**Министерство науки и высшего образования Российского Федерации Федеральное государственное автономное образовательное учреждение высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ им. Н. И. ЛОБАЧЕВСКОГО

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Направление: 02.03.01 – «Математика и компьютерные науки»

Профиль: Наука о данных

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Использование машинного обучения для решения задачи прогнозирования метеопараметров**

Студент 4 курса группы 05-804

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Р. Имамутдинов

Научный руководитель:

кандидат физ.-мат. наук, доцент

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Агафонов

Заведующий кафедрой ВмиММ

кандидат физ.-мат. наук, доцент

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Агафонов

КАЗАНЬ – 2022

**Оглавление**

Введение ..........................................................................................................… 3

Глава 1. Использование машинного обучения в метеорологии…………...... 5 1.1 Метеорология и метеопараметры ................................................................ 5 1.2 Машинное обучение и метрики качества модели ...................................... 8 1.3 Способы обработки первичных данных .………………............................ 11 1.4 Нейронные сети и обучение ......................................................................... 15

Глава 2. Программа прогнозирования метеопараметров ..………….............. 20 2.1 Описание набора данных и его предобработка .......................................... 20 2.2 Обучение нейронной сети............................................................................. 30 2.3 Результаты обучения модели ...................................................................... 35

Заключение ......................................................................................................... 40  
Список литературы ............................................................................................ 41

**Введение**

Существует проблема предсказания погоды, которая заключается в неполноте и неточности информации о текущем состоянии погоды, несовершенстве используемых прогностических методов и моделей, а также в непредсказуемом поведении отдельных циклонов, зон осадков, экстремумов температуры и в целом в изменчивости погоды[1]. Решение этой проблемы будет влиять на различные отросли, такие как: сельское хозяйство, энергетика, морской транспорт, курортное хозяйство. Погода может меняться как в течение дня, так и на протяжении недели или месяца[2]. Простые наблюдения за погодой с утра не всегда гарантируют неизменность погоды, поэтому многие пользуются современными прогнозами на метеорологических сервисах, таких как: ‘gismeteo’, ’Яндекс. Погода’, ’accuweather’ и другие. Синоптики делят прогнозирование погоды на несколько этапов: краткосрочные (до двух суток), долгосрочные (от десяти дней), промежуточные (от двух до десяти дней), и в целом успешность прогнозирования зависит от предсказания тенденции погоды. Причем самые успешные результаты в прогнозировании погоды получаются на краткосрочных этапах.

Все они основываются на наблюдениях за погодой на больших участках территорий и прогнозировании погоды исходя из прошлых данных, используя машинное обучение. Для этого используются наблюдения за метеостанциями, спутниковый мониторинг облачности территории, движение ветра и прочее, что отслеживается современными измерительными приборами. Все полученные метеопараметры (данные) непрерывно собираются в огромные массивы данных, что передается дальше в обработку, для изучения и предсказания моделью будущей погоды.

В данной работе представлен алгоритм обработки и предсказания погоды.

Цель работы: Используя машинное обучение, спрогнозировать метеопараметры на примере температуры воздуха.

Задачи:

1. Выгрузить и обработать набор данных

2. Выбрать наилучшую архитектуру многослойной нейронной сети

3. Получить близкий к реальным данным прогноз метеопараметра

Работа состоит из двух глав, введения и заключения.

Для достижения поставленной цели подразумевается работа с пропущенными значениями, которые напрямую влияют на качество обучения и предсказания модели, поэтому необходимо понять какой способ заполнения пропусков оптимальный.

План работы программы выглядит следующим образом:

1. Выгрузка набора данных с последующим заполнением пропущенных значений.

2. Выявление зависимых параметров в наборе данных с прогнозируемым параметром, чтобы отсеять нежелательную информацию от необходимой перед отправлением данных для обучения.

3. Обучение многослойного персептрона с различными архитектурами для выявления лучшей архитектуры.

4. Масштабирование алгоритма для прогнозирования погоды на других станциях, с разным количеством станций в округе, на определенное время вперед.

5. Визуальное представление результатов работы моделей, их оценки метрик.

**Глава 1. Использование машинного обучения в метеорологии**

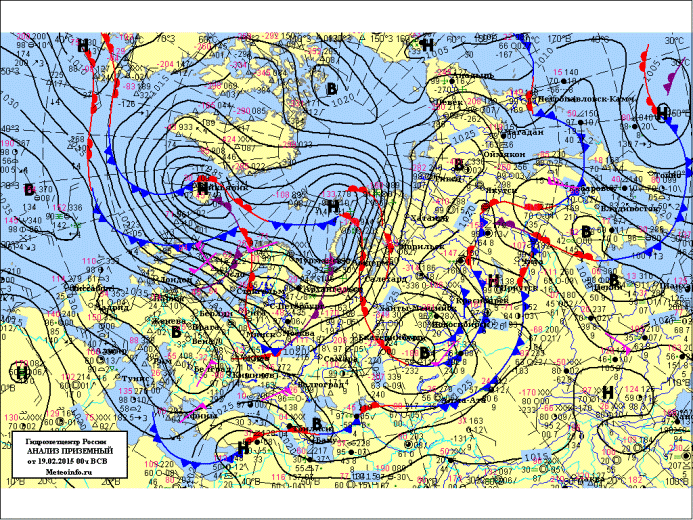
**1.1 Метеорология и метеопараметры**

Каждый день человек обращает внимание на состояние окружающей его среды. Если бы произошло наводнение или землетрясение, он был бы рад только тому, если бы узнал об этом заблаговременно. К счастью такие опасные природные явления встречаются не слишком часто, но согласно статье [3], их количество увеличивается. Поэтому даже если пойдет дождь или станет слишком жарко в течение дня, человек хочет об этом знать перед тем как уходить из дома или поливать выращиваемые культуры. Погода меняется периодически и не периодически, например, ночью всегда холоднее чем днем, зимой средняя температура дня уменьшается по сравнению с летом. Не периодические же изменения — это циркуляция воздушных масс в средних широтах планеты. Несовпадение фаз периодических изменений с не периодическими приводят к резким изменениям погоды. Изменчивость окружающей среды происходит постоянно, поэтому, чтобы избежать ненужных трат ее пытаются предсказывать, причем ценится не только качество предсказания состояния погоды, но и найденные зависимости между какими-то определенными событиями или признаками и их последствиями.

**Метеорология** — это наука, занимающаяся изучением строения и свойств физико-химических процессов в земной атмосфере, то есть пониманием закономерностей в атмосферных процессах. Именно состояние атмосферы у земной поверхности называют погодой. Ее характеристики, такие как: температура, количество осадков, давление, влажность, облачность, скорость и направление ветра и другие называют ***метеопараметрами***. В любом месте на Земле погода в разные годы может протекать по-разному, но имея определенный климат. **Климат** — это совокупность погодных условий (метеопараметров) присущих определенной местности в зависимости от ее географической обстановки, то есть не только географического положения (широты, долготы, высоты), но и характер земной поверхности, почвенный покров и др. Так, например, было подмечено что атмосферные условия меняются в годовом ходе, от лета к зиме за счет удаления Земли от Солнца и наоборот. Но от одного многолетнего периода к другому совокупность атмосферных условий меняется лишь в самых ограниченных пределах, причем эти наблюдения имеют характер колебания то в одном, то в другом направлении[4]. Климат таким образом обладает устойчивостью.

Основные атмосферные процессы развёртываются на больших пространствах, а значит их последствия тоже будут иметь большой масштаб, поэтому их движения принято рисовать на синоптических картах (Рис. 1) Такие карты могут обновляться через небольшой промежуток времени для диагностирования движения атмосферных процессов. Сопоставляя синоптические карты над одной и той же территории последовательно друг за другом можно прослеживать развитие атмосферных процессов и делать выводы о будущей погоде [5].

