ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки магистров по направлению**

**01.04.02 Прикладная математика и информатика.**

*Рухович Игорь Владимирович*

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Прогнозирование временных рядов  
с помощью рекуррентных нейронных сетей

|  |  |
| --- | --- |
|  | Научный руководитель:  д. ф.-м. н., профессор кафедры прикладной математики и  информатики НИУ ВШЭ - НН Калягин Валерий Александрович  Соруководитель:  приглашенный преподаватель кафедры прикладной математики и  информатики НИУ ВШЭ – НН Досов Санжар Музаффарович |

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ 4](#_Toc167749570)

[1.1. Актуальность темы 4](#_Toc167749571)

[1.2. Мотивация выбора данной темы 4](#_Toc167749572)

[2. ПОСТАВЛЕННЫЕ ЗАДАЧИ 7](#_Toc167749573)

[2.1. Задача 1. Обзор литературы 7](#_Toc167749574)

[2.2. Задача 2. Выбор данных, метрик и других условий 7](#_Toc167749575)

[2.3. Задача 3. Реализация RNN-моделей 8](#_Toc167749576)

[2.4. Задача 4. Измерения, сравнения и выводы 8](#_Toc167749577)

[3. ОБЗОР РЕШЕНИЙ И ЛИТЕРАТУРЫ 9](#_Toc167749578)

[3.1. Требования к данным. Обзор 9](#_Toc167749579)

[3.1.1. M-Competitions 9](#_Toc167749580)

[3.1.2. M4-Dataset 11](#_Toc167749581)

[3.2. Выбор метрик для сравнения 13](#_Toc167749582)

[3.2.1. OWA 13](#_Toc167749583)

[3.2.2. Average time 14](#_Toc167749584)

[3.3. Обзор моделей и методов 15](#_Toc167749585)

[3.3.1. ARMA и улучшения 16](#_Toc167749586)

[3.3.2. TBATS 20](#_Toc167749587)

[3.3.3. N-BEATS 21](#_Toc167749588)

[3.3.4. LTSF-Linear 23](#_Toc167749589)

[3.3.5. iTransformer 24](#_Toc167749590)

[3.3.6. PI-Transformer 26](#_Toc167749591)

[3.3.7. Naïve (baseline for M4) 28](#_Toc167749592)

[3.3.8. sNaïve (baseline for M4) 28](#_Toc167749593)

[3.3.9. Naïve2 (baseline for M4) 28](#_Toc167749594)

[4. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ 30](#_Toc167749595)

[4.1. Общая информация о проделанной работе 30](#_Toc167749596)

[4.2. Описание используемых методов 30](#_Toc167749597)

[4.2.1. RNN 31](#_Toc167749598)

[4.2.2. LSTM 33](#_Toc167749599)

[4.2.3. GRU 35](#_Toc167749600)

[4.2.4. Нормализация данных 37](#_Toc167749601)

[4.2.5. Модификация функций активации 38](#_Toc167749602)

[4.2.6. Persistence Initialization 38](#_Toc167749603)

[4.3. Полученные результаты 39](#_Toc167749604)

[ЗАКЛЮЧЕНИЕ 42](#_Toc167749605)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ЛИТЕРАТУРЫ 44](#_Toc167749606)

[ПРИЛОЖЕНИЕ 47](#_Toc167749607)

1. ВВЕДЕНИЕ
   1. Актуальность темы

Проблема прогнозирования временных рядов является одной из наиболее актуальных в современном мире. Временные ряды окружают нас повсюду, и зачастую возникает необходимость оценивать, какие значения будут принимать те или иные величины в будущем. Иногда требуется сделать акцент на качественных предсказаниях, а в других случаях помимо качества важна и высокая скорость обработки данных, особенно когда речь идет о больших объемах информации.

Людям приходится прогнозировать различные тренды, объемы продаж, пассажиропотоки, поломки оборудования, потребление электроэнергии и многое другое. Кроме того, необходимо рассчитывать нагрузку на веб-ресурсы, формировать бюджеты на основе прогнозируемых расходов. Список сфер, где требуется применение методов прогнозирования временных рядов, можно продолжать бесконечно.

Точные и своевременные прогнозы позволяют принимать более обоснованные управленческие решения, оптимизировать ресурсы, минимизировать риски и повышать эффективность деятельности организаций. Именно поэтому исследования в области прогнозирования временных рядов имеют важное практическое значение и продолжают привлекать пристальное внимание ученых и специалистов.

* 1. Мотивация выбора данной темы

Прогнозирование временных рядов – это область, в которой не существует единого, безоговорочно лучшего алгоритма, в отличие, например, от обработки естественного языка (NLP), где доминируют мощные архитектуры, такие как GPT и BERT. В случае временных рядов, различные подходы демонстрируют неоднородные результаты, в зависимости от исходных данных и требований, предъявляемых к прогнозирующим моделям.

Отсутствие универсального решения для задачи прогнозирования временных рядов свидетельствует о том, что в этой области есть потенциал для дальнейшего качественного развития. Хотя наше исследование, скорее всего, не станет прорывным или лучшим в своем роде, мы постараемся предложить решение, которое может оказаться полезным в некоторых ситуациях.

Архитектура трансформеров, активно применяемая в машинном обучении, отличается высокой сложностью: в таких моделях большое количество обучаемых параметров, а механизм внимания (attention) работает за квадратичное от размера последовательности время. Такие модели требуют колоссальных вычислительных ресурсов, и при этом, возможно, не принесут значительной пользы в нашей конкретной задаче. В то же время, простые модели, основанные на авторегрессии и скользящем среднем (ARMA) с многочисленными модификациями, могут оказаться слишком простыми и не улавливать некоторые закономерности. Таким образом, мы пришли к выбору структуры рекуррентной нейронной сети (RNN) в качестве основы для нашего исследования.

Рекуррентные нейронные сети являются мощным инструментом для решения задач, связанных с обработкой последовательных данных. Именно это и требуется в нашем случае. В отличие от традиционных нейронных сетей, RNN способны учитывать контекстную информацию, что делает их особенно эффективными при работе с данными, где важную роль играет временная зависимость.

Ключевое преимущество RNN заключается в их способности запоминать предыдущие состояния и использовать эту информацию для принятия решений в текущий момент времени. Это позволяет им лучше улавливать скрытые закономерности в данных и делать более точные прогнозы. Например, при прогнозировании курса акций RNN могут учитывать не только текущие показатели, но и историю изменений, что повышает точность предсказаний. Современные улучшенные версии RNN, такие как LSTM и GRU, довольно эффективно решают проблемы затухания и взрыва градиентов, и помогают сети запоминать долгосрочные зависимости.

В целом, рекуррентные нейронные сети представляют собой перспективный подход к решению задач прогнозирования временных рядов, позволяя извлекать ценную информацию из последовательных данных и делать более точные и обоснованные предсказания.

1. ПОСТАВЛЕННЫЕ ЗАДАЧИ

Перейдём к постановке задач работы. Для нас важно выбрать некоторый подтип задач прогнозирования временных рядов, провести обзор актуальной научной литературы, включая статьи, реализации и другие источники, посвященные современным и популярным методам прогнозирования. Затем предстоит выделить набор моделей, датасетов и метрик, которые будут использованы, воспроизвести модели и посчитать их метрики. После должны быть реализованы несколько видов RNN-моделей, добавлены модификации. В завершение мы проведём сравнение всех полученных результатов.

Ниже поставленные задачи запишем более формально, разложим по отдельным подпунктам и дадим некоторые дополнительные уточнения.

* 1. Задача 1. Обзор литературы

Провести обзор существующих решений и научной литературы, посвященных прогнозированию временных рядов. Особое внимание нужно уделить наиболее современным и востребованным моделям, а также подробно изучить принципы их функционирования. Это позволит сформировать более глубокое понимание текущего состояния предметной области.

* 1. Задача 2. Выбор данных, метрик и других условий

Необходимо подобрать наборы данных и метрики для тестирования и сравнения различных решений. Важно, чтобы набор данных был репрезентативным и включал в себя временные ряды различной природы из разных сфер деятельности человека. Это позволит найти более универсальное решение, которое будет эффективно работать с широким спектром данных. При выборе метрик следует учитывать, что мы должны уметь оценивать как качество, так и скорость работы прогнозирующих моделей. Кроме того, необходимо зафиксировать и выровнять исходные условия и ограничения для используемых моделей, чтобы обеспечить справедливое сравнение их результатов.

* 1. Задача 3. Реализация RNN-моделей

Реализовать несколько вариантов рекуррентных нейронных сетей, решающих поставленную задачу на выбранных наборах данных. Важно, чтобы модели не имели внешних настраиваемых гиперпараметров, зависящих от данных, и могли работать сразу, "из коробки". Соответственно, подбор гиперпараметров должен происходить внутри моделей автоматически на основе вводных данных. Следует также реализовать модификации для наиболее многообещающих моделей с целью повысить качество их работы.

* 1. Задача 4. Измерения, сравнения и выводы

Необходимо воспроизвести заявленные в источниках результаты всех выбранных моделей на отобранных наборах данных, обеспечив при этом равные условия для тестирования. Затем следует измерить все выбранные метрики. Аналогичным образом нужно провести оценку собственных реализаций рекуррентных нейронных сетей, разработанных на предыдущем этапе. Далее требуется сопоставить полученные результаты с показателями остальных подходов, использованных в исследовании. На основе проведенного анализа сделать выводы относительно качества и эффективности реализованных моделей прогнозирования.

1. ОБЗОР РЕШЕНИЙ И ЛИТЕРАТУРЫ
   1. Требования к данным. Обзор

Как уже упоминалось выше, в мире существует и появляется огромное множество временных рядов в самых разных сферах деятельности человека и самой разной природы. Для эксперимента в данной работе было бы не честно брать временные ряды из одной сферы (например, курсы акций), потому как высокое значение любой метрики для этого набора данных абсолютно ничего не будет говорить о качестве на других данных. Возможно, что наша модель полезна только для улавливания определенных типов зависимостей. А если речь заходит о курсах акций, валют или других биржевых инструментов, есть высокая вероятность того, что предсказать их стоимость просто невозможно – по некоторым оценкам, она практически случайна с точки зрения истории курса, без использования внешних данных.

Второй важный аспект – это периодичность данных и сезонность. В зависимости от периодичности сбора данных, модель и подход могут меняться. Если какие-то данные собираются всего раз в год, то, наверняка, этих данных будет не очень много, и нам придется оперировать небольшим количеством информации. Напротив, если данные собираются каждый час или раз в секунду, у модели будет больший простор для поиска зависимости, но придется убирать шум, выбирать из данных самое полезное.

Нами было изучено несколько публичных датасетов, использующихся в настоящее время для прогнозирования временных рядов. Среди всех претендентов наиболее выделяются и подходят под наши требования датасеты под названием M, из соревнований Makridakis Competitions.

* + 1. M-Competitions

Соревнования Makridakis Competitions по прогнозированию временных рядов являются одними из наиболее авторитетных и престижных в своей области. Они были основаны в 1982 году известным греческим специалистом по прогнозированию Спиросом Макридакисом с целью объективной оценки и сравнения различных методов прогнозирования [1]. За годы своего существования соревнования приобрели широкую известность и стали своеобразным эталоном качества для исследователей и практиков, работающих с временными рядами.

Популярность Makridakis Competitions обусловлена несколькими факторами. Во-первых, они предлагают разнообразные наборы данных, охватывающие широкий спектр областей - от экономики до демографии. Это позволяет участникам проверять эффективность своих методов на реальных, а не искусственных примерах. Во-вторых, соревнования проводятся по строго определенным правилам, что гарантирует объективность оценки результатов. Также победители получают престижные награды, что должно стимулировать исследователей к поиску инновационных решений.

Датасеты M отличаются высоким качеством, тщательно подобраны и документированы, что делает их идеальным материалом для апробации методов прогнозирования. Кроме того, использование этих данных достаточно широко распространено и позволяет сравнивать результаты исследований, проводимых разными авторами.

