**Отчёт.**

**Нейронная сеть для алгоритма торговли.**

Оглавление

[1. Исходные данные для проектирования нейросети. 3](#_Toc55824796)

[2. Постановка задачи 4](#_Toc55824797)

[3. Анализ существующих подходов к прогнозированию временных рядов. 5](#_Toc55824798)

[4. Прототипирование вариантов реализации нейросети. 9](#_Toc55824799)

[5. Программная реализация нейросети. 13](#_Toc55824800)

[6. Подготовка данных для обучения нейросети. 20](#_Toc55824801)

[7. Заключение и Выводы 22](#_Toc55824802)

# Исходные данные для проектирования нейросети.

Исходными данными для проектирования являются данные результатов торгов, фрагмент которых представлен в следующей таблице

| **Click\_ID** | **date** | **banner\_type** | **banner\_width** | | **banner\_heigth** | | **IP** | | **Device\_ID** | | **Bid** | | **OS** | | **OS\_Version** | | **Device\_mode** | | **Connection\_Type** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 35803771 | 2019-09-23 21:46:51 | simple | 768 | | 1024 | | 91.233.43.235 | | 26D00212-C469-4A3C-824D-8FAD9CA1554C | | 353.0 | | iOS | | ios 9.3 | | Apple iPad 1 | | WiFi | |
| 35803772 | 2019-09-23 21:47:17 | simple | | 320 | | 480 | | 5.18.232.72 | | e51b5d0c-e147-4c7c-a7d9-f54fdbe84918 | | 712.0 | | Android | | android 8 | | huawei honor 9 lite | | None |
| 35803773 | 2019-09-23 21:47:40 | simple | | 320 | | 480 | | 62.33.118.27 | | 5b47abce-0e7c-46c4-97cf-06e8a4746ef2 | | 353.0 | | Android | | android 5.0.2 | | lenovo a6010 | | WiFi |
| 35803774 | 2019-09-23 21:49:14 | simple | | 320 | | 480 | | 46.8.244.223 | | 19b84b3c-ddcc-4010-815b-1951f77515bf | | 353.0 | | Android | | android 4.4 | | None | | WiFi |

# Постановка задачи

Постановка задачи. Требуется разработать нейронную сеть, способную по историческим данным предсказать поведение временного ряда изменения цены торгов аукциона второй цены. Данные могут быть отфильтрованы на первом этапе по параметрам

* date
* banner\_type
* banner\_width
* banner\_heigth

Прогнозирование временного ряда требуется экстраполировать на интервал времени более суток с момента загрузки последних актуальных данных.

Задача прогнозирования должна решаться с использованием нейронной сети (обоснование на уровне интуиции).

# Анализ существующих подходов к прогнозированию временных рядов.

В классических задачах анализа данных предполагается независимость наблюдений При прогнозировании временных рядов, наоборот, мы надеемся, что значения ряда в прошлом содержат информацию о его поведении в будущем.

Компонентами временных рядов являются

* Тренд — плавное долгосрочное изменение уровня ряда.
* Сезонность — циклические изменения уровня ряда с постоянным
* периодом.
* Цикл — изменения уровня ряда с переменным периодом (экономические
* циклы, периоды солнечной активности).
* Ошибка — непрогнозируемая случайная компонента ряда.

Задача прогнозирования обычно включает в себя пять основных шагов.

Шаг 1: Определение проблемы.

Часто это самая сложная часть прогнозирования. Тщательное определение проблемы требует понимания того, как будут использоваться прогнозы, кто требует прогнозов и как функция прогнозирования наиболее адекватна задаче. Прогнозист должен тратить время на разговоры со всеми, кто будет участвовать в сборе данных, поддержании баз данных и использовании прогнозов для будущего планирования.

Шаг 2: Сбор информации.

Всегда требуется по крайней мере два вида информации:

а) статистические данные и

b) накопленный опыт людей, которые собирают данные и используют прогнозы.

Хорошие статистические модели будут обрабатывать эволюционные изменения в системе; не выбрасывайте хорошие данные без необходимости.

Шаг 3: Предварительный (исследовательский) анализ.

Рассмотрим графики данных.

Существуют ли последовательные закономерности?

Есть ли существенная тенденция?

Важна ли сезонность?

Есть ли доказательства наличия бизнес-циклов?

Существуют ли какие-либо выбросы в данных, которые должны быть объяснены теми, кто имеет экспертные знания?