**Метеорологические наблюдения** - это измерения и качественные оценки метеорологических элементов. К метеорологическим элементам относятся в первую очередь температура и влажность воздуха, атмосферное давление, ветер, облачность, осадки, туманы, метели, грозы, видимость. Наиболее полные и точные наблюдения производятся в метеорологических и аэрологических обсерваториях, имеющихся во всех странах мира. Число таких обсерваторий, однако, невелико. Кроме того, даже самые точные наблюдения в немногочисленных пунктах не могут дать исчерпывающего представления обо всей жизни атмосферы, поскольку атмосферные процессы протекают в разной географической обстановке по-разному. Поэтому, кроме метеорологических обсерваторий, наблюдения над основными метеорологическими элементами ведутся еще на многих тысячах метеорологических станций по всему Земному шару. Правда необходимо чтобы метеорологические станции в различных участках Земли вели наблюдения, по возможности, однотипными приборами, по единой методике, в определенные часы суток. Тогда они будут составлять единое целое - сеть метеорологических станций, метеорологическую сеть. В каждой стране, в том числе и в России, существует основная государственная сеть метеорологических станций, отвечающая указанному выше требованию - единообразной и согласованной работы. Метеорологические станции общегосударственной сети устанавливаются по возможности равномерно в местах, характерных для данного района. Нужно стремиться к тому, чтобы показания станции были репрезентативными, то есть характерными не только для ее ближайших окрестностей, но и для возможно большего окружающего района.

(Рис. 1 Синоптическая карта)

**1.2 Машинное обучение и метрики качества модели**

**Машинное обучение** – это обучение математической модели на основе тренировочных данных с целью нахождения общих закономерностей среди них и прогнозирования новых данных. Тем самым машинное обучение не решает задачу напрямую, а лишь предоставляет обученную математическую модель для нужных нам предсказаний признаков из тренировочных данных.

Согласно статье [6 и 7] алгоритмы машинного обучения делятся на три типа:

1. **Обучение с учителем** – описываемый метод машинного обучения подразумевает размеченные данные, то есть когда модель обучается на примерах.

Алгоритмы обучения с учителем включают в себе:

1.1. Задача *классификации* – соотнесение схожих черт данных к различным классам

1.2. *Регрессия* - в задачах регрессии программа машинного обучения должна оценить и понять взаимосвязи между переменными. Предмет регрессионного анализа – одна зависимая переменная и набор других изменяющихся переменных.

2. **Обучение без учителя** – в данном случае модель пытается выявить закономерности между данными (паттерны), чтобы можно было их разделить или сгруппировать между собой, при этом модель выполняет это самостоятельно описывая их структуру и упорядочивая их.

Алгоритмы обучения без учителя включают в себе:

2.1. Задачу *кластеризации* – группировка наборов данных, схожих между собой.

2.2. Задача *понижения размерности данных* – отделение искомой информации в данных от шума.

3. **Обучение с подкреплением** – модель учится на примере игры, где существуют правила и каждое действие модели оценивается. Методом проб и ошибок модель находит оптимальную стратегию для победы в игре исходя из запоминания проделанных действий и их последствий.

Для оценки успешности модели, набор данных заранее делят на тестовую и проверочную часть. Это необходимо делать чтобы любая модель не переобучалась на текущей выборке, и мы видели насколько модель с этим справляется, особенно если эта выборка очень мала по сравнению с генеральной совокупностью. Самый простой способ для такого – поделить набор данных на тестовую и проверочную часть в процентном соотношении, к примеру 80 на 20. Обучать модель на тестовой части, подбирая параметры сети, а проверять результаты на проверочной части и смотреть как хорошо модель работает[8].

Один из простых методов оценки модели классификации нейронной сети – использование Accuracy score. Данный метод берет соотношение правильных предсказаний ко всем предсказаниям нейронной сети. Его обычно применяют, когда распределение между классами предсказываемого параметра равномерное, в других случаях оно дает незначимый результат оценки модели.

Когда классы прогнозируемого параметра распределены не равномерно, можно воспользоваться F-мерой [9], которая учитывает в себе ошибки первого и второго рода (false positive and false negative). Она состоит из вычисления двух величин: precision(точность) и recall (полнота).

Также есть и другие метрики, которые измеряют погрешность предсказываемого результата с реальным и простые в понимании, такие как RMSE и MAPE:

RMSE – корень квадратичной ошибки, где y- правильные результаты, а y\_pred – предсказываемые моделью результаты

MAPE – средняя абсолютная ошибка, где Y – правильные результаты, а Y\_pred – предсказываемые моделью результаты

**1.3 Способы обработки первичных данных**

Ни одно обучение модели не обходится без специальных данных для обучения. Эти данные могут быть как размеченными, так и нет. К примеру: на этой картинке изображена собака, а на другой кошка, а на следующих десяти тысяч картинок изображены собаки и кошки вперемешку, но каждая картинка подписано тем животным, которое изображено на картинке. Важную роль играет корректность подписанных картинок, если будут картинки с ошибочными подписями, то и процесс обучения различия изображенных на картинке животных будет проходить дольше и тяжелее, потому что модель будет сбиваться в закономерностях, которые она вычисляет между одинаковыми животными, но подписанными по-разному. Точно также ведет себя и человек в процессе обучения, когда ему предоставляют корректную, правильную информацию, то понять ее и осмыслить проще, чем если бы ему предоставляли информацию с ошибками, где он бы находил не состыковки, ему бы потребовалось больше времени для анализа информации и заключения вывода о ней. Такой тип задач обучения называется классификацией, а само обучение - обучение с учителем, где данные для обучения изначально корректно подписаны и модель пытается отличить одни подписи от других находя закономерности в представленной ей информации. Другой случай, когда данные не размечены и модели предстоит задача кластеризации – отделение одних объектов от других по каким-то критериям. Так, например, если дать модели для обучения русский алфавит, то она отделит буквы друг от друга, да и потом на тексте, составленном из этого же алфавита найдет одинаковые буквы.  
Набором данных может называться любая такая структурированная и обработанная информация, представленная в виде таблицы, которая используется в качестве обучения. Набор данных может быть собран из картинок, текста, аудио, поскольку любую информацию можно представить в виде чисел. Также важно понимать из какой генеральной совокупности произведена выборка и каким образом, потому что это влияет на полученные результаты после решения поставленной задачи. К примеру, если задача была представлена в виде: Как люди относятся к обновлению интернет магазина, то генеральная совокупность должна покрывать всех пользователей интернет магазина которые пользовались им до обновления и после него, а выборка должна быть разнообразной, а не одного типа людей, скажем людей из Москвы или людей младше 21 года и при этом быть независима между каждыми наблюдениями\опытами [10].

Существует множество вариантов предобработки данных, где каждый вариант отличается от других и походит для определенной информации. Эта предобработка данных нужна в случае, когда данные предоставляются в первичном виде (‘сыром виде’) или с пропущенными значениями.   
‘Сырые данные’ это данные которые могут поступать в виде логов событий (см. Таблица 1 и Таблица 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | step\_id | timestamp | submission\_status | user\_id |
| 0 | 31971 | 1434349275 | correct | 158553 |
| 1 | 31972 | 1434348300 | correct | 158553 |
| 2 | 31972 | 1478852149 | wrong | 158553 |
| 3 | 31972 | 1478852164 | correct | 158553 |
| 4 | 31976 | 1434348123 | wrong | 158553 |

(Таблица 1 submission\_data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | step\_id | timestamp | action | user\_id |
| 0 | 32815 | 1434340848 | viewed | 17632 |
| 1 | 32815 | 1434340848 | passed | 17632 |
| 2 | 32815 | 1434340848 | discovered | 17632 |
| 3 | 32811 | 1434340895 | discovered | 17632 |
| 4 | 32811 | 1434340895 | viewed | 17632 |

(Таблица 2 events\_data)

Как видно из таблицы 1 и 2, эти данные нужно преобразовывать чтобы получить какую-то полезную информацию[11]. Можно заметить, что в этом может помочь связывание двух таблиц по колонке step\_id или user\_id. Так, например, после некоторых группировок или объединений, или сортировки данных по столбцам можно из этих данных получить уже что-то понятное для обучения (см. Таблицу 3)

|  |  |  |  |
| --- | --- | --- | --- |
| submission\_status | user\_id | correct | wrong |
| 0 | 2 | 2 | 0 |
| 1 | 3 | 29 | 23 |
| 2 | 5 | 2 | 2 |
| 3 | 8 | 9 | 21 |
| 4 | 14 | 0 | 1 |
| 5 | 16 | 77 | 106 |
| 6 | 17 | 0 | 2 |

(Таблица 3 Количество верных\неверных ответов)

А вот с обработкой пропущенных значений не все так просто, согласно статье [12], неправильное заполнение пропущенных значений или их игнорирование, а также отбрасывание может привести к ошибкам в моделях и в принятии решений. Для того чтобы решить, как их можно обработать надо понять, как они появились. Различают три типа появления пропущенных значений:

1. MCAR – missing completely at random
2. MAR – missing at random
3. MNAR – missing not at random

Первый тип предполагает, что вероятность пропущенного значения для каждой записи одинакова, в таком случае игнорирование или исключение пропущенных записей не ведет к искажению результата. Второй тип пропуска получается ввиду каких-то закономерностей, представленных из таблицы, то есть вероятность пропуска может быть определена исходя из другой информации в таблице. Здесь так же игнорирование или исключение пропущенных записей не существенно сказывается на результате. Но вот с третьем типом, где вероятность пропуска невозможно найти исходя из информации, представленной в таблице, все обстоит похуже. Здесь нужно анализировать что может произойти в случае того или иного способа заполнения пропуска.

