**министерство образования и науки российской федерации**

**федеральное государственное бюджетное   
образовательное учреждение высшего профессионального образования  
«Национальный исследовательский университет «МЭИ»**

Кафедра  
релейной защиты и автоматизации энергосистем

**курсовОЙ ПРОЕКТ**

по дисциплине «Применение методов ИИ в электроэнергетике»

Тема: **Прогнозирование цен на оптовом рынке электроэнергии**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент группы Э-13м-19 |  |  |  | Шмыков А.Д. |
|  | *дата сдачи* |  | *подпись* |  |
| Научный руководитель |  |  |  | *К.т.н.* Волошин А.А. |
|  |  |  | *подпись* |  |
| Консультант |  |  |  | *ассистент* Нухулов С.М. |
|  |  |  | *подпись* |  |

|  |  |
| --- | --- |
| *Дата защиты* |  |
| *Оценка* |  |
| *Подписи* |  |
| *членов комиссии* |  |

Москва 2020**Оглавление**

1. Формализация задачи машинного обучения 3

2. Сбор и формирование данных. 9

3. Анализ и предобработка данных......................................................................11

4. Обоснование выбора алгоритма машинного обучения и метрик качества 15

5. Обучение алгоритма. 15

6. Оценка качества алгоритма..............................................................................20

Заключение………………….................................................................................25

Список литературы................................................................................................26

**1. Формализация задачи машинного обучения.**

Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Чтобы запустить процесс машинного обучение, для начала необходимо загрузить в компьютер датасет (некоторое количество исходных данных), на которых алгоритм будет учиться обрабатывать запросы.

Обучение с учителем используется всякий раз, когда мы хотим предсказать целевую переменную (ответ) по данному объекту, и у нас есть пары: объект-ответ [1]. Машинное обучение с учителем часто требует вмешательства человека, чтобы получить обучающий набор данных, но потом оно автоматизирует и часто ускоряет решение трудоемких или неосуществимых задач.

Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число (2, 35, 76.454 и др.), к примеру цена квартиры, стоимость ценной бумаги по прошествии полугода, ожидаемый доход магазина на следующий месяц, качество вина при слепом тестировании.

Рекуррентные нейронные сети (RNN) - вид нейронных сетей, где связи между элементами образуют направленную последовательность, благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. RNN – это разновидность искусственной нейронной сети, в которой выходные данные одного временного интервала предоставляются в качестве входных данных для следующего временного интервала. Данное обстоятельство позволяет RNN принимать решения об объекте прогноза, основываясь как на входных данных для текущего временного интервала, так и на выходных данных предыдущих шагов [2].

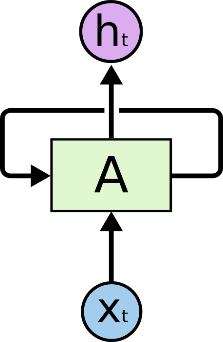


Рисунок 1. Нейрон

Фрагмент нейронной сети А принимает входное значение Xt и возвращает значение ht. Наличие обратной связи позволяет передавать информацию от одного шага сети к другому.

