ЗМІСТ

РЕФЕРАТ ..... Error! Bookmark not defined.
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ **3**
ВСТУП **4**
1 ПРОГНОЗУВАННЯ ГЕНЕРАЦІЇ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНИМИ СТАНЦІЯМИ **7**
1.1 МЕТОДИ ПРОГНОЗУВАННЯ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНИМИ СТАНЦІЙМИ **8**
1.2 ТИПОВІ МОДЕЛІ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ПРОГНОЗУВАННЯ **10**
1.2.1 НЕЙРОННІ МЕРЕЖІ ПРЯМОГО ПОШИРЕННЯ **10**
1.2.2 РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ **12**
1.3 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ **14**
2 МАТЕМАТИЧНА МОДЕЛЬ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ПРЯМОГО РОЗПОВСЮДЖЕННЯ **15**
2.1 МАТЕМАТИЧНА МОДЕЛЬ ШТУЧНОГО НЕЙРОНУ **15**
2.2 МЕТОДИ МАШИННОГО НАВЧАННЯ **18**
2.3 ВИСНОВКИ ЗА РОЗДІЛОМ **22**
3 РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ **24**
3.1 РОЗРОБКА ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ **26**
3.2 РОЗРОБКА МОДУЛЯ ДИНАМІЧНОГО ЗАВАНТАЖЕННЯ ДАНИХ **28**
3.3 ВИСНОВКИ ЗА РОЗДІЛОМ **30**
4 ПІДГОТОВКА ДАНИХ ДЛЯ НАВЧАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ **31**
4.3 АНАЛІЗ ДАНИХ **32**
4.4 ВИЗНАЧЕННЯ КРИТЕРІЇВ ФОРМАЛІЗАЦІЇ ДАНИХ **37**
4.5 РОЗРОБКА МОДУЛЯ ОБРОБКИ ДАНИХ **39**
4.6 ВИСНОВКИ ЗА РОЗДІЛОМ **43**
5 АНАЛІЗ РЕЗУЛЬТАТІВ НАВЧАННЯ **43**
5.3 ВИСНОВКИ ЗА РОЗДІЛОМ **45**
6 РОЗРОБКА СТАРТАП ПРОЕКТУ **46**
6.3 АНАЛІЗ ЗОВНІШНЬОГО ТА ВНУТРІШНЬОГО СЕРЕДОВИЩА СТАРТАПУ **50**
6.4 КЛЮЧОВІ ФАКТОРИ УСПІХУ СТАРТАПУ ЗА МЕТОДОМ ШОНФІЛДА **51**
6.5 РОЗРАХУНОК ОСНОВНИХ ТЕХНІКО-ЕКОНОМІЧНИХ ПОКАЗНИКІВ ПРОЕКТУ **52**
6.6 КАРТА БІЗНЕС-ПРОЦЕСІВ РЕАЛІЗАЦІЇ ПРОЕКТУ **54**
ВИСНОВКИ **56**
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ **58**
ДОДАТОК А **61**
ДОДАТОК Б **66**

ДОДАТОК В 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ШНМ – штучна нейронна мережа

ВС – вітряна станція

ВП – вітряний парк

ЕОМ – електронна обчислювальна машина

BNEF – Bloomberg New Energy Finance

NWP – numeric weather prediction

NN – neural network

ШІ – штучний інтелект

ШН – штучний нейрон

ML – machine learning

ВЕС – вітряна електростанція

ПЗ – програмне забезбепечення

SD - similar days (method)

РНМ – рекурентна нейронна мережа

ВСТУП

Частка відновлювальних джерел енергії у світовому енерговиробництві попри недосконаліть існуючих засобів невпинно росте. Так, за даними агентства Bloomberg New Energy Finance (BNEF) до 2040 року очікується зростання світового споживання електроенергії на 58% (що відповідає 2% середньомісячного зростання). За цими ж даними очікується зростання частки ринку «зеленої» енергії – до 48% встановленої потужності світових електростанцій і 34% обсягів випущеної електроенергії. Крім того, енергія вітру у 2018 році забезпечила 11,6% від загальної кількості енергоспоживання Європейського Союзу, а енергозабезпечення Данії взагалі є залежним від вітряних станцій на 43% (за даними 2017 року).

На сьогодні, сонячні та вітряні електростанції займають 12% встановленої потужності світової енергосистеми, і 5% - генерації, а зростання потужності вітряних електростанцій очікується в 4 рази до 2040 року. Вартість виробництва електроенергії до 2040 року скоротиться в материковій вітроенергетиці — на 47%, а в офшорній вітроенергетиці — на 71%. Проте, у сучасній вітроенергетиці існує 2 проблеми, що сповільнюють її поширення та потребують вирішення:

- вартість вітряків та комплектуючих;
- висока залежність продуктивності від погодних умов;

Проблему зменшення залежності від погодних умов можна вирішити шляхом додавання промислових акумуляторів у мережу, проте це значно збільшує сумарну вартість вітряного парку. Іншим варіантом вирішення даних проблем є впровадження інноваційних систем прогнозування електрогенерації, що дозволить застосовувати розумні алгоритми акумуляції виробленої електроенергії та покращити рівень стабільності електропостачання [1].

Існує декілька технік прогнозування електрогенерації:

- числові методи передбачення (Numeric weather prediction (NWP));

- статистичні методи передбачення;
- методи засновані на технології нейронних мереж (Neural Networks (NN) methods);
- комбінації методів описаних вище;

З розвитком комп'ютерних технологій стрімкої популярності набирають методи на базі машинного навчання із застосуванням нейронних мереж, а також комбіновані методи [2].

Дослідження процесу передбачення електрогенерації вітряними станціями із застосуванням методів машинного навчання є головною задачею магістерської дисертації.

Об'єктом дослідження є системи прогнозування електрогенерації на базі ШНМ прямого розповсюдження для навчання на статистичних та метеорологічних даних з турецької вітряної станції у місті Ялова.

Ціль роботи – розробка програмного забезпечення на базі ШНМ для передбачення генерації електроенергії вітряною станцією.

Результатом роботи є програмне забезпечення (реалізована штучна нейронна мережа), яке можна використовувати для прогнозування сумарної кількості електроенергії що буде вироблена вітряною станцією протягом майбутніх 8 годин з точністю 80-85%.

Актуальність роботи – разом із швидким поширенням та розвитком вітряної енергетики критично важливим є можливість її безпечного підключення у електромережу. При досягненні частки ВС у 25-30% від загальної потужності електромережі, виникає ризик дестабілізації мережі. Як наслідок потрібно застосовувати техніки акумуляції та прогнозування електроенергії на короткострокові терміни (1-3 доби). Крім того, в країнах Європи все більшої популярності набуває ринок електроенергії. Такий підхід дозволяє децентралізувати сферу енергетики, проте вимагає укладення угод між гравцями ринку щодо кількості енергопостачання та енергоспоживання на «добу наперед». Це дозволяє зберегти баланс між виробництвом та споживанням електроенергії. Ринок електроенергії запрацював в Україні 1

липня 2019 року. Високоточне прогнозування електропостачання ВС є життєвоважливим для гравців цього ринку, що використовують енергію вітру як джерело електроенергії.

За результатами даної магістерської роботи було опубліковано наступні статті:

1. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ЯК НЕОБХІДНИЙ ІНСТРУМЕНТ ДЛЯ ЕФЕКТИВНОГО ВИКОРИСТАННЯ БАЗ ДАНИХ СИСМЕТ SCADA.

Topical issues of the development of modern science. Abstracts of the 3rd International scientific and practical conference. Publishing House “ACCENT”. Sofia, Bulgaria. 13-15 November 2019. Pp. 422-426.

2. Бугаєва Л. М., Сидоренко І. А. СИСТЕМА ПРОГНОЗУВАННЯ ГЕНЕРАЦІЇ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНОЮ СТАНЦІЄЮ ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ.