Итак, возможные способы заполнения пропусков:

1. Исключение записей содержащие пропуски (колонок, строк) - может возрасти стандартное отклонение, используются не все данные и как следствие результаты менее репрезентативны. Когда таких данных много то это становится существенной проблемой.

2. Метод игнорирования пропущенных данных – данный метод не годится для обработки данных если они используются в машинном обучении, потому что машина не воспринимает данные, где пропущены значения. Этот метод применяется при условии выполнения гипотезы mcar, и то не всегда, потому что могут возникнуть некорректные результаты статистических величин.

3. Заполнение пропуска средним значением и модой – Интуитивно кажется, что проблем возникнуть не должно, потому что заполняем средним или модой, но ввиду различия данных, важно понимать, как они распределены, прежде чем пытаться заполнять этим способом.

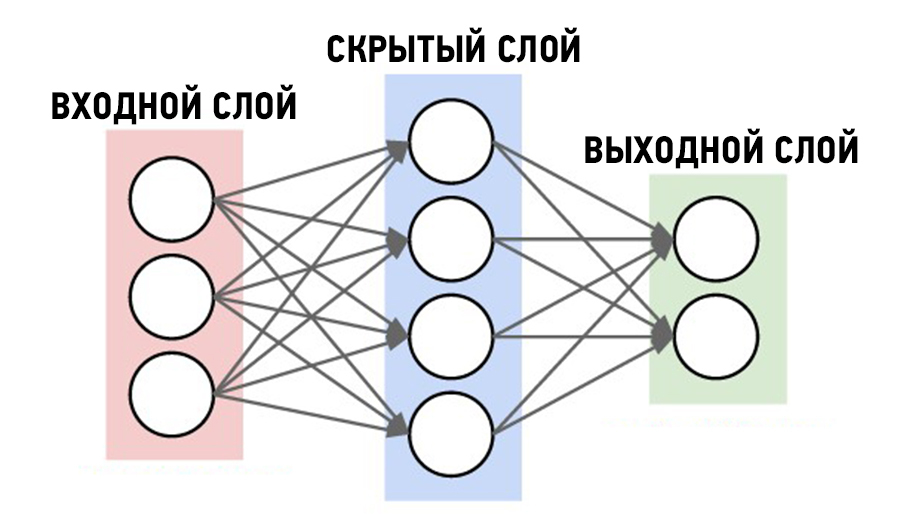
4.LOCF (pocf)- last(previous) observation carried forward – Заполнение пропусков следующим (предыдущим) значением. Хорошо подходит для временных рядов, потому что предыдущие значения связаны с последующими.

5. Восстановление пропущенных значений на основе регрессионной модели. Этот метод заполняет пропуски правдоподобными значениями. Однако корреляция между характеристиками, на которых построенная регрессия увеличивается, а вариация значений заполняемого параметра уменьшается. Причем этот метод хуже работает чем больше пропущенных значений.

Тем самым важно определить какое распределение имеют пропущенные значения и как эта характеристика коррелирует с другими характеристиками в таблице, прежде чем как начать заполнять пропуски.

**1.4 Нейронные сети и обучение.**

Нейронные сети своим строением похожи на нейроны головного мозга, где каждый нейрон принимает информацию от других нейронов, и выдает какой-то свой сигнал другим нейронам. Нейронные сети повторяют эту концепцию, нейронные сети выглядят следующим образом (см. Рис. 2)



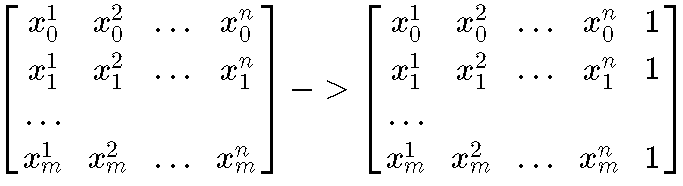
(Рис.2 Схема нейронной сети)

**Входной слой** (input layer) представляются N нейронами, где N – число столбцов или длина одной записи в наборе данных. Нейроны входного слоя не вычисляют ничего, они служат доступом к последующим слоям скрытого слоя. На **скрытых слоях** (hidden layers) происходят алгебраические и математические операции с предыдущим слоем. А **выходной слой** (output layer) – результат предсказания нейронной сети.

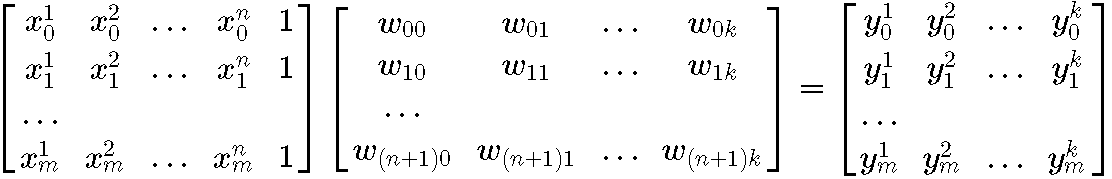
Существует множество моделей нейронных сетей, одни из них: персептрон, многослойный персептрон, сверточная нейронная сеть, рекуррентная нейронная сеть, сети долгой краткосрочной памяти, рекурсивная нейронная сеть, генеративно состязательные сети, импульсная нейронная сеть, seq2seq модель. Они отличны друг от друга во фрагментах нейронной сети, то есть нейроны могут выполнять разные математические и алгебраические действия как с данными, поступающими на них от нейронов предыдущего слоя, так и могут брать данные с других нейронов на том же слое. Ту или иную модель нейронной сети выбирают в зависимости от поставленной задачи и зависимости данных[13]. Также существует несколько методов обучения нейронной сети, помимо двух основных направлений обучения с учителем и без учителя, саму нейронную сеть нужно продолжать обучать либо говорить, что нейронная сеть уже обучена. Этого помогают нам достичь методы обучения, такие как:

1. Метод обратного распространения ошибки (back propagation error) – метод позволяющий корректировать веса модели за счет двух моментов: передачи данных от входного слоя до выходного (прямое распространение информации) и обратный метод распространения информации за счет подсчета функции ошибки на выходном слое и последовательное изменение весов сети начиная с последнего за счет дифференцирования функции ошибки используемой в вычислении отклонения предыдущего слоя нейронов.
2. Метод упругого распространения (resilient propagation) - Похож на метод обратного распространения ошибки, только используется не значение дифференцируемой функции ошибки, а ее знак, позволяющий определить изменение корректирующей переменной для весов сети.
3. Генетический алгоритм (NEAT) – Позволяет найти наилучшую минимально необходимую архитектуру сети (топологию) за счет выборки лучших двух результатов поколения исходя из их fitness function и их скрещивания с вероятностью мутации следующего поколения, которое использует результаты двух родителей, но с некоторыми изменениями.

**Прямое распространение:**  
Берутся все наблюдения в наборе данных в количестве m штук по n столбцов и добавляется столбец 1. Полученная матрица называется L0.