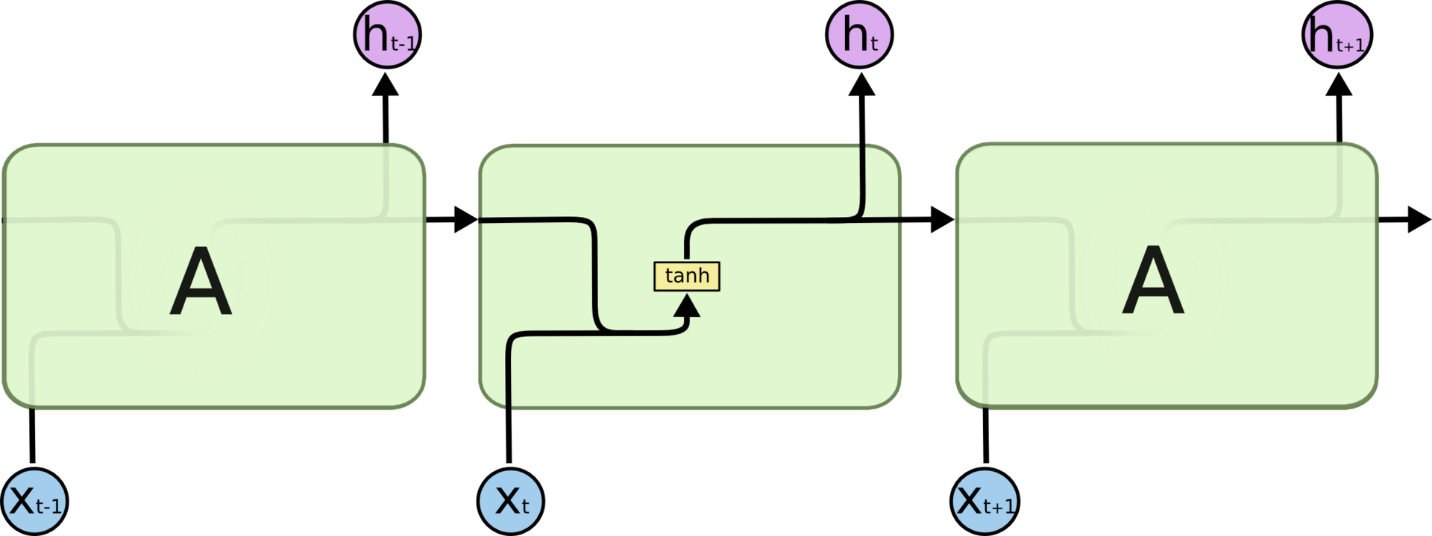


Рисунок 2. Рекуррентная нейронная сеть

Значение Xt-1 попадает на первый слой, и он делает предсказание ht-1. Первый слой сохраняет вектор активации нейронов и передает его «себе в будущем», то есть второму слою. Первое предсказание ht-1 становится вторым входным Xt. Так обеспечивается память нейросети о контексте.

Слой нейронов — это три операции:

1. умножение входного вектора X на матрицу весов
2. прибавление сдвига
3. функция активации нейронов «гиперболический тангенс» tanh

Нейроны в RNN могут активироваться сильнее или слабее, генерируя сигнал от −1 до 1

В последнее время наибольшее распространение получили сеть с долговременной и кратковременной памятью (LSTM). Они разработаны специально, чтобы избежать проблемы долговременной зависимости. В отличии от RNN они содержат четыре слоя нейронной сети.

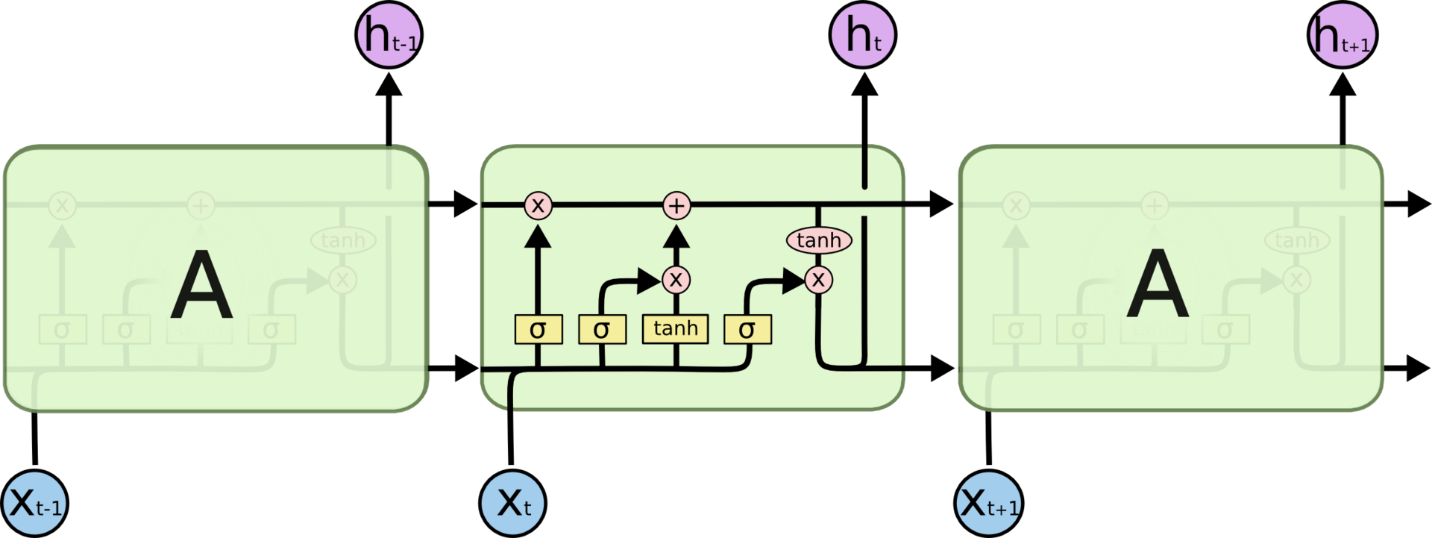


Рисунок 3. LSTM

Ключевой компонент LSTM – это состояние ячейки (cell state) – горизонтальная линия, проходящая по верхней части схемы. Она проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Информация может легко течь по ней, не подвергаясь изменениям. К трубе подключены два розовых «вентиля»: слева направо — вентиль «забывания» и вентиль «запоминания». Они контролируют, что нужно забыть, а что — запомнить.