За всё время было проведено 6 соревнований M. Рассмотрим датасеты, которые на них были предложены:

1. M-Competition (1982) – первое соревнование, 1001 временной ряд из различных областей.
2. M2-Competition (1993) – второе соревнование, 29 наборов данных, включая экономические, демографические и другие.
3. M3-Competition (2000) – 3003 временных ряда из 24 различных источников, охватывающих широкий спектр областей.
4. M4-Competition (2018) – 100000 временных рядов, включая микро-, мезо- и макроэкономические данные, а также данные из области туризма, энергетики и других сфер. Самое популярное у исследователей соревнование, поскольку содержит больше всего различных данных.
5. M5-Competition (2020) – сфокусировано на прогнозировании продаж в рознице. 42840 временных рядов, связанных с продажами в 10 магазинах сети Walmart.
6. M6-Competition (2022) – последнее из проведенных соревнований. Фокус на прогнозе финансовых данных. Включает данные о 50 акциях из S&P500 и 50 международных ETF-фондах.
   * 1. M4-Dataset

Наш выбор в итоге пал на датасет со соревнования M4. Это самый обширный набор данных за всё время проведения Makridakis Competition. Он содержит 100 тысяч временных рядов из 6 сфер: демография, финансы, индустрия, макроэкономика, микроэкономика и другие, включая данные о туризме, человеческом труде, биржевых сделках, недвижимости, заработных платах, транспорте, природных ресурсах и окружающей среде [2].

По заверениям авторов, датасет был собран с целью представить всё, что есть в мире настолько, насколько это возможно. Изначально было собрано 900 тысяч наборов данных, из которых затем для соревнования случайно были отобраны 100 тысяч наборов.

Данные также имеют разделение по периодичности сбора. Измерения проводились один раз в год, квартал, месяц, неделю, день или час. На (рис. 1) представлена сводная таблица по данным от авторов соревнования.



Рисунок . Сводная таблица по данным соревнования M4

Все наборы данных, предложенные в соревновании, уже разделены на тренировочную и тестовую выборки. Размер тестовой выборки всегда фиксирован, и составляет, в зависимости от периодичности сбора, 6, 8, 18, 13, 14 и 48 периодов соответственно. Размер тренировочной выборки в данных не фиксирован, но ограничен снизу 13, 16, 42, 80, 93 и 700 наблюдениями в зависимости от периодичности соответственно.

Все модели, обучаемые на датасете M4, не должны иметь доступ к какой-либо посторонней информации. Авторы не раскрывали точные источники, времена сбора данных и все тестовые выборки на момент проведения соревнования для чистоты эксперимента. Отдельно обратим внимание на то, что все временные ряды в датасете – одномерные (англ. unvariate). То есть не содержат никаких посторонних признаков. Единственный дополнительный признак, который разрешено использовать – это период самой выраженной сезонности. Этот признак фиксирован для каждой периодичности данных и присутствует в датасете.

Было решено, что данный датасет полностью покрывает все наши требования по данным, поэтому во всех экспериментах будет использован именно он. Организаторы Makridakis Competition 4 также предложили собственную метрику качества, общую для всего датасета, о которой поговорим в следующей главе.

* 1. Выбор метрик для сравнения
     1. OWA

Overall Weighted Average (OWA) – метрика качества прогнозирования, предложенная авторами соревнования M4, и используемая в этом соревновании. Представляет собой среднее арифметическое значений двух других метрик, MASE и sMAPE, нормированных на некоторый бейзлайн – значение этих же метрик для наивного решения (ф. 3.1).

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.1) |

Где , , – метрики модели M для произвольного подмножества данных из датасета. и – метрики бейзлайн-модели Naive2, предложенной авторами соревнования, на тех же данных. Принцип работы модели Naïve2 будет описан в будущих главах. Таким образом, OWA равное 1 будет соответствовать наивному решению. Значения, большие 1 будут говорить о слабом качестве модели, а меньшие – о лучшем качестве. Минимальное значение OWA, достигнутое во время соревнования и проверенное авторами, составляет 0.821, и было достигнуто инженером из Uber Technologies.

Метрики MASE и sMAPE считаются вначале для каждого временного ряда по отдельности, а затем усредняются. Сначала находится среднее по каждому frequency (yearly, monthly, hourly, etc.), а затем эти величины усредняются в общую метрику. Таким образом общая метрика будет равным образом зависеть от результатов моделей на разных типах данных. Ниже приведены формулы расчёта MASE (ф. 3.2) и sMAPE (ф. 3.3) для отдельно взятого набора данных.

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.2) |
|  |  | ф. (3.3) |

Где:

* – элемент тестовой выборки под номером t
* – элемент тренировочной выборки под номером t
* h – размер тестовой выборки
* n – размер тренировочной выборки
* m – интервал сезонности (закреплён для каждого типа данных; например, для monthly (ежемесячных) измерений m = 12)

Поскольку метрика OWA является базовой для соревнования M4, и её часто используют другие исследователи при публикации статей, будем считать, что эта метрика хорошо оценивает качество работы моделей, а заодно подходит нам для эксперимента.

* + 1. Average time

Помимо метрики качества в нашей работе необходимо также учитывать временные затраты. Как было упомянуто в мотивации, мы хотим найти метод, который позволит достаточно оперативно получать прогнозы достойного качества.

Поэтому предлагается дополнительно для каждого набора данных измерять следующие величины:

* Полное время тренировки модели
* Полное время, затраченное на предсказание модели

Данные метрики затем будем усреднять аналогично принципу, предложенному в OWA: усреднение по каждому frequency, а затем общее усреднение.

* 1. Обзор моделей и методов

Для сравнения с собственными реализациями был проведен анализ существующих решений по прогнозированию временных рядов. Исследование литературы показало, что на данный момент существует принципиально 3 вида моделей для решения задачи Time Series Forecasting. Перечислим их в порядке усложнения.

Это классические модели из семейства ARMA и не только, часто основанные на комбинации двух более простых моделей: авторегрессии (AR) и скользящего среднего (MA). У этого семейства моделей существуют многочисленные модификации. О некоторых из них поговорим в следующих пунктах.

Второй вид – различные нейросетевые модели, созданные без применения механизма attention. К этой группе относится большое число подходов, от простых персептронов и их комбинаций до сетей с остаточными связями или сложных рекуррентных нейронных сетей.

В отдельную подгруппу выделим модели с использованием механизма attention. Сюда входят набравшие сейчас популярность трансформеры, а также некоторые другие модели. Механизмы attention и self-attention [3] – это ключевые компоненты в современных нейросетевых архитектурах, которые позволяют модели фокусироваться на наиболее релевантных частях входных и уже сгенерированных данных при генерации новых последовательностей. Практика показывает, что это очень сильный механизм, который, однако, требует больших вычислительных ресурсов при работе. Механизм работает за квадратичное от размера последовательности время, что может быть неуместно в некоторых задачах.

В итоге было решено отобрать по две наиболее новых и перспективных на момент исследования модели каждого вида. Их описание будет представлено в следующих пунктах.

* + 1. ARMA и улучшения

Идея модели ARMA была предложена ещё в 1951 году [4]. В изначальном эта модель применима только к стационарным временным рядам, то есть рядам, статистические характеристики которых (среднее, дисперсия, автокорреляционная функция) не зависят от времени. Другими словами, модель подразумевает отсутствие тренда в данных. Для работы с нестационарными рядами используются расширенные версии модели, такие как ARIMA (состоит из авторегрессии, интегрирования и скользящего среднего) и SARIMA (сезонная ARIMA), их мы рассмотрим в следующих абзацах.

В основе модели ARMA(p, q) лежит предположение, что значение временного ряда в текущий момент времени может быть представлено как линейная комбинация предыдущих значений ряда (авторегрессионная компонента) и случайных возмущений (компонента скользящего среднего). Математически это можно записать следующим образом (ф. 3.4):

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.4) |

Где:

* – значение временного ряда в момент времени t
* – котоэффициенты авторегрессионной компоненты
* – коэффициенты компоненты скользящего среднего
* – случайное возмущение в момент времени t

Параметры p и q определяют порядок авторегрессионной и скользящей компонент соответственно и подбираются вручную, перебором или на основе анализа автокорреляционной и частичной автокорреляционной функций исходного ряда.

В дальнейшем появилась модель ARIMA(p, d, q). Она была описана в книге [5] в 1970-м. I – сокращение от англ. integrated. ARIMA включает в себя дополнительный этап интегрирования (а точнее, взятия производной), позволяющий преобразовать нестационарный ряд в стационарный путем взятия разностей определенного порядка. Идея заключается в том, что перед началом метода ARMA над временным рядом последовательно d раз производится операция взятия разностей первого порядка. При d = 1 наша изначальная формула примет следующий вид (ф. 3.5):

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.5) |

Такая модель будет предсказывать уже не значение целевой переменной, а некоторую прибавку к предыдущему значению. Тем не менее, она всегда позволяет восстановить предсказание оригинального целевого значения.

Параметр d подбирается таким образом, чтобы после d операций дифференцирования ряд стал стационарным. Для проверки на стационарность обычно используют статистические тесты. Например, тест Дики-Фуллера [6] на нестационарность ряда. Также используют расширенный тест Дики-Фуллера, предложенный позднее теми же авторами, и тест Филлипса-Перрона [7].

Модель SARIMA(p, d, q, P, D, Q, s) была предложена изобретателями ARIMA позднее в 1970-х годах. Эта модель добавляет в предыдущие компоненту сезонности s. s – некоторое натуральное число, характеризующее количество периодов в данных, через которое сезонный эффект должен повториться.

Сезонностью обладают многие временные ряды. Например, посещения сайтов в интернете и различные взаимодействия в этой сети чаще происходят днём, чем ночью. Новогодние украшения чаще покупают под новый год, а потребление газа и электричества в северных странах уменьшается весной, когда отключается отопление. Есть множество примеров сезонности в данных, а иногда в них бывает даже несколько сезонностей. Более сложные модели позволяют это учесть. SARIMA позволяет учитывать лишь одну (самую выраженную) сезонность. Тем не менее, это бывает очень полезно.

Компоненты P, Q, D в SARIMA отвечают за сезонные авторегрессию, скользящее среднее и сезонное дифференцирование. Таким образом, с появлением сезонности в модели значительно возрастает число параметров. Это повышает вероятность переобучения, поэтому использовать такую модель стоит с большей осторожностью. Полная формула SARIMA (ф. 3.6) выглядит намного сложнее из-за нескольких дифференцирований:

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.6) |

Где:

* – значение временного ряда в момент времени t
* – оператор сдвига назад во времени на i позиций.
* – оператор взятия разностей порядка d (используются для устранения нестационарности)
* – оператор взятия сезонных разностей порядка D
* – несезонная авторегрессионная компонента порядка p
* – несезонная компонента скользящего среднего порядка q
* – сезонная авторегрессионная компонента порядка P
* – сезонная компонента скользящего среднего порядка Q
* s – длина сезонного периода
* – порядки несезонных компонент
* – порядки сезонных компонент

В нашем эксперименте мы попробуем использовать все 3 описанных алгоритма (ARMA, ARIMA, SARIMA) и сравнить их качество.

* + 1. TBATS

Алгоритм TBATS (Trigonometric, Box-Cox, ARMA errors, Trend, and Seasonal components) [8] был разработан для прогнозирования временных рядов со сложными сезонными паттернами. В отличие от ранее рассмотренной SARIMA, TBATS позволяет учитывать более сложные и разнообразные компоненты сезонности и тренда, что делает его подходящим для анализа временных рядов с несколькими сезонными периодами и нелинейными особенностями.

Основные компоненты TBATS включают: тригонометрическое представление сезонных компонентов, Box-Cox [9] трансформацию (ф. 3.7) (приближение распределения к нормальному) для стабилизации дисперсии и обеспечения аддитивности сезонных и трендовых компонентов, а также моделирование остаточных компонентов с помощью ARMA моделей.

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.7) |

В модели TBATS реализованы многочисленные улучшения. Во-первых, алгоритм позволяет моделировать множественную и нецелочисленную сезонность, в то время как SARIMA ограничена одним целочисленным сезонным периодом. Во-вторых, TBATS использует меньшее количество параметров, что снижает риск переобучения и улучшает стабильность модели. Также в методе уже встроено использование Box-Cox трансформации, что позволяет обрабатывать нелинейные зависимости, в отличие от линейных предположений SARIMA. Эти преимущества делают TBATS особенно полезным для прогнозирования в областях с данными, характеризующимися сложными и многоуровневыми сезонными паттернами.