Матеріали V Міжнародної науково-технічної конференції «Комп'ютерне моделювання та оптимізація складних систем», м. Дніпро, 6-8 листопада 2019 року. С. 111-113.

3. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ЗАСТОСУВАННЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДО БАЗ ДАНИХ SCADA СИСТЕМ.

Всеукраїнська наукова Інтернет-конференція " Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення", 14 листопада 2019 р, Тернопіль. С.19-21

1 ПРОГНОЗУВАННЯ ГЕНЕРАЦІЇ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНИМИ СТАНЦІЯМИ

Однією з негативних властивостей вітряної енергетики є нестабільність електропостачання через високу залежність від погодних умов. Є декілька шляхів мінімізації цього фактору. Найголовнішим критерієм є регіон розміщення вітряного парку та вітряків у парку між собою. Зони для розміщенні вітряних парків можна поділити на 2 види:

- материкова зона (пологі рівнини з постійним вітрозабезбеченням, висоти, прибережні морські та океанічні території);
- нематерикова зона (вітряні турбіни розміщуються у морях або океанах на висоті 18-25 метрів над рівнем моря).

Перед початком побудови вітряного парку спочатку досліджується вітряна активність в регіоні, розраховується теоретичний середньорічний ККД вітряного парку. Далі визначається положення турбін у просторі відносно одна одної з метою захоплення вітру будь-якого напрямку та мінімізації впливу зміни вітряного потоку турбіною на сусідні станції.

При підключенні турбін у мережу використовуються перетворювачі напруги та частоти для досягнення промислових характеристик струму. Також використовується акумулюючий шар з метою поглинання нестабільності електропостачання. Проте ці методи дають тільки наближені значення можливого очікуваного електропостачання та переважно застосовуються для передбачення генерації електроенергії на довго- та середньострокові проміжки часу (середньорічна, сезонна потужність ВС).

Під короткостроковим прогнозуванням зазвичай розуміють кількісне значення електричної потужності певної енергосистеми з горизонтом упередження 1-48 годин.

Для прогнозування електрогенерації на короткострокові проміжки часу застосовуються програмні засоби прогнозування. Такі засоби можна класифікувати за методами які вони використовують.

1.1 МЕТОДИ ПРОГНОЗУВАННЯ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНИМИ СТАНЦІЙМИ

Існує декілька методів прогнозування з різною точністю та періодом передбачення: числові, статистичні та методи зі застосуванням засобів машинного навчання.

Числові методів передбачення (Numeric weather prediction (NWP)) використовують фізичні поняття, такі як тиск, температура, місцевість для подальшого прогнозування швидкості вітру. Відповідно до таких методів будуються математичні моделі стану атмосфери та океанів для прогнозування погоди великомасштабного регіону. Такі математичні моделі будуються на основі фізичних законів та дозволяють передбачати як короткострокові зміни в погоді, так і довгострокові кліматичні зміни. Для представлення топографії будуються цифрові тривимірні моделі для конкретного регіону.

На сьогодні ці методи швидше застосовуються на стадії проектування вітряного парку для визначення кліматичних умов регіону, аніж для передбачення генерації електроенергії існуючим вітряним парком на короткострокові проміжки часу. За результатами різних досліджень визначено, що середня похибка прогнозу становить від 10 до 20 % потужності досліджуваної ВЕС.

Статистичні методи базуються на інформації, яку можна одержати, знаючи тенденції зміни потужності ВС базуючись на попередніх числових значеннях або маючи статистично достовірні залежності потужності турбіни від внутрішніх параметрів стану. Для цього використовуються такі методи,як:

- Аналіз числових рядів.
- Казуальні методи прогнозування.
    - Багатовимірні регресійні моделі.
    - Економетричні моделі.
    - Комп’ютерна імітація.

Проте статистичні методи мають певні недіолки, наприклад, вони не враховують механізми, що визнчають прогнозовані дані; незахищені від випадкових коливань; невраховують сезонні коливання і тенденції. Зазвичай статистичні методи комібнують з чисельними та/або матодами машинного навчання.

Методи засновані на технології штучних нейронних мереж зазвичай націлені на короткострокове прогнозування. В основі таких методів лежать штучні нейронні мережі (ШНМ), моделі нечіткої логіки, направляючих векторів та ін. Ці моделі базуються на використанні історичних даних з вітряних станцій та метеорологічних даних.

Значної популярності здобув метод прогнозування Similar Days (SD) method (метод «Схожих днів»), який використовує ШНМ для прогнозування потужності вітряних станцій на базі історичних даних потужності у дні зі схожими погодними умовами. За результатами різних досліджень визначено, що середня похибка прогнозу становить від 5 до 15 %. Проте значним недоліком даного методу є необхідність великої кількості навчальних даних (інтервалом у 3-4 роки з частотою зняття показань від 10 до 30 хв) для можливості точного виявлення великої множини статистичних даних відповідних вхідним критеріям [2].

В магістерській дисертації для прогнозування генерації електроенергії вітряною станцією застосовується метод із використанням нейронних мереж прямого розповсюдження навчених на історичних даних ВЕС та метеорологічних даних регіону.

1.2 ТИПОВІ МОДЕЛІ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ПРОГНОЗУВАННЯ

На сьогодні існує 2 типи нейронних мереж які найкраще підходять для вирішення задач прогнозування:

- Нейронні мережі прямого поширення.
- Рекурентні нейронні мережі.

Існують альтернативні популярні топології нейронних мереж, такі як автоенкодер, машина Больцмана та мережа Хопфілда, проте вони застосовуються для іншого роду завдань, такого як розпізнавання зображень, музики і т.д.

На рис. 1.1-1.2 наведені графічні моделі даних нейронних мереж, проте більш детально на них зупинятись не є доцільним.

Рисунок 1.1 - Популярні топології нейронних мереж: 1- мережа Хопфілда, 2 – машина Больцмана

Рисунок 1.2 – Топологія автоенкодера

Мережі Хопфілда та Больцмана мають зв’язок «усього з усім». Такі типи мереж називають повністю зв’язаними [3].

1.2.1 НЕЙРОННІ МЕРЕЖІ ПРЯМОГО ПОШИРЕННЯ

Особливістю нейронних мереж прямого поширення (Feedforward) є однонаправлений зв’язок нейронів.

Рисунок 1.3 – Багатошаровий перцептрон

Принцип побудови багатошарового перцептрона базується на утворенні шарів – групи нейронів що не зв’язаних між собою. Виходи нейронів n-го шару є входами для нейронів n+1 шару. Кількість входів у штучний нейрон відповідає кількості нейронів попереднього шару, а кількість виходів – кількості наступного. Існує 3 типи шарів:

- Вхідний.
- Проміжні (приховані).
- Вихідний.

Розмір вхідного шару відповідає кількості параметрів на вході у ШНМ. Розмір вихідного шару залежить від класу задачі яка поставлена штучній нейронній мережі. Для прогнозування - це зазвичай 1 нейрон, виходом якого є інформація прогнозу; для класифкації – кількість можливих значень; розпізнавання – 1 логічний вихідний нейрон.

Модель багатошарового перцептрона побудована таким чином, що на вхід початкового шару передаються вхідні параметри (плейсхолдери), а виходи зв’язані з першим прихованим шаром. Проміжні шари зв’язані між собою, а останній прихований шар зв’язаний з вихідним. Дані на виході з останнього шару є рішенням прийнятим штучною нейронною мережею.

Кожен нейрон має свою функцію активації, параметрами якої є входи, а результатом – значення на виході з нейрону (рис 2.3) [4,5,6].

Рисунок 1.4 - Модель ШНМ прямого розповсюдження

В ШНМ прямого розповсюдженння існую поняття нейрону «зміщення» - додаткової складової кожного шару. Це дозволяє коригувати значення на виході з функції активації. Нейрон зміщення має вплив на ступінь активації всіх нейронів свого шару. Необхідність у цьому виникає через умову обов’язкової трансформацій даних перед «згодовуванням» їх мережі та через необхідність зберігання чисельного порядку даних між входами та виходами, а також між проміжними шарами ШНМ.