LSTM может удалять информацию из состояния ячейки; этот процесс регулируется структурами, называемыми фильтрами (gates).

Фильтры позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения.

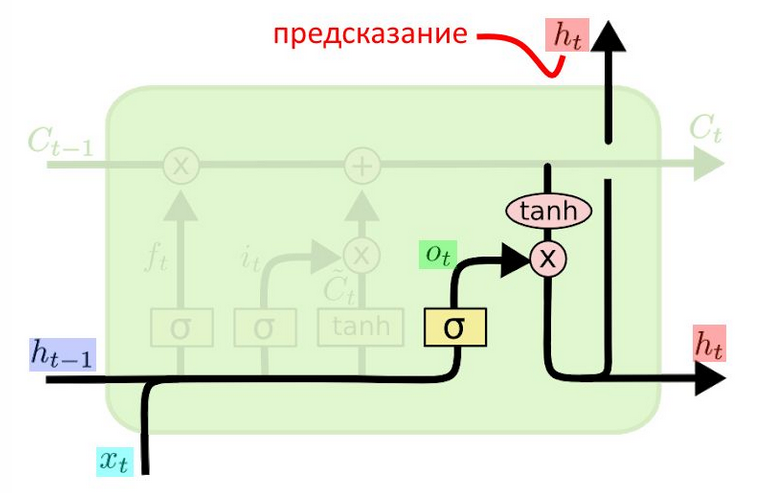


Рисунок 4. LSTM

Пошаговый разбор LSTM:

1. Определить, какую информацию можно выбросить из состояния ячейки.



- веса забывания

 - оценочный вектор

1. Решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений-кандидатов, которые можно добавить в состояние ячейки.



– матрица весов, полученная в процессе тренировки.

– вектор новых значений

- выученная матрица весов (веса входа)

 - «оценочный» вектор

1. Запись новых значений в контекст.



 - внутренняя память блока. Умножаем старое состояние на , забывая то, что решили забыть. Затем прибавляем новые значения-кандидаты, умноженные на t – на сколько мы хотим обновить каждое из значений состояния.

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



- предсказание, результат работы слоя LSTM.

Количество предыдущих дней, используемых в качестве входных данных, определяет одномерную последовательность данных, которая будет подаваться на вход для изучения и выделения признаков. Несколько вариантов:

 Все предыдущие дни за несколько лет.

 Предыдущие семь дней (одна неделя).

 Предыдущие две недели.

 Предыдущий месяц.

 Предыдущий год.

Нет однозначного ответа на вопрос, какой размер использовать. Лучшим решением будет протестировать каждый из них вместе с другими подходами, а по результатам выполнения выбрать тот, который показал наилучшие результаты.

В работе будет использоваться метод скользящего окна - алгоритм трансформации, позволяющий сформировать из членов временного ряда набор данных, который может служить обучающим множеством для построения модели прогнозирования. Под окном понимается временной интервал, содержащий набор значений, которые используются для формирования обучающего примера. В процессе работы алгоритма окно смещается по временной последовательности на единицу наблюдения, и каждое положение окна образует один пример.

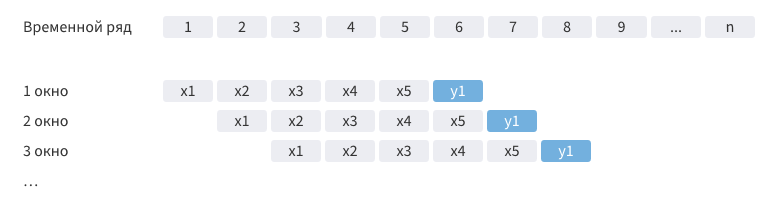


Рисунок 5. Пример скользящего окна

**2. Сбор и формирование данных**

Набор данных: Russian Wholesale Electricity (https://www.kaggle.com/irinachuchueva/russian-wholesale-electricity-market/).На оптовом рынке применяется узловая система ценообразования: каждый узел получает индивидуальное значение цены за определенный час. Для анализа цен и задачи прогнозирования аналитик использует средневзвешенное значение узловых цен для определенной территории, такой как ценовая зона.

На оптовом рынке электроэнергии актуальна проблема прогнозирования цен и потребления электроэнергии. Возникают вопросы:

* Какую цену ожидать завтра, на следующей неделе, в следующем году?
* Какое потребление электроэнергии следует ожидать завтра, на следующей неделе, в следующем году?