Насколько прочны взаимосвязи между переменными, доступными для анализа?

Шаг 4: Выбор и установка моделей.

Лучшая модель для использования зависит от наличия исторических данных, прочности взаимосвязи между переменной прогноза и любыми пояснительными переменными, а также от способа использования прогнозов. Часто можно сравнить две или три потенциальные модели. Каждая модель сама по себе является искусственной конструкцией, которая основана на наборе предположений (явных и неявных) и обычно включает в себя один или несколько параметров, которые должны быть оценены с использованием известных исторических данных. Были рассмотрены регрессионные модели, экспоненциальные методы сглаживания, модели Box-Jenkins ARIMA, модели динамической регрессии, иерархическое прогнозирование, а также несколько передовых методов, включая нейронные сети и векторную авторегрессию.

Шаг 5: Использование и оценка модели прогнозирования.

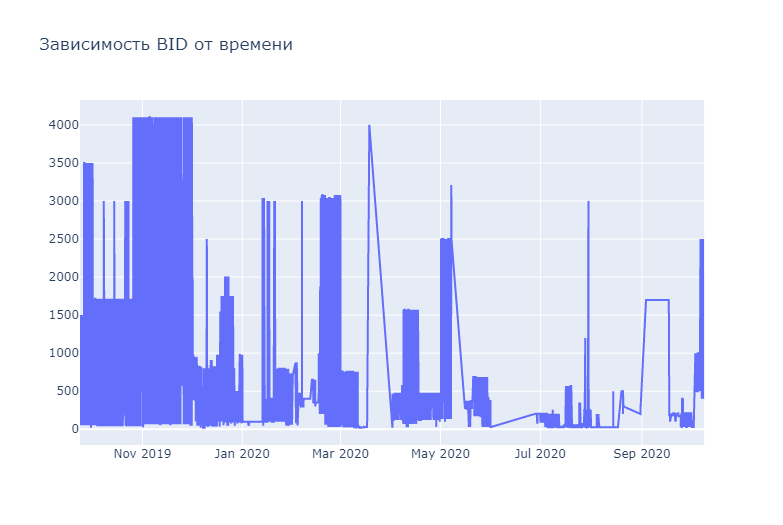
После того, как модель была выбрана и ее параметры оценены, модель используется для прогнозов. Производительность модели может быть правильно оценена только после того, как станут доступны данные за прогнозный период. Существует ряд методов, которые помогают оценить точность прогнозов. Существуют также организационные вопросы в использовании и действиях по прогнозам. При использовании модели прогнозирования на практике возникают многочисленные практические вопросы, такие как способ обработки недостающих значений и выбросов, или как справиться с короткими тайм-рядами.

Все эти подходы и методы реализованы в тетради BYYD\_analis\_cll0.ipynb,

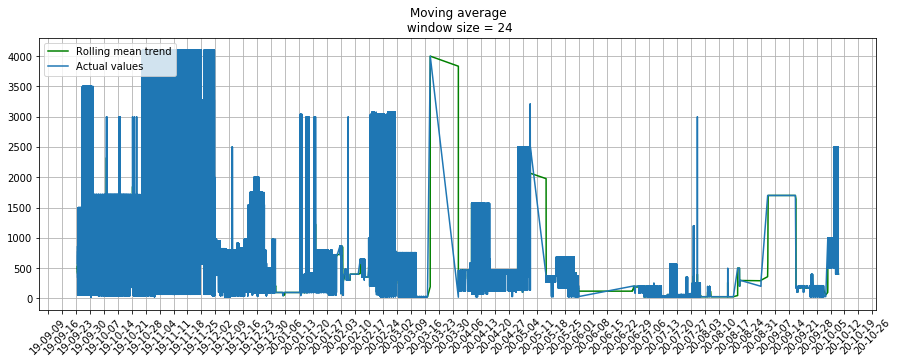
который находится на гит хабе <https://github.com/523a/BYYD_TimeSeries>

вместе с требуемыми для работы данными cll.csv.

Построим график прогнозируемой функции от времени



Сглаживание этой функции информативности не добавляет.



Экспоненциальное сглаживание тоже не дает сколь либо значимого эффекта на представленных данных

Изображение выглядит как цветной, окрашенный, карандаш, сидит

Автоматически созданное описание

Из вида графика видно, что интересующая зависимость выглядит как последовательность дискретных сигналов, поэтому к ней подходят больше методы моделирования последовательностей.