Рисунок 1.5 - Модель ШНМ прямого розповсюдження із застосуванням нейрону зміщення на шарах

ШНМ які мають 2 і більше прихованих шари називаються глибинними нейронними мережами, а навчання таких мереж називають глибинним навчанням (Deep learning) [7,8].

1.2.2 РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ

Особливість рекурентних нейронних мереж (РНМ), в порівнянні з мережами прямого поширення заключається в наявності зворотніх зв'язків. За рахунок цього стан нейрона у РНМ в певний момент часу може впливати на його стан в майбутньому. Деякі типи таких мереж уможливлюють зв'язок нейронів самого з собою, інші – тільки опосередковані зв'язки (рис. 1.6).

Рисунок 1.6 – Архітектура рекурентної нейронної мережі

Рекурентні мережі не завжди мають явно визначені вхідні та вихідні нейрони.

Наявність у РНМ зворотніх зв'язків створює іммітацію короткотривалої «пам'яті». Тобто значення на виході мережі частково залежить від певної кількості попередніх станів. Такі мережі гарно справляються з задачами прогнозування, проте мають обмежену кількість «запам'ятовування» попередніх значень. Технічно реалізувати пам'ять більше ніж на 3-4 попереді стани для класичних рекурентних мереж неможливо. Цю проблему вирішує

модифіковона РНМ – мережа довгої короткотривалої пам’яті або Long Short-Term Memory Neureal Network (LSTM). Модель LSTM-мережі зображена на рис. 1.7.

Рисунок 1.7 – Модель LSTM-мережі

LSTM-мережі відрізняються від звичайних рекурентних мереж наявністю довгострокової пам’яті, тобто стан нейрону LSTM-мережі залежить не тільки від вхідних параметрів та стану нейрону на попередній ітерації, а й від станів нейрону на і – n ітерацій, де і – номер поточної ітерації, а змінна n знаходиться у межах $1 < n < K$. Змінну K називають коефіцієнтом тривалості пам’яті і він залежить від налаштування нейрону (так званої функції «гейту» - пропускної функції).

Математична модель нейрону LSTM-мережі зображена на рис. 1.8- 1.9 та формулах 1.1.

Рисунок 1.8 – Функції втрати (зліва) та збереження (справа) LSTM-нейрону

Рисунок 1.9 – Функції збереження нового стану (зліва) та виходу (справа) LSTM-нейрону

Математична модель LSTM-нейрону виглядає наступним чином:

$$\begin{cases} f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ C'_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t = f_t * C_{t-1} + i_t * C'_t \\ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t = o_t * \tanh(C_t) \end{cases} \quad (1.1)$$

LSTM-мережі часто застосовуються у прогнозуванні тексту і є одними з найбільш інноваційних ШНМ, проте мають високу складність у налаштуванні та навчанні на даних з великою кількістю факторів. Крім того, прогнозування генерації електроенергії вітряною станцією на термін від кількох годин потребує аналізу такої кількості станів турбіни і метеопрогнозів, котра значно перевищує допустимо можливу межу пам'яті LSTM-мереж [9,10].

Тому, зважаючи на ці фактори, і широку розповсюдженість багатошарового перцептрону для вирішення подібного класу задач, було обрано саме цю нейронну мережу для прогнозування генерації електроенергії вітряними станціями.

1.3 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Було проаналізовано та досліджено процес прогнозування генерації електроенергії вітряними станціями, розглянуто типові методи прогнозування, умови їх роботи, основні переваги та недоліки, межі точності прогнозування, типові моделі нейронних мереж для задач прогнозування.

На основі проведеного аналізу було зроблено висновки і поставлено наступний перелік задач:

- Дослідження існуючих систем прогнозування електроенергії вітряними станціями;
- Скласти та описати математичну модель нейрону для ШНМ прямого розповсюдження;
- Обрати та описати метод машинного навчання для цього типу мереж;

- Розрахувати параметри моделі мережі;
- Проаналізувати та обробити дані, використовувані для навчання багатошарового перцептрону;
- Побудувати нейронну мережу прямого розповсюдження згідно визначеним параметрам;
- Розробти програмне забезпечення для навчання ШНМ;
- Верефікувати роботу програмного комплексу на тестових даних;
- Проаналізувати отримані результати;
- Розробити та оформити стартап проект.

2 МАТЕМАТИЧНА МОДЕЛЬ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ПРЯМОГО РОЗПОВСЮДЖЕННЯ

Штучно нейронною мережею називають математичну модель імітації структури та функціональних можливостей біологічних нейронних мереж. Основним будівельним блоком типової ШНМ є штучний нейрон - проста математична модель.

2.1 МАТЕМАТИЧНА МОДЕЛЬ ШТУЧНОГО НЕЙРОНУ

Така модель містить в собі три правила: сумування, множення та активація. Типовий нейрон має декілька входів та один або більше виходів. На вхід подаються фактори - величини що впливають на стан нейрону. Кожен фактор помножується на вагу зв’язку. В нейронних мережах інформація (знання) зберігається у вагах, а процес навчання зводиться до оптимізації вагів відповідно поставленим вимогам задачі. Вага характеризує силу зв’язку між неронами – як значення виходу нейрону попереднього шару впливає на вхід наступного.

Всередині нейрону значення з входів сумуються між собою та передаються на функцію активації.

Математичну модель нейрону можна записати у наступному вигляді:

$$y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n + z), \quad (2.1)$$

де y – значення на виході з нейрону , f – функція активації, w_i – вага і-го зв'язку, i = 1 .. n, n – кількість входів, x_i –значення і-го входу, z – значення на вході нейрону суміщення [11].

Графічно математичну модель нейрону можна представити у вигляді рис. 2.1.

Рисунок 2.1 - Графічна інтерпретація штучного нейрону; y – значення на виході нейрону , f – функція активації, w_1, w_2 – вага 1-го і 2-го зв'язку відповідно, x_1, x_2 –значення 1-го і 2-го входу відповідно, e – вплив попереднього шару

Функція активація є визначаючим елементом стану нейрону. Зазвичай роль функції активації виконує одна з нелінійних функцій наступного виду:

Таблиця 2.1 – Функції активації

Найбільш поширеним типом функції активації є логістична функція - сигмоїд (Sigmoid).

Рисунок 2.2 - Типові функції активації

На рис 2.2 зображено деякі популярні функцій активації. Проте, роль функції активації може виконувати будь-яка функція. Існують рекомендацій, згідно яким функції активації мають задовольняти наступні умови:

- Нелінійність.

Якщо функція активації нелінійна, можна стверджувати що двошарова ШНМ є апроксиматором універсальної функції. Якщо мережа використовує функцію нелінійності, можна стверджувати що вся мережа еквівалентна одношаровій моделі. Пояснення цього положення у спрощеному вигляді наведено в рівняннях (2.2) – (2.5).

$$a^{(1)} = x, \tag{2.2}$$

$$a^{(2)} = f(W^{(1)} \cdot a^{(1)} + z^{(1)}), \tag{2.3}$$

$$a^{(3)} = f(W^{(2)} \cdot a^{(2)} + z^{(2)}). \tag{2.4}$$

де $a^{(i)}$ – значення на виході з нейрону, $f(x)$ – функція активації, $W^{(i)}$- вага вхідного зв’язку нерону, $z^{(i)}$- значення на виході нейрону зміщення, x – значення на вході в нейрон першого шару.

За умови що функція активації лінійна, наприклад $f(x) = x$, маємо:

$$a^{(3)} = W^{(2)} \cdot a^{(2)} + z^{(2)} = W^{(2)} \cdot \left(W^{(1)} \cdot a^{(1)} + z^{(1)}\right) + z^{(2)} = Wx + z \tag{2.5}$$

- Постійно диференційована.

Загалом ця властивість є бажаною, проте є необхідною при використання градієнтних методів для навчання, адже однією з умов є диференційованість функції на всьому проміжку.