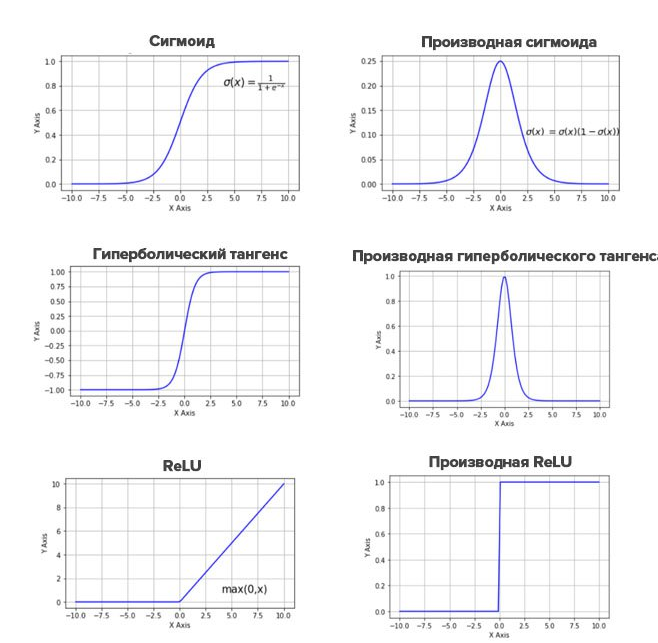


Эта матрица L0 умножается на заранее выбранные (рандомно) веса матрицы W0, матрица, находящаяся между L0 слоем и L1.



Применяется функция активации к получившейся матрице (к каждому элементу матрицы):

Где δ – дифференцируемая функция активации. В качестве функции активации можно использовать (см. Рис.3)



(Рис. 3 Функции активации)

Потом снова приписывается столбец 1 к матрице Z, и эта матрица умножается на веса матрицы W1 (заранее выбранные рандомно). Этот процесс повторяется столько раз, сколько скрытых слоев в нейронной сети. На последнем слое – выходном, получим ответ – предсказание нашей сети.

**Обратное распространение[14]:**

Вычисляются отклонения между предсказываемой переменной и прогнозом сети.

Строится матрица обратных ошибок:

После чего можно вычислить:

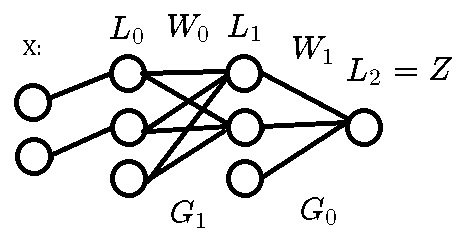
И соответствующую матрицу ошибок:

Также продолжаются строиться матрицы ошибки вплоть до номера n. После этого корректируются веса нейронной сети по правилу:

,

Где оператор [:,:-1] – все строки кроме последней строки.

Пример многослойного перцептрона с 1 скрытым слоем (см. Рис. 4)



(Рис.4 Пример многослойного персептрона с 1 скрытым слоем)

Разберем метод обучения прямого и обратного распространения ошибки в модели многослойного персептрона, который используется в данной работе для предсказания метеопараметров погоды, на примере многослойного персептрона с 1 скрытым слоем. Отделяем предсказываемую переменную от набора данных.

**Прямое распространение:**

К входным параметрам X приписывается 1, получаем слой L0.

Далее: , , где веса (матрицы) W0 и W1 изначально выбраны рандомно (отличны от 0).

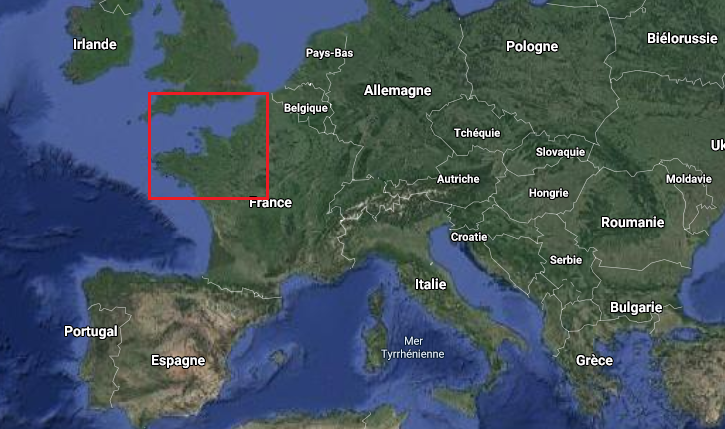
**Обратное распространение:**

Где оператор [:,:-1] означает все строки кроме последней строки.

**Глава 2. Программа прогнозирования метеопараметров**

**2.1 Описание набора данных и его предобработка**

Прогнозирование метеопараметров погоды должно основываться на данных метеостанций или снимков спутников, причем они должно быть в большом количестве. В этом случае за основу взят набор данных MeteoNet North-West France на kaggle.com [15]. Набор данных – это набор данных, который берется для обучения и предсказания на нем нейронной сети. Этот набор данных включает в себе: Снимки со спутников на территории северо-восточной Франции на протяжении 3 лет с 2016 по 2018 года, а также данные более чем 500 метеостанций, расположенных на этой же территории 550х550 км. за те же 3 года (см. Рис. 5)



(Рис. 5 Территория)

Но использовать в данной работе будем лишь данные метеостанций на этой территории.

В данном наборе данных есть несколько столбцов (см. Таблицу 4)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | number\_sta | lat | lon | height\_sta | date | dd | ff | precip | hu |
| 0 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:00 | 210.0 | 4.4 | 0.0 | 91.0 |
| 1 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:06 | 210.0 | 4.0 | 0.0 | 91.0 |
| 2 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:12 | 210.0 | 3.7 | 0.0 | 90.0 |
| 3 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:18 | 210.0 | 3.4 | 0.0 | 92.0 |
| 4 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:24 | 200.0 | 2.8 | 0.0 | 91.0 |
| 5 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:30 | 200.0 | 2.7 | 0.0 | 91.0 |
| 6 | 14066001 | 49.33 | -0.43 | 2.0 | 20160101  00:36 | 200.0 | 2.6 | 0.0 | 92.0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | td | t | psl |
| 0 | 278.45 | 279.85 | NaN |
| 1 | 278.05 | 279.45 | NaN |
| 2 | 277.85 | 279.35 | NaN |
| 3 | 277.95 | 279.15 | NaN |
| 4 | 277.95 | 279.35 | NaN |
| 5 | 277.95 | 279.35 | NaN |
| 6 | 278.05 | 279.25 | NaN |

(Таблица 4 Часть набора данных)

number\_sta – number station, номер станции с которой происходило измерение

lat – latitude, широта в десятичных градусах

lon – longitude, долгота в десятичных градусах

height\_sta – height station, высота станции в метрах над уровнем моря

date – date, дата когда происходило измерение в формате 'ГГ-ММ-ДД ЧЧ: мм :сс'

dd – direction wind, направление ветра в градусах 0<=dd<360

ff – wind speed, скорость ветра в м.с^-1

precip – precipitation the reporting period, осадки за выбранный период в кг.м^2

hu – humidity, влажность в %

td – dew point, точка росы в кельвинах

t – temperature, температура в кельвинах

psl – pressure reduced to sea level, давление относительно уровня моря в паскалях

Не все данные полны, данный набор данных не исключение, в нем присутствуют записи с NaN (not a number).

В качестве языка программирования выбран язык python, потому что он максимально востребованный и используемый в машинном обучении. В качестве среды программирования будет использоваться бесплатная среда ‘kaggle’ из-за удобства в доступности набора данных. Бесплатная среда ‘kaggle’ имеет следующие характеристики: 16 гб оперативной памяти, 73.1 гб внутренней памяти и выполнение программы до 12часов.