В датасете время указано в часовом поясе Европы / Москвы.:

* объем и цена на сутки вперед для ценовых зон Европы и Сибири (RU\_Electricity\_Market\_PZ\_dayahead\_price\_volume.csv)
* Timestamp-время (с 01.09.2006 по 22.11.2011) с шагом один час.
* consumption\_eur – рассчитываемый объем потребления в Европейской ценовой зоне в мВт
* consumption\_sib – рассчитываемый объем потребления в Сибирской ценовой зоне в мВт
* price\_eur – цена на мВт в Европейской ценовой зоне
* price\_sib – цена на мВт в Сибирской ценовой зоне

Всего 45816 измерений.

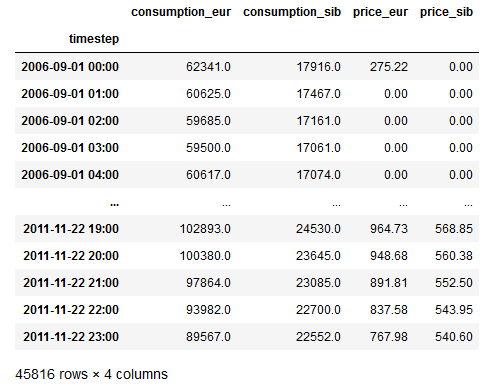


Рисунок 6. RU\_Electricity\_Market\_PZ\_dayahead\_price.csv датасет

Для прогнозирования выберем европейский регион.