- Має обмежений діапазон значень.

Завдяки цій умові ми маємо чіткий діапазон значень функції активації, тим самим значно пришвидшується процес навчання [12, 13].

2.2 МЕТОДИ МАШИННОГО НАВЧАННЯ

Труднощі, з якими стикаються системи, що спираються на жорстко-кодовані знання, дозволяють стверджувати, що системи штучного інтелекту потребують можливості навчатися шляхом виділення шаблонів з вихідних даних. Задачі, які покликане виконувати машинне навчання зазвичай не піддаються алгоритмізації. Такі типи задач можна розділити на:

- Задачі регресії – прогнозування на основі вибірки об’єктів з різноманітними ознаками (характеристиками). Тут технології машинного навчання дозволяють визначити нечітку залежність (логіку) певних характеристик об’єкту. Приклад задач: розрахунок ціни квартири, очікуваний дохід магазину на наступний місяць, і т.д.
- Задачі класифікації – категоризація об’єктів на основі певних ознак. Приклади задач: розпізнавання об’єктів на фото, визначення браку продукції.
- Задачі кластеризації – розподіл вхідних даних на групи за нечіткими ознаками.
- Задача зменшення розмірності – зменшення кількості вагомих ознак об’єкту. Приклад задач: стиснення даних.

Алгоритми машинного навчання класифікують на основі бажаного результату. Такі алгоритми можна поділити на:

- Навчання з учителем (Supervised learning) – метод машинного навчання оснований на порівнянні дійсних результатів з прогнозованими та розповсюдженні похибки передбачення на зв’язки з вхідними параметрами (факторами). Таким чином алгоритм знаходить функцію залежності вихідних параметрів (результатів) від вхідних.

- Навчання без вчителя (Unsupervised learning) – метод навчання без людського фактору, коли цілю навчання системи є саме навчання. Таким чином системі не надаються реальні результати які хочуть від неї отримати, тим самим не обмежується можливість розвитку системи. Такий метод навчання є найбільш схожим до біологічних методів навчання нейронних мереж.
- Навчання з підкріпленням (Reinforcement learning) – алгоритм вивчає правила як навколишнє середовище відкликається на його дію. Воно поєднує як позначені, так і непозначені дані для створення відповідної функції або класифікатора. Цей тип можна назвати середнім між навчанням з учителем та навчанням без учителя. В такому типі навчання існують фактори що заохочують навчання в правильному напрямку, але не надають системі реально очікуваних даних [14].

Процес навчання ШНМ виконується засобами алгоритмізації і завжди є ітераційним. Існує багато різних алгоритмів навчання, що мають різні характеристики та продуктивність.

Проблема навчання в нейронних мереж методом «навчання з учитилем» сформульована з точки зору мінімізації функції втрат f (рис. 2.3). Ця функція в цілому складається з помилки та умов регуляризації. Поняття помилки характеризує стан відповідності ваг нейронної мережі. З іншого боку, термін регуляризації використовується для запобігання перенавчання шляхом контролю ефективної складності нейронної мережі. Функція втрат залежить від адаптивних параметрів (відхилень та синаптичних ваг) в нейронній мережі. Ці параметри можна об’єднати у єдиний n-мірний ваговий вектор w. На рис. 2.3 представлена функція втрат f (w).

Рисунок 2.3 – Візуалізація функції втрат. f – значення функції активації (значення виходу ШНМ), w_1, w_2 – ваги входів 1 і 2 функції активації f, Hf – патриця Гессіана, w^* - оптимум, ∇f – вектор-градієнт мінімізації функції втрат, A – точка відповідності ваг

З граіфку, зображеному на рис. 2.3, точка w* - мінімум функції втрат. У будь-якій точці А ми можемо обчислити 1-у та 2-гу похідні функції втрат. Перші похідні згруповані у векторі-градієнті функції втрат (2.6).

$$\nabla_i f(w) = \frac{df}{dw_i} \tag{2.6}$$

Аналогічно, друга похідна від функції втрат може бути згрупувати в матрицю Гессіана (2.7).

$$H_{i,j} f(w) = \frac{d^2 f}{dw_i dw_i} \tag{2.7}$$

Проблема мінімізації безперервних і диференційованих функцій багатьох змінних була широко вивчена. Багато загальноприйнятих підходів до цієї проблеми безпосередньо застосовуються при навчанні штучних нейронних мереж [15].

Для навчання використовувався метод зворотнього розповсюдження помилки, так як цей метод є найбільш поплуяярним для навчання нейронних мереж прямого розповсюдження.

Алгоритм методу зворотного розповсюдження помилки заключається в пошуку помилки зпрогнозованого результату від реального значення та розповсюдженні цієї помилки на всі нейрони попередніх шарів.

Рисунок 2.4 – Визначення помилки передбачення.

δ – середньоквадратична помилка

Визначення помилки прогнозування відбувається шляхом визначення середньоквадратичної похибки між зпрогнозованими та реальними даними.

$$\delta = \sqrt{(B\sigma_{\text{сер}})^2 + (\text{C}\sigma_{\text{сер}})^2} \quad (2.8)$$

де $B\sigma_{\text{сер}}$ – стандартне середнє відхилення вибірки реальних значень, $\text{C}\sigma_{\text{сер}}$ – середнє стандартне відхилення вибірки зпрогнозованих значень.

Розповсюдження похибки на нейрони мережі виконується шляхом перемноження похибки на ваги зв'язку, рис. 2.5.

Рисунок 2.6 – Розповсюдження середньоквадратичної похибки

Для штучних нейронів більш глибоких шарів значення середньоквадратичного відхилення є сума добутку ваг зв'язків на середньоквадратичну похибку відповідного нейрону наступного .

Рисунок 2.7 – Розповсюдження похибки на нейрони глибших шарів

Після визначення середньоквадратичної похибки усіх нейронів проміжних шарів починається процес балансування вагів. Таким чином, починаючи зі зв'язків між вхідним та першим прихованим шаром визначається значення нових вагів визначається за формулою 2.9.

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1 \quad (2.9)$$

Де η – коефіцієнт навчання, $\eta \in [0,1]$.

Рисунок 2.8 – Процес коригування вагів штучної нейронної мережі

Процес оптимізації (навчання) відбувається шляхом скінченної кількості повторень процесу навчання. Тривалість процесу та властивості інерційності процесу визначається налаштуванням розміру пакетів та кількості епох.

Епохою називають число, що характеризує кількість проходжень усього нобору даних через нейронну мережу. Під час навчання дані між епохами залишаються сталими.

Пакетом в даному методі навчання називають набір факторів який буде «прогнаний» через штучну нейронну мережу без корегування вагів. Тобто, згідно алгоритму методу зворотнього розповсюдження похибки, процес корегування вагів ШНМ відбувається після кожного проходження пакету. Існує обмеження на розмір пакету: він має бути цілочисельним дільником кількості вхідних записів. Це обмеження існує для того щоб упевнитись у тому що всі дані використовуються для навчання.

Кількість епох є довільним числом, та визначається емпірично. Зміна цього числа регулює ступінь навчання. При ознаках перенавчання необхідно зменшити це число, а також, можливо, скорегувати розмір пакету [16].

2.3 ВИСНОВКИ ЗА РОЗДІЛОМ

Отже, вирішенням задач які не мають чіткої логіки займаються методи машинного навчання. Для конкретного типу задач було розроблено кращі методи навчання, особливу популярність зайняли штучні нейронні мережі, проте і вони мають конкретну специфіку вирішення задач в залежності від топології. Важливим етапом вирішення поставлених задач є процес проектування нейронної мережі. Проте цьому етапу передує не менш важливий процес пошуку, аналізу та обробки даних для навчання, визначення впливових факторів. Після цього слідує процес вибору алгоритму навчання та зведення процесу навчання до вирішення задачі оптимізації, знаходження помилок, вирішення проблем перенавчання, оцінка якості прогнозування та

тестування нейронної мережі на реальних даних. Усі ці етапи вже проведено для задачі даної магістерської дисертації та детально описані в наступних розділах.