Прогнозировать погоду, в данном случае температуру будем на какой- то конкретной станции используя записи с других станций, ближайших к нашей в округе. Для начала зафиксировали глобальные параметры с номером станции.

years = [2016,2017,2018]

zone = ‘NW’

around = 10

STATION = 14066001

mindist = 10

После чего нужно из нашего набора данных отобрать данные 10 ближайших станций включая нами выбранную для дальнейшей работы с этой информацией, используя библиотеку ‘pandas’. Этот процесс называется предобработка. Ближайшие станции к нашей можно отобрать, используя столбец ‘lat’ и ‘lon’.

import pandas as pd

fname = ‘/kaggle/input/meteonet/’+zone+’\_Ground\_Stations/’+zone+’\_Ground\_Stations/’+zone+’\_Ground\_Stations\_’+str(2016)+’.csv’

df = pd.read\_csv(fname)

new\_df = df.drop([‘height\_sta’,’date’,’dd’,’ff’,’precip’,’hu’,’td’,’t’,’psl’],axis=1).drop\_duplicates(‘number\_sta’).reset\_index()

lat = new\_df[new\_df.number\_sta==STATION].lat[0]

lon = new\_df[new\_df.number\_sta==STATION].lon[0]

neighbours = np.zeros(around)

neighbours[0] = STATION

for station in range(1,around):

for i in range(0,new\_df.shape[0]):

currdist = np.abs(lat-new\_df.loc[i].lat)+np.abs(lon-new\_df.loc[i].lon)

if(mindist>currdist):

mindist = currdist

index = i

neighbours[station] = new\_df.loc[index][‘number\_sta’]

mindist=10

del new\_df

del df

Так найдя индексы нужных нам станций можно выгрузить все данные нужных нам 10 станций заново прочитав csv файл

weather = pd.DataFrame()

for year in years:

fname = ‘/kaggle/input/meteonet/’+zone+’\_Ground\_Stations/’+zone+’\_Ground\_Stations/’+zone+’\_Ground\_Stations\_’+str(year)+’.csv’

df = pd.read\_csv(fname)

for i in range(0,around):

dataframe = df[df[‘number\_sta’]==int(neighbours[i])]

weather = weather.append(dataframe,ignore\_index=True)

del df

del dataframe

Размеры данного набора данных составляют 2575236 rows × 12 columns. Это данные 10 станций с промежутками в 6 минут каждой в течение 3 лет, но не у каждой станции есть все зафиксированные в таблице наблюдения, есть пропуски, и для их заполнения воспользуемся методом fillna(method = ‘bfill’) из библиотеки pandas (см. Рис. 6).



(Рис. 6 Станции соседи)

Данный метод заполняет пропуски значением последующего валидного значения в столбце. Для этого нужно перегруппировать набор данных по номерам станций после чего применить метод fillna(method = ‘bfill’). Но и таким образом все пропуски не заполнились, чтобы их заполнить до конца перегруппировали набор данных по ‘date’ и снова применив метод fillna(method=’bfill’).

weather.date = pd.to\_datetime(weather.date)

weather = weather.sort\_values(‘number\_sta’).reset\_index().drop(‘index,axis=1)

weather = weather.fillna(method=’bfill’)

dataset = weather.set\_index(‘date’)

dataset = dataset.fillna(method=’bfill’)

del weather

Чтобы нейронная сеть лучше прогнозировала результаты работы, нужно иметь в виду что данные должны быть нормализованы и для прогнозируемой переменной отбираются данные коррелирующие с ней. Иначе если существуют разные разбросы интервалов среди данных в наборе, нейронной сети придется делать больше вычислений (проходить эпох) чтобы хорошо лучше оперировать данными представленными в n мерном пространстве в виде разных столбцов таблицы. Зависимые данные же необходимы, чтобы выловить какую-то общую закономерность между прогнозируемым параметром и остальными столбцами, нет смысла давать нейронной сети независимые данные с прогнозируемым параметром, при обучении у независимых параметров будут обнуляться соответствующие веса нейронной сети потому что они не играют роль в прогнозе нужного нам параметра.

Нормализация осуществляется, например, методом ‘MinMaxScaler’ из библиотеки ‘sklearn.preprocessing’.

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(-1,1))

dataset[‘dd’]=scaler.fit\_transform(dataset[‘dd’].values.reshape(-1,1))

dataset[‘ff’]=scaler.fit\_transform(dataset[‘ff’].values.reshape(-1,1))

dataset[‘precip’]=scaler.fit\_transform(dataset[‘precip’].values.reshape(-1,1))

dataset[‘hu’]=scaler.fit\_transform(dataset[‘hu’].values.reshape(-1,1))

dataset[‘td’]=scaler.fit\_transform(dataset[‘td’].values.reshape(-1,1))

dataset[‘psl’]=scaler.fit\_transform(dataset[‘psl’].values.reshape(-1,1))

dataset[‘t’]=scaler.fit\_transform(dataset[‘t’].values.reshape(-1,1))

Чтобы понять какие параметры будем использовать для обучения нейронной сети, можно составить матрицу корреляций методом ‘corr()’ из библиотеки ‘pandas’ (Таблица 5).

dataset.reset\_index().drop([‘date’,’year’,’yday’,’hour’,’minute’,’number\_sta’],axis=1).corr()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | lat | lon | height\_sta | dd | ff | precip | hu |
| lat | 1 | 0.17 | -0.66 | -0.006 | 0.03 | 0.002 | -0.001 |
| lon | 0.17 | 1 | -0.03 | 0.003 | -0.07 | -0.003 | 0.01 |
| height\_sta | -0.66 | -0.03 | 1 | 0.007 | -0.07 | 0.006 | -0.0004 |
| dd | -0.006 | 0.003 | 0.007 | 1 | 0.12 | 0.02 | 0.06 |
| ff | 0.03 | -0.07 | -0.07 | 0.12 | 1 | 0.06 | -0.15 |
| precip | 0.002 | -0.003 | 0.006 | 0.02 | 0.06 | 1 | 0.07 |
| hu | -0.001 | 0.01 | -0.0004 | 0.06 | -0.15 | 0.07 | 1 |
| td | 0.009 | -0.001 | -0.03 | 0.1 | -0.11 | 0.01 | 0.07 |
| t | 0.02 | 0.002 | -0.04 | 0.04 | -0.06 | -0.02 | -0.37 |
| psl | -0.0003 | 0.002 | -0.005 | -0.07 | -0.3 | -0.1 | -0.04 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | td | t | psl |
| lat | 0.009 | 0.02 | -0.02 |
| lon | -0.001 | 0.002 | -0.02 |
| height\_sta | -0.03 | -0.04 | -0.08 |
| dd | 0.1 | 0.04 | -0.02 |
| ff | -0.11 | -0.06 | -0.004 |
| precip | 0.01 | -0.02 | -0.04 |
| hu | 0.07 | -0.37 | 0.01 |
| td | 1 | 0.87 | 0.003 |
| t | 0.87 | 1 | -0.003 |
| psl | 0.003 | -0.003 | 1 |

(Таблица 5 Матрица корреляций)

Исходя из данной таблицы и нескольких раз обучений на различных параметрах были выбраны 9 параметров участвующие в прогнозировании температуры на выбранной нами станции: lat, lon, dd, ff, precip, hu, td, t, psl. Теперь их нужно снова отобрать и представить в виде удобного для нейронной сети, чтобы сложить их в input layer. Опишем функции преобразования нужных нам столбцов набора данных с разделением на data и target и преобразованием их в ‘torch.Tensor’ и разделением на тестовую и проверочную части.

def make\_tensors(table,predicted\_par):

data=[]

target=[]

periods = table.index.unique().astype(‘str’)

label=table[table.number\_sta==STATION][predicted\_par]

target = label.values.tolist()

new\_table=table.loc[table[‘number\_sta’]!=STATION].drop([‘lat’,’lon’,’height\_sta’,’year’,’yday’,’hour’,’minute’,’number\_sta’],axis=1)

for period in periods:

if period in label:

data.append(new\_table[period:period].values.tolist())

return data,target

X,y = make\_tensors(dataset,’t’)

Здесь происходило создание data и target из нашего набора данных, путем отбора нужных нам значений из набора данных в data при условии, что на нужной нам станции была запись за конкретный промежуток времени столбца predicted\_par.

import numpy as np

columns = len(X[0][0])

X\_ = np.empty((0,(around-1)\*columns),dtype=’f’)

index = []

for i in range(len(X)):

if len([a for b in X[i] for a in b])==(around-1)\*columns and i!=len(X)-1:

X\_=np.append(X\_,[[a for b in X[i] for a in b]],axis=0)

else:

index.append(i)

index.pop(len(index)-1)

y=np.delete(y,index,None)

y=np.delete(y,len(y)-1,None)

Во втором преобразовании отбирались только те данные из X, где количество наблюдений за текущий временной момент соответствовало (around-1) \*columns, т.е. если в текущий промежуток времени есть полные данные по 9 станциям в 9 колонках.

В третьем преобразовании для наших target выделили прогнозируемый нами параметр в отдельный список, это соответствует температуре в каждый конкретный момент времени.

y = [[] for k in range(len(y))]

for i in range(len(y)):

y\_[i]=[y[i]]

y=y\_

Теперь, когда данные для обучения готовы разделяем их на проверочную и тестовую часть, чтобы приступить к обучению, размеры этих данных уменьшились по сравнению с изначальным набором данных, потому что не за каждый момент времени были полные данные на нужных нам around станциях.