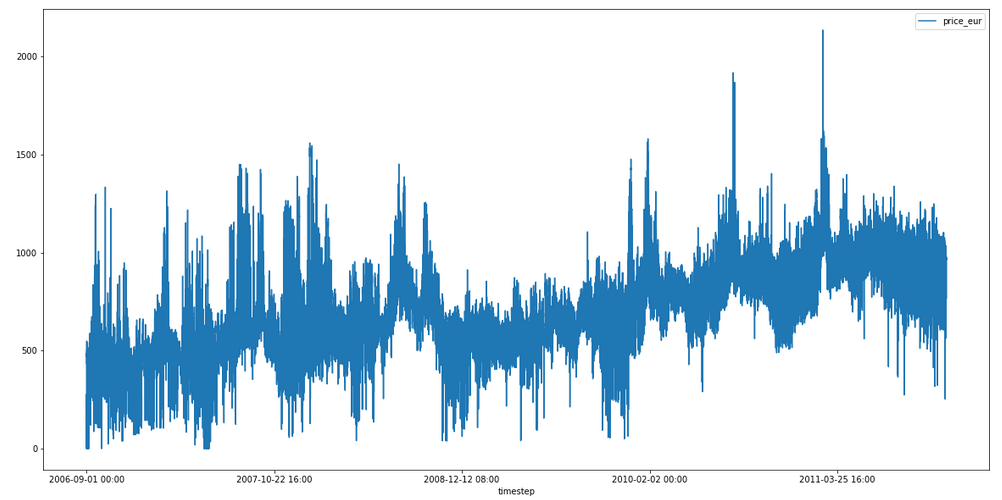


Рисунок 7. Изменение цены в европейском регионе

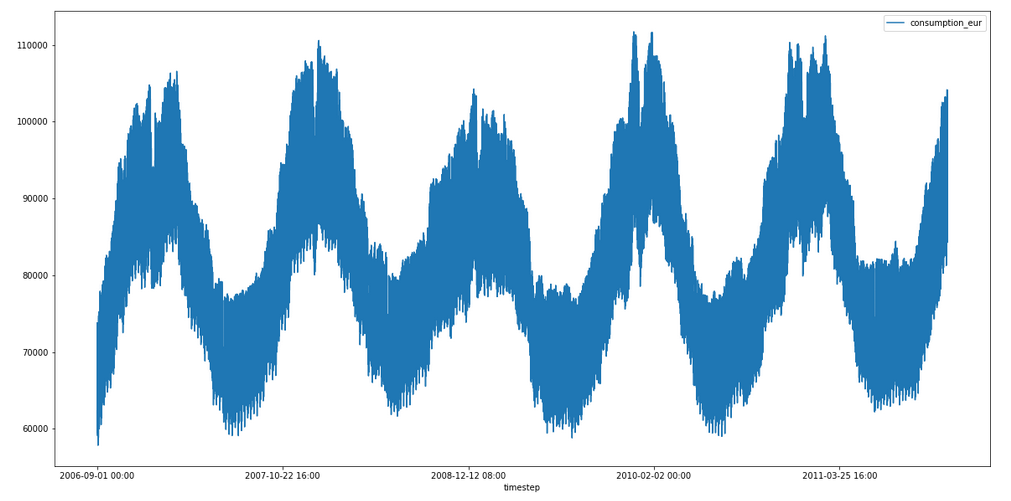


Рисунок 8. Изменение потребления в европейском регионе

**3. Анализ и предобработка данных**

Для начала следует убедиться, что отсутствуют пропущенные данные, если такие есть, то нужно использовать данные за тоже время за предыдущий день и провести нормализацию. Нормализация - это процедура предобработки входной информации (обучающих и валидационных выборок), при которой значения признаков во входном векторе приводятся к некоторому заданному диапазону, например, [0…1] или [-1…1]. Исходные значения признаков могут изменяться в очень большом диапазоне и отличаться друг от друга на несколько порядков. Необходимость нормализации вызвана тем, что разные признаки обучающего набора данных могут быть представлены в разных масштабах и изменяться в разных диапазонах. Также в слое нейросети используются функции активации «сигмоида» и «гиперболический тангенс», которые принимают значения от 0 до 1 и от -1 до 1 соответственно. Без нормализации на них будут подаваться относительно большие значения, и на выходе этих функций активации всегда будет 1. После нормализации все числовые значения входных признаков будут приведены к одинаковой области их изменения – некоторому узкому диапазону. Выборка была разделена на 60% - train;

20% - validation;

20% - test;

#Чтение датасета

df=pd.read\_csv(r"C:\Users\1\Downloads\RU\_Electricity\_Market\_PZ\_dayahead\_price\_volume.csv")

df=df.drop(columns=['consumption\_sib','price\_sib'])

df=df.set\_index('timestep')

#Обработка пропущенных значений

one\_day = 60 \* 24

for row in range(df.values.shape[0]):

for col in range(df.values.shape[1]):

if np.isnan(df.values[row, col]):

df.values[row, col] = df.values[row - one\_day, col]

#Нормализация

scaler=StandardScaler()

data\_scaled=scaler.fit\_transform(df)

means=scaler.mean\_

std=scaler.scale\_

lookback=168 #кол-во временных участков,

#на основании которых делаем прогноз

step=1 #шаг окна

delay=0 #кол-во прогнозируемых временных участков

batch\_size=128 #Размер пакета

#Разбивка данных на выборки

train\_split=int(0.6\*len(data\_scaled))

train=train\_split-(train\_split%batch\_size)

val\_split=int(0.8\*len(data\_scaled))

val=val\_split-(val\_split%batch\_size)

test\_split=(len(data\_scaled))

test=test\_split-(test\_split%batch\_size)

train\_gen = generator(data\_scaled,

lookback=lookback,

delay=delay,

min\_index=0,

max\_index=train,

shuffle=False,

step=step,

batch\_size=batch\_size)

val\_gen = generator(data\_scaled,

lookback=lookback,

delay=delay,

min\_index=train+1,

max\_index=val,

step=step,

batch\_size=batch\_size)

test\_gen = generator(data\_scaled,

lookback=lookback,

delay=delay,

min\_index=val+1,

max\_index=test+1,

step=step,

batch\_size=batch\_size)

#Сколько раз нужно обратится к вал.выборке

val\_steps=(val-train-1-lookback)//batch\_size

#Сколько раз нужно обратится к тест.выборке

test\_steps=(test+1-val-1)//batch\_size

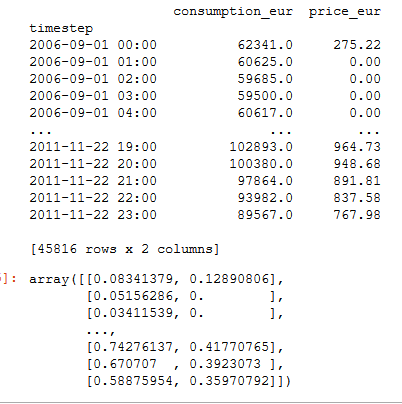


Рисунок 9. Изначальные данные и их нормализированные значения

**4. Обоснование выбора алгоритма машинного обучения и метрик качества**

Необходимость использовать нейросеть с запоминанием состояния-суточная периодичность данных. Потребление электроэнергии различно в разное время в течении дня (днём и ночью).

Среднеквадратическая ошибка - самая популярная метрика, используемая для задач регрессии (среднеквадратичная ошибка между прогнозируемыми и фактическими значениями).



Оптимизатор используем ADAM для замены стохастического градиентого спуска для обучения модели. Это позволит быстрее обучаться и не застревать в локальных минимумах. Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков.

**5. Обучение алгоритма**

Тензоры — многомерные массивы. Тензором может быть, как отдельное число, вектор признаков из решаемой задачи или изображение, так и целый батч описаний объектов или массив из изображений. Вектор - это одномерный тензор или тензор первого порядка, а матрица - двумерный тензор или тензор второго порядка [3].