3 РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ

Не основі сформульованих вимог до системи прогнозування генерації електроенергії вітряною станцією було розроблено програмний комплекс реалізації поставленої задачі, що складається з 3-ох програмних модулів:

- Модуль навчання штучної нейронної мережі;
- Модуль підготовки та обробки данних;
- Модуль динамічного завантаження метеопрогнозу (з віддаленого серверу).

До вибору засобів, середовища та платформи було поставлено наступні вимоги:

- Наявність зручних програмних рішень (фреймворків, бібілотек);
- Кросплатформеність;
- Легкість встановлення кінцевого додатку та зручність інтеграції;
- Динамічність системи.

На основі описаних вище критеріїв було обрано наступний стек технологій. Для побудови та навчання штучної нейронної мережі було обрано мову програмування Python, так як вона задовільняє усі поставлені вище вимоги. Вибір був зупинений на ній, оскільки ця мова є найбільш популярною серед конкурентів через те, що:

- Python – скриптова мова програмування;
- Вже давно використовується спеціалістами з області Big Data, а тому має широкий набір реалізованих бібліотек для роботи з даними;
- З самого початку розроблявся з можливістю зручно та гнучко працювати з великими даними;
- Найбільш популярний фреймворк Tensorflow 2.0 реалізований вчасності на мові програмування Python, та має велике ком’юніті користувачів;

Побудова та навчання ШНМ проводилась із використанням фреймворку Tensorflow версії 2.0 – найбільш поплуярним засобом з великим набором функціоналу для роботи з нейронними мережами.

Втілення вимог кросплатформенності було виконане шляхом контейнеризації додатку засобами Docker. Через велику складність таких додатків, вони мають велику кількість залежностей, які потрібно встановлювати на ЕОМ перед використанням нейронних мереж. Docker містить підготовлені зображення («знімки») програмного оточення з превстановленим інтерпретатором мови програмування Python та загальним набором бібліотек та фремворкім для роботи з machine learning. Це дозволяє легко мігрувати реалізовані програмні засоби між різними ЕОМ та використовувати готові засоби інтеграції Docker.

Програмні модулі обробки та завантаження даних були виконані на мові програмування Java через її зручність роботи з веб-серверами, великими даними та файлами.

Таблиця 3.1 – Модулі прогрмного комплексу

Таблиця 3.1 – Модулі прогрмного комплексу

Діаграму залежностей програмних модулів та потоків даних можна побачити на рис. 3.1.

Рисунок 3.1 – Діаграма зв’язків програмних комплексів та потоків даних

Модулі обробки та динамічного завантаження даних не мають користувацього інтерфейсу, так як є допоміжними додатками, які не мають зв’язку з функціоналом користувача. Користувачу достатньо притримуватись

наступного формату формування таблиць в файлах з даними для навчання/прогнозування:

- Розширення файлу - .csv;
- Розділ стовпців – кома «,»;
- Назва стовпців розміщується в першому рядку файлу через кому
- Кількість стовпців кожного рядку має бути однаковою.

Формат прикладу CSV таблиць з назвами стовпців можна знайти в Додаткі Г.

3.1 РОЗРОБКА ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Загальних рекомендацій щодо кількості шарів багатошарового перцептрону не існує, проте пропонується наступна початкова модель структури нейронної мережі. Оптимальна кількість прихованих шарів залежить набору вхідних параметрів і розраховується за формулою 3.1.

$$N = \frac{\sqrt[p+1]{k}}{p+1} = \frac{\sqrt[2]{212}}{2} = 7, \quad (3.1)$$

де N – кількість прихованих шарів, k – кількість вхідних факторів, p – кількість нейронів вихідного шару.

Проте, під час емпіричного дослідження структури нейромережі було встановлено оптимальну кількість шарів N = 4.

Для складних систем з великою кількістю факторів пропонується використання наступної моделі кількості нейронів у проміжних шарах:

$$n_1 = 0{,}1..0{,}8 * k \quad (3.2)$$

$$n_i = \frac{n_{i-1}}{1{,}5..2{,}9} \quad (3.3)$$

де k – кількість вхідних факторів, n_i – кулькість нейронів і-го шару.

За формулами 3.2, 3.3 кількість нейронів багатошарового перцептрона є наступною: n1 = 32, n2 = 16, n3 = 6, n4 = 2, n_виходу = 1.

Початкові ваги зв’язків ШНМ мають значний вплив на швидкість процесу навчання, проте цей вплив помітний тільки на початку процесу навчання. Розрахунок початкових вагів є дуже специфічним для кожної прикладної задачі, проте враховуючи незначний вплив на загальний процес навчання, існує 3 емпіричні методи встановлення початкових вагів. Кожен з цих методів має випадкову складову. Перший метод (fan_in) базується на кількості вхідних факторів, другий (fan_out) на кількості вихідних значень, а третій (fan_avg) на генерації рандомної величини за нормальним розподілом значень. В своїй роботі я використовував третій метод.

У Tensorflow функція ініціалізації вагів визначається наступним чином:

Рисунок 3.2 – Функція ініціалізації вагів Tensorflow 2.0

Функцією активації штучних нейронів обрано Сигмоїду:

Рисунок 3.2 – Сигмоїдальна функція активації

Метод вибору функції активації – емпіричний.

Кінцевий вигляд структури ШНМ зображено на рис. 3.3.

Рисунок 3.3 – Візуалізація структури багатошарового перцептрона зі знайденими параметрами мережі

В рамках задачі, на мові програмуванні Python було розроблено скрипт для читання, форматування та злиття даних, побудови та навчання ШНМ на прочитаних даних. Результатом роботи скрипту є представлення користувачу

готової шутчної нейронної мережі для прогнозування генерації електроенергії вітряними станціями.

Блок-схему роботи скрипту наведено на рис. 3.4.

Рисунок 3.4 – Блок-схема роботи Python-скрипту

Програмна реалізація побудови нейронної мережі наведена на рис. 3.5.

Рисунок 3.5 – Програмний код побудови багатошарового перцептрону

Код побудови нейронної мережі розділений на 8 етапів. Етап 1 – ініціалізація кількості та розміру шарів. Етап 2 – стоверння матриці вхідних факторів та вихідних значень. Етап 3 – визначення функції ініціалізації початкових вагів. Етап 4 – визначення функції початкової ініціалізації вагів для нейронів зміщення. Етап 5 – розміщення вагів (ініціалізація початкових вагів функцією з етапу 3) на шарах нейронної мережі. Етап 6 – ініціалізація вагів для вихідного та нейрону зміщення. Етап 7 – визначення метаматичної моделі нейрону. Етап 8 – встановлення математичної моделі вихідного нейрону.

3.2 РОЗРОБКА МОДУЛЯ ДИНАМІЧНОГО ЗАВАНТАЖЕННЯ ДАНИХ

Так як дані про погоду знаходяться на віддаленому сервері, для обробки їх «модулем обробки даних» та передачі акутального стану погоди набором вхідних параметрів для «модуля прогнозування» необхідно мати засоби

завантаження та збереження цих даних на локальній ЕОМ. Саме з цією метою було реалізовано модуль динамічного завантаження даних. Діаграму залежностей між класами даного модуля наведено на рис. 3.6.

Рисунок 3.6 – Схема залежностей класів модуля динамічного завантаження даних

Результатом роботи даного модуля є CSV-файл з інформацією про погоду на заданий проміжок часу відповідно даним отриманим з сервера збереження метеопрогнозу.

Так як всі дані прогнозу погоди приходять у текстовому вигляді, вони мають бути переформатовані у відповідний формат. Приклад формату даних що надходять з серверу наведено нижче.

Рисунок 3.7 – Формат данних з серверу інформації про погоду

Дані з сереверу надходять в форматі JSON та містять у собі інформацію про погоду за 1 день з інтервалом зняття показань у 30 хвилин. При запускі програми задається URL-адреса віддаленого серверу, що містить інформацію про погоду, діапазон дат, прогноз для яких необхідно отримати та назву файлу, куди необхідно зберігати дані. Результати роботи даного модуля наведені на рис. 3.8.