Проектирование нейросети прогнозирования цены аукционных торгов предлагается реализовать на основе рекуррентной нейросети с длинной короткой временной памятью.

Нейронная сеть, как универсальный аппроксиматор способна к решению поставленной задачи, поэтому реализация подхода не должна вызвать особых трудностей.

**Выводы:**

1. На основе анализа вида временного ряда с учетом требуемых фильтраций приходим к заключению, что классические подходы для прогнозирования данного вида временного ряда не подходят.
2. Наиболее подходящим подходом является один из методов анализа и прогнозирования символьных последовательностей, нашедший реализацию на основе нейронных сетей архитектуры LSTM, RNN и CNN. Этот вывод обосновывается тем, что предлагаемый временной ряд больше похож на символьную последовательность, нежели на временную функцию, позволяющую ее дифференцировать.
3. В качестве первого варианта реализации предлагается реализация на основе LSTM а именно архитектуры Seq2Seq.

# Прототипирование вариантов реализации нейросети.

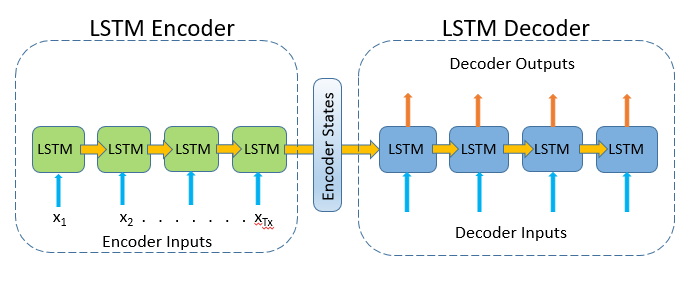
Нейронная сеть, как универсальный аппроксиматор способна к решению поставленной задачи. В рамках выбранного подхода были разработаны и проанализированы результаты работы трех 4(четырех) конкурентных моделей нейросети, построенных по методологии прогнозирования временных рядов с помощью однонаправленных и двунаправленных LSTM с глубоким стеком.

Модели от последовательности к последовательности, как следует из названия, принимают последовательность функций в качестве входных данных и выводят целевую последовательность в качестве продолжения входной целевой последовательности (она может предсказать «n» временных шагов вперед в будущее).

В основном он состоит из двух частей: кодировщика, который выводит вектор контекста (кодирование) входной последовательности, который затем передается декодеру для декодирования и прогнозирования целей.

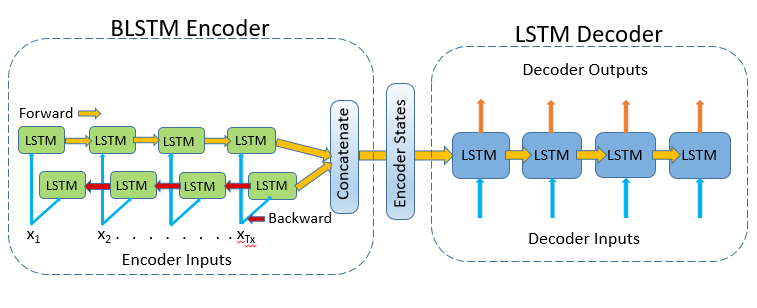
Базовая архитектура кодировщика-декодеровщика представлена на рисунке. Затем мы сможем постепенно добавлять к ней новые функции и уровни для создания более сложных архитектур.

1. С однонаправленные LSTM в качестве кодировщика



Здесь кодер LSTM принимает последовательность временных рядов в качестве входных данных (один временной шаг на ячейку LSTM) и создает кодирование входной последовательности. Эта кодировка представляет собой вектор, состоящий из скрытых состояний и состояний ячеек всех ячеек LSTM кодировщика. Затем кодирование передается декодеру LSTM в качестве начальных состояний вместе с другими входными данными декодера для создания наших прогнозов (выходных данных декодера). Во время обучения модели мы устанавливаем целевую выходную последовательность в качестве выходных данных декодера для модели для обучения.