X\_train = X\_[0:int(len(X\_)\*0.9)]

X\_test = X\_[int(len(X\_)\*0.9):]

y\_train = y[0:int(len(y)\*0.9)]

y\_test = y[int(len(y)\*0.9):]

**2.2 Обучение нейронной сети**

Как говорилось в главе 1.5 моделью нейронной сети будет являться многослойный персептрон, но, чтобы понять какая архитектура лучше с какими параметрами подготовим 8 различных нейронной сети и посмотрим на их результаты, решая пример прогнозирования температуры на выбранной станции в текущий момент, исходя из данных в текущий момент на станциях соседях.

Заимствуя из библиотеки ‘torch’ сами модели и функции активации нам остается только переопределить класс с его функциями, а также задать условия к обучению и перевести наши данные в формат ‘torch.Tensor’.

import torch

import torch.nn as nn

import torch.nn.functional as F

X\_train=torch.from\_numpy(X\_train).type(torch.Tensor)

X\_test=torch.from\_numpy(X\_test).type(torch.Tensor)

y\_train=torch.from\_numpy(np.array(y\_train)).type(torch.Tensor)

y\_test=torch.from\_numpy(np.array(y\_test)).type(torch.Tensor)

class MLP(nn.Module):

def \_\_init\_\_(self,input\_dim,output\_dim):

super().\_\_init\_\_()

self.input\_fc=nn.Linear(input\_dim,40)

self.hidden\_fc=nn.Linear(40,20)

self.output\_fc=nn.Linear(20,output\_dim)

def forward(self,x):

batch\_size=x.shape[0]

x=x.view(batch\_size,-1)

h\_1=torch.tanh(self.input\_fc(x))

h\_2=torch.tanh(self.hidden\_fc(h\_1))

y\_pred=self.output\_fc(h\_2)

return y\_pred

Таким образом видно, что создалась нейронная сеть с 1 скрытым слоем, а функция активации является ‘tanh’.

import time

hist=np.zeros(num\_epochs)

start\_time=time.time()

model=MLP(input\_dim=X\_train[0].shape[0],output\_dim=len(y\_train[0]))

criterion=torch.nn.MSELoss(reduction=’mean’)

optimiser=torch.optim.Adam(model.parameters(),lr=0.03)

num\_epochs=100

for epoch in range(num\_epochs)

y\_train\_pred=model(X\_train)

loss=criterion(y\_train\_pred,y\_train)

print(“Epoch “,epoch,”MSE: “,loss.item(),’Time: “,time.time()-start\_time)

hist[epoch]=loss.item()

optimiser.zero\_grad()

loss.backward()

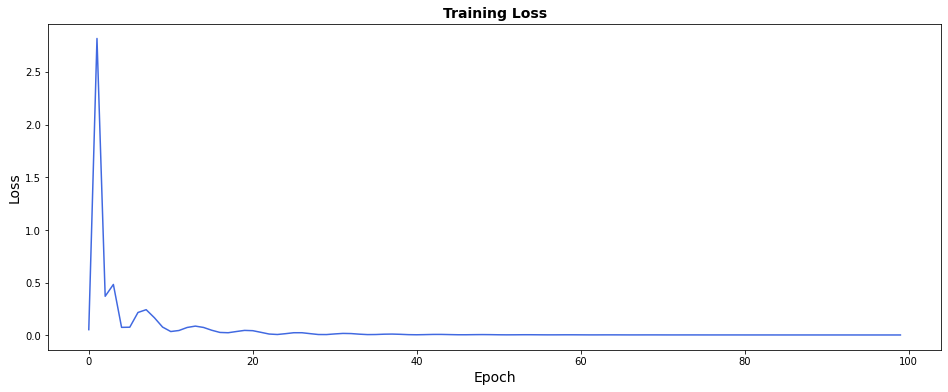
optimiser.step()

training\_time=time.time()-start\_time

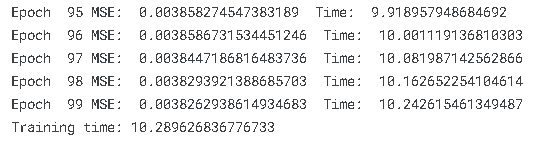
print(“Training\_time : {}”.format(training\_time))

Здесь описана инициализация нейронной сети с нашими параметрами набора данных, с функцией ошибки ‘MSE’ и оптимазейром ‘Adam’ с ‘learning rate ‘=0.03 и количеством эпох =100. В течении каждой эпохи вычисляется ошибка ‘loss’ между прогнозируемыми величинами из ‘X\_train’, на основе которой оптимизируются веса с использованием градиента на каждом слое нейронной сети, тем самым изменяя веса нейронной сети на каждом слое.

Для данного примера получилось, что функция ошибки с течением увеличения количества эпох уменьшается, что свидетельствует о процессе обучения нейронной сети (подбора нужных весов) (Рис. 7).

(Рис. 7 Уменьшение функции ошибки)

Причем ошибка на тренировочном наборе данных на этом примере составила (Рис.8)



(Рис. 8 Ошибка)

Менее 1%, а именно 0.38%. Теперь нужно проверить теперь как себя ведет нейронная сеть на тестовых данных.

y\_test\_pred=model(X\_test)

y\_test\_predict=scaler.inverse\_transform(y\_test\_pred.detach().numpy().reshape(-1,1))

y\_test\_target=scaler.inverse\_transform(y\_test.detach().numpy().reshape(-1,1))

y\_test\_predict=pd.DataFrame(y\_test\_predict)

y\_test\_target=pd.DataFrame(y\_test\_target)

После чего можно построить график (рис. 9) используя библиотеку ‘seaborn’ и ‘matplotlib.pyplot’.

import matplotlib.pyplot as plt

import seaborn as sns

fig=plt.figure()

ax=sns.lineplot(x=y\_test\_target.index,y=y\_test\_target[0],label=”Data”,color=’tomato’)

ax=sns.lineplot(x=y\_test\_predict.index,y=y\_test\_predict[0],label=”Prediction”,color=’royalblue’)

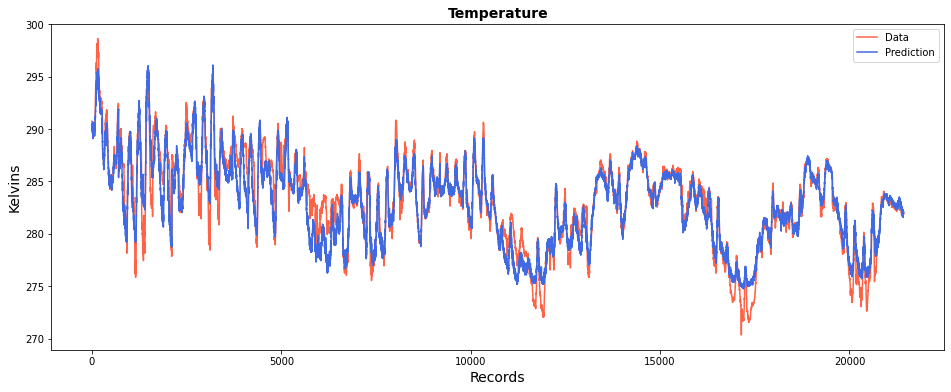
ax.set\_title(‘Temperature’,size=14,fontweight=’bold’)

ax.set\_xlabel(“Records”,size=14)

ax.set\_ylabel(“Kelvins”,size=14)

fig.set\_figheight(6)

fig.set\_figwidth(16)

(Рис. 9 Прогноз)

На данном примере прогноз нейронной сети близок к настоящим данным. Для оценки качества модели воспользуемся RMSE и MAPE из библиотеки ‘sklearn.metrics’ (рис. 10).

import math

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_percentage\_error

testScore=math.sqrt(mean\_squared\_error(y\_test\_target,y\_test\_predict))

print(‘Test Score: %.2f RMSE’ %(testScore))

testScore=mean\_absolute\_percentage\_error(y\_test\_target,y\_test\_predict)

print(‘Test Score: %.2f MAPE’ % (testScore))