Рисунок 3.8 - частина CSV-файлу з інформацією про погоду за 2018 рік

При порівнянні рис. 3.7-3.8 можна зробити висновок що даний модуль зберігає інформацію що надходить з серверу без попередньої обробки. Для використання отриманих даних для навчання ШНМ вони спершу мають бути

направлені на модуль обробки даних, а тільки звідти на модуль навчання штучної нейронної мережі.

3.3 ВИСНОВКИ ЗА РОЗДІЛОМ

В рамках розробки магістерської дисертації було написано програмний комплекс із використанням мов програмування Java та Python із застоуванням фреймворків та бібілотек наведених в табл. 3.2.

Таблиця 3.2 – Перелік фремйморків та бібліотек використовуваних при розробці додатків

Написаний додаток має консольний інтерфейс, а також використовує чітко визначені шляхи директорій в операційній системі для роботи з файлами даних. Додаток є повністю платформонезалежний, оскільки під час розробки використовувались кросплатформаенні мови програмування та Docker. Архітектура системи в подальшому дозволяє реалізувати графічний інтерфейс додатку, проте для такого типу систем це скоріше виключення, аніж необхідність.

Програмний комплекс дозволяє динамічно прогнозувати генерацію електроенергії вітряними станціями з використанням технології нейронних мереж шляхом отримання доступу в реальному часі до прогнозу погоди в регіоні, а також історичних даних про роботу вітряної станції в минулому.

Було проведено комп’ютерні розрахунки на основі написаного програмного забезпечення. Використано різні значення вхідних наборів даних та дослідженно для них результати пронозування.

4 ПІДГОТОВКА ДАНИХ ДЛЯ НАВЧАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Одним з ключових факторів які впливають на результати роботи штучних нейронних мереж є дані, які використовувались для навчання. Діапазон даних має бути досить великим, щоб сисетма могла бути успішно на ньому навчана та протестована, а самі дані мають бути консистентними, та в повному обсязі чисельно описувати предметну область, яку вони відображають. Першим етапом підготовки був пошук даних відповідних до вище описаних критеріїв.

На цьому етапі було проведено аналіз вітряних парків України і було визначено що сьогодні існуючі вітропарки мають загальну потужність більше 706 МВт. Це такі парки, як Ботнієвська ВЕС (200 МВт), Приморська ВЕС (200 МВт), Дмитрівська ВЕС (35 МВт) та інші. Також у 2019 році відбувається будівництво нових вітряних парків сумарною потужністю маже 1 ГВт.

Нажаль статистичних даних з цих станцій немає у вільному доступі, а спроба отримати їх від власника (ТОВ «ДТЕК») не закінчилася успіхом.

Тому для навчання ШНМ було використано дані з турецької ВЕС у м. Ялова, які знаходились у вільному доступі і містять інформацію за весь 2018 рік з інтервалом фіксації інформації кожні 10 хвилин. Далі, у форматі листування було визначено точне місцезнаходження ВС (X:668478 Y:4494833 UTM ED 50, 6°). Ця інформація дала змогу визначити метеостанцію, найближчу за розташуванням та приступити до пошуку історичних даних прогнозу погоди в регіоні на 2018 рік. Загалом, ця інформація є платною (близько 80-100 євро), проте, в решті решт було знайдено безкоштовний який містив цю інформацію.

В рамках під-задачі було розроблено модуль програмного забезпечення на мові програмування Java для отримання, форматування, фільтрації та збереження у відповідний формат даних про погоду з описаного вище ресурсу.

4.3 АНАЛІЗ ДАНИХ

Статистична інформація з ВС була завантажена з відкритого ресурсу Kaggle для спеціалістів з області Великих даних (Data Science/Big Data).

Однією з головних проблем спеціаліста з області Machine Learning є:

- аналіз даних та визначення впливових факторів
- адаптація необроблених даних для можливості навчання ШНМ.

На початку роботи дані з ВС мали вигляд таблиці на рис. 4.1.

Рисунок 4.1 - Необроблені дані отримані з ВС за 2018 рік

Формат набору даних – CSV-файл, що містить таблицю розміром 5 стовпчиків та 50530 записів. Інтервал запису інформації – 10 хвилин. Дані є «брудними» та, як мінімум, потребують фільтрації і відкидання несуттєвих факторів. Наприклад, теоретична потужність – очікувана потужність ВС за заданої швидкості вітру, може здаватись одним з найбільш безкорисних показників для передбачення, так як відома вже реальна потужність ВС у конкретний момент часу і в процесі прогнозування електрогенерації краще покластись на навички ШНМ у знаходженні залежностей між швидкістю/напрямком вітру і вже реальною потужністю ВС, а не теоретичною. Проте, якщо провести більш глибинний аналіз, можна встановити нові корисні фактори для передбачення, базуючись на значенні теоретичної потужності.

Для знаходження залежностей між факторами було проведено первинний аналіз даних і побудовано наступні діаграми (рис.4.2 – 4.3).

Рисунок 4.2 – Теоретична і реальна середня потужність ВС від напрямку вітру

Рисунок 4.3 – Теоретична і реальна кількість виробленої енергії ВС від напрямку вітру

З рис 4.2-4.3 можна зробити висновок що найбільша потужність вітряної станції досягається при наявності південного та південно-західного вітру, проте найбільшу кількість електроенергії ВС генерує при східно-північному та північно-східному вітрі.

Рисунок 4.4 – Втрати потужності від напрямку вітру

Рис. 4.4 зображує втрати потужності за напрямком вітру - середню різницю між теоретичною та реальною потужністю ВС відносно напрямку вітру. При порівнянні значень теоретичної та реальної потужності для різних швидеостей східно-північного вітру (рис. 4.5), можна помітити турбулентний характер реальної потужності на проміжку швидкостей більше 13 м/с. Таку нелінійність та великі втрати можна пояснити припущенням що східно-північний вітер має досить поривчастий характер. Рис. 4.5 підтверджує наше припущення.

Виявлення таких факторів та використання їх під час навчання значно покращує результат прогнозування кількості генерації електроенергії методами штучного інтелекту.

Рисунок 4.5 - Залежність потужності ВС від швидкості східно-північного вітру

Для визначення ступеня лінійної залежності між факторами та потужністю ВС визначимо коефіцієнт кореляції Пірсона. Коефіцієнт кореляції Пірсона між двома змінними дорівнює коваріації двох змінних, або сумі добутків відхилень, поділеній на добуток їх стандартних відхилень. Нехай, існує дві вибірки $x^m = (x_1, \dots, x_m)$, $y^m = (y_1, \dots, y_m)$, тоді:

$$r_{xy} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})\cdot(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{m}(y_i - \bar{y})^2}}, \quad (4.1)$$

де $\bar{x}, \bar{y}$ – вибіркові середні x^m, y^m, $(x_i - \bar{x})^2$, $(y_i - \bar{y})^2$ – вибіркові дисперсії, $r_{xy} \in [-1, 1]$.

За формулою (4.1) побудуємо матрицю кореляції факторів та потужністю вітряної станції. Значення матриці кореляції наведені в таблиці 4.1.

Таблиця 4.1 – Коефіцієнти кореляцій Пірсона між факторами набору даних.

Таблиця 4.2 – Значущість кореляції Пірсона

Зі значеннь таблиць 4.1-4.2 можна зробити висновок що всі фактори окрім напрямку вітру мають високу лінійну залежність і мають залишитись в наборі даних. Щоб вирішити чи потрібно залишати фактор напрямку вітру побудуємо матрицю щільності розподілу значень напрямку вітру від кожного з факторів.

Рисунок 4.6 - Матриця щільності розподілу значень напрямку від інших факторів

Проаналізувавши матрицю щільності можна виділити по дві ділянки щільного розподілу даних для кожного з факторів.