В нашем эксперименте мы будем обучать модель TBATS с помощью пакета в языке Python. Сравним результаты этой модели с SARIMA и попробуем заметить отличия.

* + 1. N-BEATS

Алгоритм N-BEATS (Neural Basis Expansion Analysis Time Series) (2020) – это модель глубокого обучения, специально разработанная для прогнозирования временных рядов [10]. Ее архитектура основана на использовании глубокой нейронной сети с остаточными связями и блочной структурой, что позволяет эффективно моделировать временные зависимости и компоненты временных рядов.

В основе модели лежат специально разработанные авторами резервные и прогностические блоки (Backcast and Forecast Blocks). Каждый блок состоит из полностью связанных слоев (Fully Connected layers), выполняющих две функции - резервную (обработка входных данных для выделения информации, не связанной с будущими прогнозами) и прогностическую (предсказание будущих значений временного ряда).

Между блоками используются, в том числе, остаточные связи (Residual Connections). Использование остаточных связей позволяет передавать необработанные сигналы на следующие слои, улучшая процесс обучения и прогнозирования [11]. Модель использует двойное остаточное стекание (Doubly Residual Stacking, DRESS), включающее резервную связь и частичные прогнозы на уровне блоков, которые затем агрегируются для получения окончательного прогноза [12].

Конечная архитектура состоит из нескольких стеков, каждый из которых содержит несколько блоков. Это позволяет модели быть глубокой и гибкой, обеспечивая возможность учитывать сложные временные зависимости. В разных стеках могут использоваться различные типы функций для улучшения качества прогноза. На (рис. 2) представлена предложенная авторами схема архитектуры N-BEATS.

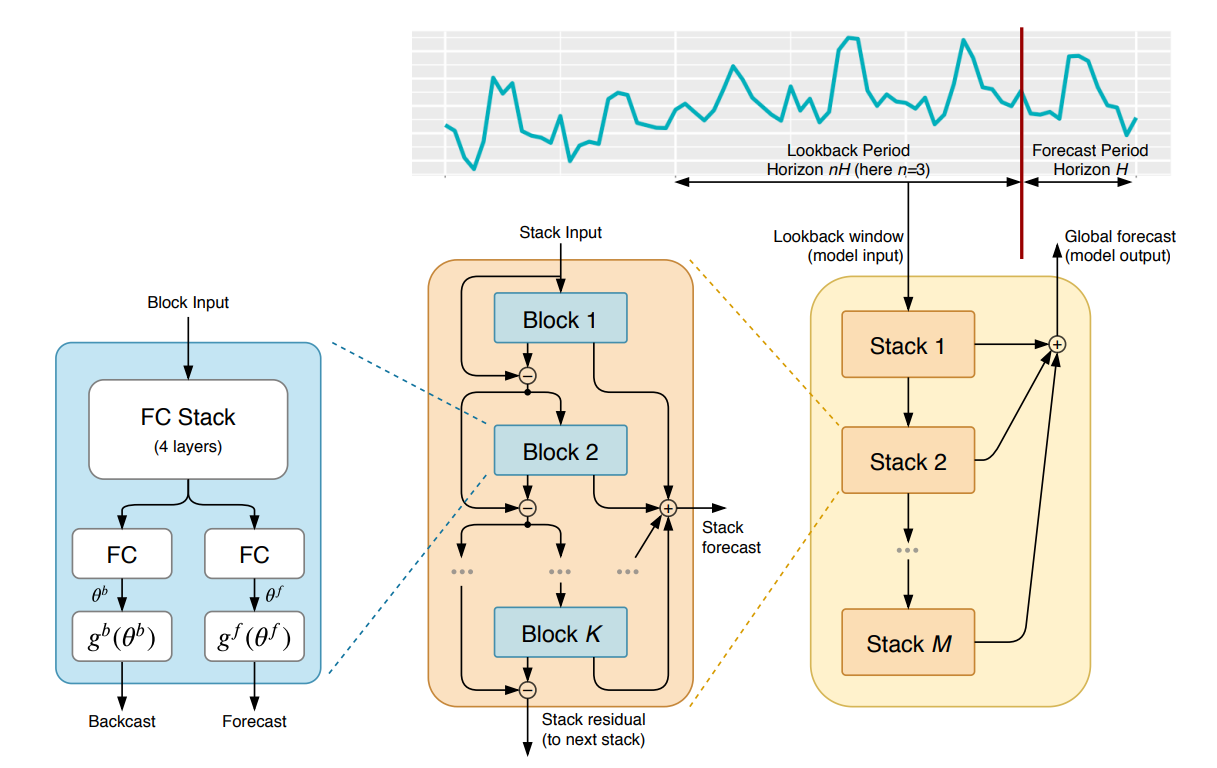


Рисунок . Архитектура модели N-BEATS

Поговорим о принципе работы алгоритма. Входной временной ряд передается в первый блок, который выполняет резервную и прогностическую функции. Резервный выход первого блока передается на вход следующего блока вместе с остаточной связью, а прогностический выход используется для частичного прогноза. Прогностические выходы всех блоков в стеке агрегируются для формирования общего прогноза. По заявлениям авторов, это позволяет учитывать как краткосрочные, так и долгосрочные зависимости во временных рядах.

Для повышения точности прогноза, создатели предлагают использовать модель в ансамбле с несколькими экземплярами, каждый из которых обучается на разных подвыборках данных. Это должно улучшить общее качество за счет снижения возможных ошибок отдельных моделей и повышения устойчивости к различным типам временных рядов.

По словам авторов, архитектура N-BEATS доказала свою эффективность на множестве наборов данных, обеспечивая высокую точность и надежность прогнозов без необходимости использования дополнительных признаков. По собственным расчетам создателей модели, N-BEATS опережает по OWA всех участников соревнования M4.

* + 1. LTSF-Linear

В этой довольно свежей статье авторы предлагают и проверяют очень смелую гипотезу. Утверждается, что простая линейная модель, состоящая из одного FC слоя, способна обогнать по производительности современные трансформенные алгоритмы [13].

Авторы предлагают обучать линейный слой, входами которого будет являться известная последовательность наблюдений, а на выходе будет происходить линейная регрессия в неизвестные точки (рис. 3). Обучаться модель должна на похожих разбиениях тренировочной выборки, а потом, как утверждается, удастся “сдвинуть” модель вперёд по временной шкале и успешно предсказывать будущее.

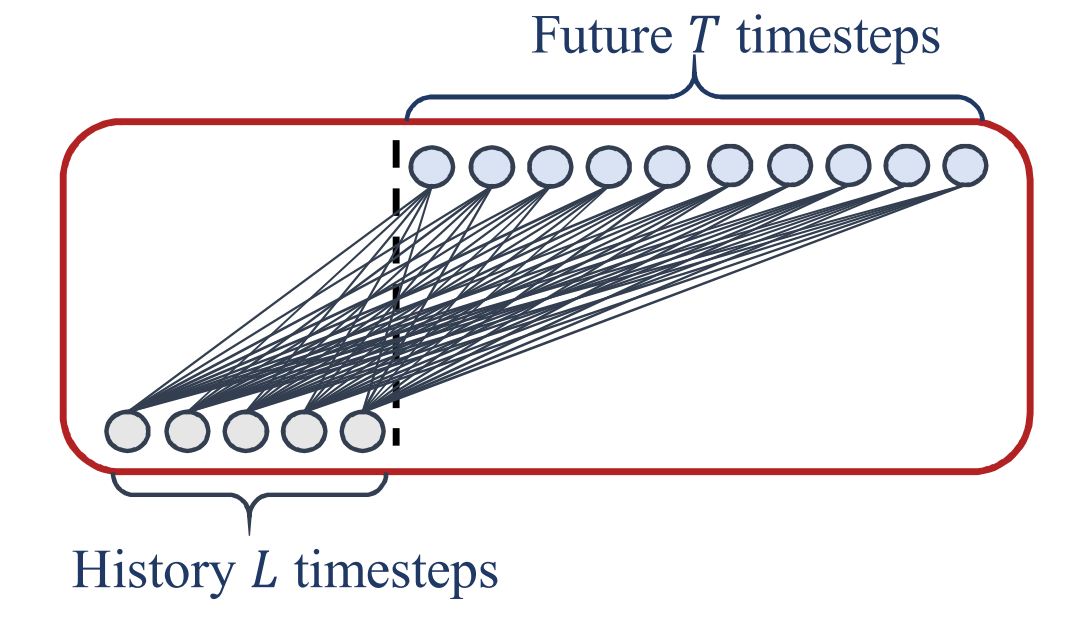


Рисунок . Архитектура одной модели LTSF-Linear

Авторы также предлагают попробовать использовать ансамбль таких линейных регрессий. Утверждается, что качество работы в таком случае возрастает. Однако, если читатель обратится к простым математическим знаниям, можно вспомнить, что ансамбль (или линейная комбинация) линейных моделей – это по-прежнему линейная модель, которую можно выразить проще через единственную матрицу.

Детальнее изучив исходный код [14] данной реализации, мы пришли к пониманию, что для данного типа моделей создатели действительно не используют функций активации, а значит они всегда обучали линейную модель (впрочем, в статье это не скрывается).

Поскольку создатели этой модели для экспериментов использовали другие наборы данных, отличные от выбранного нами M4, нам понадобится воспроизвести результаты этой довольно смелой статьи и разобраться, действительно ли такая простая модель выдаёт заявленный результат.

* + 1. iTransformer

Первая трансформерная архитектура, которую мы хотим рассмотреть в эксперименте – iTransformer (2024). Алгоритм iTransformer (Inverted Transformer) представляет собой модифицированный подход к применению архитектуры трансформеров для прогнозирования временных рядов. В отличие обычной модели, которая токенизирует временные ряды построчно (по временным меткам), iTransformer фактически транспонирует временные ряды и строит токены переменных (variate tokens) (рис. 4), что, по заверениям автором, позволяет лучше выявлять корреляции между различными временными рядами [15].

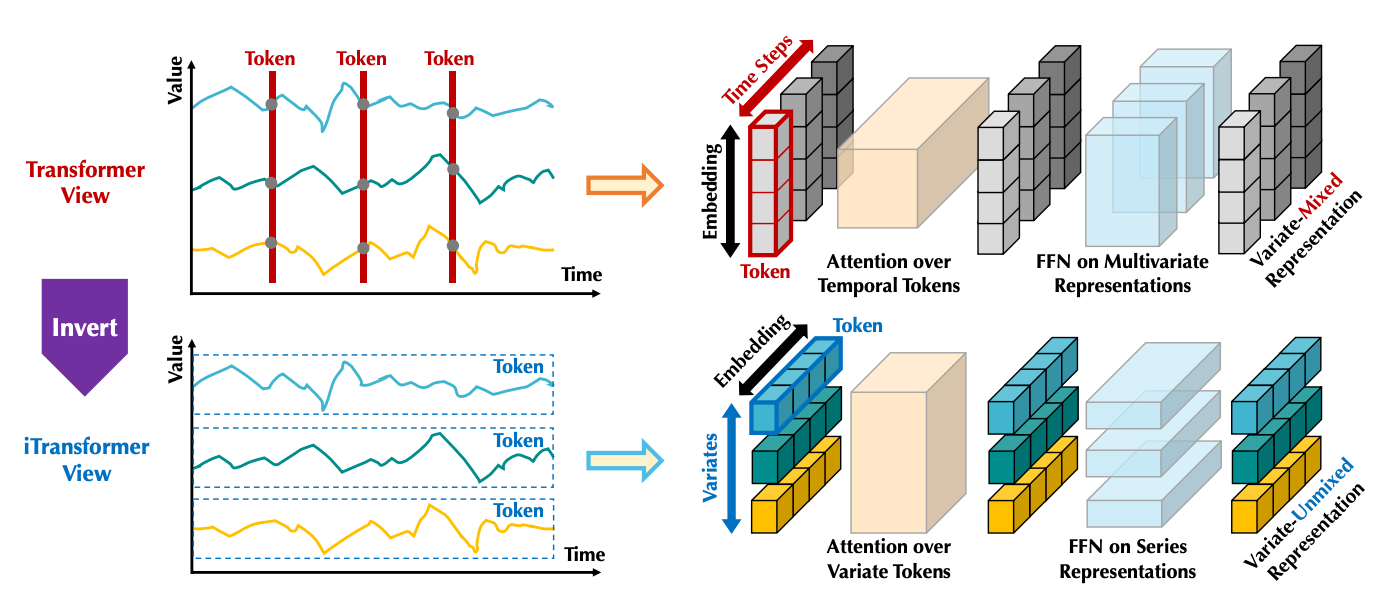


Рисунок . Схема восприятия входных данных алгоритмом iTransformer

Архитектура iTransformer включает те же ключевые компоненты, что и обычный трансформер: эмбеддинг временных рядов, несколько слоев внимания (multivariate attention), нормализации, сеть прямого распространения и проекцию. В процессе прогнозирования модель получает эмбеддинги входных временных рядов, их обработку в блоках Transformer и проекцию токенов обратно в прогнозируемые временные ряды (рис. 5).