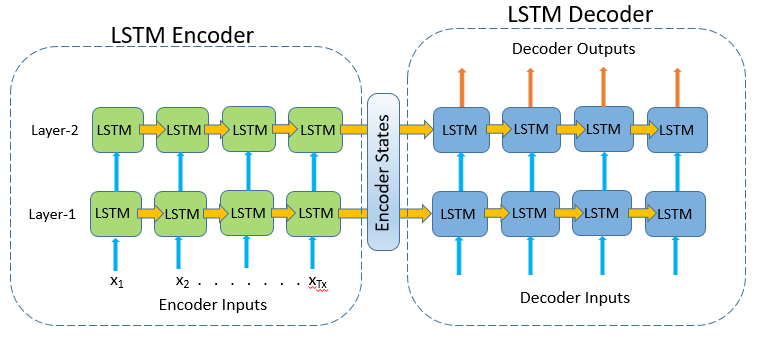
1. С двунаправленные LSTM в качестве кодировщика



Двунаправленные LSTM имеют два повторяющихся компонента: прямой повторяющийся компонент и обратный повторяющийся компонент. Прямой компонент вычисляет скрытые состояния и состояния ячеек аналогично стандартному однонаправленному LSTM, тогда как обратный компонент вычисляет их, принимая входную последовательность в обратном хронологическом порядке, то есть начиная с временного шага Tx до 1. Интуиция использования обратного компонента заключается в том, что мы создаем способ, с помощью которого сеть видит будущие данные и соответственно узнает их веса. Это может помочь сети захватить некоторые зависимости, которые в противном случае не были бы зафиксированы стандартным (прямым) LSTM. BLSTM также является стартовым алгоритмом для большинства задач НЛП из-за его способности достаточно хорошо фиксировать зависимости во входной последовательности.

В LSTM состояния скрытых и ячеек прямого компонента отличаются от состояний обратных компонентов. Таким образом, чтобы получить кодировку, скрытые состояния и состояния ячеек прямого компонента должны быть объединены с состояниями обратного компонента соответственно.

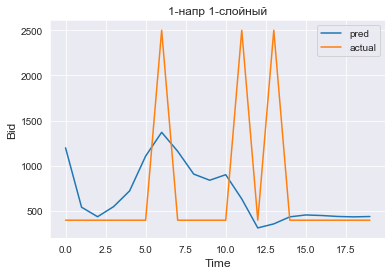
1. Со сложенными однонаправленными LSTM в качестве кодировщика.



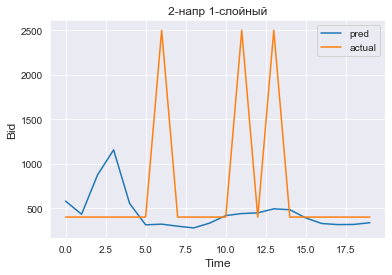
Когда слои складываются вместе, выходные данные (состояния ячеек) ячеек LSTM первого уровня как кодера, так и декодера передаются в ячейки LSTM второго уровня в качестве входных данных. Глубокие архитектуры LSTM с несколькими скрытыми слоями могут эффективно изучать сложные шаблоны и могут постепенно создавать более высокие уровни представлений данных входной последовательности.

Аналогичным образом можно складывать двунаправленные LSTM. Выходы прямого и обратного компонентов первого уровня передаются в прямые и обратные компоненты второго уровня соответственно.

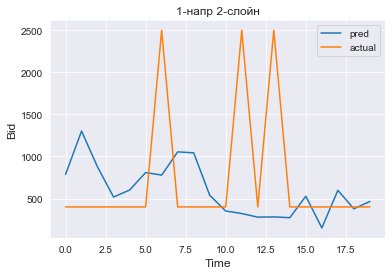
Реализованы все 4 вида моделей на представленных данных. Тест программы приведен в приложении. Результаты работы моделей представлены на рисунках.



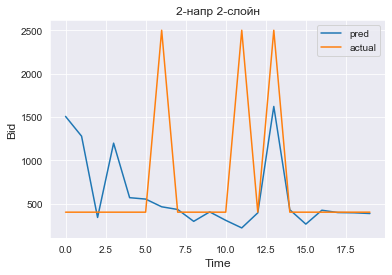
Однонаправленный однослойный



Двунаправленный однослойный



Однонаправленный двухслойный



Двунаправленный двухслойный

Все модели были обучены только для 100 эпох (с одинаковыми параметрами), и двунаправленные LSTM выделялись в изучении сложных паттернов в данных достаточно хорошо по сравнению с однонаправленными LSTM. Таким образом, описанные модели могут применяться ко многим другим сценариям прогнозирования временных рядов даже для многомерных входных случаев, когда вы можете передавать данные с несколькими функциями в виде трехмерного тензора.

**Выводы:**

Лучшим и наиболее точным вариантом является вариант использования двунаправленной двухуровневой сети. Поэтому используем этот вариант сети для дальнейшей реализации.

# Программная реализация нейросети.

Для программной реализации предлагаемой нейросети выбран язык Python. Текст программы с необходимыми для понимания логики ее работы комментариями приведен на гитхабе, ссылка на который приведена выше, в файле s2s.py. Текст программы снабжен всеми необходимыми комментариями и пояснениями для понимания логики его работы.

Конфигурация сети представляет собой двунаправленную рекурсивную сеть с двумя слоями:

Model: "functional\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

=======================================================================================

input\_1 (InputLayer) [(None, None, 1)] 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional (Bidirectional) [(None, 200), [(None 242400 input\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

input\_2 (InputLayer) [(None, None, 1)] 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

concatenate (Concatenate) (None, 200) 0 bidirectional[0][1]

bidirectional[0][5]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

concatenate\_1 (Concatenate) (None, 200) 0 bidirectional[0][2]

bidirectional[0][6]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

concatenate\_2 (Concatenate) (None, 200) 0 bidirectional[0][3]

bidirectional[0][7]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

concatenate\_3 (Concatenate) (None, 200) 0 bidirectional[0][4]

bidirectional[0][8]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

rnn\_1 (RNN) [(None, None, 200), 482400 input\_2[0][0]

concatenate[0][0]

concatenate\_1[0][0]

concatenate\_2[0][0]

concatenate\_3[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, None, 1) 201 rnn\_1[0][0]

=======================================================================================

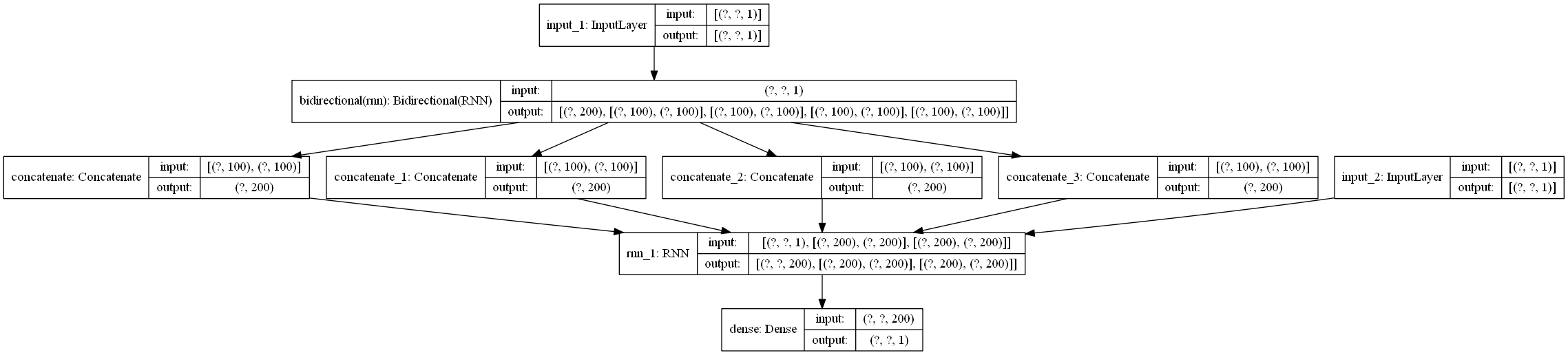
Total params: 725,001

Trainable params: 725,001

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Схема сети представлена на следующем рисунке



Нейронная сеть работает в двух режимах – режиме обучения и режиме предсказания. Для использования режима предсказания сеть необходимо сначала запустить в режиме обучения. В противном случае в режиме предсказания работа сети ничем не будет отличатся от случайного предсказания. Режим обучения делится на эпохи. От количества эпох обучения существенно зависит точность выдаваемого во втором режиме пронгноза. Количество эпох обучения выбирается произвольно, но в общем случае зависит от параметра ошибки прогноза. Чем больше количество эпох, тем меньше ошибка. График зависимости ошибки от эпох обучения приведен на рисунке.



Режим обучения сопровождается информацией о динамике прохождения каждой эпохи и результатах обучения по окончании каждой из них. Пример приведен ниже

Epoch 1/100

2/227 [..............................] - ETA: 24s - loss: 0.1574WARNING:tensorflow:Callbacks method `on\_train\_batch\_end` is slow compared to the batch time (batch time: 0.0410s vs `on\_train\_batch\_end` time: 0.1730s). Check your callbacks.

227/227 [==============================] - 8s 36ms/step - loss: 0.1440 - val\_loss: 0.1351

Epoch 2/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1405 - val\_loss: 0.1339

Epoch 3/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1402 - val\_loss: 0.1342

Epoch 4/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1394 - val\_loss: 0.1331

Epoch 5/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1382 - val\_loss: 0.1334

Epoch 6/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1378 - val\_loss: 0.1326

Epoch 7/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1374 - val\_loss: 0.1324

Epoch 8/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1353 - val\_loss: 0.1315

Epoch 9/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1354 - val\_loss: 0.1304

Epoch 10/100

227/227 [==============================] - 8s 36ms/step - loss: 0.1362 - val\_loss: 0.1312

Epoch 11/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1336 - val\_loss: 0.1300

Epoch 12/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1326 - val\_loss: 0.1298

Epoch 13/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1301 - val\_loss: 0.1278

Epoch 14/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1281 - val\_loss: 0.1314

Epoch 15/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1293 - val\_loss: 0.1276

Epoch 16/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1274 - val\_loss: 0.1279

Epoch 17/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1268 - val\_loss: 0.1258

Epoch 18/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1236 - val\_loss: 0.1280

Epoch 19/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1244 - val\_loss: 0.1261

Epoch 20/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1270 - val\_loss: 0.1297

Epoch 21/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1229 - val\_loss: 0.1234

Epoch 22/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1201 - val\_loss: 0.1252

Epoch 23/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1230 - val\_loss: 0.1241

Epoch 24/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1151 - val\_loss: 0.1192

Epoch 25/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1215 - val\_loss: 0.1258

Epoch 26/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1127 - val\_loss: 0.1202

Epoch 27/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1107 - val\_loss: 0.1172

Epoch 28/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1140 - val\_loss: 0.1189

Epoch 29/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1091 - val\_loss: 0.1158

Epoch 30/100

227/227 [==============================] - 8s 34ms/step - loss: 0.1073 - val\_loss: 0.1123

Epoch 31/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1043 - val\_loss: 0.1129

Epoch 32/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1096 - val\_loss: 0.1173

Epoch 33/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1069 - val\_loss: 0.1119

Epoch 34/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1052 - val\_loss: 0.1146

Epoch 35/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1033 - val\_loss: 0.1110

Epoch 36/100

227/227 [==============================] - 8s 35ms/step - loss: 0.1000 - val\_loss: 0.1090

Epoch 37/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0991 - val\_loss: 0.1127

Epoch 38/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0988 - val\_loss: 0.1081

Epoch 39/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0969 - val\_loss: 0.1057

Epoch 40/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0978 - val\_loss: 0.1112

Epoch 41/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0938 - val\_loss: 0.1056

Epoch 42/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0927 - val\_loss: 0.1069

Epoch 43/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0951 - val\_loss: 0.1136

Epoch 44/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0937 - val\_loss: 0.1054

Epoch 45/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0907 - val\_loss: 0.1033

Epoch 46/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0909 - val\_loss: 0.1142

Epoch 47/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0884 - val\_loss: 0.1053

Epoch 48/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0870 - val\_loss: 0.0985

Epoch 49/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0855 - val\_loss: 0.1012

Epoch 50/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0862 - val\_loss: 0.1027

Epoch 51/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0847 - val\_loss: 0.1029

Epoch 52/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0831 - val\_loss: 0.1017

Epoch 53/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0804 - val\_loss: 0.1005

Epoch 54/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0801 - val\_loss: 0.0936

Epoch 55/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0798 - val\_loss: 0.0977

Epoch 56/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0773 - val\_loss: 0.0931

Epoch 57/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0768 - val\_loss: 0.0923

Epoch 58/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0766 - val\_loss: 0.0962

Epoch 59/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0812 - val\_loss: 0.0964

Epoch 60/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0752 - val\_loss: 0.0955

Epoch 61/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0750 - val\_loss: 0.0888

Epoch 62/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0682 - val\_loss: 0.0880

Epoch 63/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0754 - val\_loss: 0.0981

Epoch 64/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0759 - val\_loss: 0.0887

Epoch 65/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0710 - val\_loss: 0.0855

Epoch 66/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0658 - val\_loss: 0.0960

Epoch 67/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0695 - val\_loss: 0.0897

Epoch 68/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0641 - val\_loss: 0.0946

Epoch 69/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0636 - val\_loss: 0.0809

Epoch 70/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0701 - val\_loss: 0.0929

Epoch 71/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0703 - val\_loss: 0.0867

Epoch 72/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0645 - val\_loss: 0.0828

Epoch 73/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0669 - val\_loss: 0.0900

Epoch 74/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0683 - val\_loss: 0.0919

Epoch 75/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0674 - val\_loss: 0.0898

Epoch 76/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0681 - val\_loss: 0.0848

Epoch 77/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0637 - val\_loss: 0.0938

Epoch 78/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0681 - val\_loss: 0.0894

Epoch 79/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0644 - val\_loss: 0.0882

Epoch 80/100

227/227 [==============================] - 8s 37ms/step - loss: 0.0594 - val\_loss: 0.0802

Epoch 81/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0601 - val\_loss: 0.0965

Epoch 82/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0603 - val\_loss: 0.0868

Epoch 83/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0599 - val\_loss: 0.0917

Epoch 84/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0608 - val\_loss: 0.0845

Epoch 85/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0591 - val\_loss: 0.0804

Epoch 86/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0608 - val\_loss: 0.0832

Epoch 87/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0597 - val\_loss: 0.0811

Epoch 88/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0538 - val\_loss: 0.0753

Epoch 89/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0536 - val\_loss: 0.0765

Epoch 90/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0542 - val\_loss: 0.0808

Epoch 91/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0541 - val\_loss: 0.0775

Epoch 92/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0590 - val\_loss: 0.0854

Epoch 93/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0546 - val\_loss: 0.0832

Epoch 94/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0496 - val\_loss: 0.0826

Epoch 95/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0519 - val\_loss: 0.0788

Epoch 96/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0574 - val\_loss: 0.0720

Epoch 97/100

227/227 [==============================] - 8s 36ms/step - loss: 0.0537 - val\_loss: 0.0820

Epoch 98/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0518 - val\_loss: 0.0868

Epoch 99/100

227/227 [==============================] - 8s 35ms/step - loss: 0.0527 - val\_loss: 0.0762

Epoch 100/100

227/227 [==============================] - 8s 34ms/step - loss: 0.0482 - val\_loss: 0.0775

Кроме номера отрабатываемой эпохи можно отслеживать время работы каждой эпохи и ошибки на обучаемом и тестовом наборе данных. О делении данных на тестовый и обучаемый будет описано в следующем разделе.

Второй режим, как правило запускает сразу за первым и выдает решения в графическом и числовом виде.

Прогноз на день

[[65.19128 ]

[65.78185 ]

[66.44975 ]

[67.097046]

[67.67772 ]

[68.17407 ]

[68.58392 ]

[68.91324 ]

[69.171616]

[69.36977 ]

[69.51813 ]

[69.6261 ]

[69.701805]

[69.75201 ]

[69.78226 ]

[69.79697 ]

[69.79963 ]

[69.793 ]

[69.77918 ]

[69.759834]

[69.73619 ]

[69.70925 ]

[69.679756]

[69.648285]

[69.61531 ]

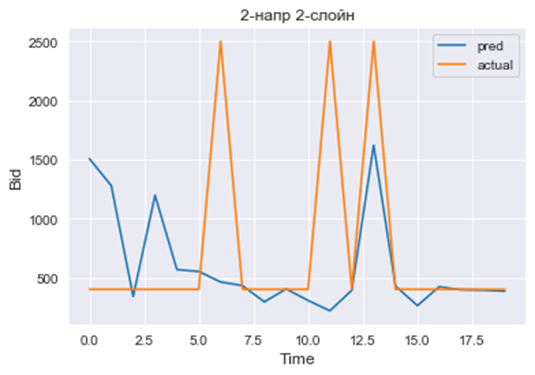
[69.58119 ]

[69.54622 ]

[69.510635]

[69.47464 ]

[69.4384 ]]



Графическое представление информации является более удобным и позволяет человеку более быстро и качественно интерпретировать полученные результаты.

**Выводы.**

Для быстрой и правильной интерпритации данных следует провести дополнительные исследования и реализовать API для возможности использования результатов в автоматическом режиме.