В моей задаче форма тензора, подающегося на вход нейросети имеет следующий вид:

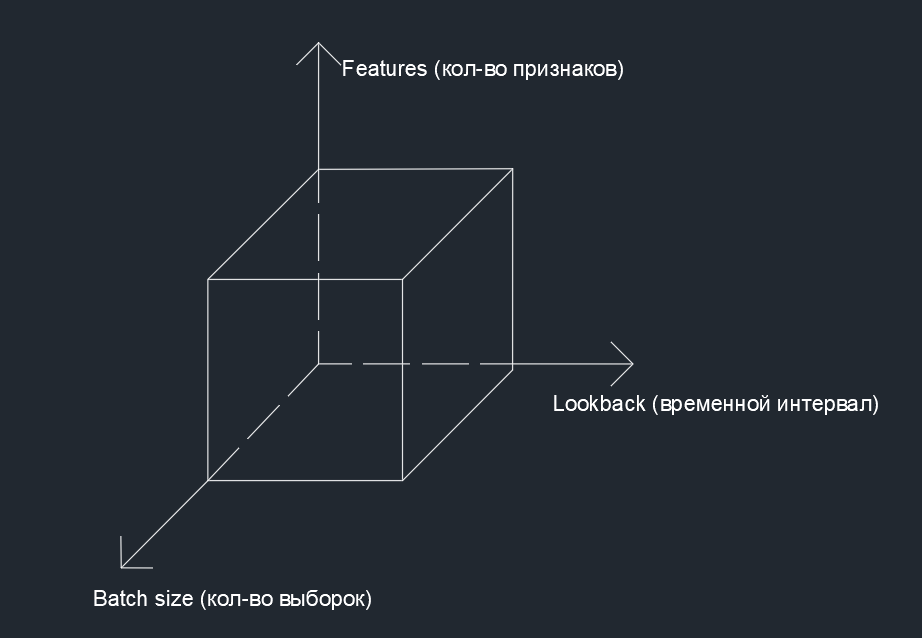


Рисунок 10. 3D-тензор на выходе функции-генератора.

В случае большого набора данных следует генерировать данные на ходу при помощи функции-генератора, чтоб не было переполнения оперативной памяти [4].

def generator(data, lookback, delay, min\_index, max\_index,

shuffle=False, batch\_size=128, step=1):

if max\_index is None:

max\_index = len(data) - delay - 1

i = min\_index + lookback

while 1:

if shuffle:

rows = np.random.randint(

min\_index + lookback, max\_index, size=batch\_size)

else:

if i + batch\_size >= max\_index: #если превысили, считаем заново(когда наступает след.эпоха)

i = min\_index + lookback

rows = np.arange(i, min(i + batch\_size, max\_index)) #окно 168-311(=144)

i += len(rows) #+=144

samples = np.zeros((len(rows),

lookback // step,

data.shape[-1]))

targets = np.zeros((len(rows),))

for j, row in enumerate(rows): # j=0-batch\_size row=168...end

indices = range(rows[j] - lookback, rows[j], step)#168-lookback,168

#169-lookback,169

samples[j] = data[indices] #окно 0-168 1-169 2-170... кол-во окон=batch\_size

targets[j] = data[rows[j] + delay][1] #цена на след.день после 7ми дней

yield samples, targets

Генератор последовательно возвращает тензор 3х мерной формы, так как РНС работают с последовательностями (sequences). Это означает, что подаваемые на вход модели данные должны иметь следующую форму:

[выборки, временной интервал, кол-во признаков]

Сгенерированный тензор с наблюдениями имеет форму (128, 168, 2), а тензор целевой переменной - (128,).

128- batch\_size (размер пакета).

168 – lookback. Количество наблюдений за предыдущую неделю.

2 – количество признаков (потребление и цена)

Обучающие данные подаются как перекрывающиеся окна (0-168, 1-169, 2-170 и.т.д.). В пакете, подающимся за раз в сеть batch\_size окон.

Размер пакета должен отражать периодичность данных, так как РНС сопоставляет состояния соответственных элементов соседних пакетов, а это значит, что при правильном размере пакета сеть будет обучаться быстрее, поэтому размеры обучающей и тестовой выборки должны быть кратны размеру пакета.

Модель сети последовательна и состоит из двух слоёв – LSTM и Dense.

LSTM сеть состоит из нейронов с функцией сохранения состояния (stateful). Внутреннее состояние, полученное после обработки очередного пакета данных, повторно используется в качестве начальных состояний для выборок следующей серии для обработки длинных последовательностей. Выходной вектор имеет размерность (128,8) и поступает на слой Dense.

Dense - это линейная модель, состоящая из одного нейрона, на выходе которой мы получаем 128 окна с предсказанием цены (одно число, поэтому один нейрон).

Рассмотрим нейронную сеть, не сохраняющую состояние от пакета к пакету (stateless) с перемешиваем выборки.

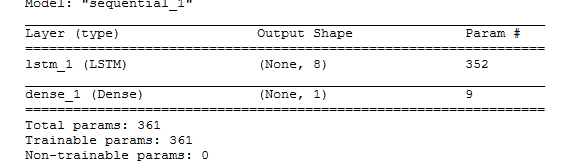


Рисунок 11. Двухслойная stateless НС с восемью нейронами.