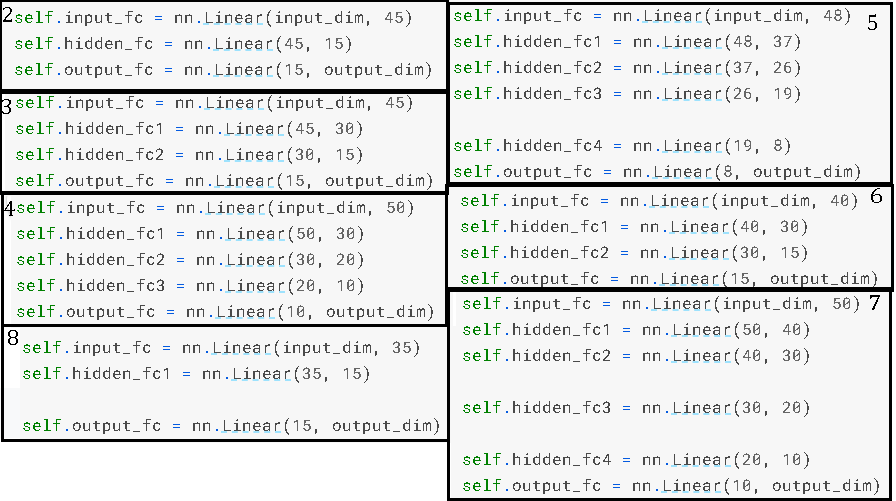


(Рис. 10 Метрики)

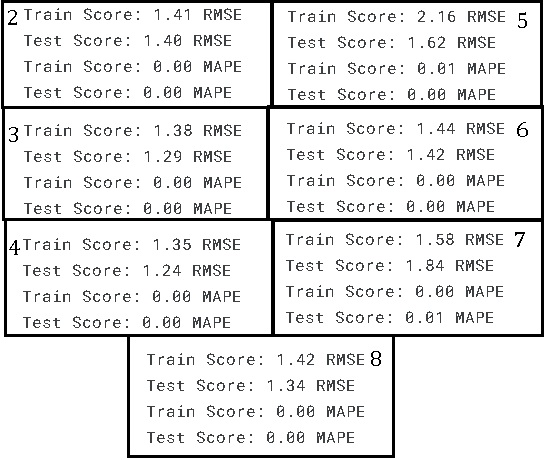
Модель обучается и дает неплохие результаты, чтобы их улучшить возьмем 7 моделей с различными архитектурами и параметрами в них, после чего выберем лучшую модель по оценкам метрик.

**2.3 Результаты обучения модели**

Чтобы подобрать лучшую архитектуру были созданы 8 моделей включая описанную выше. Они обучались на том же наборе данных что и представленная выше модель. Они имеют следующую архитектуру (Рис.11).

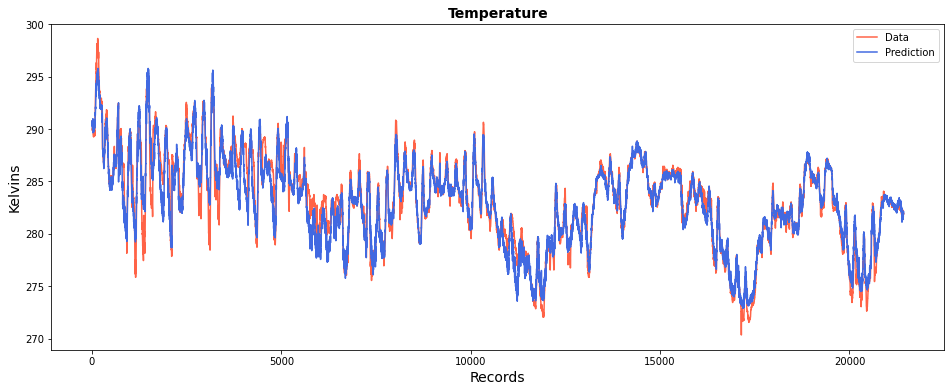
(Рис. 11 Архитектуры нейронной сети)

Функции активации и порядок обучения этих нейронной сети был такой же, как и у модели 1. Но эти модели дали другие результаты тех же метрик (Рис. 12)

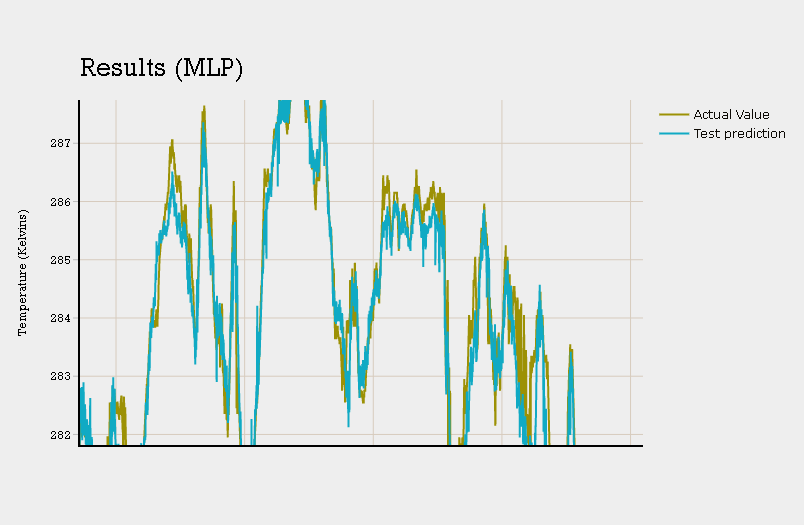


(Рис.12 Результаты различных моделей)

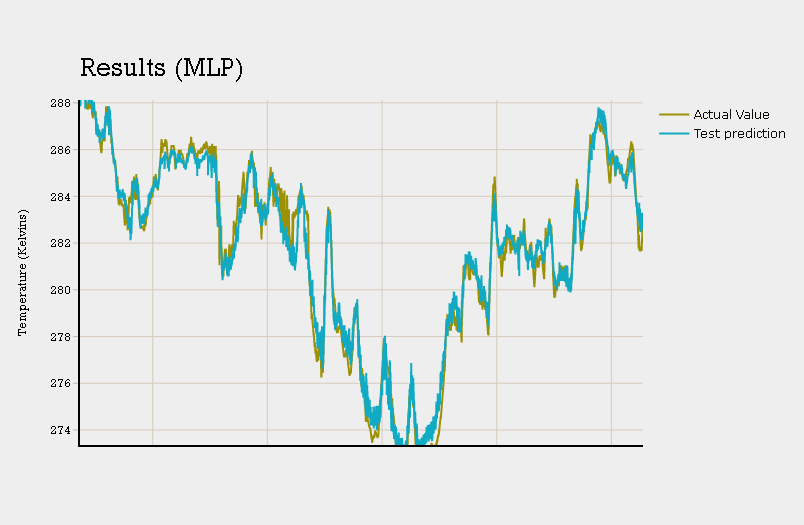
Исходя из значений оценок метрик лучшая модель среди представленных оказалась четвертая (см. Рис 17). Её прогноз почти совпадает с тестовыми данными (Рис. 13).

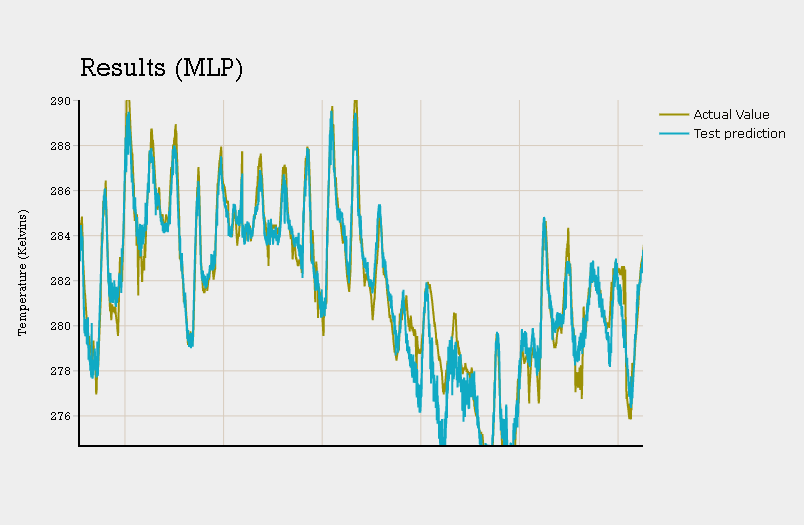
(Рис. 13 Прогноз 4ой модели)

Детальнее (Рис. 14 и Рис.15 и Рис.16):



(Рис. 14 Прогноз 4ой модели 2)

(Рис. 15 Прогноз 4ой модели 3)

(Рис. 16 Прогноз 4ой модели 4)



(Рис. 17 Метрики 4ой модели)

Её прогноз совпадает с реальными значениями, максимальное расхождение до 2 кельвинов.