# Подготовка данных для обучения нейросети.

Подготовка данных для обучения нейросети состоит из нескольких этапов. На первом этапе выполняется фильтрация данных с использованием библиотеки Pandas или запроса SQLith. Данные фильтруются

по времени,

типу баннера,

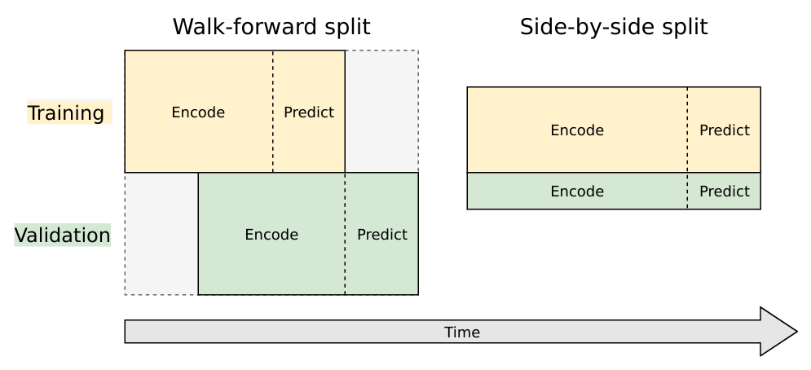
вертикальному и

горизонтальному размеру баннера.

Временной ряд имеет временную структуру, и случайно перемешивать в фолдах значения всего ряда без сохранения этой структуры нельзя, иначе в процессе потеряются все взаимосвязи наблюдений друг с другом.

Есть два способа разделить тайм серии на наборы данных для обучения и проверки:

1. Разделение выборки вперед. На самом деле это не разделение: мы тренируемся на полном наборе данных и проверяем его на полном наборе данных, используя разные временные рамки. Временной интервал для проверки сдвигается вперед на один интервал прогнозирования относительно временного интервала для обучения.



1. Поперечный разрез . Это традиционная сплит-модель для массового машинного обучения. Набор данных разбивается на независимые части, одна часть используется исключительно для обучения, а другая - только для проверки.

Второе предпочтительнее, потому что оно напрямую связано с целью решаемой задачи: прогнозировать будущие значения с использованием исторических значений. Но это разделение потребляет точки данных в конце временного ряда, что затрудняет обучение модели для точного прогнозирования будущего.

У нас есть исторические данные за 300 дней, и мы хотим спрогнозировать следующие 100 дней. Если мы выберем разбиение вперед, нам придется использовать первые 100 дней для реального обучения, следующие 100 дней для прогнозирования в режиме обучения (запустить декодер и вычислить потери), следующие 100 дней для проверки и следующие 100 дней для фактического прогнозирования более ценны. Таким образом, мы фактически можем использовать только 1/3 доступных точек данных для обучения, и между последней точкой данных обучения и первой точкой данных прогноза будет 200-дневный промежуток. Это слишком много, потому что качество предсказания экспоненциально падает по мере удаления от обучающих данных (неопределенность растет). Модель, обученная с перерывом в 100 дней (вместо 200), будет иметь значительно лучшее качество.

Параллельное разделение более экономично, так как в конце не расходуются точки данных. для наших данных производительность модели на наборе данных проверки сильно коррелирует с производительностью на наборе обучающих данных и почти не коррелирует с реальной производительностью модели в будущем.

**Выводы.**

Для актуализации результатов прогнозирования требуется периодично пополнять обучающий набор данных и пробудить до обучение сети.

# Заключение и Выводы

В настоящем документе описана программа прогнозирования временного ряда изменения цены аукционных торгов интернет-рекламы.

Анализ полученных результатов позволяет сделать вывод о работоспособности реализованной программы и достаточной практической точности полученных результатов. Результаты работы программы могут быть использованы как в ручном режиме для анализа и настройки параметров торгового алгоритма, так и для дальнейшего использования в автоматическом режиме для автоматической подстройки параметров торгов в режиме полностью автоматического функционирования алгоритма торговли. Для этого требуется верификация и более глубокое тестирование правильности получаемых прогнозов.

Для дальнейшего увеличения точности прогнозирования временного ряда возможна реализация алгоритма на основе сверхточных нейронных сетей с различными модификациями, позволяющая также увеличить интервал прогнозирования без снижения точности.