Средняя квадратическая ошибка MSE = 0.1384619627171737

Если установить параметр stateful = True, чтоб состояния не сбрасывались и последнее состояние сети на последнем элементе пакета было использовано как начальное состояние сети при подаче на него следующего пакета, при этом не следует перемешивать выборку.

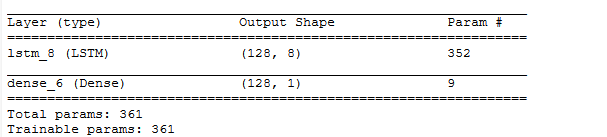


Рисунок 11. Двухслойная НС с восемью нейронами.

Средняя квадратическая ошибка MSE = 0.1394514555887576

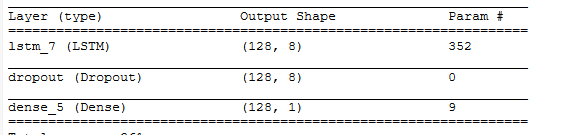


Рисунок 12. Двухслойная НС с восемью нейронами и прореживаем.

Средняя квадратическая ошибка MSE = 0.15574186333012607

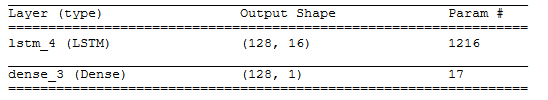


Рисунок 13. Двухслойная НС с шестнадцатью нейронами.

Средняя квадратическая ошибка MSE = 0.1514417056302534

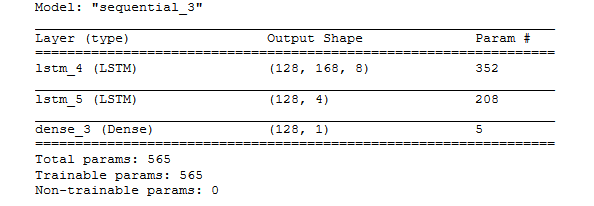


Рисунок 14. Трехслойная НС.

Средняя квадратическая ошибка MSE = 0.1358666393600728. Это несущественно лучше двухслойной сети, однако время обучения существенно больше, поэтому стоит отдать предпочтение двухслойной сети.

**6. Оценка качества алгоритма**

По результатам обучения в течении 40-ти эпох модели удалось получить минимальную среднеквадратическую ошибку, равной 0.14

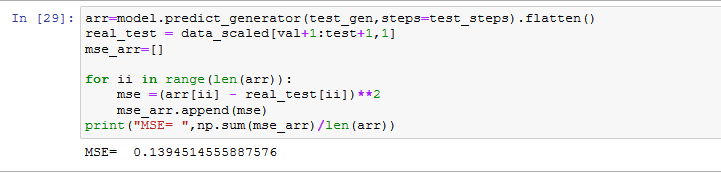


Рисунок 15. Оценка MSE между обученной моделью и тестовыми данными.

Также для оценки модели был выбран тестовый день, на графике ниже представлено оригинальное и предсказанное изменение цены в течении недели.

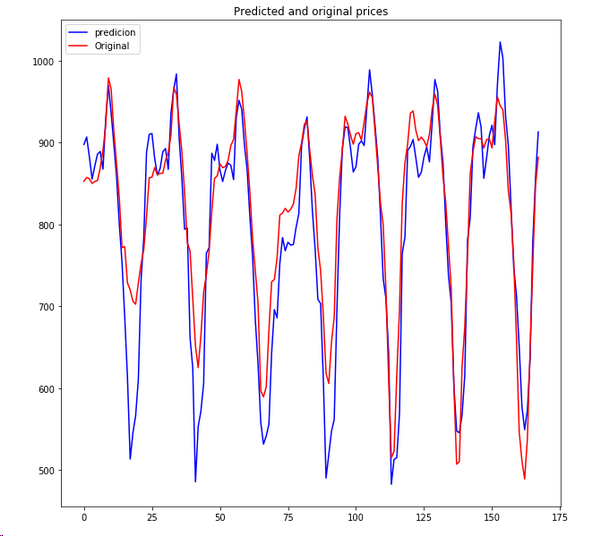


Рисунок 16. График сравнения спрогнозированной и действительной цены за неделю.