***Использование обученной модели***

Используем обученную модель для предсказания температуры на другой станции, не участвовавшей в обучении модели:

STATION=36093002

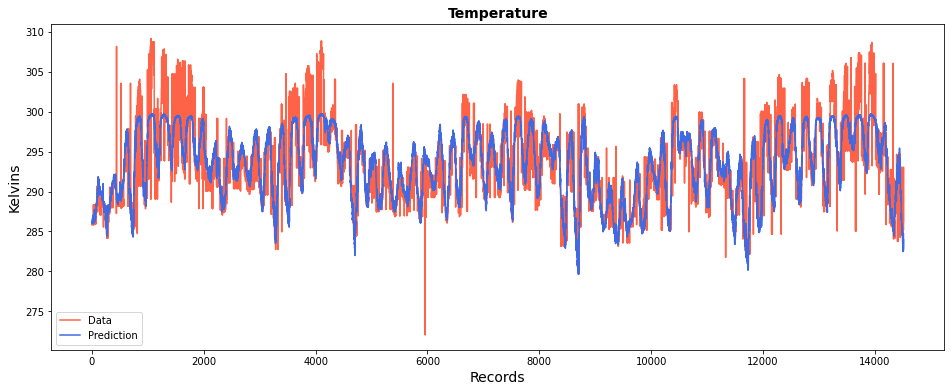
around=10

С временным промежутком:

dataset = weather[weather[‘date’]<’2017-09-01’]

dataset = weather[dataset[‘date’]>’2017-07-01’]

В результате получаем (см. Рис 18):



(Рис. 18 Прогноз на другой станции обученной моделью)

Качество прогноза существенно поменялось. Возможно это следует из-за переобученной нейронной сети так как заметна верхняя граница предсказания в 300 кельвинов.

***Прогнозирование температуры на 30 минут***

Обучим модель на смещенных данных по температуре, то есть, со смещением в 30минут.

STATION = 14066001

around = 15

timepredict = (around-1)\*5

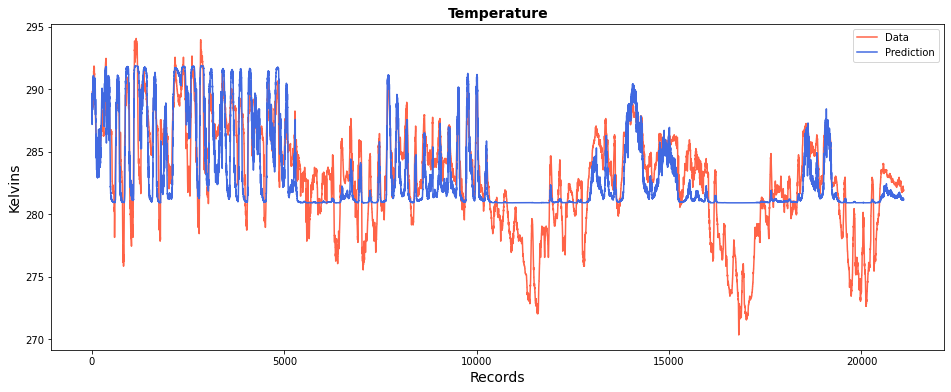
X\_train=X\_[0:int(len(X\_)\*0.9) ]

X\_test=X\_[int(len(X\_)\*0.9):len(X\_)-int(timepredict/(around-1))]

y\_train=y[int(timepredict/(around-1)):int(len(y)\*0.9)+int(timepredict/(around-1))]

y\_test=y[int(len(y)\*0.9)+int(timepredict/(around-1)):]

В результате имеем (см. Рис. 19):



(Рис. 19 Прогноз на 30 минут)

Здесь тоже качество прогноза плохое, четко видна нижняя граница в ~282 кельвина, что наводит на мысль о том, что нейронная сеть переобучилась.

**Заключение**

В рамках этой работы был написан алгоритм прогнозирования метеопараметров погоды, на примере прогнозирования температуры воздуха. С помощью его можно предсказывать метеопараметры, основываясь на данных с соседних станций, как на текущий момент, так и на определенное время вперед, если имеется соответствующий набор данных.

Программа была протестирована на моделях с разными архитектурами для выявления лучшей, и представлены результаты вариаций прогнозирования температуры воздуха на ней.

В дальнейшем можно реализовать обучение на других типах моделей, таких как LSTM или использовать данные не только с метеостанций, но и данные в виде картинок со спутников.

**Список литературы**

1. Гидрометцентр России [Электронный ресурс]/ Откуда возникают ошибки прогнозов? Режим доступа: https://meteoinfo.ru/faq, свободный. Загл. с экрана.
2. meteonovosti.ru [Электронный ресурс]/ Трудности долгосрочного прогнозирования. Режим доступа: <http://www.hmn.ru/index.php?index=14&value=7>, свободный. Загл. с экрана.
3. Национальная ассоциация нефтегазового сервиса [Электронный ресурс] / Анастастия Н. Число природных катастроф выросла в 2 раза за 20 лет, и это не предел. Режим доступа: <https://nangs.org/news/ecology/oon-chislo-prirodnyh-katastrof-vyroslo-v-dva-raza-za-20-let-i-eto-ne-predel>, свободный. Загл. с экрана.
4. Святский Д. О., Кладо Т. Н. Занимательная метеорология. – Общество с ограниченной ответственностью Издательство ЮРАЙТ, 2020. – С. 212-212.
5. Персональная страница Андрея Братцева [Электронный ресурс] / Андрей Б. Учение об атмосфере. Введение. Режим доступа: <http://abratsev.ru/atmosphere/intro.html>, свободный. Загл. с экрана.
6. SAS Аналитические решения и ПО [Электронный ресурс]/ Катрина Уэйкфилд: Алгоритмы машинного обучения и их типы. Режим Доступа: <https://www.sas.com/ru_ru/insights/articles/analytics/machine-learning-algorithms-guide.html#/> ,свободный. Загл. с экрана.
7. Шалев-Шварц Ш., Бен-Давид Ш. Идеи машинного обучения. – 2019.
8. Чайка К. В., Шестопалов Р. П. Оценивание качества обучения нейросетевых алгоритмов обработки информации //Информационно-экономические аспекты стандартизации и технического регулирования. – 2021. – №. 2 (60). – С. 17.
9. Сайт разработчика Дениса Баженова [Электронный ресурс]/ Оценка классификатора (точность, полнота, F-мера). Режим доступа: <http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html>, свободный. Загл. с экрана.
10. Сайт школы bigdataschool [Электронный ресурс] /Анна Вичугова: Как сформировать датасет для машинного обучения. Режим доступа: <https://www.bigdataschool.ru/blog/dataset-data-preparation.html>, свободный. Загл. c экрана.
11. Github.com: среда для разработчиков [Электронный ресурс]. Репозиторий: <https://github.com/taerd/DSIntro/tree/master/dropout_course>, свободный. Загл. с экрана.
12. Loginom.ru : Аналитическая платформа [Электронный ресурс] / Oleg Glushko Обработка пропусков в данных. Режим доступа: <https://loginom.ru/blog/missing>, свободный. Загл. с экрана.
13. Шестопалов Р. П. О выборе предпочтительного алгоритма подстройки весов при обучении нейросетевых алгоритмов обработки информации //Информационно-экономические аспекты стандартизации и технического регулирования. – 2021. – №. 2. – С. 60.
14. Романов Д. Е. Нейронные сети обратного распространения ошибки //Инженерный вестник дона. – 2009. – Т. 9. – №. 3. – С. 19-24.
15. Kaggle.com : среда для датасаентистов [Электронный ресурс]/ Открытый набор данных: <https://www.kaggle.com/datasets/katerpillar/meteonet>, свободный. Загл. с экрана.