests
import re
from bs4 import BeautifulSoup
```

```
API_TOKEN = 'API_TOKEN_API_TOKEN_API_TOKEN_API_TOKEN_API_TOKEN'
```

```
# Запуск сервера телеграм-бота
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
```

```
class Form(StatesGroup):
    currencyState = State()
```

```
#-----
```

```
# команда help
help_message = '''
    Доступные команды:
    /start - приветствие
    /help - список команд

    /links - полезные ссылки
    /kurs_now [интересующая валюта]- нынешний курс интересующей валюты,
        вызов без параметра выдаст инлайн кнопки для популярных валют
    /kurs [интересующая валюта интересующая дата] - предсказание курса
        валют на указанную дату, вызов без параметра
        выдаст предсказание на завтра
    ...
```

```
@dp.message_handler(commands=['help'])
async def process_help_command(message: types.Message):
    await message.reply(help_message)
```

```
#-----
```

```
# команда start
@dp.message_handler(commands=['start'])
async def send_welcome(message: types.Message):
    await message.reply('Добро пожаловать!\nЯ FinAnalysisBot!
        \nМоё предназначение, помогать всем нуждающимся узнать актуальную
        информацию про финансовые инструменты.
        \nДля знакомства с моим функционалом вызовите /help')
```

```
#-----
```

```

# команда links
links = InlineKeyboardMarkup(row_width=1)
linksButton1 = InlineKeyboardButton(text='Официальные курсы валют',
                                     url='https://www.cbr.ru/currency_base/daily/')
linksButton2 = InlineKeyboardButton(text='Акции и котировки на сегодня',
                                     url='https://www.banki.ru/investment/shares/?source=submenu_share')
linksButton3 = InlineKeyboardButton(text='Вклады',
                                     url='https://www.banki.ru/products/deposits/?source=submenu_deposits')
links.add(linksButton1, linksButton2, linksButton3)

@dp.message_handler(commands='links')
async def url_command(message: types.Message):
    await message.answer('Основные ссылки:', reply_markup=links)

#-----

# убирает все скобочки, тэги и прочую ерунду
def html_stripper(text):
    return re.sub('<[<]+?>', '', str(text))

# находит нынешний курс запрошенной валюты
def currentCurrency(currency):
    # return "???????????????" + currency
    url = 'https://www.cbr.ru/currency_base/daily/'

    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')

    cleanedSoup = html_stripper(soup.findAll('table', class_='data'))
    listCleanedSoup = cleanedSoup.split('\n\n')

    if __name__ == '__main__':
        print('Курс ' + currency)
        f = False
        all_res = ''

        for el in listCleanedSoup:
            if re.findall(currency.lower(), el.lower()) != []:
                res = el.split('\n')
                all_res += res[2] + ' ' + res[4] + ': ' + res[5] + ' к ' + res[3] + '\n'

                f = True

        if f:
            return all_res
        else:
            return 'Такая валюта не найдена'

kurs_cb = CallbackData("currency_prefix", 'currency')

```

```

def get_kb():
    kurs_now = InlineKeyboardMarkup(row_width=1)
    knButton1 = InlineKeyboardButton(text=emoji.emojize("us") + ' USD',
                                      callback_data=kurs_cb.new(currency = 'USD'))
    knButton2 = InlineKeyboardButton(text=emoji.emojize('eu') + ' EUR',
                                      callback_data=kurs_cb.new(currency = 'EUR'))
    knButton3 = InlineKeyboardButton(text=emoji.emojize("cn") + ' CNY',
                                      callback_data=kurs_cb.new(currency = 'CNY'))
    kurs_now.add(knButton1, knButton2, knButton3) # , knButton4
    return kurs_now;

# команда kurs_now
@dp.message_handler(commands='kurs_now')
async def url_command(message: types.Message):
    if int(len(message.text)) < 10:
        await message.answer('Узнать нынешний курс',
                              reply_markup=get_kb())
    else:
        print(message.text)
        await message.answer(text = currentCurrency(message.text[10:]))

@dp.callback_query_handler(kurs_cb.filter())
async def kbhandler(query: types.CallbackQuery, callback_data: dict):
    await bot.send_message(text = currentCurrency(
        callback_data.get("currency")), chat_id=query.from_user.id)

#-----

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)

```