Рисунок 4.7 – Ділянки щільного розподілу даних

Виходячи з доказу існування нелінійних залежностей між напрямком вітру та іншими параметрами є доцільним залишити фактор напрямку вітру в наборі даних.

З історичними даними з вітряної станції є ще низка проблем (періодична відсутність консистентності даних і т.д) які були вирішені програмним модулем форматування та оброки даних написаного на мові програмування Java. Більше детальної інформації про цей модуль описано в наступному підрозділі.

Дані з метеостанції мали первинний вигляд зображений на рис. 4.8.

Рисунок 4.8 – історичні дані прогнозу погоди в регіоні

Так як походження та природа такого типу даних давно відома у зв’язку з гарно дослідженою області метеорології, встановлення залежностей та коефіцієнтів кореляції для цього набору даних не несе додаткової інформаційної цінності. Тому з моєю точки зору було доцільним залишити всі фактори зображені на рис. 4.8, а саме:

- Час прогнозу.
- Температуру повітря.
- Словесна інтерпретація погодних умов (сонячно, похмуро, дощ, гроза і т.д.)
- Швидкість вітру.
- Напрямок вітру.
- Вологість повітря.
- Атмосферний тиск.
- Видимість.

Проте отримані дані є «брудними», та потребують формалізації і обробки. Крім того була виявлена проблема з інтервалом періодичності записів та відсутності консистентності показань (деякі записи не містили

повного спектру факторів). Для вирішення цих та інших проблем з наборами даних мною було написано програмний модуль для обробки та трансформації «брудних» даних у формат, що підходить для навчання ШНМ.

4.4 ВИЗНАЧЕННЯ КРИТЕРІЇВ ФОРМАЛІЗАЦІЇ ДАНИХ

Не беручи до уваги (на даному етапі) проблеми «брудних» записів, перед нами стоїть задача вирішення питання відповідності даних між вітряною та метеостанцією, правильної формалізації даних для можливості використання їх для навчання штучної нейронної мережі. Так як задача програмного комплексу стоїть у передбаченні кількості електрогенерації на 6-8 годин вперед, запис одного рядку датасету має містити усю необхідну інформацію для того щоб була змога зробити такий прогноз.

Одним з критеріїв злиття даних (з ВС та метоостанції) в один набір є їх відповідність між собою. Таким чином, мають співпадати розмірності величин, що характеризують одні і ті ж або спів розмірні параметри, а також, фактичний розмір набору даних (кількість записів). Розмір датасету з інформацією про погодні умови, після його завантаження «програмним модулем завантаження даних» склав 17 038 записів. У ньому зберігаються дані про погоду за весь 2018 рік. Розмір набору даних статистичної інформації з ВС за 2018 склав 50 530 записів. Проблема невідповідності кількості записів першочергово залежить від інтервалу фіксації показань. Для набору даних з ВС цей інтервал складає 10 хвилин, а для набору даних метеорологічного прогнозу – всередньому 30 хвилин. Домноживши кількість записів на часовий інтервал отримаємо цифри досить схожі за значеннями.

$$\text{Кількість записів}_{\text{з вітряної станції}} = 50\,530 \cdot 10 \text{ хв} = 505\,300 \text{ хв} \quad (4.2)$$

$$\text{Кількість записів}_{\text{з метео станції}} = 17\,038 \cdot 30 \text{ хв} = 511\,140 \text{ хв} \quad (4.3)$$

Приймемо до уваги що у невисокосному році 525 600 хвилин, різницю в кількості отриманих даних компенсуємо за допомогою програмного модуля обробки даних. Беручи до уваги кількість хивилн у році, інтервали зняття показань та отриману кількість записів, можна зробити наступні висновки про втрати даних в датасетах:

- Набір даних з ВС (датасет №1) – 2 030 записів.
- Набір даних метеорологічного прогнозу (датасет №2) – 482 записи.

Таким чином після формалізації таблиць (компенсації втрачених показань) ми отримаємо на кожен запис метеорологічного прогнозу 3 записи стану ВС. Тобто, при злитті датасетів нам потрібно створювати рядки шляхом комбінації n рядків з датасету №2 і 3*n рядків з датасету №1.

Крім того, для навчання ми використовуватимемо тільки цифрові показники, тому необхідним є вилучення текстової інформації та дат з датасетів шляхом перетворення їх в чисельний еквівалент.

При відсутності деяких факторів у датасеті необхідністю є їх заміна на числову константу яка не відповідає природі характеризуючого явища, наприклад, -1 для швидкості вітру або атмосферного тиску.

Останнім критерієм формалізації є перетворення деяких даних у іншу величину. Цей критерій стосується підрахунку кількості згенерованої електроенергії ВС впродовж наступних 8 годин. Прогнозування цієї величини є цільовою задачею для даної нейронної мережі, тому враховуючи що ми використовуємо метод навчання з учителем, нам необхідно інтерпретувати ці записи, як дані відносно яких буде корегуватись похибка прогнозування.

Для прогнозування електрогенерації на наступні 8 годин, для навчання потрібно використовувати записи з інформаціїєю про погоду за попередні та наступні 8 годин, а також інформацію про стан ВС за попердні 8 годин та сумарну кількість згенерованої електроенергії впродовж майбутніх 8 годин (рис. 4.9).

Рисунок 4.9 – Структурна схема рядка датасету

Рис. 4.9 описує формат даних використовуваних для навчання. Формат даних для вже описаної ШНМ наведений на рис. 4.10.

Рисунок 4.10 – Структурна схема вхідного набору даних для вже навченої ШНМ

З рис. 4.10 видно, що на вхід вже навченої системи подається записи з ВС за попередні 8 годин та прогноз погоди за попередні та майбутні 8 годин.

4.5 РОЗРОБКА МОДУЛЯ ОБРОБКИ ДАНИХ

Однією з проблем навчання та використання вже навченої системи є доступ до даних метеопрогнозу які зазвичай знаходяться на віддаленому сервері. Таким чином постає питання у створенні програмного додатку який зможе отримувати дані метеопрогнозу на заданий проміжок часу та динамічно їх формалізуватиме.

В рамках роботи над магістерською дисертацією було створено програмний додаток який надає можливість отримувати частково формалізовані дані, які надалі будуть використовуватись для навчання та/або прогнозування кількості генерації електроенергії на вже навченій системі.

Діаграма залежностей класів даного модуля наведена на рис. 4.11.

Рисунок 4.11 – Діагарма залежностей класів модуля обробки даних

Результатом роботи даного ПЗ є CSV-файл з набором записів прогнозу погоди за вказаний проміжок часу.

Цей додаток було застосовано для отримання даних для навчання.

Так як всі дані прогнозу погоди приходять у текстовому вигляді, вони мають бути переформатовані у відповідний формат. Приклад формату даних що надходять з серверу наведено на рис. 4.12.

Рисунок 4.12 – Формат данних з серверу даних про погоду

Результати роботи ПЗ наведено в таблиці 4.3.

Таблиця 4.3 - Частина даних прогнозу погоди на 2018 рік у CSV-файлі

Після того як дані завантажені та збережені, інший програмний модуль, відповідальний за форматування починає зводити дані з двох CSV-файлів у один кінцевий датасет. Під час зведення вирішуються усі проблеми формалізації наведені у пункті 4.2.

Проблема втрачених даних вирішується шляхом знаходження найближчого значення в датасеті та дублюванням його значень. Схематичний опис цього процесу зображений на рис. 4.13.

Рисунок 4.13 – Схематичний опис вирішення проблеми втрачених даних

В результаті ми маємо два датасети з консистентною інформацією відносно часу.

На етапі запуску python-скрипта створення і навчання ШНМ на вхід подається 2 датасети (історичних даних погоди та стану ВС). Датасет погоди має 8 факторів що описують погодні умови. Інший датасет – 5 факторів про стан ВС у певний момент часу.

Для реалізації поставленої задачі (прогнозування генерації електроенергії вітряною станцією на майбутні 8 годин) було обрано наступний підхід формування даних.