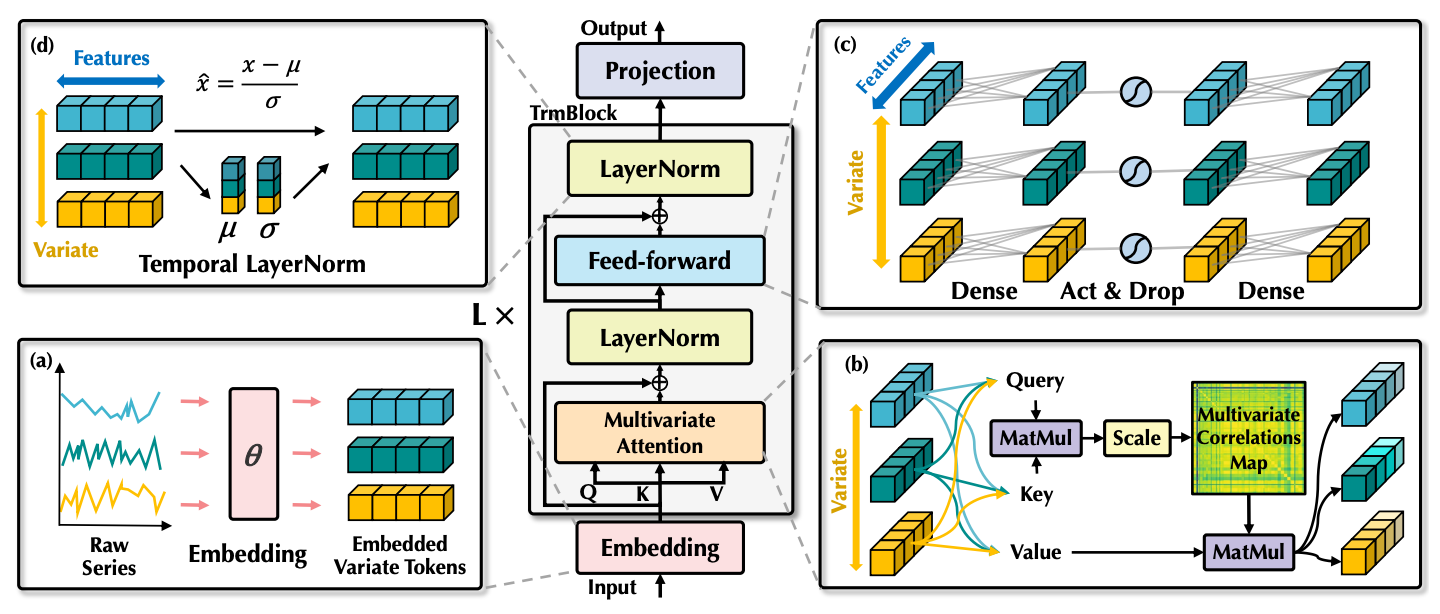


Рисунок . Приблизительная схема работы iTransformer

Экспериментальные результаты авторов статьи показали, что iTransformer достигает передовых показателей на реальных данных, значительно улучшая производительность по сравнению с традиционными методами на основе Transformer. Модель эффективно выявляет корреляции и использует большие окна наблюдений для прогнозирования.

Авторы статьи не проводили эксперимент на датасете М4, поэтому нам придется самостоятельно провести это измерение. Данная модель больше подходит для прогнозирования многомерных временных рядов, но, из-за своей новизны и относительно простой идеи, она интересна и в нашем эксперименте.

* + 1. PI-Transformer

Модель PI-Transformer (2023) является ещё одним представителем трансформеров в нашем исследовани. PI-Transformer (Persistence Initialization Transformer) представляет собой модификацию классического трансформера, специально адаптированную авторами для задач прогнозирования временных рядов. Архитектура модели в целом схожа с обычным трансформером, но имеет ряд важных нововведений [16].

На первом этапе происходит особая нормализация входных данных. Данные делятся на среднее значение за последние H наблюдений (H – гиперпараметр), а затем к ним применяется логарифмическое преобразование. По словам создателей, это позволяет лучше учитывать тренд временного ряда вблизи окна прогнозирования. Для получения предсказаний используется обратное преобразование:

Для генерации прогнозов модель использует архитектуру Transformer-декодера, аналогичную генеративным языковым моделям в обработке естественного языка. Важная особенность тут в том, что вместо стандартной техники Layer Normalization используется более новая ReZero нормализация. Также используется RoPE (Rotary Positional Encoding) вместо стандартного синусоидального кодирования.

Для учёта позиции элементов во временном ряду используется относительное позиционное кодирование Rotary Positional Encodings вместо абсолютного синусоидального кодирования. Это обеспечивает лучшую индуктивную предрасположенность для задач прогнозирования, так как временные ряды часто не имеют особого значения для абсолютного положения внутри окна предсказания.

Основная техника, которая делает наибольший вклад в качество работы модели – это Persistence Initialization. Она обеспечивает, чтобы начальные прогнозы (до начала обучения) модели были равны прогнозам модели наивного подхода (Persistence Model), то есть предыдущим значениям временного ряда. Для реализации этого добавляются две компоненты: остаточная связь и обучаемый множитель , на который изначально умножаются выходы трансформера. В результате обеспечивается хороший старт для дальнейшего обучения.

Авторы тестировали эту модель на датасете М4, и в том числе сравнивали результаты с методом N-BEATS. Исходя из приложенных в статье измерений, PI-Transformer показывает отличное качество. Добавим его в наш эксперимент.

* + 1. Naïve (baseline for M4)

Ниже приведём формулы нескольких простых бейзлайн-методов, которые реализованы авторами M4 Competition. Один из них понадобится для расчета целевой метрики OWA.

Формула Naïve проста: предсказание модели на всём горизонте равно последнему доступному наблюдению (ф. 3.8)

|  |  |  |
| --- | --- | --- |
|  |  | ф. (3.8) |

Где – наблюдение под номером i. При этом n и m – размеры тренировочной и тестовой выборок соответственно.

* + 1. sNaïve (baseline for M4)

sNaive – сокращение от Seasonal Naïve. Метод, аналогичный предыдущему, но предсказание равно последнему известному наблюдению за тот же период. Например, если измерения ежемесячные, то , где k такое натуральное число, что .

Этот метод, скорее всего, будет отлично работать для стационарных временных рядов. В общем случае мы получаем хороший baseline.

* + 1. Naïve2 (baseline for M4)

Последний метод является некоторой усложненной комбинацией двух предыдущих. Вначале проверяется наличие автокорреляции (ACF-test в R или Python) для определения выраженности сезонности. Если сезонность не обнаружена – результат равен результату модели Naïve. Иначе к данным применяется мультипликативная декомпозиция (пакет stats в R, statsmodels в Python) для выделения сезонной компоненты. Данные разделяются на сезонную и остаточную часть. Ко второй части применяется модель Naïve (берётся последнее известное значение), а затем результат умножается на сезонную компоненту.

Модель Naïve2 была использована как базовое решение в соревновании M4. На метрики этой модели нормировались MASE и sMAPE всех остальных. Значение OWA для этой модели равно 1.

1. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ
   1. Общая информация о проделанной работе

Выбрав в предыдущей главе модели, данные и метрики для экспериментов, мы воспроизведём результаты из соответствующих статей и рассчитаем недостающие показатели. Помимо этого, реализуем несколько вариантов RNN-моделей и их модификаций, о которых поговорим далее. Для каждой из реализаций будут подобраны гиперпараметры на основе экспериментов или заданы инструкции для автоматического подобра на основе входных данных. В итогах для каждой реализации будем отображать единственный результат – лучший из всех исследованных.

Для экспериментов будем использовать язык программирования Python3, среду разработки VSCode. Реализации базовых блоков нейронных сетей, а также вспомогательную инфраструктуру для обучения на GPU, проведения операции обратного распространения ошибки, будем использовать из пакета PyTorch, разработанного компанией Facebook. Исходный код будем запускать удалённо на виртуальной машине с операционной системой Ubuntu v22.04, CPU c 32 логическими ядрами, 256GB RAM, с доступом к 3 GPU NVIDIA Tesla A100. Такая конфигурация необходима, чтобы в разумные сроки проводить обучение большого количества моделей, как запланировано в работе. Вычислительные мощности предоставлены работодателем для написания данной работы. Исходный код, использованный в работе, а также итоговые таблицы будут выложены в git-репозитории по ссылке **(Приложение 2)**.

* 1. Описание используемых методов

Итак, перед нами стоит задача прогнозирования одномерных временных рядов с помощью рекуррентных нейронных сетей (RNN). Прогнозировать придется на несколько значений вперёд, имея доступ только к тренировочным данным. Ниже перечислим модели, реализованные в эксперименте, и их модификации. Дополнительно кратко описана их архитектура.

Общая архитектура используемых моделей будет похожей. Вначале мы подаём временной ряд в RNN-блок (их может быть несколько), затем добавляем 2 полносвязных слоя, и на выходе у сети будет 1 нейрон, поскольку мы решаем задачу регрессии (рис. 6). Для простоты схемы опустим функции активации, нормализацию и подробное содержимое блоков. Опишем эти детали позже. Полносвязные слои должны постепенно сузить пространство, в котором оперирует RNN, до единственного числа. За функцию потерь возьмём MSE (средний квадрат ошибки). Дополнительно в некоторые версии добавим остаточную связь Persistence Initialization (PI).