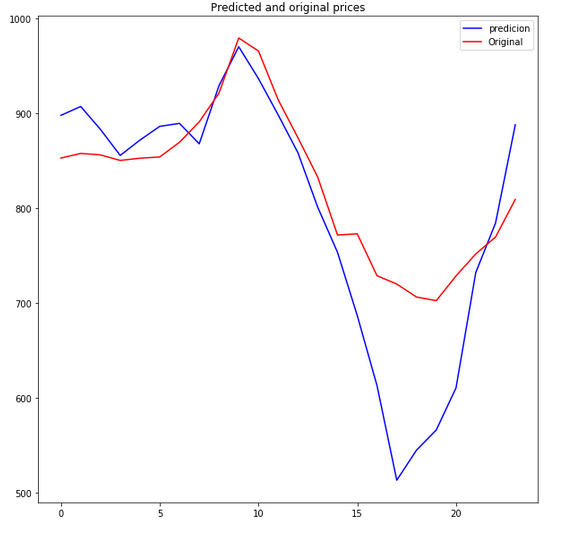


Рисунок 17. График сравнения спрогнозированной и действительной цены за день.

Видно, что модель не очень точно предсказывает выбросы (резкое изменение цены).

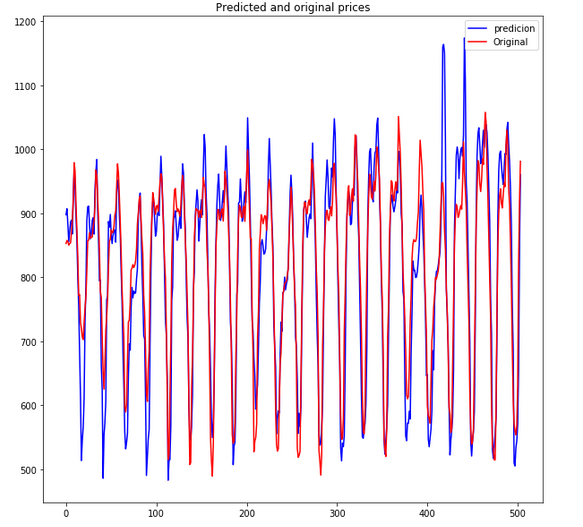


Рисунок 18. График сравнения спрогнозированной и действительной цены за месяц.

Можно сделать вывод, что модель научилась запоминать выбросы, и на следующих неделях разница между предсказанными и оригинальными значениями во время резких изменений цен уменьшилась.

Для сравнения приведём обучение без машинного обучения, предполагая, что следующее значение временного ряда равно фактическому значению предыдущего шага.

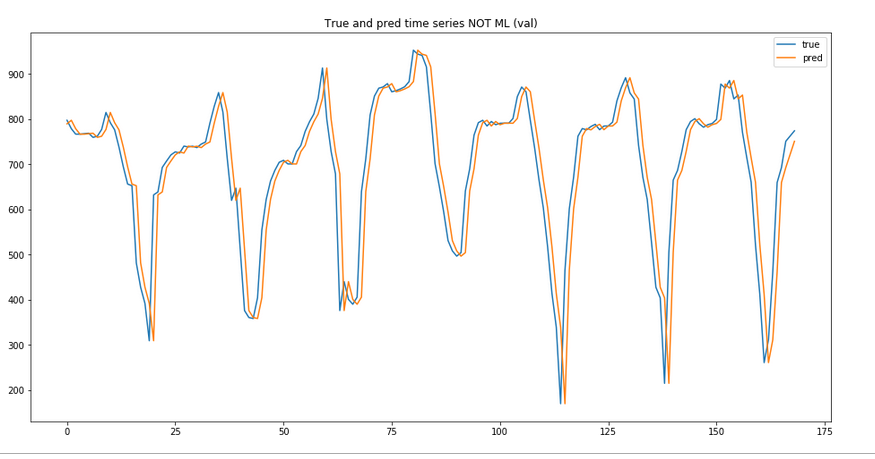


Рисунок 19. График сравнения спрогнозированной и действительной цены за неделю без машинного обучения.

**Заключение**

В курсовом проекте был рассмотрен принцип работы рекуррентной нейронной сети с долговременной и кратковременной памятью (LSTM) для задачи регрессии. В процессе работы была разобрана функция-генератор, которая при больших входных данных позволит избежать переполнения оперативной памяти. Оценивать работу сети пришлось на двумерной временной последовательности, а путем выбора различных архитектур удалось найти оптимальную с точки зрения точности и скорости нейронную сеть. По результатам обучения были сделаны графики, наглядно показывающие предсказательные способности сети, то есть отличие реальных данных от предсказанных.

**Список литературы**

1. Введение в машинное обучение с помощью Python / Мюллер А., Гидо С. – 2017.
2. Understanding LSTM Networks // Портал « https://colah.github.io/» Просмотрено: 1.12.2020
3. TensorFlow для глубокого обучения / Рамсундар Бхарат. -2019.
4. Глубокое обучение на Python / Шолле Франсуа. -2018.
5. Библиотека Keras – инструмент глубокого обучения/ Антонио Д, Суджит П. -2018.