Один запис датасету містить 249 факторів, з них:

128 – кількість факторів про погоду. 128 це сумарна кількість факторів 16-ти об’єднаних записів про погоду. З цих 16-ти, 8 записів – дані про погоду до поточного моменту часу, інші 8 – прогноз погоди на майбутні 8 годин.

121 – кількість факторів про стан вітряної станції. З них 120 – кількість факторів про попередній стан ВС до поточного моменту, 1 фактор – сумарна кількість електроенергії виробленої за майбутні 8 годин (розрахована з відомих історичних даних). Цей 121-й фактор є «ціллю» навчання нашої ШНМ, це значення буде порівнюватись з передбаченим нашою мережею, а помилка передбачення буде розповсюджуватись на усі попередні шари. 120 факторів містять у собі інформацію з 24 записів датасету. Відношення 24/8 = 3 відповідає критеріям формалізації наших даних з пункту 4.2.

Перевірка відповідності даних до критеріїв формалізації наведена на рис. 4.14 - 4.15.

Рисунок 4.14 – Перевірка формалізації даних у консолі Python

Далі дані були зведені до відповідного типу та злиті в одну таблицю.

Рисунок 4.15 – Технічна інформація зведеного датасету

Наступним етапом підготовки даних до навчання було позбавлення датасету дублюючої або зайвої інформації. Наприклад, враховуючи що на цьому етапі у нас один запис таблиці містить в собі інформацію 16 записів датасету про погоду та 24 записи датасету про стан ВС можна праналізувати фактори на наявність зайвої інформації. Таким чином ми можем позбутися 39

лишніх факторів дат, які або дублюють одна одну (випадок однакових дат 2-х різних таблиць), або містить значення дат з чітко визначеним інтервалом, тим самими нівелюють корисність даної інформації. Після видалення 39 факторів дат (з 40) та переформатування останнього фактору дати у 2 фактори - фактор місяця та фактор дня ми отримаємо датасет на 213 факторів (212 + 1 фактор кількості реально виробленої електроенергії протягом наступних 8 годин). Програмний код форматування злиття та адаптації даних наведено в Додатках.

Після проведених маніпуляцій з даними кінцевий розмір датасету складає 1094x213 (рис. 4.16).

Рисунок 4.16 – Кінцевий розмір датасету готового до навчання

На даному етапі вже можна підтвердити що дані повністю формалізовано і є готовими для навчання ШНМ.

Наступним важливим кроком є розподіл отриманих даних на дані що будуть використовуватись для навчання, та дані що будуть використовуватись для тестування.

Зазвичай, для тестування відбирають 20% даних, а для навчання решту 80%. Дуже важливим етапом є правильний розподіл даних, так як він буде прямим чином впливати на висновки щодо правильності побудови, навчання і роботи ШНМ. Тому, для тестування ШНМ було відібрано 20% даних на всьому проміжку записів. Таким чином ми усуваємо імовірність впливу локалізованих факторів під час тестування ШНМ. Наприклад, відібравши 20% даних з початку або кінця датасету ми скоріш всього зустрілись би з проблемою невірного передбачення, так як ШНМ не змогла б навчитись на даних зимового періоду. Крім того, тестування відбувалось би саме на цих даних. Досить імовірно що в зимовий період часу електрогенерація ВС відрізняється від генерації електроенергії, наприклад, у літній або весняний період, так як тут має місце фактор значної зміни погодних умов.

За умови наявності більшої кількості даних можна було б зробити критерії формалізації більш жорсткими, наприклад, видалити записи які мають активну потужність 0 кВт при ненульовій швидкості вітру у зв’язку з проведенням реглаентних робіт або несправністю датчиків. Покращення якості вибірки даних збільшує точність прогнозування.

4.6 ВИСНОВКИ ЗА РОЗДІЛОМ

У ході виконання магістерської дисертації було проведено підготовку та обробку даних використовуваних для навчання штучної нейронної мережі. Під час виконання цього процесу було визначено критерії формалізації даних, для можливості використання їх у процесі навчання. Такими критеріями були:

- Відповідність розмірності величин що описують однакові або схожі фізичні характеристики;
- Консистентність;
- Відповідність розміру датасетів.

Таким чином до проведення формалізації даних було отрмано 50 530 записів про стан вітряної станції та 17 038 записів прогнозу погоди.

Після проведення формалізації розмір датасету склав 1094 рядків та 213 факторів. 1 рядок датасету містить інформацію про 16 станів погоди, 8 станів ВС та 1 фактору сумарної кількості електрогенерації за майбутні 8 годин.

5 АНАЛІЗ РЕЗУЛЬТАТІВ НАВЧАННЯ

Висновок по результатам навчання проводиться шляхом аналізу точності передбачення на тестових даних.

За результатами навчання було досягнуто точність передбачення у 86%.

Найбільш оптимальна кількість епох – 30, при розміру пакету – 5 записів.

Слід зазначити що дані перед «згодовуванням» в нейронну мережу трансформуються у діапазон значень [-1,1]. Середньоквадратичне відхилення для трансформованих значень склало 0.0025067409.

Після зворотної трансформації середньоквадратичне відхилення склало 184.653 кВт для середнього значення кількості згенерованої електроенергії на 8 годин вперед 1293.335 кВт. Середня похибка прогнозування склала 14,2%.

Результати роботи модуля візуалізації навчання наведено на рис. 5.1-5.4.

Рисунок 5.1 – Візуалізація навчання 1 епоха, 10 пакет

Рисунок 5.2 – Візуалізація навчання 3 епоха, 10 пакет

З рис. 5.1-5.3 можна спостерігати поступове наближення апроксимуючої функції до реальних значень функції електрогенерації. Слід зауважити що на деяких епохах можна спостерігати погіршення значеня прогнозуваннь, проте цей факт не є сигналом для припинення процесу навчання. Це пов’язано з використанням градієнтних методів оптимізації ваг (навчання) і можливістю їх потрапляння в локальний оптимум.

Рисунок 5.3 - Візуалізація навчання 8 епоха, 140 пакет

Важливим фактором підбору оптимальної кількості пакетів та епох базується, головним чином, на результатах навчання. Тому, для визначення найкращих показників грогнозування було проведено декілька експериментів за результатами яких було досягенено:

- результатів з меншою точністью прогнозування;

- перенавчання – коли ШНМ ідеально апроксимує дані для навчання, проте робить значні помилки на тестових даних;
- результатів з різною точністю прогнозування серед яких було обрано найкращий.

Після визначення найркащого випадку навчання було запам’ятовано та збережено відповідні налаштування ШНМ - кількість прихованих шарів, кількість нейронів у шарах, значення ваг та нейронів зміщення.

Рисунок 5.4 - Візуалізація навчання 30 епоха, 170 пакет

По завершенню процесу навчання можна спостерігати місця де значення передбачення не співпадає з реальним значенням потужності ВС. Це частково пов’язано з наявністю недостовірної інормації у даних для навчання, такої як 0 значення потужності ВС для ненульової швидкості вітру під час проведення регламентних ремонтних робіт. З покращенням якості даних для навчання очікується ріст точності прогнозування ШНМ.

5.3ВИСНОВКИ ЗА РОЗДІЛОМ

Провівши навчання ШНМ прямого розповсюдження було отримано досить точні результати прогнозування, які задовільнять поставлені очікування. Точність прогнозування склала 86,8%. Модуль візуалізації навчання відображає процес поступового покращення результатів. Крім того, спираючись на отримані графіки з модуля візуалізації навчання було розраховано оптимальну кількість нейронів у шарах та кількість необхідних епох. Також на визначення цих характеристик впливали результати навчання, а саме точність прогнозування. Проте, ітераційно запускаючи навчання ШНМ з однаковими характеристиками, було отримано різні результати. Це пов’язано з довільним визначенням початкових ваг, перемішуванням датасету перед розподілом даних на тестові та навчальні, а також самою природою штучних нейронних мереж. