Курсовая работа защищена с оценкой _

Ученый секретарь кафедры Теории вероятностей Ряднова Е.М.

Ученый секретарь кафедры Вычислительной математики Валединский В.Д.

Московский государственный университет имени М.В. Ломоносова

Механико-математический факультет

Кафедра Теории вероятностей

КУРСОВАЯ РАБОТА

Прогнозирование финансовых временных рядов с помощью нейронных сетей с памятью

Выполнена студенткой 309 группы

Маховой Анастасией Геннадьевной

Научный руководитель:

д.ф.-м.н М.И. Кумсков

Москва, 2023

Аннотация

В работе предложена модель нейросети с памятью (LSTM), прогнозирующая тренд финансового временного ряда на примере цен акций компании Apple, дан обзор на оснновные теоритические вопросы, возникающие по данной теме, проведены вычислительные эксперименты по поиску оптимальных параметров нейросети, изучено влияние тиков временного ряда и длины тренда на точность прогноза и проведено сравнение результатов работы модели с классическими методами прогнозирования временных рядов.

Содержание

Введение **4**

1 Задачи прогнозирования финансовых временных рядов **5**

1.1 Постановка задачи 6

2 Временные ряды **7**

2.1 Определения 7

2.2 ETS models 8

2.3 Методы прогнозирования временных рядов 9

2.3.1 AR(p) 9

2.3.2 MA(q) 10

2.3.3 ARMA(p,q) 10

2.3.4 ARIMA(p,d,q) 10

2.3.5 SARIMA 10

3 LSTM **11**

3.1 Архитектура 11

3.1.1 слой фильтра забывания 14

3.1.2 слой входного фильтра 14

3.1.3 обновляем состояние ячейки 14

3.1.4 слой выходного фильтра 15

3.1.5 Общая формула 15

4 Модель **16**

4.1 Функция потерь 16

4.1.1 Кросс-энтропия 17

4.1.2 Бинарная классификация 17

4.1.3 Мульти-классовая классификация 18

4.2 Алгоритм оптимизации 18

4.2.1 Stochastic Gradient Descent (SGD) 18

4.2.2 Momentum 20

4.2.3 Nesterov's accelerated gradient method . . . 20
4.2.4 AdaGrad . . . 20
4.2.5 RMSProp . . . 21
4.2.6 Adam . . . 21
4.3 Метрика оценки качества . . . 22
4.3.1 Accuracy . . . 24
4.3.2 Precision . . . 25
4.3.3 Recall . . . 25
4.3.4 F-measure . . . 25
4.4 Функции активации . . . 26
Вывод . . . 26

5 Вычислительные эксперименты **27**
5.1 Параметры LSTM в TensorFlow . . . 27
5.2 Как влияет units на прогноз? . . . 30
5.3 Как влияет timesteps на прогноз? . . . 33
5.4 Как влияет тик на прогноз? . . . 35
5.5 Как влияют features на прогноз? . . . 36
5.6 Как влияет вид модели на прогноз? . . . 37

6 Сравнение с альтернативными методами **39**

Заключение **41**

Список литературы **42**

Введение

Целью данной работы является построение модели для предсказания тренда временного ряда с помощью нейронных сетей с памятью, изучение параметров нейросети, влияющих на прогноз, и зависимости результатов работы модели от тика временного ряда.

Был решен ряд задач: изучение теории о временных рядах и их прогнозировании, о устройстве нейросетей, функциях потрерь, алгоритмах оптимизации и метриках оценки качества, а также о архитектуре LSTM-нейросетей.

В результае была создана модель, предсказывающая тренд финансового временного ряда на примере цен акций Apple, проведены вычислительные эксперименты с целью поиска параметров для наилучшего прогноза, проведено сравнение с широко известными методами из теории случайных процессов (ARIMA).

В главе 1 дан обзор на существующие варианты задач прогнозирования финансовых временных рядов, строго сформулировано условие задачи прогнозирования тренда.

Глава 2 посвящена временным рядам, определениям и следующим методам их анализа и прогнозирования: ETS, AR, MA, ARIMA и др.

Глава 3 содержит подробное описание архитектуры LSTM модели.

В главе 4 дан обзор на варианты функций потерь, алгоритмов оптимизации, метрик оценки качества и функций активации - важнейших компонентов, необходимых для работы нейросети. Кроме того, в этой главе отмечены те функции и алгоритмы, которые подходят для решения задачи прогноза тренда временного ряда.

В главе 5 проведены вычислительные эксперименты, приведены рассуждения о настройках нейросети, даны ответы на вопросы из постановки задачи.

Глава 6 посвящена сравнению результатов работы полученной LSTM-модели с методом ARIMA.

В Заключении подведены итоги, сформулированы возможные направления развития задачи.

1 Задачи прогнозирования финансовых временных рядов

Задачи о прогнозировании финансовых временных рядов можно разделить на 2 группы в зависимости от ожидаемых выходных данных[3]:

1. прогноз цены - необходимо предсказать стоимость на некоторое время вперед как можно точнее.
2. прогноз тренда - направления движения графика стоимости:
   (a) 2-class problem – предсказать восходящий и нисходящий тренд
   (b) 3-class problem - предсказать восходящий, нисходящий и боковой тренды

Входные данные для прогнозирования можно брать из:

- Цен за предыдущие периоды

  «цены устанавливаются отнюдь не на основе объективных показателей, таких как рентабельность продаж или прибыль. Курс акций может вдруг резко подскочить, но связано это будет лишь с улучшением ожиданий инвесторов, а вовсе не с ростом продаж, рентабельности или прибыли компании»

  (c)Роджер Мартин

То есть на цены влияет и эмоциональный аспект, который можно отследить с помощью:

- Фундаментального анализа

  Благодаря методам фундаментального анализа можно проанализировать справедливую стоимость на данный момент и предсказать повышение/понижение спроса

- Технического анализа

  Анализ графиков на характерные предпосылки к изменению направления движения стоимости. На рисунке 1 изображены основные виды графиков, которые используются для отображения цен.

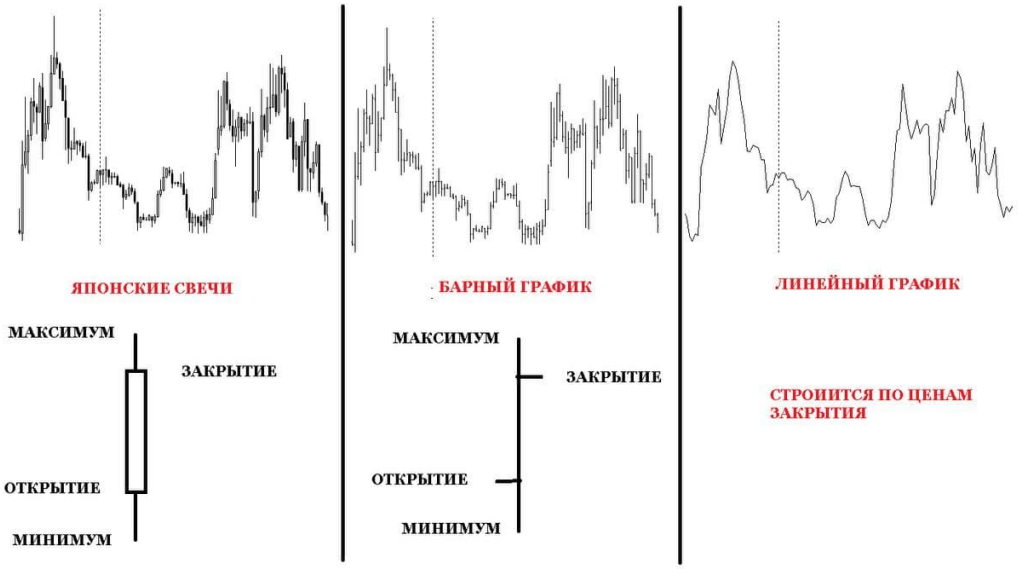


Рис. 1: виды графиков

- Текстовой информации

  Определение эмоционального настроя и мнений на основе постов и комментариев в соцсетях также помогает спрогнозировать поведение инвесторов.

1.1 Постановка задачи

Дан финансовый временной ряд $X = \{X_1, X_2, ..., X_N\}$

Составим прогноз тренда основываясь только на ценах, то есть $X_i \in \mathbb{R}^n_+$

Решим задачу классификации: сопоставим временному интервалу $[t_M, t_L]$, $N < M \leq L$ вектор со значениями из $\mathbb{Z}_3$, так как тренд бывает трех типов: восходящим, нисходящим и боковым.

2 Временные ряды

2.1 Определения

Дадим несколько определений, связанных с временными рядами:

Временной ряд - это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные промежутки.

t - настоящее время,

$t-1, t-2, t-3\ldots$ - прошлое,

$t+1, t+2, t+3\ldots$ -будущее

Лаг – временной период из прошлого (задержка), например, лаг h соответствует $t-h$, где h, характеризующая разницу во времени между элементами временного ряда, называется лаговой переменной или запаздыванием.

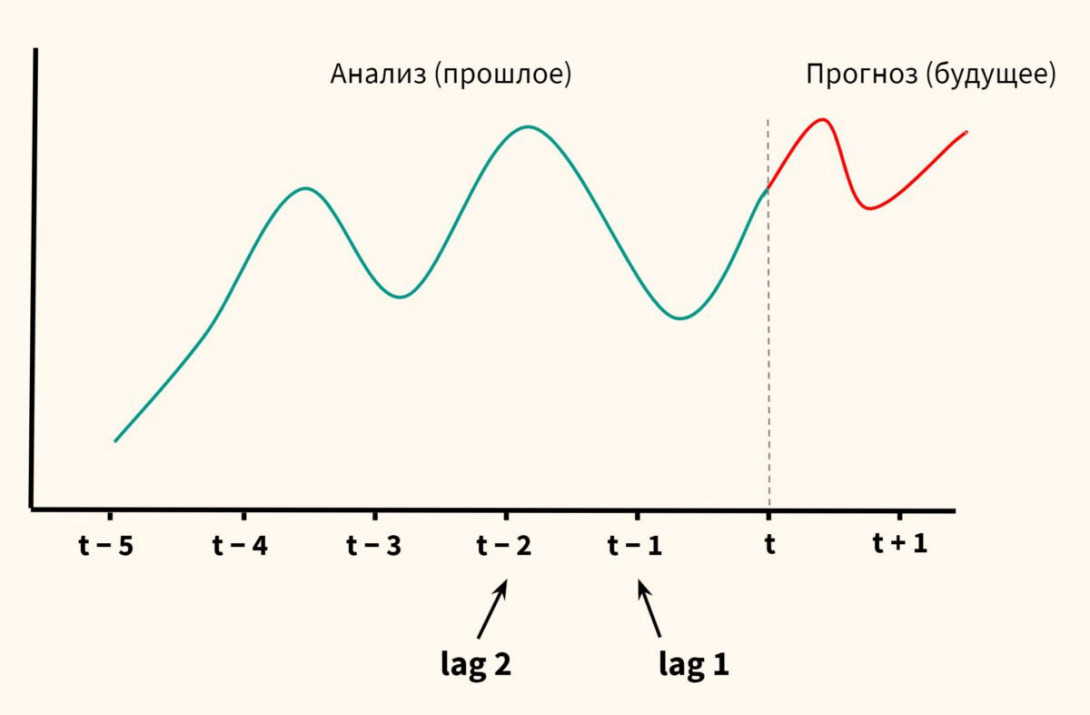


Рис. 2: временной ряд и лаг

Тик - шаг по оси времени $\Delta t = t_i - t_{i+1} = const\ \forall i$

Тренд – долгосрочное изменение уровня временного ряда

1. Нисходящий(медвежий) – каждый локальный максимум цены ниже предыдущего, как и локальный минимум
2. восходящий (бычий) – каждый локальный минимум цены выше предыдущего, как и локальный максимум
3. боковой (флэт) – локальные максимумы и минимумы примерно на одном значении

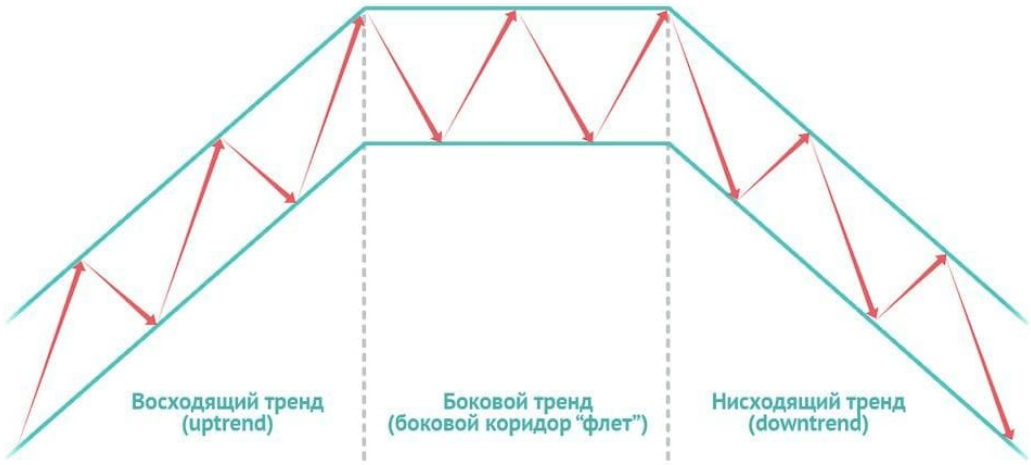


Рис. 3: виды тренда

Сезонность – циклические изменения уровня ряда с постоянным периодом

Шум - случайное изменение в ряду, не коррелирующее с другими данными

Финансовый временной ряд отличается от обычного временного ряда своей нестационарностью

2.2 ETS models

ETS (Error-Trend-Seasonality) модель представляет временной ряд как композицию тренда, сезонности и ошибки(шума). Существует аддитивная модель, в общем случае представляющая уровень как сумму компонент:

$$y(t) = Error + Seasonality + Trend$$

Ее стоит использовать, когда величина изменяется линейно (например, у авиакомпании +1000 пассажиров в год), а мультипликативную

$$y(t) = Error * Seasonality * Trend$$

когда величина изменяется нелинейно (например, количество пассажиров увеличивается каждый год в 2 раза). На рисунке 4 изображено разложение временного ряда на соответствующие компоненты:

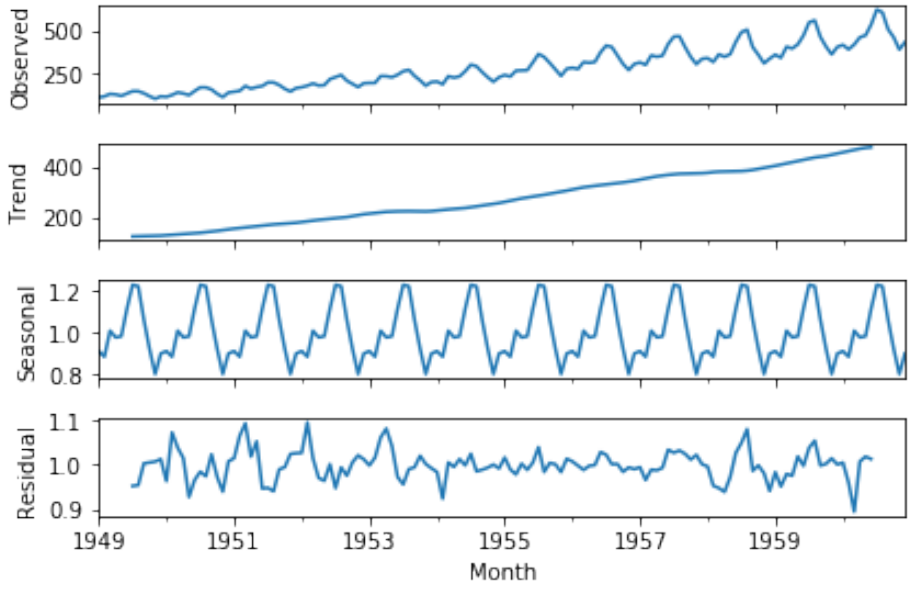


Рис. 4: декомпозиция временного ряда (мультипликативная модель)

2.3 Методы прогнозирования временных рядов

2.3.1 AR(p)

Авторегрессионные модели (AR-модели) используют прошлые значения ряда для прогнозирования его будущих значений. Эта модель предполагает, что текущее значение ряда зависит от его предыдущих значений. Наиболее популярной AR-моделью является модель первого порядка (AR(1)), которая предполагает, что текущее значение ряда зависит только от его предыдущего значения. [23] Аналогично, AR(p):

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \epsilon_t$$

где ϵ_t - белый шум, а α_i - коэффициент [25]

2.3.2 MA(q)

Следующий метод моделирования временных рядов - скользящее среднее (MA-модели). Эта модель использует прошлые значения ошибок - разницу между фактическими значениями ряда и его прогнозируемыми значениями, для прогнозирования будущих значений. Модель q-ого порядка (MA(q)) предполагает, что текущая ошибка зависит только от ее q предыдущих значений.[23]

$$y_t = \epsilon_t + \sum_{i=1}^{q} \beta_i \epsilon_{t-i}$$

2.3.3 ARMA(p,q)

ARMA - это модель, комбинирующая AR- и MA-модели:

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \beta_i \epsilon_{t-i}$$

2.3.4 ARIMA(p,d,q)

ARIMA позволяет моделировать данные, не являющиеся стационарными, как это требует для AR- и MA-модели. ARIMA включает три параметра: параметр авторегрессии (p), параметр скользящего среднего (q) и параметр интегрирования (d).

Можем привести ряд к стационарному виду с помощью Δ^d— оператора разности временного ряда порядка d (производим последовательное взятие d раз разностей первого порядка — сначала от временного ряда, затем от полученных разностей первого порядка, затем от второго порядка и т. д.)

$$\Delta^d y_t = c + \sum_{i=1}^{p} \alpha_i \Delta^d y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \beta_i \epsilon_{t-i}$$

2.3.5 SARIMA

SARIMA - это модель, комбинирующая ARIMA с сезонностью. SARIMA используется для моделирования сезонного поведения временных рядов. SARIMA - это расширение ARIMA с тремя дополнительными параметрами, определяющими сезонность: период сезонности (P), параметр авторегрессии со сезонностью (S) и параметр скользящего среднего со сезонностью (Q)

3 LSTM

Исследования в области прогнозирования временных рядов ведутся в течение многих лет, но с открытием применения глубокого обучения для решения данной задачи интерес к теме невероятно возрос. За последние десятилетия вышло огромное количество работ о прогнозировании финансовых временных рядов с использованием глубокого обучения (Deep Learning (DL)), рекуррентных нейронных сетей (Recurrent Neural Networks (RNN)), в частности, LSTM (Long Short-Term Memory) нейронных сетей с долгой краткосрочной памятью.

Минус RNN в невозможности смотреть далеко в прошлое, поэтому, работая в 1991 году над решением проблемы затухающего(исчезающего) градиента (the vanishing gradient problem) уже в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером была представлена новая архитектура рекуррентной нейронной сети - LSTM - которая смогла эффективно решать следующие задачи[2]:

1. Распознавание долгосрочных закономерностей в зашумленных входных последовательностях
2. Определение в зашумленных входных потоках порядка событий, находящихся во времени далеко друг от друга.
3. Извлечение информации, передаваемой расстоянием между событиями
4. Точная генерация периодических событий, закономерностей.
5. Надежное и длительное хранение действительных чисел

LSTM является самой цитируемой нейронной сетью 20 века [1], а современные алгоритмы LSTM разрабатываются и по сей день и используются для решения широкого спектра задач[2]: распознавание речи, машинный перевод, распознавание видео, распознавание рукописного ввода, прогнозирование временных рядов. LSTM-сети используются в робототехнике, видеоиграх, чат-ботах, в сфере здравоохранения и тд.

3.1 Архитектура

Описание принципа работы LSTM взято из [4]

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети:

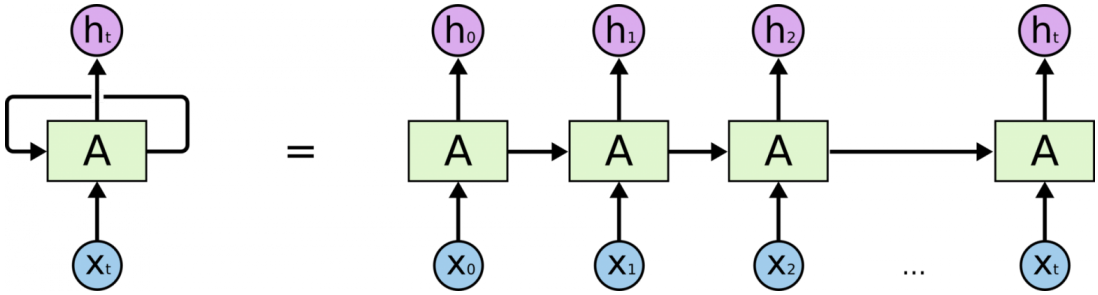


Рис. 5: общий вид рекуррентной нейронной сети (RNN)

Структура LSTM также напоминает цепочку, но модули выглядят иначе. Расмотрим один из них:

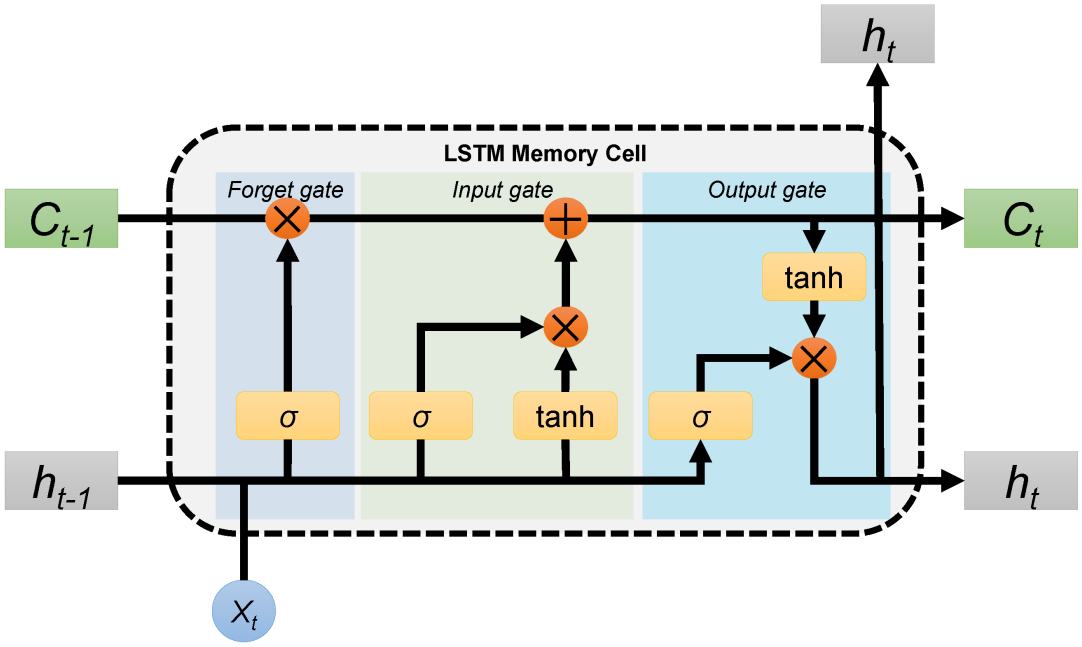


Рис. 6: модуль LSTM

На схеме выше каждая линия переносит целый вектор от выхода одного узла ко входу другого.Сливающиеся линии означают объединение, а разветвляющиеся стрелки говорят о том, что данные копируются и копии уходят в разные компоненты сети. Красными кружочками обозначены поточечные операции:

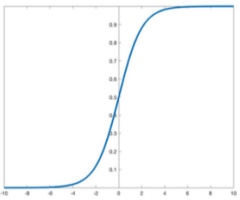
$+$ - сложение векторов,

$\times$ - произведение Адамара (поточечное умножение)

Желтые прямоугольники – это обученные слои нейронной сети.

Для более строгого описания схемы напомним вид функций активации:

Sigmoid $S(x) = \dfrac{1}{1+e^{-x}}$



Tanh $\tanh x = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

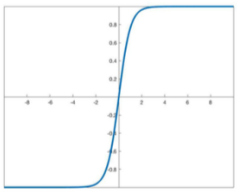


Рис. 7: функции активации

Ключевой компонент LSTM – это состояние ячейки (cell state) – горизонтальная линия, проходящая по верхней части схемы. LSTM может удалять информацию из состояния ячейки; этот процесс регулируется структурами, называемыми фильтрами (gates). Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения. Сигмоидальный слой возвращает числа от нуля до единицы, которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети. Ноль в данном случае означает “не пропускать ничего”, единица – “пропустить все”. В LSTM три таких фильтра, позволяющих защищать и контролировать состояние ячейки.

3.1.1 слой фильтра забывания

Первый шаг – определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый “слоем фильтра забывания” (forget gate layer). Он смотрит на h_{t-1} и x_t и возвращает число от 0 до 1 для каждого числа из состояния ячейки C_{t-1}. 1 означает “полностью сохранить”, а 0 – “полностью выбросить”.

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

где W, b - матрица и вектор коэффициентов

3.1.2 слой входного фильтра

Следующий шаг – решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов $\tilde{C}_t$, которые можно добавить в состояние ячейки.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{3}$$

3.1.3 обновляем состояние ячейки

Настало время заменить старое состояние ячейки C_{t-1} на новое состояние C_t. Что нам нужно делать — мы уже решили на предыдущих шагах, остается только выполнить это.

Мы умножаем старое состояние на f_t, забывая то, что мы решили забыть. Затем прибавляем $i_t * \tilde{C}_t$. Это новые значения-кандидаты, умноженные на t – на сколько мы хотим обновить каждое из значений состояния.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

3.1.4 слой выходного фильтра

Наконец, нужно решить, какую информацию мы хотим получать на выходе. Выходные данные будут основаны на нашем состоянии ячейки, к ним будут применены некоторые фильтры. Сначала мы применяем сигмоидальный слой, который решает, какую информацию из состояния ячейки мы будем выводить. Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

3.1.5 Общая формула

Итого, простейший LSTM-модуль можно представить в виде системы уравнений (1-6)

$$\begin{cases} f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \\ i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \\ C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \\ h_t = o_t * tanh(C_t) \end{cases} \tag{7}$$

4 Модель

4.1 Функция потерь

Функция потерь помогает оптимизировать параметры нейронных сетей. Наша цель - минимизировать потери для нейронной сети путем оптимизации ее параметров (весов). Потери рассчитываются с использованием функции потерь путем сопоставления целевого (фактического) значения и прогнозируемого значения нейронной сетью. Затем мы используем метод градиентного спуска для обновления весов нейронной сети таким образом, чтобы потери были сведены к минимуму. Так мы обучаем нейронную сеть.[6]

Нейронные сети пытаются минимизировать потери[10]

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(X_i, y_i; \theta) \to \min_{\theta}$$

где θ - параметры(веса) нейронной сети

$l(X_i, y_i; \theta)$ - функция, которая показывает как хорошо нейросеть с параметрами θ предсказывает y_i для X_i

N - число элементов в датасете

Функция потерь является мерой расхождения между истинным значением оцениваемого параметра (y) и оценкой параметра$(\hat{y})$.

Основной результат, который даёт функция потерь — оценка того, насколько хорошо классификатор работает на обучающей выборке. Но мы помним, что главная цель машинного обучения — заставить алгоритм правильно работать на тестовой выборке. Поэтому хорошо справляющийся с обучающими данных метод может совершенно не работать на новых объектах. Это явление называется «переобучение».[5]

В обучении с учителем (supervised learning) есть 2 основных типа функций потерь, которые соответствуют 2 основным типам нейронных сетей: регрессионные и классификационные функции потерь[7]

1. Регрессионная функция потерь: Mean Squared Error, Mean Absolute Error.
2. Класификационная функция потерь: Binary Cross-Entropy, Categorical Cross-Entropy

4.1.1 Кросс-энтропия

Кросс-энтропия (или логарифмическая функция потерь – log loss): Кросс-энтропия измеряет расхождение между двумя вероятностными распределениями. Если кросс-энтропия велика, это означает, что разница между двумя распределениями велика, а если кросс-энтропия мала, то распределения похожи друг на друга.[8]

Кросс-энтропия определяется как:

$$L(P, Q) = -\sum_{x} P(x) \log(Q(x))$$

где P – распределение истинных ответов, а Q – распределение вероятностей прогнозов модели.

Давайте упростим это для нашей модели:

1. N – количество наблюдений
2. K – количество возможных меток класса (в нашем случае - разновидности тренда: K = 2 или 3)
3. I – двоичный индикатор (0 или 1) того, является ли метка класса C правильной классификацией для наблюдения y
4. p – прогнозируемая вероятность модели

4.1.2 Бинарная классификация

В случае бинарной классификации (K=2), формула бинарной кросс-энтропии(Binary Cross-Entropy или BCE) имеет вид:

$$BCE = -(I \log(p) + (1 - I) \log(1 - p))$$

При двоичной классификации каждая предсказанная вероятность сравнивается с фактическим значением класса (0 или 1), и вычисляется оценка, которая штрафует вероятность на основе расстояния от ожидаемого значения.[8]

Если мы используем BCE, нужен один выходной узел, чтобы классифицировать данные на два класса. Выходное значение должно быть передано через сигмоидальную функцию активации(28) и диапазон выхода $[0, 1]$.[5]

4.1.3 Мульти-классовая классификация

В случае мульти-классовой классификации (K>2) мы берем сумму значений логарифмических функций потерь для каждого прогноза наблюдаемых классов. Получаем формулу категоральной кросс-энтропии (Categorical Cross-Entropy или CCE)[8]

$$CCE = -\sum_{c=1}^{K}(I_{y,c}\log(p_{y,c}))$$

Если мы используем CCE, должно быть столько же выходных узлов, сколько классов. И окончательный выходной слой должен быть пропущен через SoftMax(27) функцию активации, чтобы каждый узел выводил значение вероятности в диапазоне $[0, 1]$.[5]

4.2 Алгоритм оптимизации

Мы говорили, что функция потерь показывает, насколько хорошо или плохо выбранные веса справляются с поставленной задачей. В двумерном пространстве значения функции можно представить в виде цветного «ландшафта», где наиболее холодный (синий) — наименьшая величина потерь, к которой мы стремимся; а тёплый регион (красный) — наибольшая. На рисунке 8 черная линия - траектория движения алгортима оптимизации, которая стремится к мимимуму функций потерь.
приведем ниже самые популярные алгоритмы оптимизации.

4.2.1 Stochastic Gradient Descent (SGD)

SGD или Стохастический градиентный спуск обновляет параметры модели (θ) в отрицательном направлении градиента g взяв подмножество или мини-пакет (mini-batch) данных размером (m):

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{8}$$

$$\theta = \theta - (\epsilon_k \times g) \tag{9}$$

Нейронная сеть представлена $f(X_i;\theta)$, градиент функции потерь L вычисляется относительно параметров модели θ

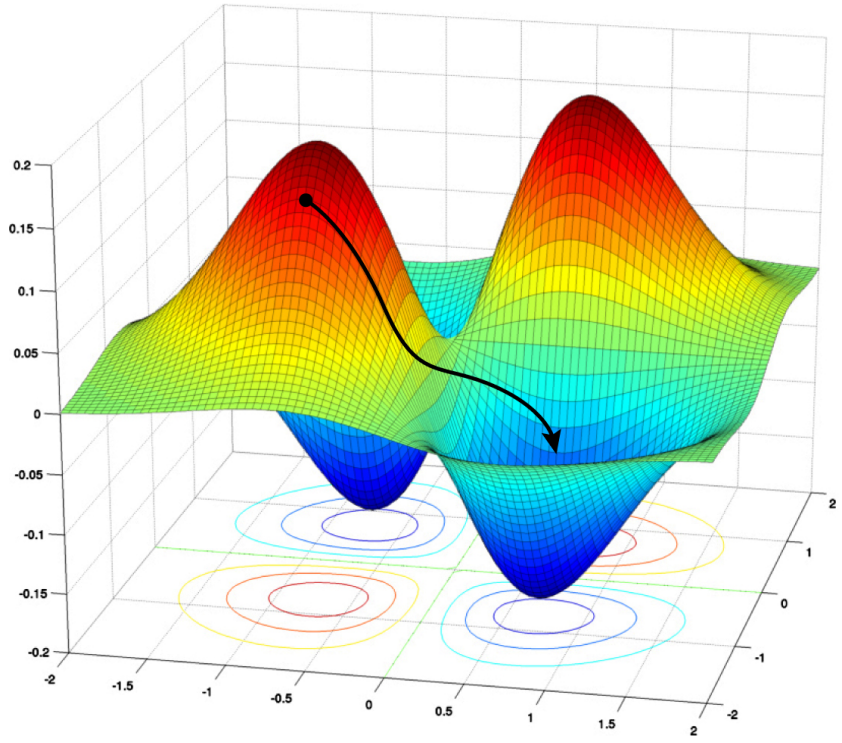


Рис. 8: Loss landscape или Ландшафт функции потерь для 2х весов

Скорость обучения (Learing rate) ϵ_k определяет размер шага, который алгоритм выполняет вдоль градиента (в отрицательном направлении в случае минимизации и в положительном направлении в случае максимизации). [11]

ϵ_k является функцией k-ой итерации алгоритма оптимизации и является чрезвычайно важным гиперпараметром. Слишком высокая скорость обучения (например, $> 0,1$) может привести к обновлениям параметров, которые пропускают оптимальное решение, а слишком низкая скорость обучения (например, $< 1e-5$) приведет к длительному времени обучения или же к сходимости в ближайшем локальном минимуме, который вряд ли окажется искомым ответом. [14]

Мы хотим, чтобы ϵ_k удовлетворяло условиям Роббинса-Монро (10). Первое равенство гарантирует, что алгоритм найдет оптимальное решение внезависимости от начальной точки, а второе неравенство регулирует колебания:

$$\sum_k \epsilon_k = \infty \qquad \sum_k \epsilon_k^2 < \infty \tag{10}$$

4.2.2 Momentum

Momentum(Импульс) - Импульс накапливает экспоненциально затухающую скользящую среднюю прошлых градиентов и продолжает двигаться в их направлении. Идея методов с накоплением импульса проста: «Если мы некоторое время движемся в определённом направлении, то, вероятно, нам следует туда двигаться некоторое время и в будущем»[12]

$$v = \alpha v - \epsilon \Delta_\theta (\frac{1}{m} \sum_i L(f(X_i; \theta), y_i)) \tag{11}$$

$$\theta = \theta + v \tag{12}$$

Широкоиспользуемые значения параметра α: 0.5 и 0.9

4.2.3 Nesterov's accelerated gradient method

$$v = \alpha v - \epsilon \Delta_\theta (\frac{1}{m} \sum_i L(f(X_i; \theta + (\alpha \times v), y_i)) \tag{13}$$

$$\theta = \theta + v \tag{14}$$

Разница между Нестеровым и стандартным импульсом заключается в том, когда оценивается градиент. С импульсом Нестерова градиент оценивается после применения текущей скорости, поэтому в импульс Нестерова добавляеся поправочный коэффициент к градиенту.

4.2.4 AdaGrad

$$g = \frac{1}{m} \Delta_\theta \sum_i L(f(X_i; \theta), y_i) \tag{15}$$

$$s = s + g^T g \tag{16}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s + eps}}) \tag{17}$$

AdaGrad(adaptive gradient) решает проблему того, что скорость обучения не зависит от градиента. То есть мы можем застревать в каих-то местах, напрмер, когда

градиент почти не меняется (рисунок 9, левый график) или проскакивать слишком быстро, колеблясь, но не достигая оптимального решения, когда градиент большой (рисунок 9, справа).

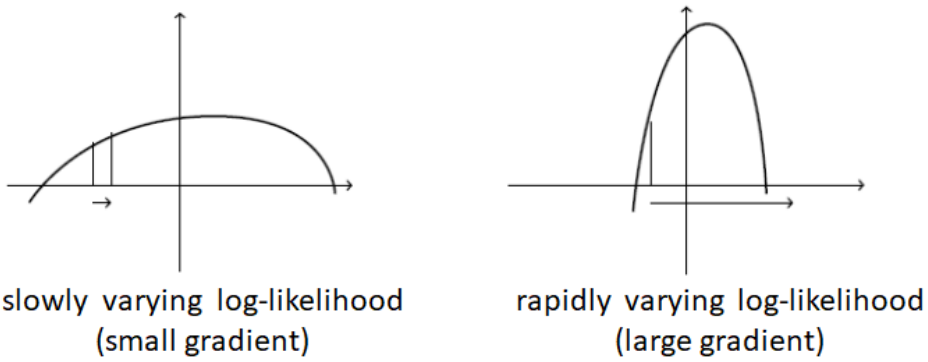


Рис. 9: примеры графиков с маленьким(слева) и большим(справа) градиентом

4.2.5 RMSProp

RMSProp(Root Mean Square Propagation или же Среднеквадратичное распространение корня) - модифицирует AdaGrad, изменяя накопление градиента на экспоненциально взвешенную скользящую среднюю, то есть отбрасывает историю из отдаленного прошлого. Из-за этого RMSProp является эффективным алгоритмом оптимизации и часто используется в глубоком обучении.

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{18}$$

$$s = decay\_rate \times s + (1 - decay\_rate)g^T g \tag{19}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s + eps}}) \tag{20}$$

4.2.6 Adam

Adam(adaptive moments) - один из самых популярных оптимизаторов благодаря своей скорости сходимости. Является ваиантом комбинации RMSProp (4.2.5) и Momentum (4.2.2). Итерации выглядят как RMSProp, за исключением того, что вместо необработанного стохастического градиента используется плавная версия гради-

ента, также полное обновление Адама включает механизм коррекции смещения.

$$g = \frac{1}{m}\Delta_\theta \sum_i L(f(X_i;\theta), y_i) \tag{21}$$

$$m = \beta_1 m + (1-\beta_1)g \tag{22}$$

$$s = \beta_2 s + (1-\beta_2)g^T g \tag{23}$$

$$\theta = \theta - (\epsilon_k \times \frac{g}{\sqrt{s+eps}}) \tag{24}$$

Рекомендованные значения параметров [13]

$$\beta_1 = 0.9 \qquad \beta_2 = 0.999 \qquad eps = 1e-8 \tag{25}$$

Вывод

Во многих случаях градиентный спуск может привести к оверфиттингу (переобучению), упасть в слишком глубокий минимум функции ошибки, который плохо будет обобщаться на новые данные. Собственно, алгоритмам градиентного спуска неоткуда знать о том, насколько хорошо обобщается то, что они делают, они просто оптимизируют ошибку на тренировочном множестве. Поэтому мы сами должны следить за качеством обобщения, которое в простейшем случае соответствует ошибке на валидационном множестве.

Когда данные достаточно разрежены, то однозначно стоит сделать выбор в сторону адаптивных моментов. Так как Adam является улучшенной версией AdaGrad, RMSProp, то в силу скорости сходимости стоит выбрать его. Но существуют задачи, когда Adam не справляется, тогда можно обратиться и к другим алгоритмам. [14]

Мой выбор пал на Adam и SGD - наиболее известные и используемые методы оптимизации.

4.3 Метрика оценки качества

Метрика оценки качества - третий важный элемент в построении нейрорсети. С помощью нее можем оценить на тестовом множестве качество предсказаний обученной нейросети, отследить переобучение. Метрики, как и функции потерь, для задач классификации и регрессии различаются. Рассмотрим метрики оценки качества для задач классификации.

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — confusion matrix (матрица ошибок). Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Таблица 1: матрица ошибок(confusion matrix) для двух классов

Поясним обозначения в таблице.

True Positive (TP):

1. Прогнозируемое значение равно фактическому значению ($\hat{y} = y$).
2. Фактическое значение было положительным ($y = 1$), и модель предсказала положительное значение ($\hat{y} = 1$).

True Negative (TN):

1. Прогнозируемое значение равно фактическому значению ($\hat{y} = y$).
2. Фактическое значение было отрицательным ($y = 0$), и модель предсказала отрицательное значение ($\hat{y} = 0$).

False Negative (FN) - ошибка 1-ого рода:

1. Прогнозируемое значение было предсказано неверно ($\hat{y} \neq y$).
2. Фактическое значение было положительным ($y = 1$), но модель предсказала отрицательное значение ($\hat{y} = 0$).

False Positive (FP) - ошибка 2-ого рода:

1. Прогнозируемое значение было предсказано неверно ($\hat{y} \neq y$).
2. Фактическое значение было отрицательным ($y = 0$), но модель предсказала положительное значение ($\hat{y} = 1$).

Случай нескольких классов отображен на рисунке 10.

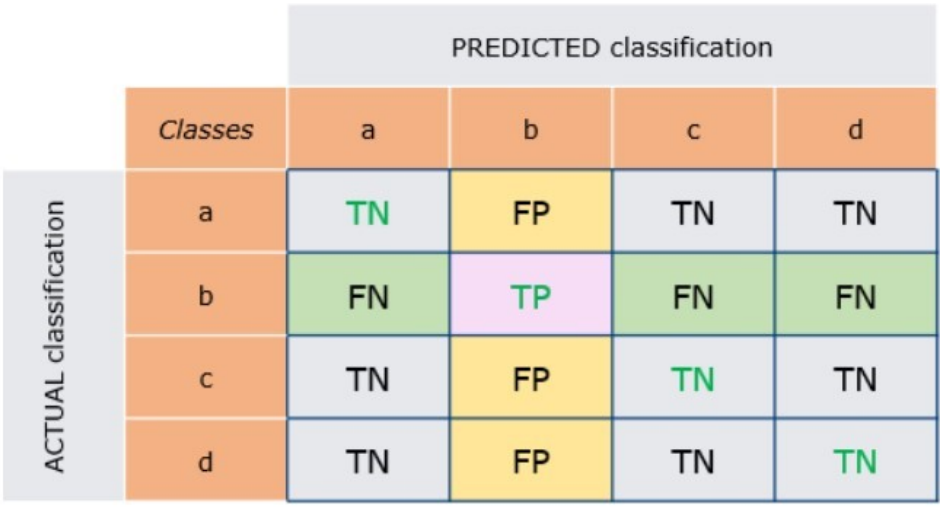


Рис. 10: confusion matrix для мультиклассовой классификации относительно класса b.

4.3.1 Accuracy

Accuracy (точность) - одна из самых простых метрик - доля правильных ответов алгоритма[16]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

В задачах с неравными классами accuracy бесполезна, так как не выполняет главную задачу метрики - адекватную оценку точности работы нейросети. Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно ($TN = 90, FP = 10$), и 10 спам-писем, 5 из которых классификатор также определил верно ($TP = 5, FN = 5$). Тогда accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4\%$$

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9\%$$

Для случая мультиклассовой классификации рассматривается accuracy относительно 1 из классов. [15]

4.3.2 Precision

Precision (точность) можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными. То есть мы можем оценить возможность алгоритма отличать данный класс от других.

$$Precision = \frac{TP}{TP + FP}$$

Именно введение precision не позволяет нам записывать все объекты в один класс, как в случае с accuracy, так как в мы получаем рост уровня False Positive.

4.3.3 Recall

Recall (полнота) показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Recall демонстрирует способность алгоритма обнаруживать данный класс.

$$Recall = \frac{TP}{TP + FN}$$

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.

4.3.4 F-measure

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае F_β) — среднее гармоническое precision и recall :

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

β в данном случае определяет вес точности в метрике. Часто используестя вариант

$$\beta = 1 \qquad\qquad F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

В случае когда количество классов $K > 2$ есть 2 разновидности F-меры. Макро F-мера $\in [0; 1]$. В ней нет связи с размером классов, то есть большие и маленькие

равнозначны. Чем больше Macro F1 мера, тем лучше алгоритм предсказывает все классы.[15]

$$MacroF_1 = \frac{\sum_{k=1}^{K} F_{1k}}{K}$$
$$MicroF_1 = \frac{\sum_{k=1}^{K} TP_k}{N}$$

индекс k обозначает класс относительно которого считается метрика,
N - общее число наблюдений.

Заметим, что Micro F1 мера равна accuracy, то есть преимущества и недостатки у нее такие же.

4.4 Функции активации

Ниже приведены основные функции активации, которые упоминаются в этой работе:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} x_j} \in [0; 1] \tag{27}$$

$$sigmoid = \frac{1}{1 + e^{-x}} \in [0; 1] \tag{28}$$

$$ReLU = \max(0; x) \in [0; 1] \tag{29}$$

$$tanh = \max(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}) \in [-1; 1] \tag{30}$$

Вывод

В этой части приведен обзор на основные моменты, которые используются при создании нейросети: функции потерь, алгоритмы оптимизации, метрики оценки качества и функции активации. Выбор каждого из методов зависит от задачи, типа данных и отображается на правильности, скорости работы алгоритма.

В случае прогнозирования тренда финансовых временных рядов стоит выбать:

1. функция потерь: Categorical Cross-Entropy 4.1.3
2. алгоритм оптимизации: Adam 4.2.6
3. метрика: Precision и Recal 4.3.2

5 Вычислительные эксперименты

Для проведения вычислительных экспериментов воспользуемся языком Python [26] и Google Colaboratory [27] — это среда разработки на Python с использованием инструмента для разработки Jupyter Notebook.

Возьмем датасет цен акций Apple из библиотеки yfinance [28].

Работу с данными проведем с помощью библиотек Numpy [29] и Pandas [30]. Используем MinMaxScaler из библиотеки scikit-learn для нормализации fechures [31].

Для построения модели воспользуемся TensorFlow (Keras)[32].

Все графики сделаны с помощью библиотеки Matplotlib [33].

5.1 Параметры LSTM в TensorFlow

Что обозначают входные параметры LSTM в TensorFlow? Этот вопрос важен, так как от подготовки данных и определения парамеров зависит точность и скорость работы модели.

Для начала приведем наш датасет к размерности
$[batch, timesteps, feature]$,

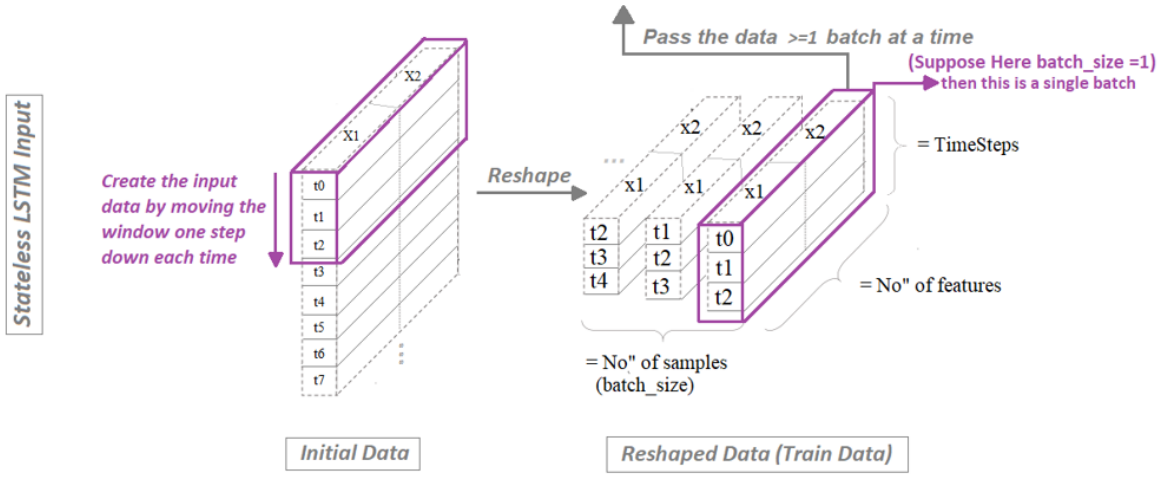


Рис. 11: Представлене датасета в размерности вида [batch, timesteps, feature]

где $feachure$ - количество столбцов-признаков, на основе которых строим прогноз
$timesteps$ - количество предыдущих значений в последовательности (длина окна) (см. рис. 12)

batch - количество получившихся последовательностей (samples) размера $[timesteps, feature]$

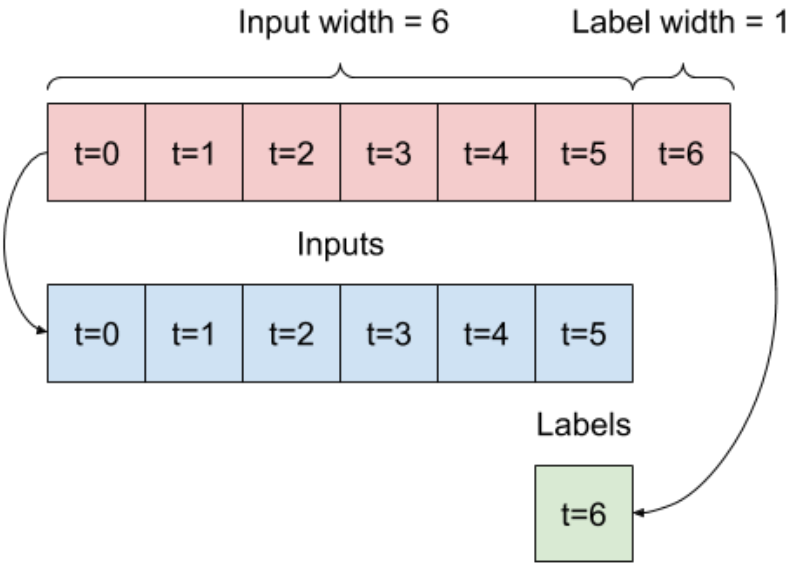


Рис. 12: Пример последовательности, которая делает предсказание на 1 час вперед, основываясь на 6-часовой истории

Важным аргументом конструктора для всех слоев Keras RNN, таких как tf.keras.layers.LSTM , является аргумент $return\_sequences$. Этот параметр может настроить слой одним из двух способов:

Если $return\_sequences = False$ (по умолчанию), слой возвращает только выходные данные последнего временного шага, давая модели время, чтобы прогреть свое внутреннее состояние (warmup), прежде чем делать один прогноз (рис. 13)

Если $return\_sequences = True$, слой возвращает результат для каждого входа. Этот способ используется, когда в модели есть несколько RNN слоев, или же в обущении можели на нескольких временных шагах одновременно.[18]

Еще один немаловажный параметр - $units$. Это размерность выходного пространства. То есть наша модель берет на вход тензор $[batch, timesteps, feature]$ и на выход дает тензор $[batch, timesteps, units]$, если $return\_sequences = True$, и [batch, units], если $return\_sequences = False$, соответственно [19]. Так же $units$ отвечает за размер скрытого слоя LSTM, размера матрицы весов. То есть, чем больше $units$, тем точнее будет модель, но при этом скорость работы будет увеличиваться из-за количества вычислений.[17]

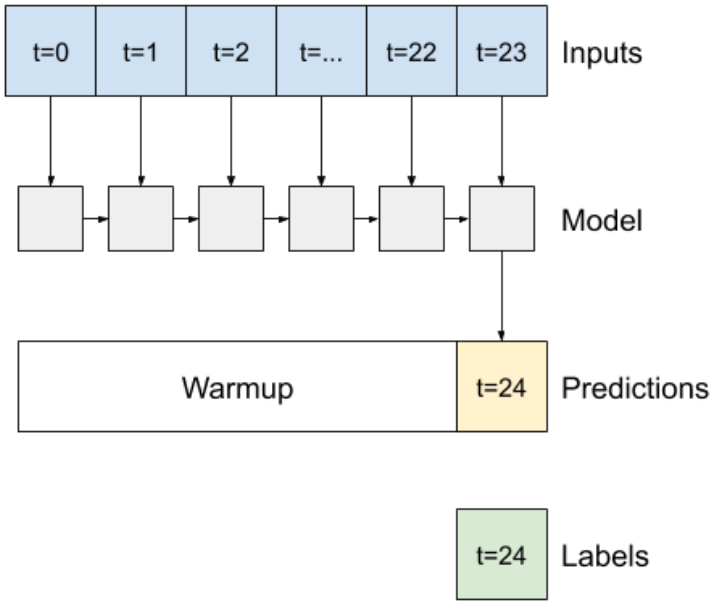


Рис. 13: $return\_sequences = False$. Возвращаются выходные данные только для последнего временного шага

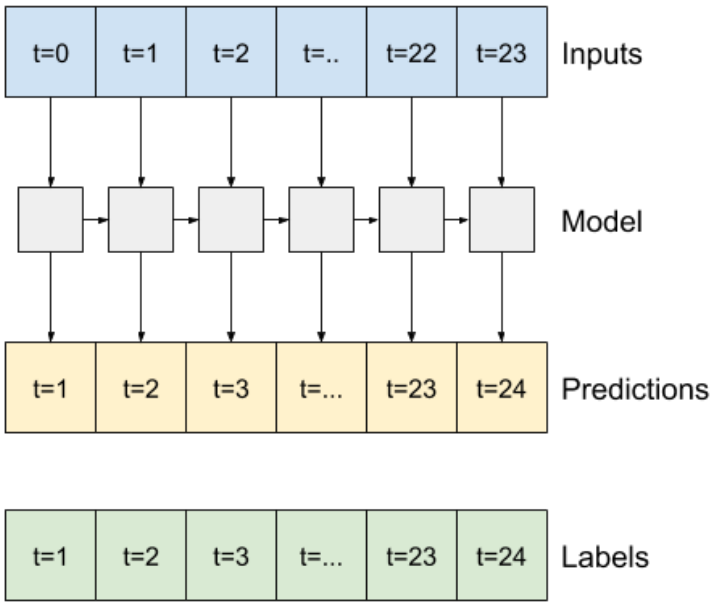


Рис. 14: $return\_sequences = True$. Возвращаются выходные данные для каждого временного шага

5.2 Как влияет units на прогноз?

Как упоминалось выше, *units* отвечает за размер матрицы весов, поэтому, чем больше *units*, тем лучше прогноз. Проверим это на следующей модели:

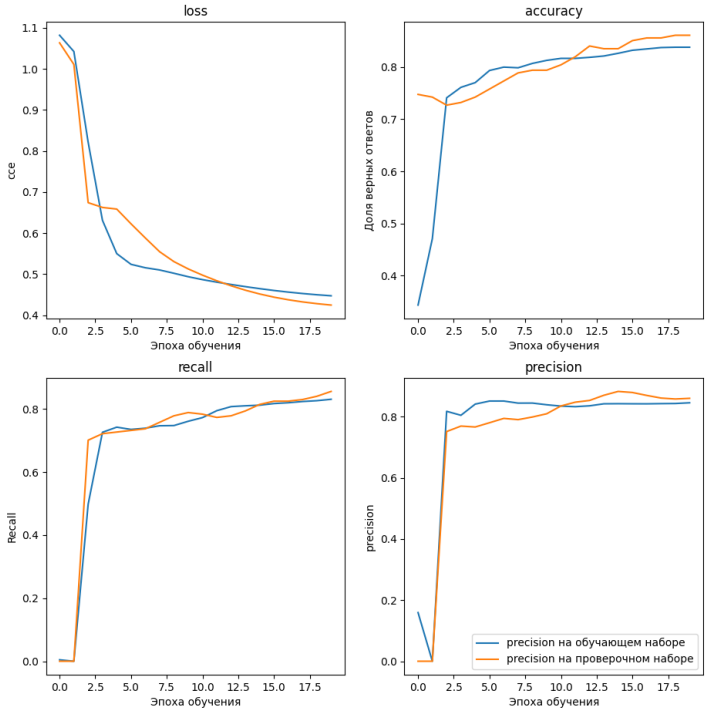
```
model1 = Sequential([layers.LSTM(units,input_shape =
                    (train_X.shape[1], train_X.shape[2])),
                     layers.Dense(3,  activation = 'softmax')])

model1.compile(loss='categorical_crossentropy',
               optimizer= Adam(learning_rate = 0.01),
               metrics=['accuracy',Recall(name='recall'),
               Precision(name='precision')])

m1 = model1.fit(train_X, train_y, validation_data=(val_X, val_y),
                 epochs= 20, shuffle = False)
```

Листинг 1: Модель1

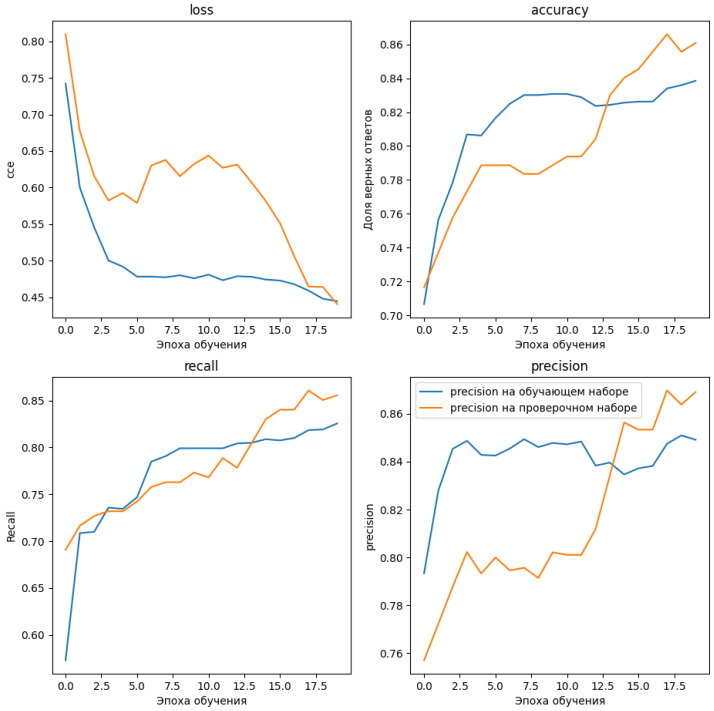
Будем менять параметр *units* при фикированных $window\_size = 20$, на поминутном датасете, используя для прогноза все доступные нам данные. То есть спрогозируем 30-минутный тренд для следующего момента времени, используя последние 20 наблюдений. На рисунках 5.2 - 5.2 отображены графики функции потерь (*loss*) и метрик *accuracy*, *recall* и *precision* относительно эпохи обучения.

Видим, что при больших *units* данная модель переобучается. Поэтому, оставим $units = 3$, так как 3 - размер наших выходных данных.

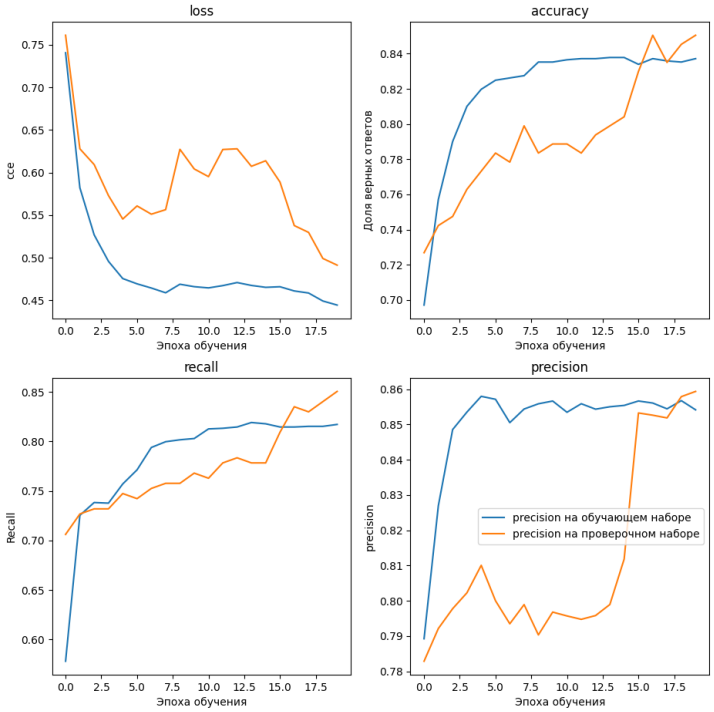
Модель1: window_size =20, units = 3,



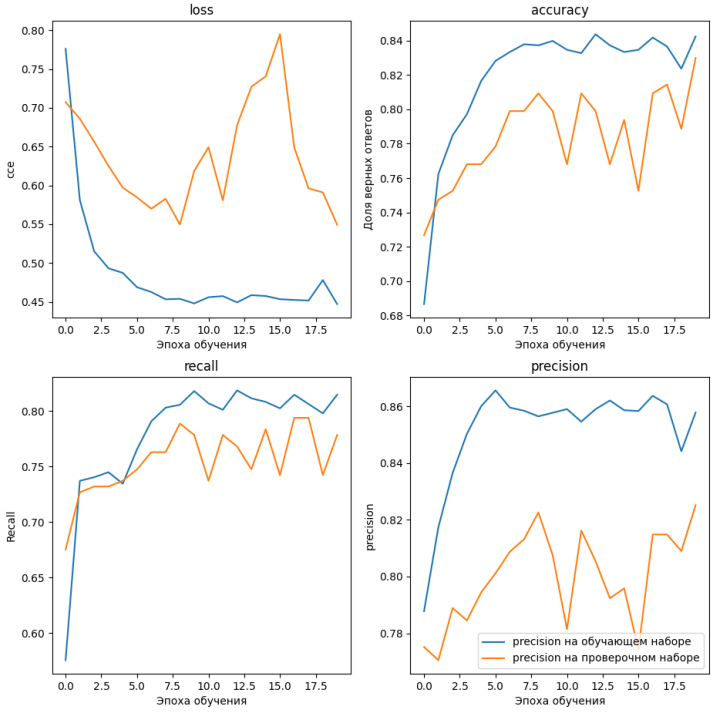
Модель1: window_size =20, units = 20,

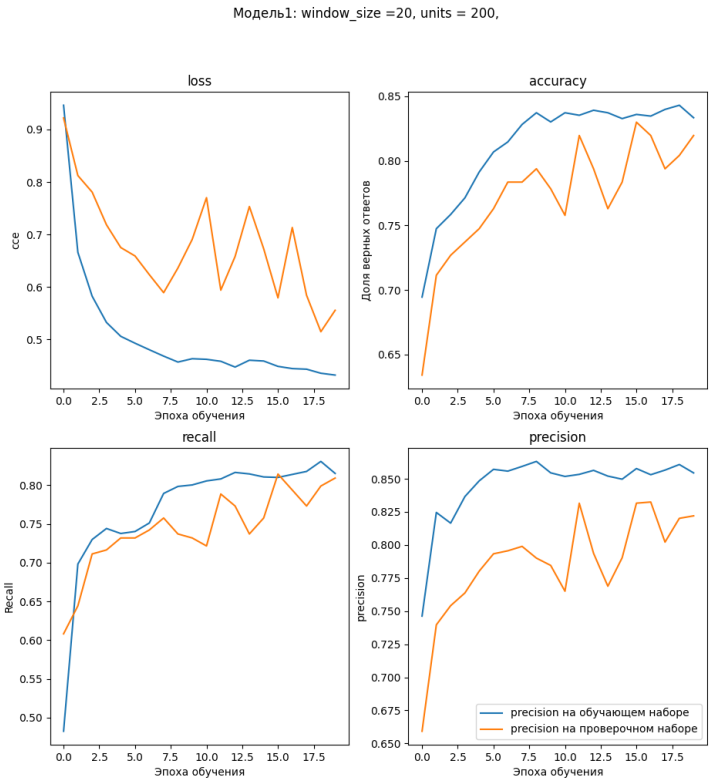


Модель1: window_size =20, units = 40,



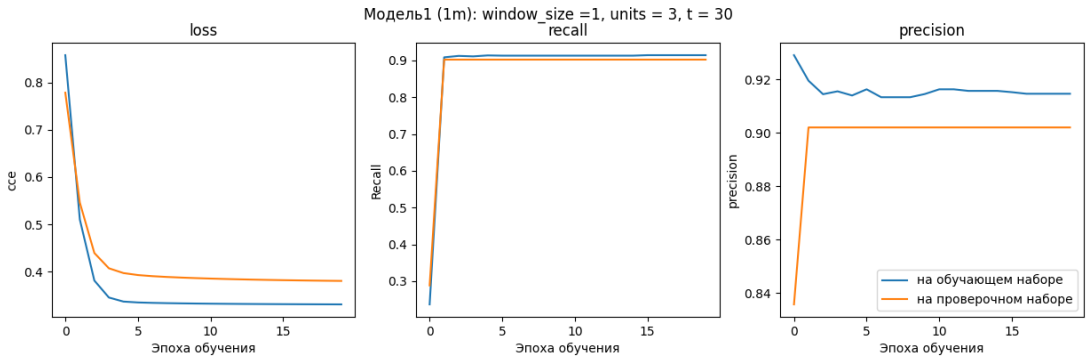
Модель1: window_size =20, units = 100,

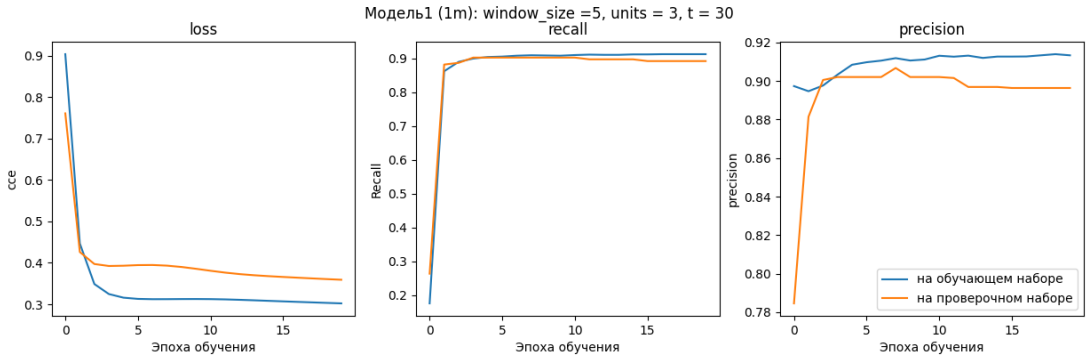


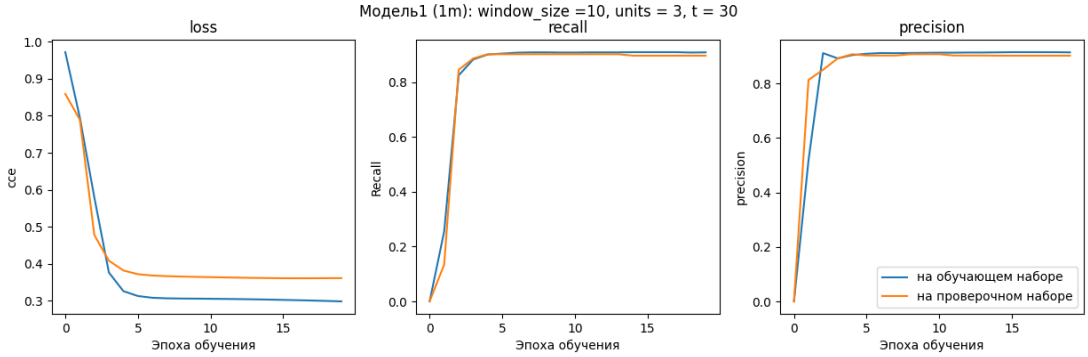


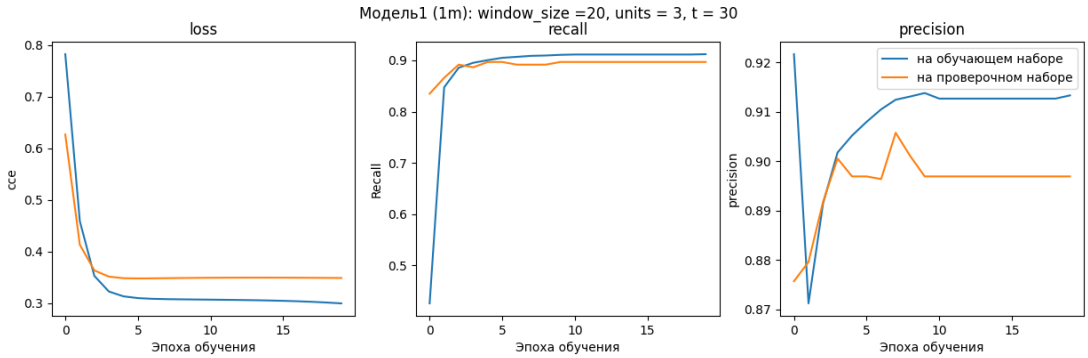
5.3 Как влияет timesteps на прогноз?

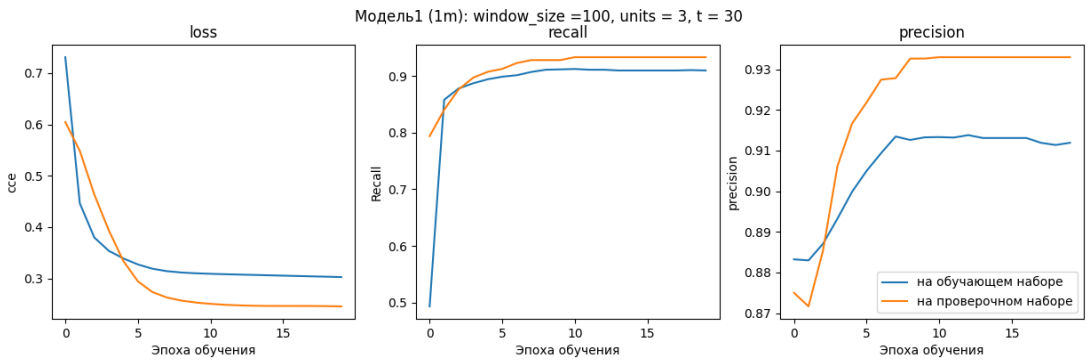
timesteps - размер скользящего окна (*window_size*). Как и в случае с *units*, большее окно дает большую точность, но и замедляет обучение из-за увеличения количества операций. Обучим модель 1 с разными *timesteps* (см. *window_size* = ...).




Модель1 (1m): window_size =5, units = 3, t = 30
loss
recall
precision
cce
Recall
precision
Эпоха обучения
на обучающем наборе
на проверочном наборе


Модель1 (1m): window_size =10, units = 3, t = 30
loss
recall
precision
cce
Recall
precision
Эпоха обучения
на обучающем наборе
на проверочном наборе


Модель1 (1m): window_size =20, units = 3, t = 30
loss
recall
precision
cce
Recall
precision
Эпоха обучения
на обучающем наборе
на проверочном наборе


Модель1 (1m): window_size =100, units = 3, t = 30
loss
recall
precision
cce
Recall
precision
Эпоха обучения
на обучающем наборе
на проверочном наборе

На практике видим, что с увеличением окна растет $precision(> 0,91)$, и заметно увеличивается время обучения: 1 минута в случае $window\_size = 100$ и 13 секунд при $window\_size = 5$. В последующих вычислениях остановимся на $window\_size = 10$.

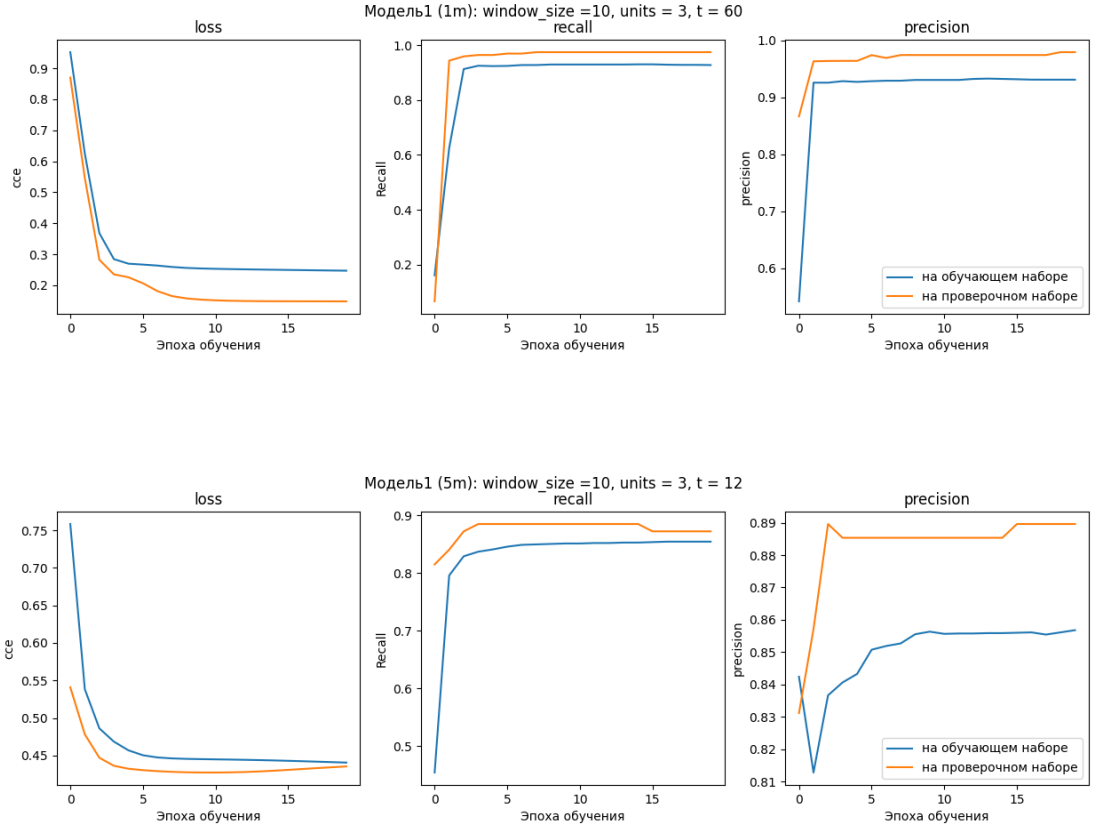
5.4 Как влияет тик на прогноз?

Кроме того мы можем выбирать тики - частоту обновления данных в датасете, а так же период времени t, который определяет период относительно которго находим тренд для i-ого наблюдения по формуле

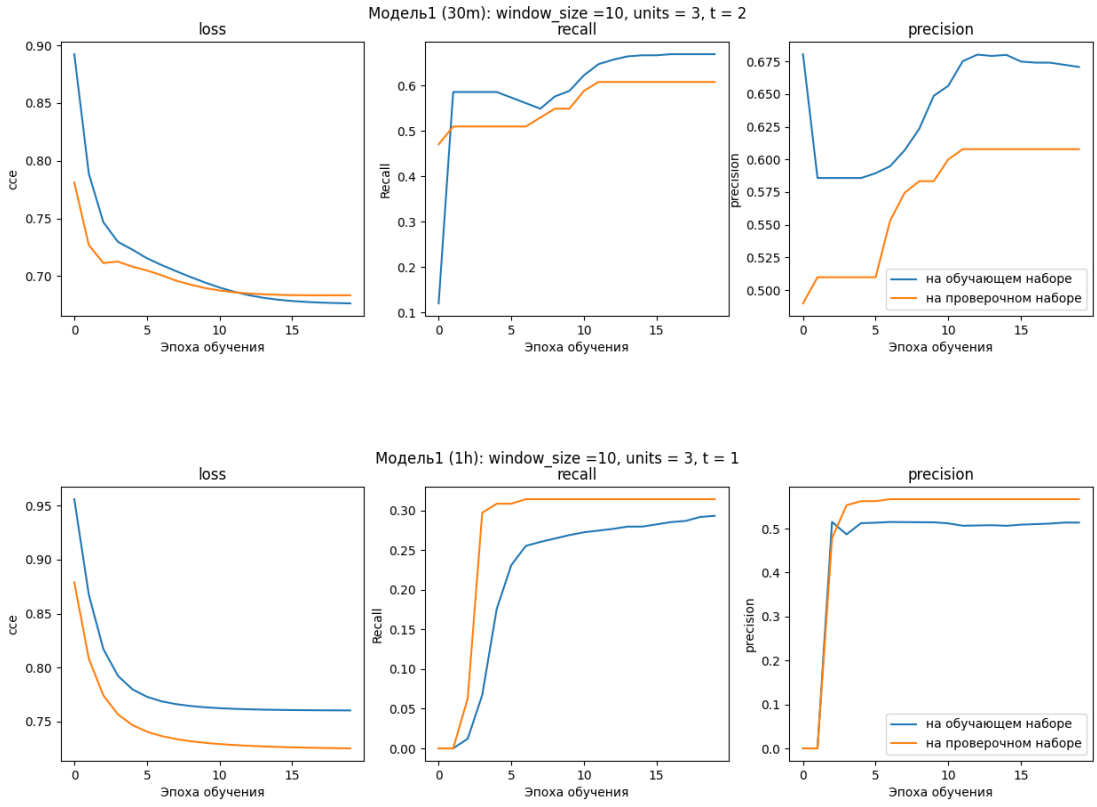
$$trend(i) = close(i + t) - open(i)$$

где *close* - цена акции в конце интересующего нас периода, а *open* - цена в начале.

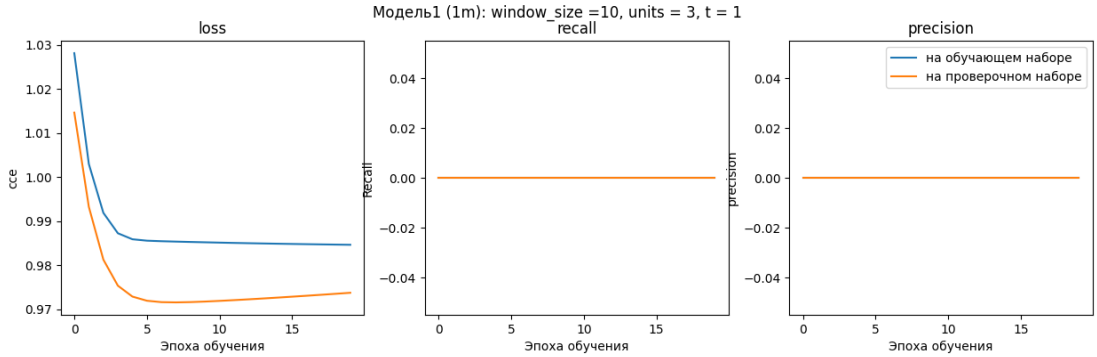
Предскажем тренд на час вперед, используя тики: 1 минута, 5 мнут, 30 минут, 1 час.



Видим, что прогноз на часовых тиках по предыдущим 10 часам - самый худший по всем показателям, а прогноз на минутных тиках по предыдущим 10 минутам -



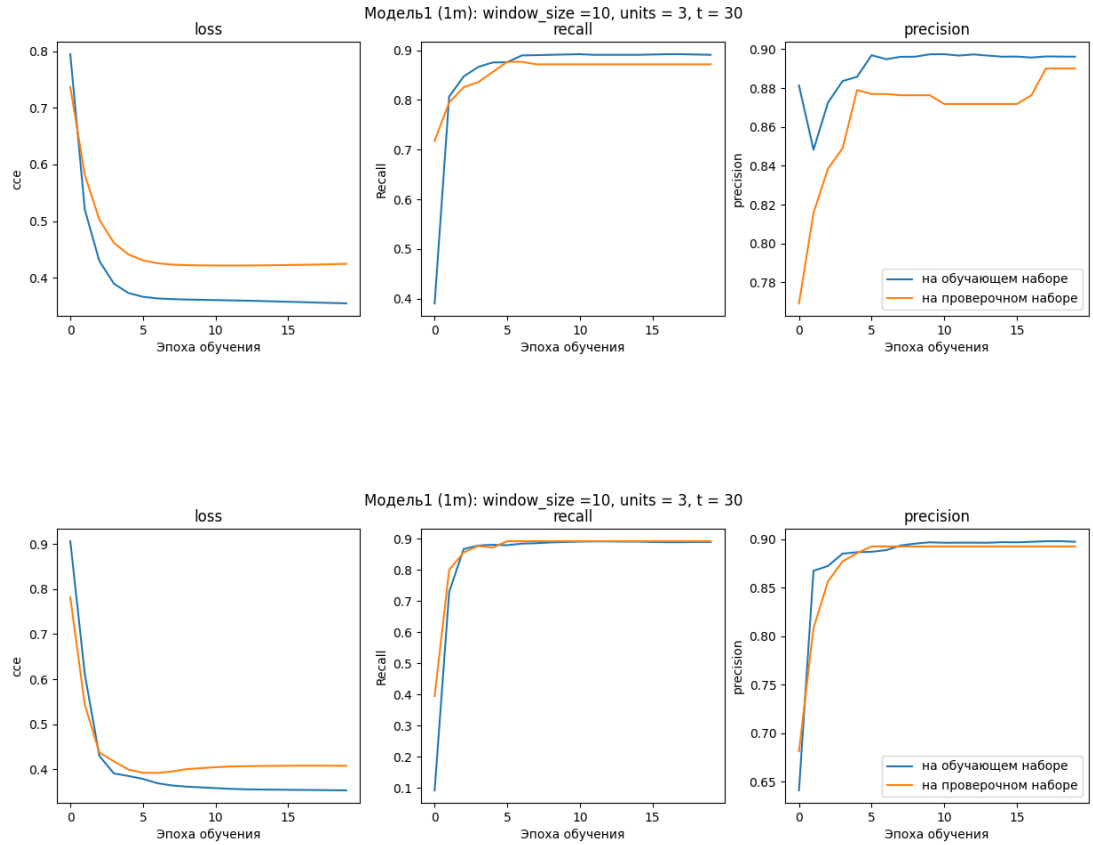
наилучший. Но у нас нет тиков меньше минуты, поэтому прогноз минутного тренда будет неточным без возможности его улучшения:



5.5 Как влияют features на прогноз?

Можем делать прогноз только на предыдущих значениях тренда или же добавить $features$. Сравним прогнозы основанные на тренде (рис. 5.5) и на тренде, цене

закрытия (Close), открытия (Open), максимальной (High) и минимальной (Low)(рис. 5.5) Видим, что дополнительные данные улучшают прогноз.



5.6 Как влияет вид модели на прогноз?

Выше была исследована модель 1 состоящяя из 1 LSTM слоя и 1 Dense слоя - простого полносвязного слоя, когда каждый нейрон предыдущего слоя связан с каждым нейроном данного - с функцией активации Softmax (27), которая требуется для правильной работы выбраной loss-функции: Categorical Cross-entropy (4.1.3).

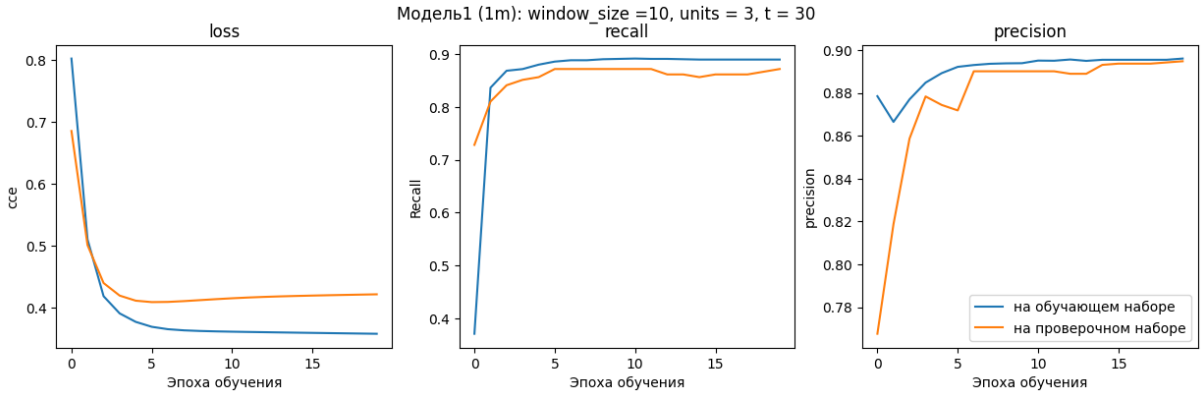
Построим модель без Dense слоя, вместо него добавив Activation слой с Softmax:

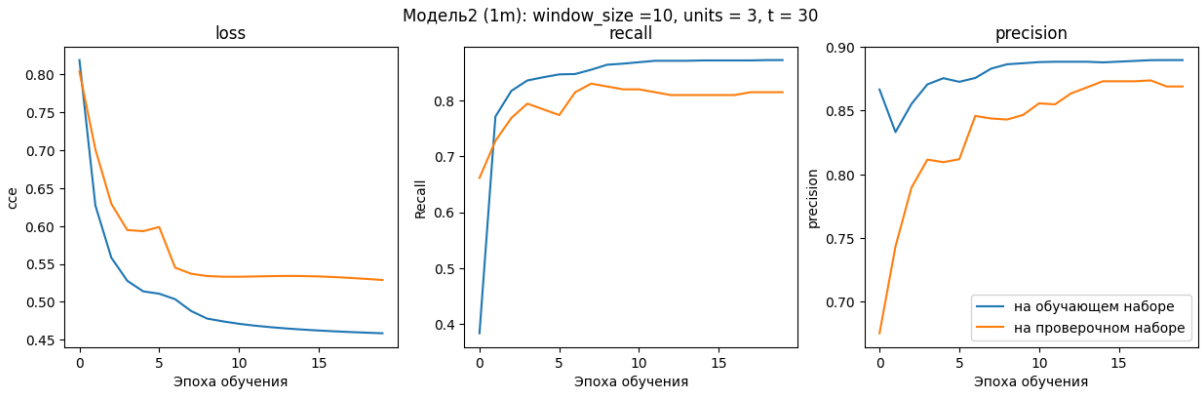
```
model2 = Sequential([layers.LSTM(3,input_shape =
                    (train_X.shape[1], train_X.shape[2])),
                    layers.Activation('softmax')])

model2.compile(loss='categorical_crossentropy',
               optimizer= Adam(learning_rate = 0.01),
```

```
                metrics = [Recall(name='recall'),
                Precision(name='precision')])

m2 = model2.fit(train_X, train_y, validation_data=(val_X, val_y),
                  epochs= 20, shuffle = False)
```

Листинг 2: Модель2

На практике видим, что модель 1 точнее модели 2





6 Сравнение с альтернативными методами

Нейросети стали использоваться относительно недавно, до их изобретения строились модели ARIMA и др. Остановимся на модели ARMA, так как полученный ряд стационарный (проверили с помощью обобщенного теста Дикки-Фуллера на наличие единичных корней [22]). Проверим точность предсказаний модели ARMA. Для простоты определим тренд как: 1 - восходящий и 0 - нисходящий. На минутных тиках прогноз минутного тренда не сделать - автокорреляция стремится к нулю, то есть значение каждого тика не зависит от предыдущих:

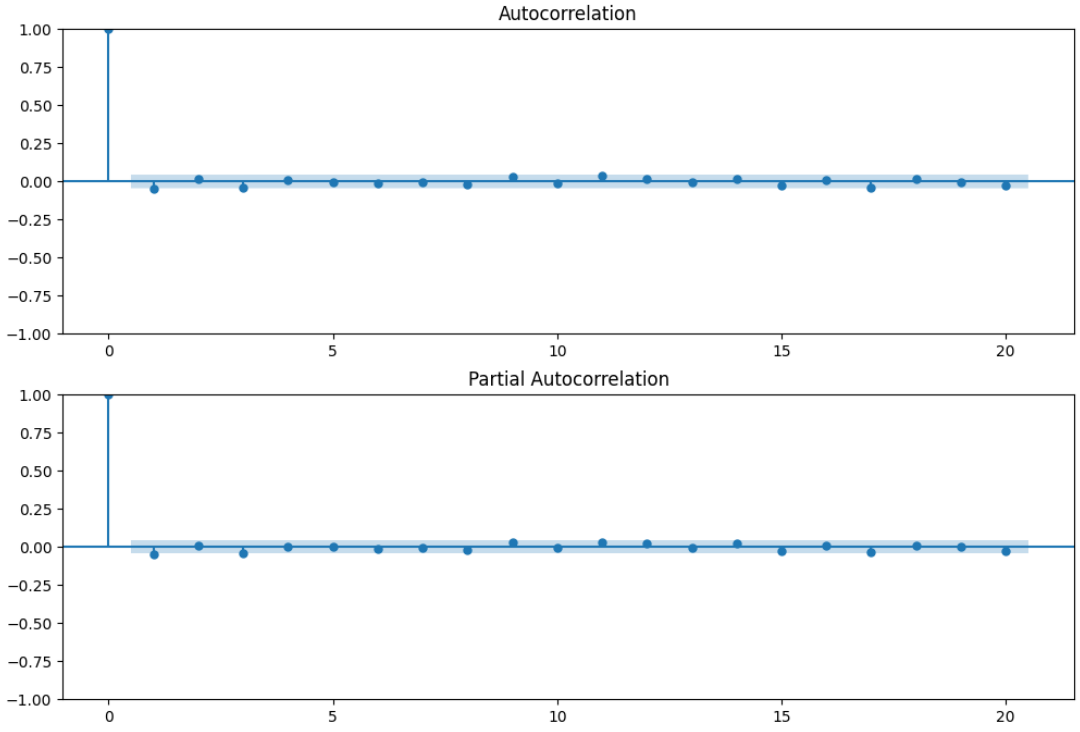


Рис. 15: На верхнем графике отображена автокорреляционная функция для минутного тренда

Для 30-минутного тренда на минутных тиках построим ARMA(1,17) - именно эти параметры дают наилучшие результаты (см.рис 16: по графику автокорреляции(ACF) определяется q, а по частичной автокорреляции(PACF) – p)

Получили следующие результаты:

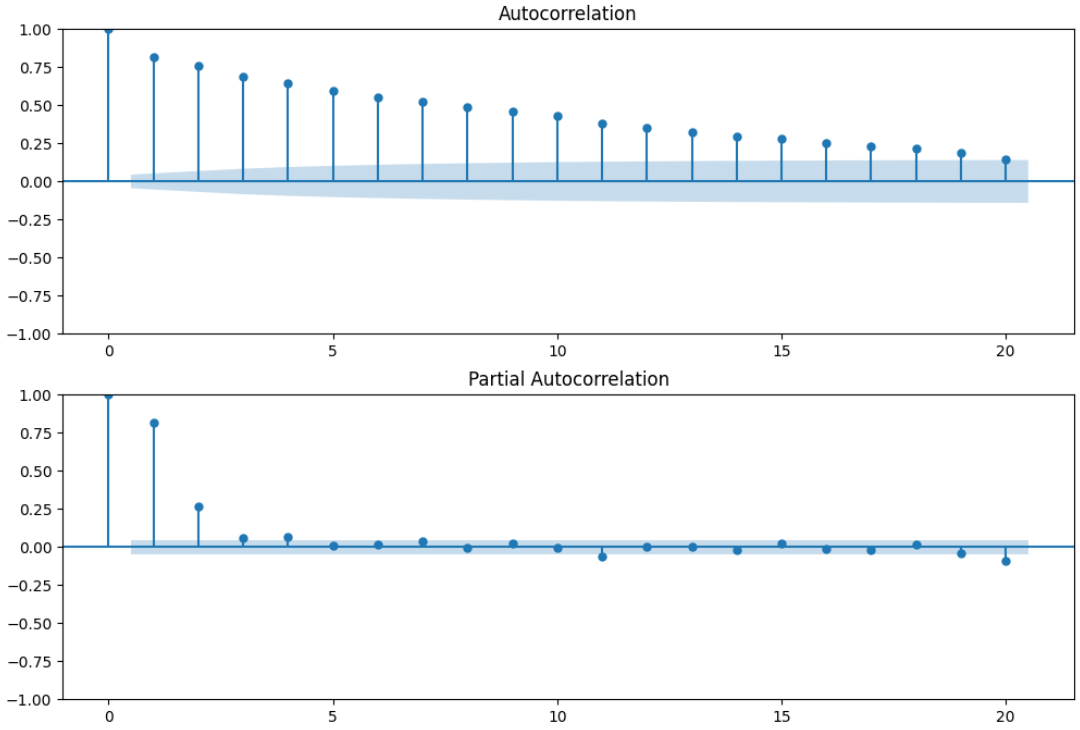


Рис. 16: На верхнем графике отображена автокорреляционная функция(ACF) для 30-минутного тренда, на нижнем - функция частичной автокорреляции (PACF). По ним находятся коэффициенты p и q

метрика	ARMA	LSTM
accuracy	0.4635416666666667	0.8967286944389343
precision	0.4635416666666667	0.9178082346916199
recall	1.0	0.9586129784584045

Таблица 2: значения метрик для ARMA и LSTM

По таблице 2 видно, что LSTM предсказывает тренд финансового временного ряда примерно в 2 раза лучше чем ARMA.

Заключение

В работе получен результат поставленной задачи - постоена модель нейронной сети с памятью, которая предсказывает тренд временного ряда, также были даны ответы на вопросы о влиянии параметров на точность прогноза, дано сравнение с классическим методом прогнозирования временных рядов.

За время работы над курсовой работой я научилась с помощью языка программирования Python и библиотек Pandas, TensorFlow работать с данными, создавать нейросети, оценивать качество их работы и усовершенствовать их для более точных результов. Кроме того, изучила и систематизировала базовые понятия функции потерь, алгоритма оптимизации и метрики качества, вдобавок к алгоритму работы нейронных сетей с памятью и нюансов их работы в TensorFlow.

Дальнейшие направления исследований:

1. В главе 1 упоминалось, что прогноз можно сторить на анализе текстовой информации, которая отображает эмоциональный аспект влияния на цены акций.

2. “Свечу” можно представить в виде набора символов и провести аналогию с прогнозированием строения белков, воспользовавшись известными для этого методами, применив их для прогнозирования следующих свечей

3. Временной ряд можем рассмотреть как граф и использовать для его прогнозирования GNN

4. В последние годы создаются более эффективные архитектуры нейросетей, которые тоже могут помочь в прогнозировании финансовых временных рядов

Список литературы

[1] The most cited neural networks all build on work done in my labs.Jürgen Schmidhuber (2021, slightly updated 2022) https://people.idsia.ch/~juergen/most-cited-neural-nets.html (27.11.2022)

[2] Jürgen Schmidhuber's page on Recurrent Neural Networks (updated 2017) https://people.idsia.ch/~juergen/rnn.html (27.11.2022)

[3] Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019 https://arxiv.org/pdf/1911.13288.pdf (27.11.2022)

[4] LSTM – сети долгой краткосрочной памяти https://habr.com/ru/companies/wunderfund/articles/331310/ (перевод Christopher Olah Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/) (14.05.2023)

[5] Fei-Fei Li, Justin Johnson, Serena Yeung Loss Functions and Optimization (перевод: https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-3-funkciya-poter-i-optimizaciya/?utm_source=yandex.ru&utm_medium=organic&utm_campaign=yandex.ru&utm_referrer=yandex.ru) (14.05.2023)

[6] Shiva Verma Understanding different Loss Functions for Neural Networks https://shiva-verma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718 (14.05.2023)

[7] Vishal Yathish Loss Functions and Their Use In Neural Networks https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9 (14.05.2023)

[8] Javaid Nabi Estimators, Loss Functions, Optimizers —Core of ML Algorithms (перевод: https://id-lab.ru/posts/developers/funkcii/) (14.05.2023)

[9] Fei-Fei Li, Justin Johnson, Serena Yeung Training Neural Networks II https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-7-obuchenie-nejrosetej-chast-2/?utm_source=yandex.ru&utm_medium=organic&utm_campaign=yandex.ru&utm_referrer=yandex.ru (14.05.2023)

[10] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein Visualizing the Loss Landscape of Neural Nets 2018 https://arxiv.org/abs/1712.09913 (14.05.2023)

[11] Vadim Smolyakov Neural Network Optimization Algorithms https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61d (14.05.2023)

[12] Павел Садовников Методы оптимизации нейронных сетей https://habr.com/ru/articles/318970/ (14.05.2023)

[13] Kingma D., Ba J. Adam: A Method for Stochastic Optimization // arXiv, 2014. http://arxiv.org/abs/1412.6980 (14.05.2023)

[14] Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).

[15] Margherita Grandini, Enrico Bagli, Giorgio Visani Metrics for Multi-Class Classification: an Overview, arXiv, 2020 https://arxiv.org/abs/2008.05756 (14.05.2023)

[16] Egor Labintcev Метрики в задачах машинного обучения https://habr.com/ru/companies/ods/articles/328372/ (14.05.2023)

[17] Jiyang Wang What is "units"in LSTM layer of Keras? https://zhuanlan.zhihu.com/p/58854907 (20.05.2023)

[18] Time series forecasting https://www.tensorflow.org/tutorials/structured_data/time_series?hl=ru#recurrent_neural_network (20.05.2023)

[19] Shiva Verma Input and Output shape in LSTM (Keras) https://www.kaggle.com/code/shivajbd/input-and-output-shape-in-lstm-keras (20.05.2023)

[20] Ran Aroussi Welcome to the yfinance wiki https://github.com/ranaroussi/yfinance/wiki (20.05.2023)

[21] Mohhamad Fneish Keras_LSTM_Diagramhttps://github.com/MohammadFneish7/Keras_LSTM_Diagram (20.05.2023)

[22] Анализ временных рядов с помощью python https://habr.com/ru/articles/207160/ (20.05.2023)

[23] Max Rokatansky Анализ временных рядов https://habr.com/ru/companies/otus/articles/732080/ (20.05.2023)

[24] Marco Cerliani ARIMA for Classification with Soft Labelshttps://towardsdatascience.com/arima-for-classification-with-soft-labels-29f3109d9840 (20.05.2023)

[25] Brian Christopher Time Series Analysis (TSA) in Python - Linear Models to GARCH https://www.blackarbs.com/blog/time-series-analysis-in-python-linear-models-to-garch/11/1/2016#AR (20.05.2023)

[26] Python https://www.python.org/ (20.05.2023)

[27] Google colab https://colab.research.google.com/ (20.05.2023)

[28] yfinance https://pypi.org/project/yfinance/ (20.05.2023)

[29] NumPy https://numpy.org/ (20.05.2023)

[30] Pandas https://pandas.pydata.org/ (20.05.2023)

[31] sklearn.preprocessing.minmax_scale https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html (20.05.2023)

[32] TensorFlow https://www.tensorflow.org/?hl=ru (20.05.2023)

[33] Matplotlib: Visualization with Python https://matplotlib.org/ (20.05.2023)

Программная реализация: https://colab.research.google.com/drive/14NMK1WhhxmyxD63QCijKtWvdhTzViJaR?usp=sharing