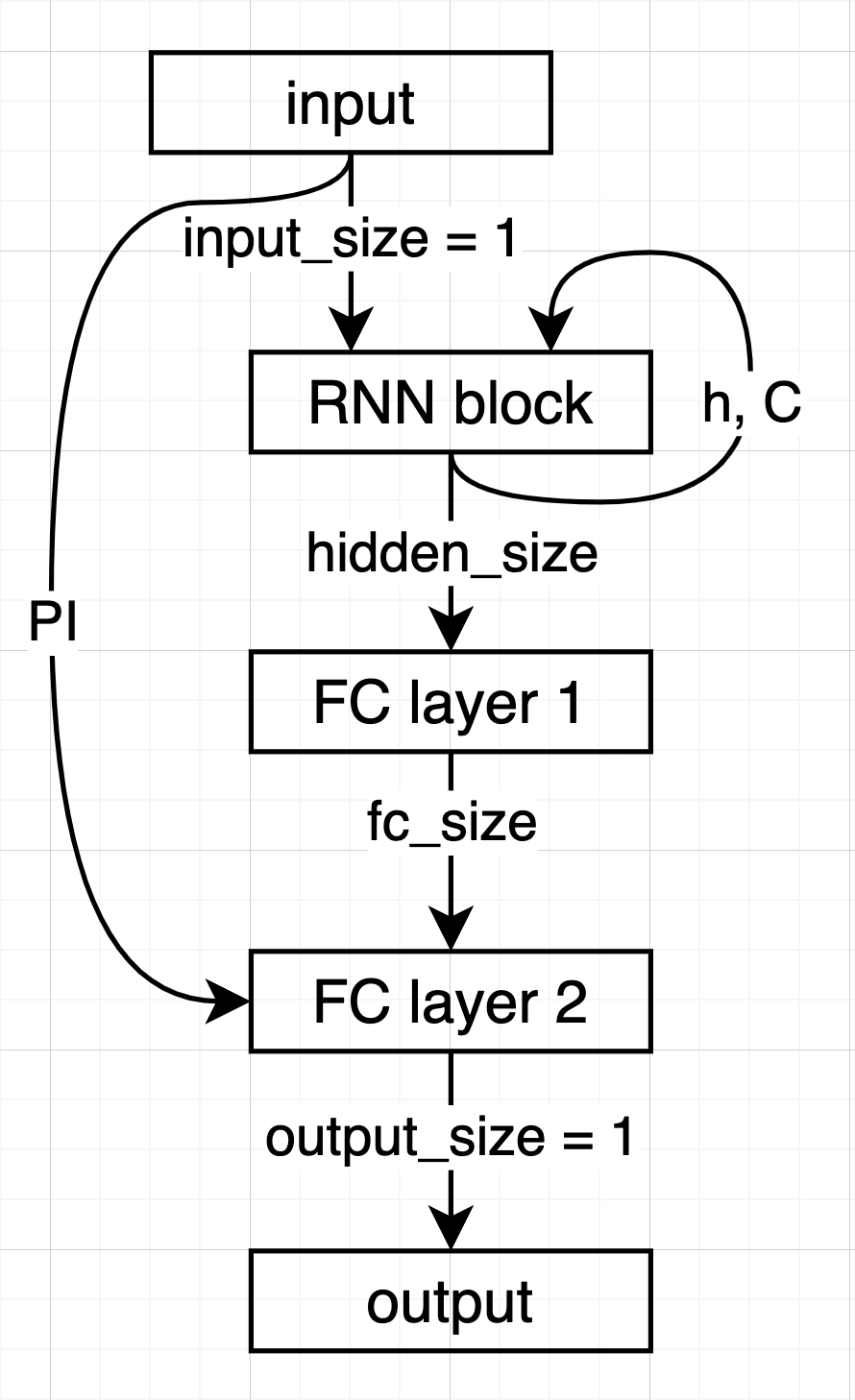


Рисунок . Общая схема RNN-сети

* + 1. RNN

Рекуррентные нейронные сети являются одним из ключевых направлений в области глубокого обучения, которое активно развивается на протяжении последних десятилетий. Данная архитектура была впервые предложена в 1980-х годах [17] и с тех пор нашла широкое применение в задачах обработки последовательных данных, таких как распознавание речи, машинный перевод и прогнозирование временных рядов.

В отличие от традиционных нейронных сетей с прямой связью, RNN также обладают внутренним состоянием, которое позволяет им учитывать предыдущие входные данные при обработке текущих. Это достигается за счет рекуррентных связей, которые передают информацию от одного временного шага к другому. Таким образом, RNN способны эффективно моделировать зависимости в последовательных данных, что делает их незаменимыми в задачах, где важен контекст, таких как наша задача.

Одним из ключевых преимуществ RNN является их способность обрабатывать входные данные произвольной длины. Кроме того, RNN отличаются высокой гибкостью и могут быть адаптированы под решение широкого спектра задач путем модификации архитектуры и обучения. Такая структура станет отличной базой для наших экспериментов.

Математически, работу RNN можно описать формулами ниже (ф. 4.1, 4.2).

|  |  |  |
| --- | --- | --- |
|  |  | ф. (4.1) |
|  |  | ф. (4.2) |

Где:

* – вектор скрытого состояния сети на шаге t
* – вектор входных данных на шаге t
* – обучаемые матрицы весов
* – обучаемые векторы сдвигов (bias)
* – функции активации (нелинейные, кусочно-дифференцируемые)

Несмотря на свою простоту, RNN демонстрируют впечатляющие результаты в широком спектре задач и продолжают оставаться одним из ключевых инструментов в арсенале исследователей и разработчиков в области искусственного интеллекта. Для начала попробуем использовать эту модель без модификаций.

* + 1. LSTM

В 1997 году классическую архитектуру рекуррентных нейронных сетей было предложено улучшить с помощью модели с долгой краткосрочной памятью (Long Short-Term Memory, LSTM) [18]. В данный момент это одно из наиболее известных и широко применяемых расширений, которое позволяет моделировать долгосрочные зависимости в последовательных данных.

Отличие LSTM от классической RNN состоит в использовании специальных ячеек памяти (cells), позволяющих эффективно запоминать и извлекать информацию из предыдущих шагов. LSTM хранит состояние в виде вектора C, которое регулируется с помощью трех управляющих вентилей: забывающего (forget gate), входного (input gate) и выходного (output gate). Вентили в виде линейных слоев реализуют сохранение, обновление и извлечение информации из ячейки памяти на каждом временном шаге (рис. 7).

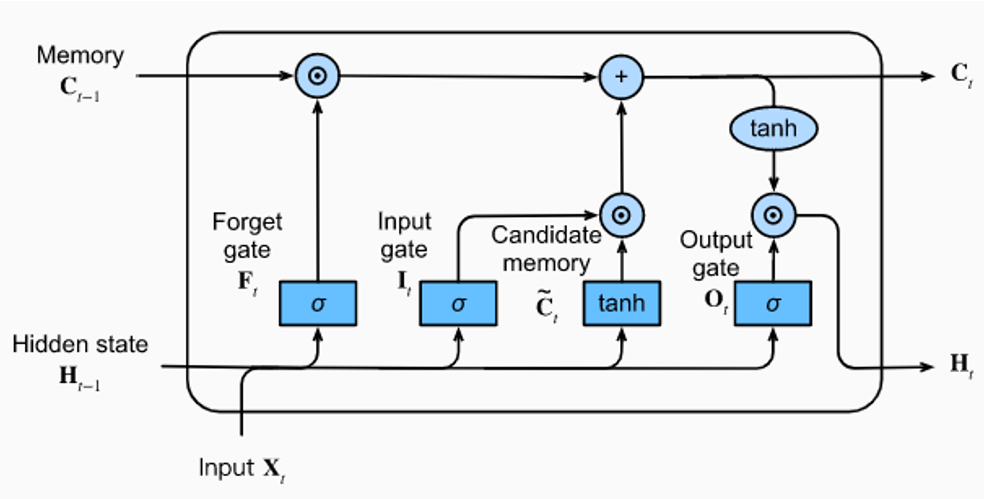


Рисунок . Схема одного блока LSTM

Обычно LSTM демонстрируют превосходство перед классическими RNN в задачах, требующих моделирования долгосрочных зависимостей. Они лучше справляются с проблемой исчезающего градиента и способны более эффективно запоминать важную информацию на протяжении длительных промежутков времени.

Формальное описание работы одного блока LSTM (ф. 4.3-7).

|  |  |  |
| --- | --- | --- |
|  |  | ф. (4.3) |
|  |  | ф. (4.4) |
|  |  | ф. (4.5) |
|  |  | ф. (4.6) |
|  |  | ф. (4.7) |

Где:

* – значения забывающего, входного и выходного вентилей соответственно в момент времени t
* – состояние ячейки (“память”) на шаге t
* – выходное значение на шаге t
* – соответствующие обучаемые матрицы весов и векторы смещений
* – функции активации: сигмоида и гиперболический тангенс

Такая структура сети обладает намного большим потенциалом для обучения, по сравнению с RNN. Проверим, как это скажется на качестве предсказаний. Возрастающее число параметров также увеличивает и шанс переобучения.

* + 1. GRU

Еще одним важным расширением классической архитектуры рекуррентных нейронных сетей является модель GRU (Gated Recurrent Unit), предложенная в 2014 году. GRU представляет собой более компактную и упрощенную версию LSTM, сохраняя при этом ключевые преимущества последней [19].

В отличие от LSTM, GRU имеет только два управляющих вентиля - обновляющий (update gate) и сбрасывающий (reset gate). Обновляющий вентиль определяет, какая часть предыдущего скрытого состояния должна быть сохранена, а сбрасывающий вентиль контролирует, какая информация из текущего входа должна быть добавлена к скрытому состоянию. Также у этой модели отсутствует отдельное скрытое состояние C. Всё состояние хранится в векторе h (рис. 8).

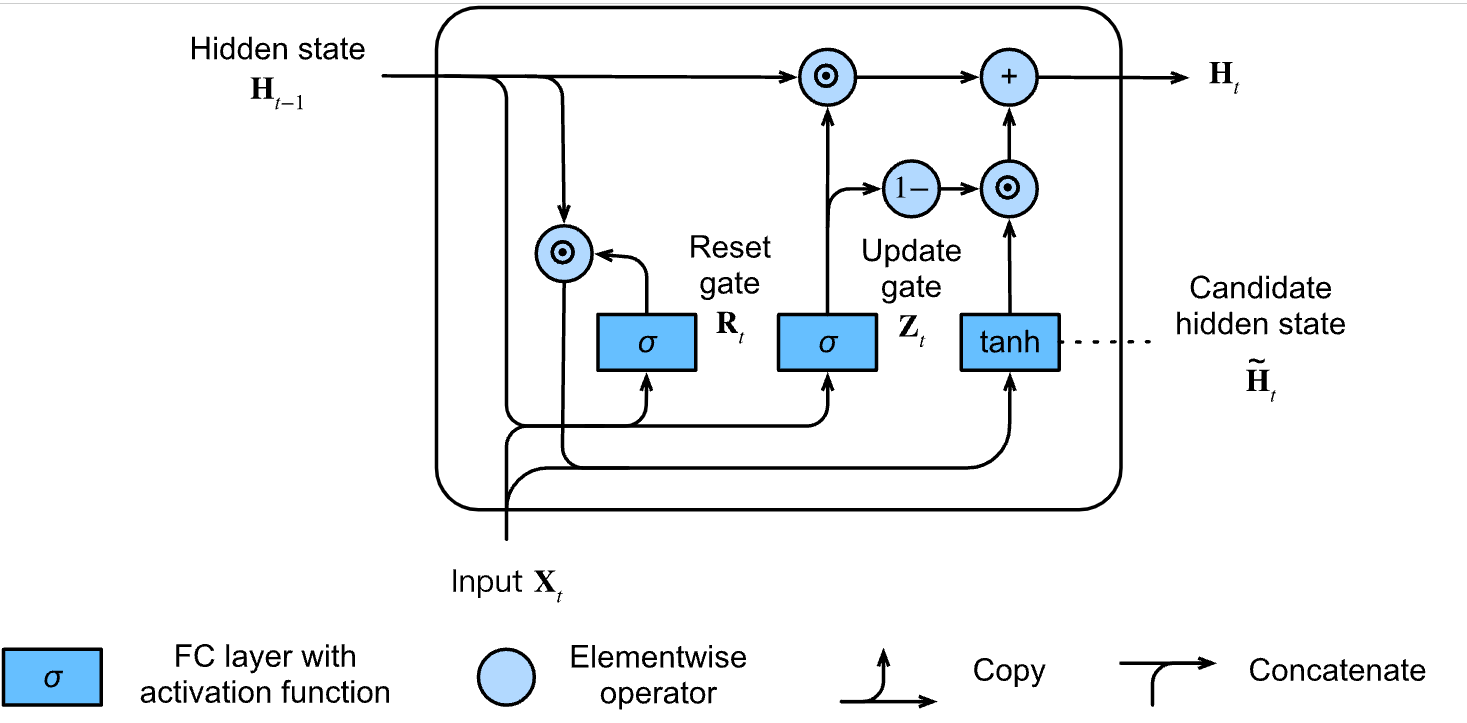


Рисунок . Схема одного блока GRU

Ключевым преимуществом GRU перед LSTM является более простая структура, что позволяет быстрее обучать модель и экономить больше ресурсов, а также понижает вероятность переобучения. При этом GRU сохраняет способность LSTM к моделированию долгосрочных зависимостей в последовательных данных, демонстрируя сопоставимые или даже превосходящие результаты.

Приведём также формальное описание работы одного блока GRU (ф. 4.8-10).

|  |  |  |
| --- | --- | --- |
|  |  | ф. (4.8) |
|  |  | ф. (4.9) |
|  |  | ф. (4.10) |

Где:

* – значения обновляющего и сбрасывающего вентилей соответственно в момент времени t
* – выходное значение на шаге t
* – соответствующие обучаемые матрицы весов и векторы смещений
* – функции активации: сигмоида и гиперболический тангенс

Также протестируем эту модель в базовом виде. Проверим, как меньшее число параметров скажется на качестве, времени работы и переобучении.

* + 1. Нормализация данных

Одна из очевидных модификаций для повышения качества – это нормализация входных данных. Во-первых, большинство нейросетевых алгоритмов начинают обучение с нулевых или случайных околонулевых значений параметров. Во-вторых, за разумное количество эпох, с учетом небольшого множителя обучения (learning rate), бывает даже теоретически невозможно достигнуть некоторых значений целевых переменных. Поэтому для обеспечения сходимости и повышения качества работы модели полезно делать нормализацию.

Наиболее распространенным методом является стандартизация, при которой данные приводятся к нулевому среднему значению и единичной дисперсии. Это позволяет сбалансировать вклад различных значений временного ряда в обучение модели и избежать перекоса в сторону больших значений. Обычно стандартизацию используют, когда распределение близко к нормальному.

Другим подходом является нормализация по минимуму и максимуму, при которой данные масштабируются в диапазон от 0 до 1. Этот метод особенно полезен, когда необходимо сохранить информацию об относительных различиях между значениями.

В эксперименте попробуем добавить оба способа нормализации к данным и посмотрим, как это скажется на значениях целевой метрики.

* + 1. Модификация функций активации

Попробуем также изменять функции активации в RNN. Из сводок выше ясно, что в наших моделях используются сигмоида и гиперболический тангенс. У функции сигмоиды есть известный недостаток – её производная практически всегда очень мала. Это может приводить к затуханию градиента во время его подсчёта для первых вводов, поскольку произойдет произведение большого числа очень маленьких чисел. Попробуем заменить оставшиеся сигмоиды на tanh, диапазон значений производных которого несколько шире (рис. 9).

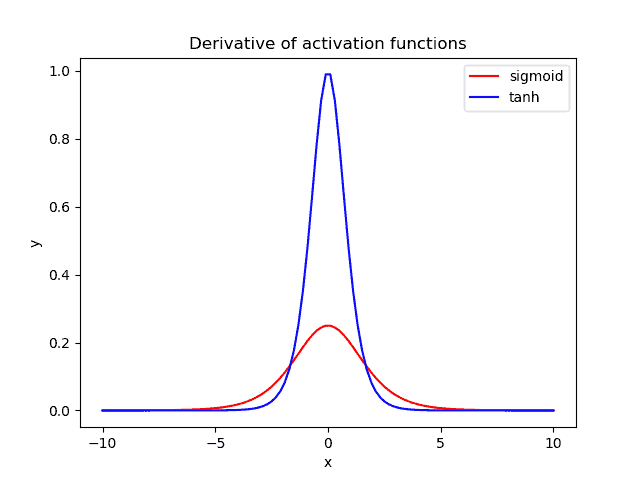


Рисунок . Сравнение производных функции сигмоиды и гиперболического тангенса

* + 1. Persistence Initialization

Последнее улучшение, которое мы хотим протестировать – это использование Persistence Initialization из PI-Transformer. Как уже описано выше, это такая инициализация весов модели, при которой до начала обучения фиксируется предсказание входного значения. По сути, это старт с модели Naïve, что является неплохим началом и в теории позволит модели лучше и легче обучаться.

* 1. Полученные результаты

Перейдём к результатам. В **приложении 1** прикреплены таблицы с объединенными результатами всех экспериментов (рис. 10-15). Для всех моделей и модификаций были измерены OWA (Overall Weighted Average), среднее время тренировки и среднее время предсказания. Дополнительно к каждой метрике приведены разбиения по периодичности (frequency) и по сфере (domain), а также представлены общие значения (total). На всех таблицах светло-зеленым фоном выделены собственные реализации (RNN). Серым фоном выделен базовый алгоритм (Naïve2). В каждой таблице все строки отсортированы по возрастанию суммарной метрики: выше – лучше.

Рассмотрим первые две таблицы (рис. 10, 11) с метрикой качества OWA. В базовом варианте все 3 модели (RNN, LSTM, GRU) показали неудовлетворительный результат, сильно хуже Naïve. Это объясняется тем, что данные никак дополнительно не подготавливались, и параметры моделей технически не могли достигнуть высоких по модулю значений. Линейная модель LTSF-Linear не показала высоких результатов и едва ли обогнала Naïve. Модель, действительно, очень проста.

Далее заметим, что с появлением нормализации все RNN-модели значительно улучшили свои результаты, как и ожидалось. При этом “чистые” RNN значительно отстают от своих модернизаций – LSTM и GRU. Это легко объяснить малым числом параметров и незамысловатостью архитектуры простой RNN. Разные виды нормализации показали себя по-разному. В нашем случае больше подошла стандартизация. С ней целевая метрика улучшается сильнее во всех случаях. Скорее всего это связано с тем, что при стандартизации среднее значение ряда обнуляется, а это намного удобнее для модели, если ряд стационарный или близкий к таковому.

Простая замена всех сигмоидных функций активации на гиперболический тангенс не принесла улучшений. Результаты измерений приблизительно похожи. Действительно, с точки зрения RNN, эти функции активации довольно схожи. Возможно, разница появилась бы при ещё большем размере входной последовательности.

Алгоритм iTransformer сработал хуже, чем мы этого ожидали. Результат едва ли лучше простого Naïve2, при намного более сложной модели. Скорее всего, этот алгоритм больше применим для многомерных временных рядов с большим количеством сопутствующих данных. В нашем одномерном случае модели не удалось качественно обнаружить зависимости в данных.

Методы ARMA, ARIMA, SARIMA и TBATS сработали лучше, чем ожидалось. Модели показали отличные результаты при всей своей простоте. Каждое улучшение базовой ARMA сказывается на целевой метрике. Отдельно отметим, что метод TBATS, в отличие от предшественников, обучается практически без входных параметров, что делает его ещё и очень удобным в использовании.

Последним и важным улучшением RNN моделей стал Persistence Initialization (PI). Присоединение остаточной связи от входного значения к последнему слою помогло моделям правильнее обучаться, показывая хорошее стартовое значение, от которого можно отталкиваться. С этим улучшением модели LSTM и GRU оказались на 3-м и 4-м местах в общем топе, немного уступая лишь N-BEATS и PI-Transformer.

Метод N-BEATS занимает второе место в нашем списке. Авторы действительно придумали полезную архитектуру, которая неплохо справляется с данными в М4.

На первом месте по качеству PI-Transformer, из которого мы взяли идею Persistence Initialization. Вместе с трансформерной архитектурой и другими улучшениями от авторов, этот метод показывает отличный результат на нашем датасете.

Обратимся теперь к временным метрикам. Рассмотрим таблицы по временам тренировки (рис. 12, 13) и предсказания (рис. 14, 15). Все таблицы достаточно похожи. Первые строчки с отрывами в несколько порядков занимают классические модели (ARMA, ARIMA, SARIMA, TBATS), а также простейшие модели с соревнования (Naïve, sNaïve, Naïve2). Затем идёт линейная LTSF-Linear, а за ней с небольшой разницей все RNN-реализации. Заметим только, что версии с PI в среднем работают немного дольше, а классические RNN тренируются незначительно быстрее. Замыкают таблицы N-BEATS и трансформеры. Наблюдаются отставания от RNN в несколько раз. Это легко объяснить сложной архитектурой и большим числом параметров в этих моделях.

Объединив всю информацию в этой главе, мы можем сделать заключение, что хотя RNN-реализации и не являются лучшими по качеству предсказаний моделями, они показывают вполне неплохие результаты, а также работают сравнительно быстро, в разы быстрее их более качественных конкурентов.

ЗАКЛЮЧЕНИЕ

Подведём итоги проделанной работы.

Мы провели широкий обзор существующих решений задачи прогнозирования временных рядов. Были выбраны и изучены 6 архитектур моделей для прогнозирования: две классических модели и четыре нейросетевых подхода. Это позволило сформировать достаточно глубокое понимание темы работы, а также устройства первых и современных решений. В дополнение, данный обзор помог привнести значимое улучшение в реализованные модели (инициализация предыдущим значением).

Был проведён поиск наилучших вариантов датасетов для задачи прогнозирования временных рядов. Выбор осуществлялся как среди изученной литературы, так и отдельно в интернете. Мы выбрали очень большой и широко используемый набор данных M4 со соревнования Makridakis Competition. Это действительно универсальный способ проверить качество работы алгоритмов для одномерных временных рядов. Помимо этого, была выбрана метрика качества OWA, как основная для M4, а также зафиксированы параметры эксперимента и метрики времени. Такой выбор позволил справедливо сравнить разные модели и оценить созданное в рамках работы решение.

Были реализованы несколько вариантов рекуррентных нейронных сетей: классическая RNN, LSRM и GRU. Получены достаточно широкие знания о работе этих моделей. В дополнение к моделям реализованы их модификации с изменением входных данных, функций активации, добавлением нового параметра (PI). Модификации позволили существенно улучшить качество работы моделей.

Реализован и проведён эксперимент, в котором были сравнены все рассмотренные в работе модели, а также собственные реализации. Было произведено справедливое сравнение в равных условиях и получены весьма интересные результаты. Исходный код выложен в git-репозиторий и доступен по ссылке (Приложение 2).

Предложенные модели с улучшениями оказались весьма неплохим решением задачи прогнозирования одномерных временных рядов, если в задаче требуется хорошее качество и одновременно быстрое время работы. Реализованные модели дают более качественный результат, чем классические (не нейросетевые), незначительно уступая сложным архитектурам. При этом по времени работы ситуация обратная: сложные трансформерные модели значительно уступают более простым архитектурам, показывая предложенные RNN в лучшем свете.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ЛИТЕРАТУРЫ

1. Makridakis Competitions [Электронный ресурс]. - URL: https://www.unic.ac.cy/iff/research/forecasting/m-competitions/ (дата обращения: 01.02.2024).
2. M4 Competitors Guide [Электронный ресурс]. - URL: https://www.unic.ac.cy/test/wp-content/uploads/sites/2/2018/09/M4-Competitors-Guide.pdf (дата обращения: 01.02.2024).
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. Attention is all you need // Advances in neural information processing systems. - 2017. - Т. 30.
4. Whittle, P. Hypothesis testing in time series analysis. - 1951.
5. Box, G.E.P., Jenkins, G. Time Series Analysis, Forecasting and Control. - San Francisco, CA : Holden-Day, 1970.
6. Dickey, D. A., Fuller, W. A. Distribution of the Estimators for Autoregressive Time Series with a Unit Root // Journal of the American Statistical Association. - 1979. - № 74. - P. 427-431.
7. Phillips, P. C. B., Perron, P. Testing for a Unit Root in Time Series Regression // Biometrika. - 1988. - Vol. 75, № 2. - P. 335-346. doi:10.1093/biomet/75.2.335.
8. De Livera, A. M., Hyndman, R. J., Snyder, R. D. Forecasting time series with complex seasonal patterns using exponential smoothing // Journal of the American Statistical Association. - 2011. - Vol. 106, № 496. - P. 1513-1527.
9. Box, G. E., Cox, D. R. An analysis of transformations // Journal of the Royal Statistical Society: Series B (Methodological). - 1964. - Vol. 26, № 2. - P. 211-243.
10. Oreshkin, B. N., Carpov, D., Chapados, N., Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting // arXiv preprint arXiv:2006.10739. - 2020.
11. He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition // Proceedings of the IEEE conference on computer vision and pattern recognition. - 2016. - P. 770-778.
12. Smyl, S., Ranganathan, A., Pasqua, A. Doubly Residual Stacking for Ensemble Forecasting // International Journal of Forecasting. - 2020. - Vol. 36, № 4. - P. 1452-1459.
13. Zeng, A., Chen, M., Zhang, L. et al. Are Transformers Effective for Time Series Forecasting? // arXiv preprint arXiv:2205.13504. - 2022. - URL: https://arxiv.org/abs/2205.13504 (дата обращения: 01.04.2024).
14. Cure-Lab. LTSF-Linear. GitHub repository. - 2022. - URL: <https://github.com/cure-lab/LTSF-Linear>
15. Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M. Inverted Transformers Are Effective for Time Series Forecasting // arXiv preprint arXiv:2310.06625. - 2023. - DOI: 10.48550/arXiv.2310.06625.
16. Haugsdal, E., Aune, E., Ruocco, M. Persistence Initialization: A novel adaptation of the Transformer architecture for Time Series Forecasting // arXiv preprint arXiv:2208.14236. - 2022. - <https://doi.org/10.48550/arXiv.2208.14236>
17. Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning representations by back-propagating errors // nature. - 1986. - Vol. 323, № 6088. - P. 533-536.
18. Hochreiter, S., Schmidhuber, J. Long short-term memory // Neural computation. - 1997. - Vol. 9, № 8. - P. 1735-1780.
19. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation // arXiv preprint arXiv:1406.1078. - 2014.

ПРИЛОЖЕНИЕ

**Приложение 1. Результаты экспериментов**

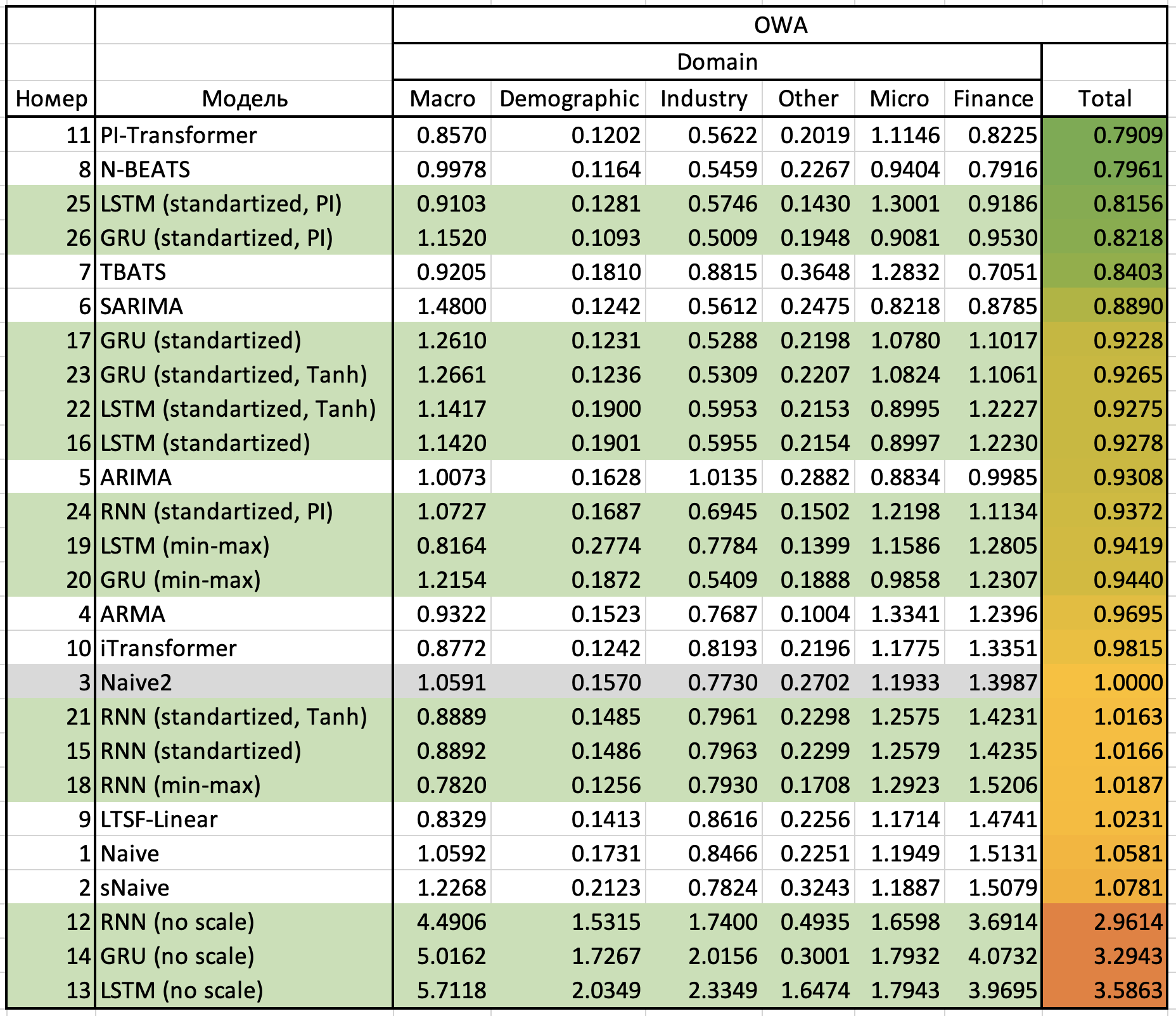


Рисунок . Сводная таблица по метрике OWA с разделением по сферам деятельности (domain)

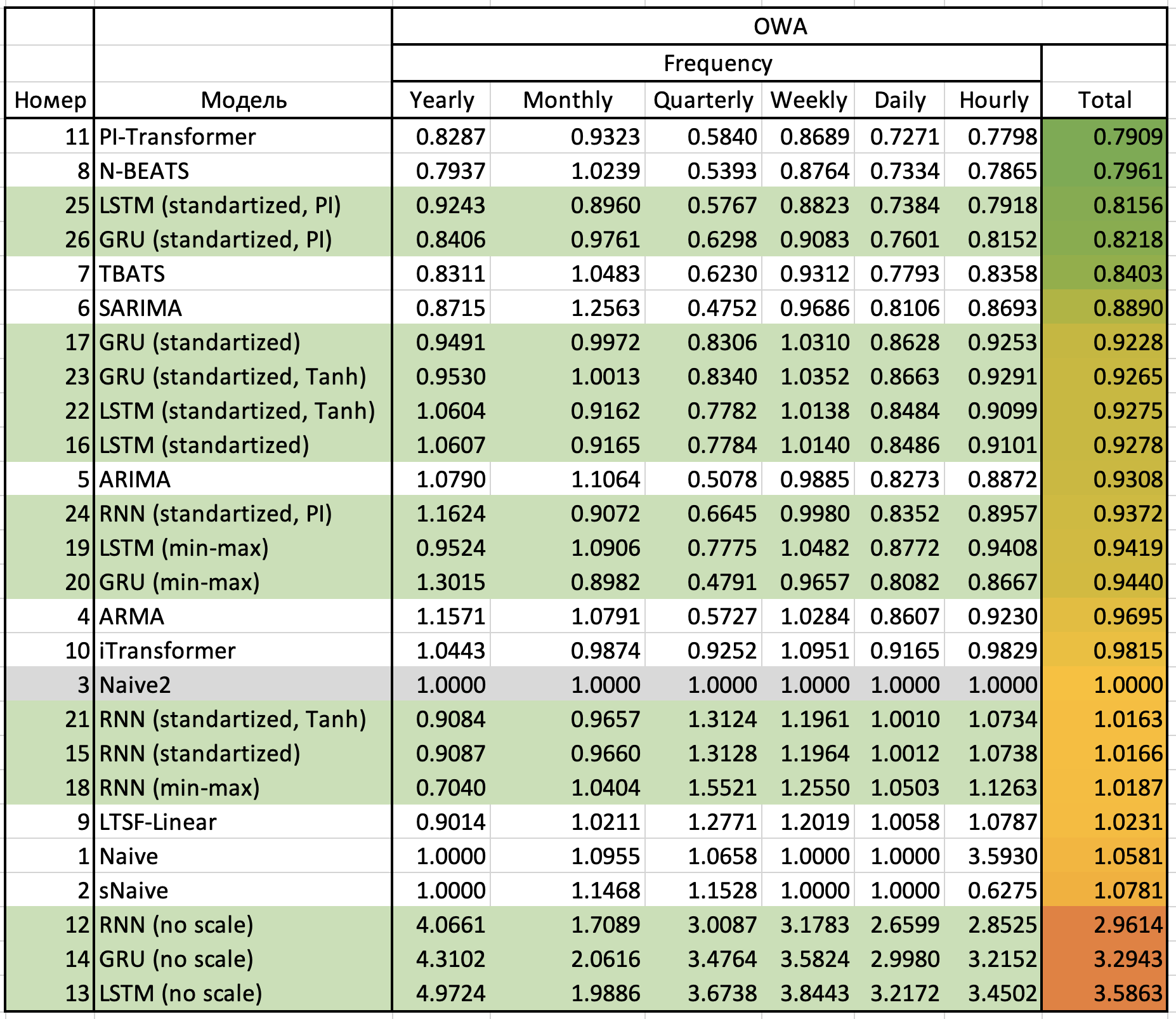


Рисунок . Сводная таблица по метрике OWA с разделением по периодичности событий (frequency)

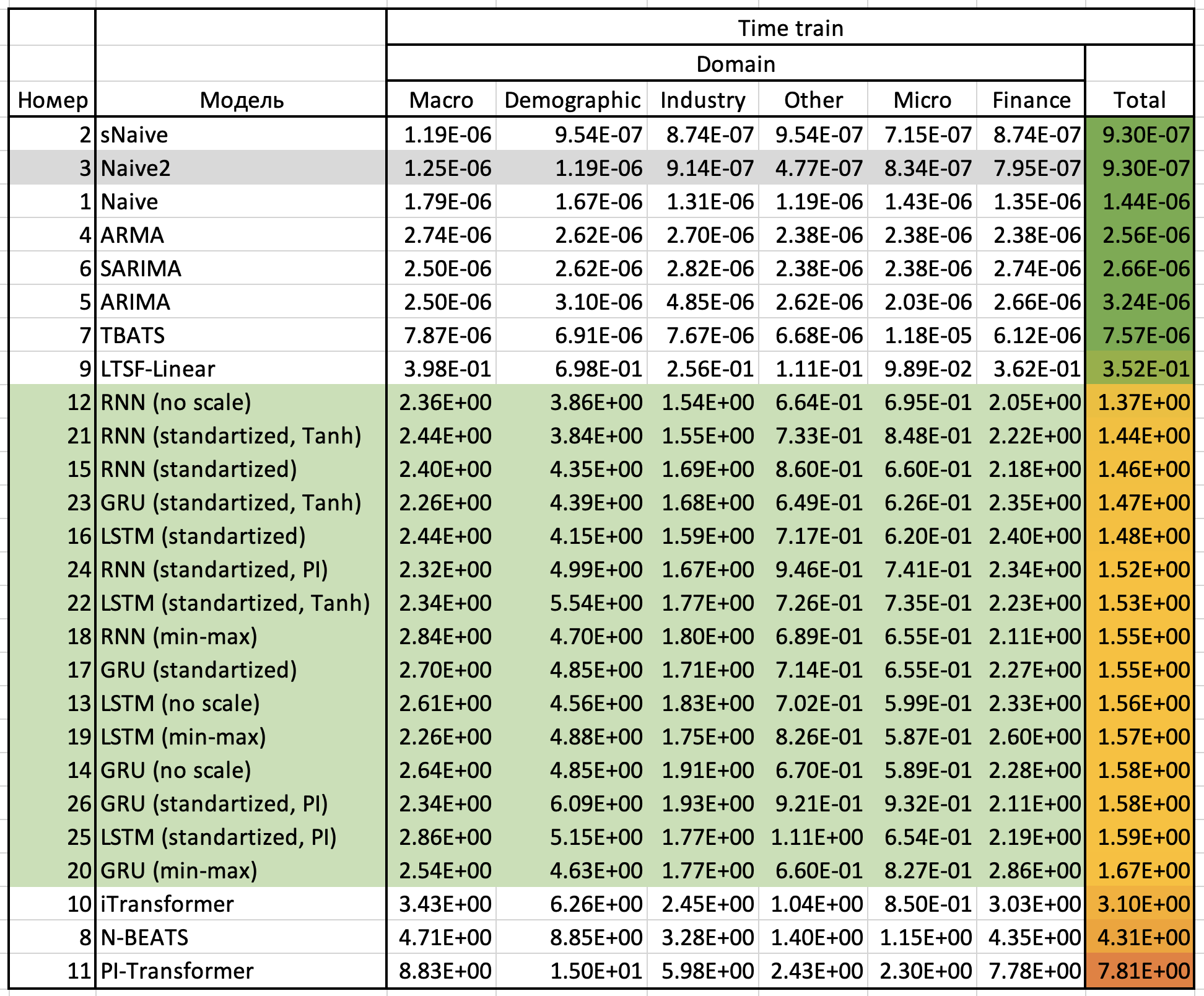


Рисунок . Сводная таблица по времени тренировки с разделением по сферам деятельности (domain)

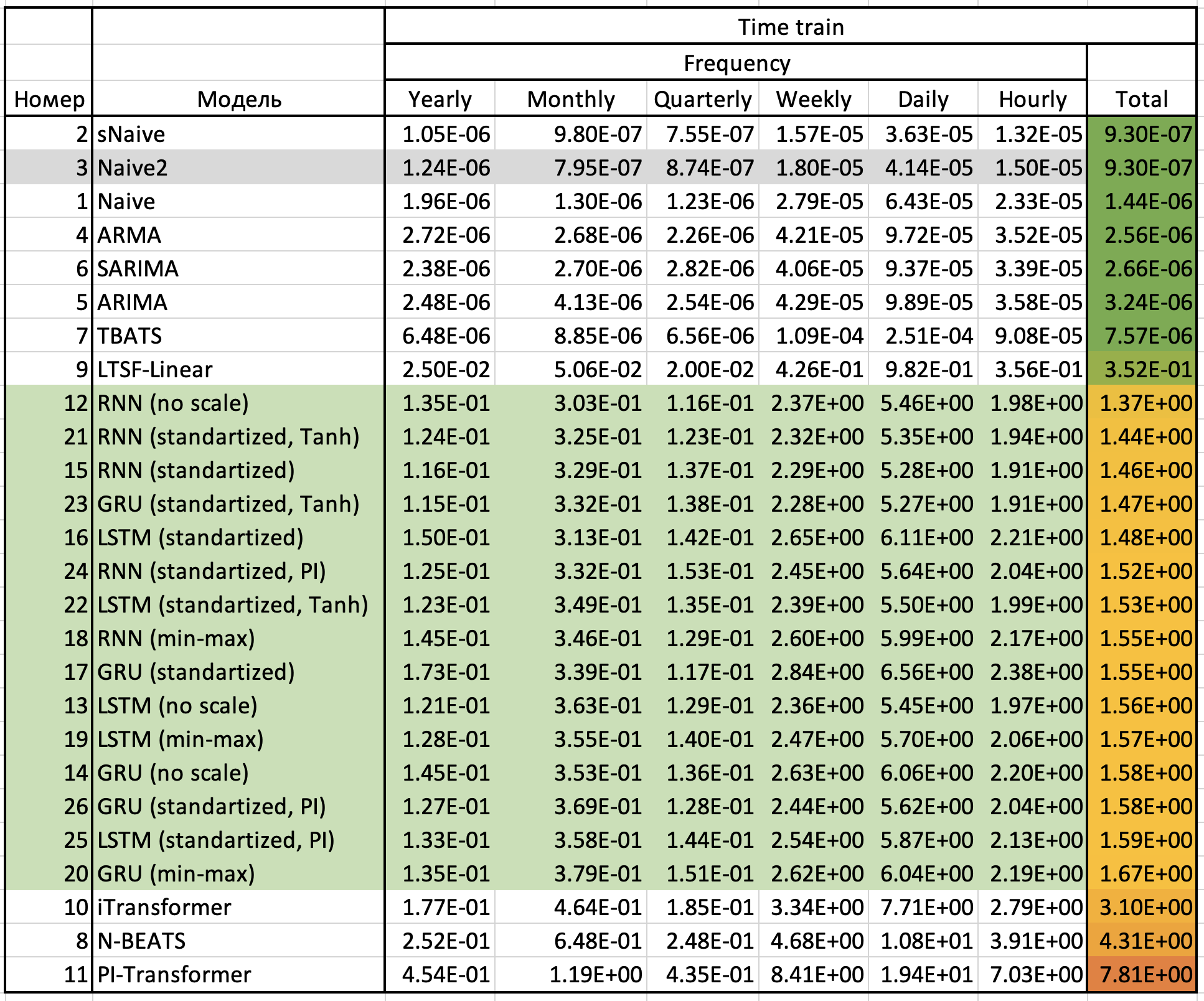


Рисунок . Сводная таблица по времени тренировки с разделением по периодичности событий (frequency)

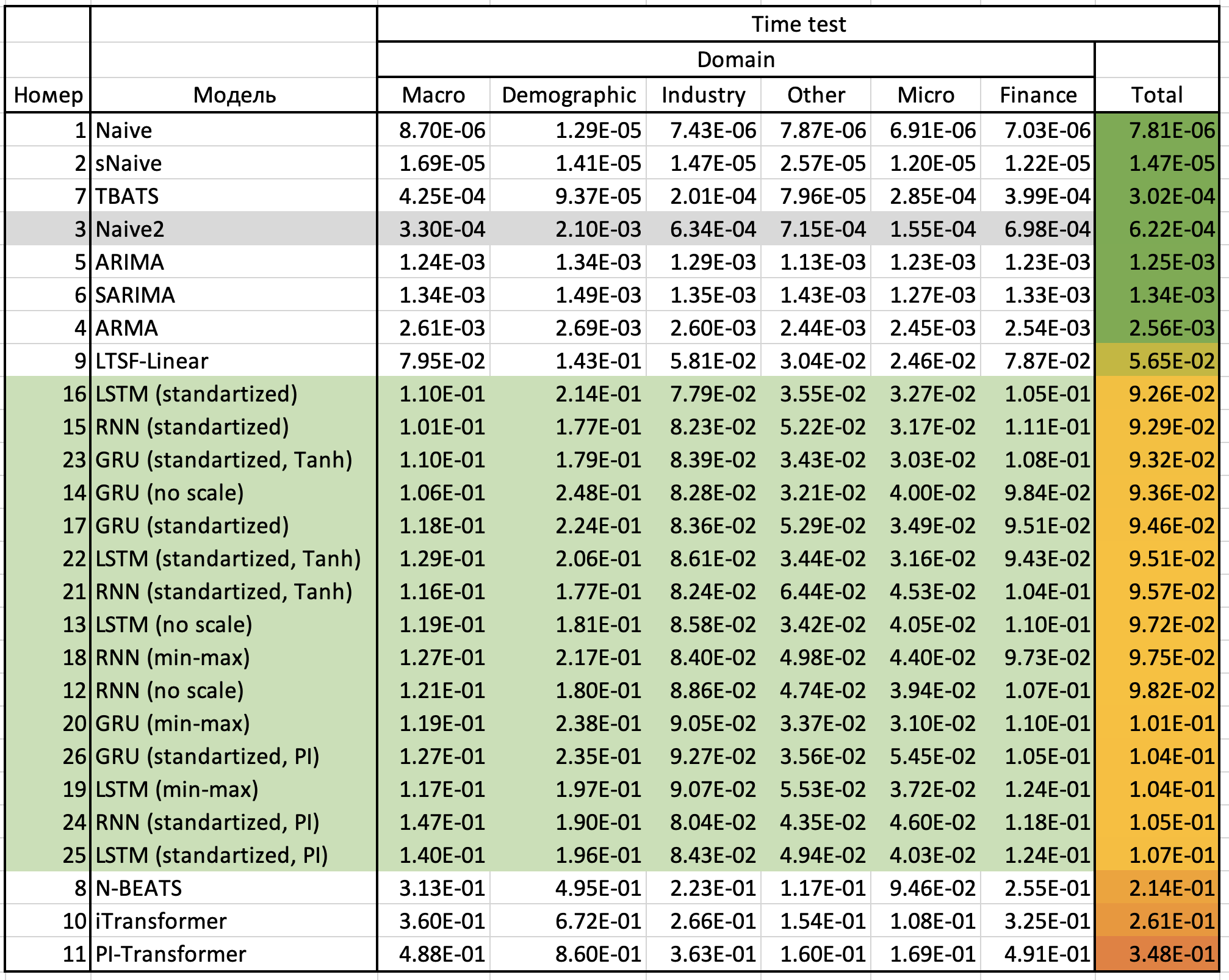


Рисунок . Сводная таблица по времени предсказания с разделением по сферам деятельности (domain)

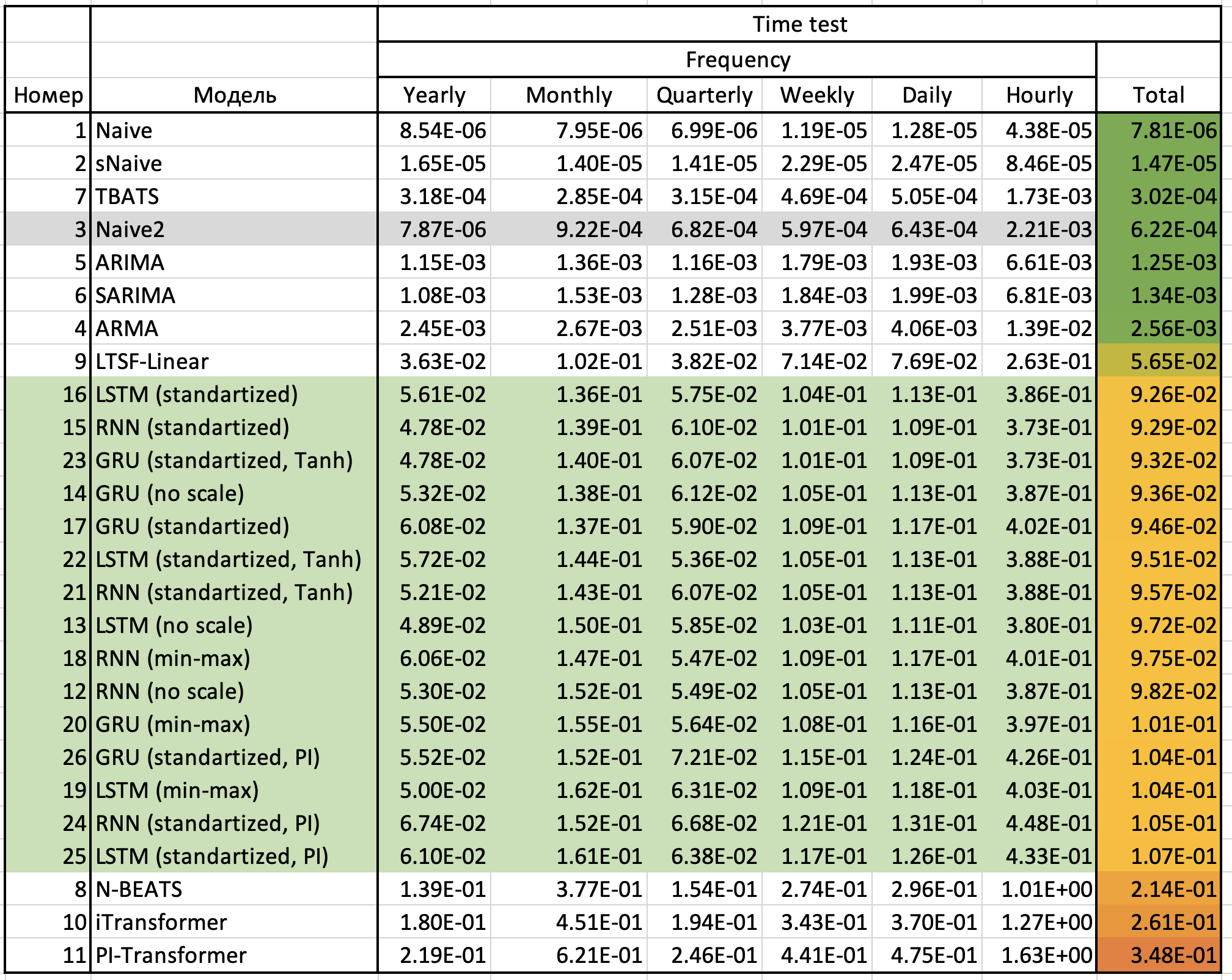


Рисунок . Сводная таблица по времени предсказания с разделением по периодичности событий (frequency)

**Приложение 2. Ссылка на исходный код**

Исходный код, реализованный и использованный в ходе работы, доступен в репозитории на сайте GitHub по ссылке:

<https://github.com/RukhovichIV/HSE-CS-Courses>

Необходимо перейти в директорию с дипломной работой по пути .courses/Diploma/. Полный путь: <https://github.com/RukhovichIV/HSE-CS-Courses/tree/master/courses/Diploma>

Модификации исходного кода внешних моделей для измерения метрик производились в копиях их собственных репозиториев. Эти модификации очень незамысловаты и могут быть легко воспроизведены любым исследователем.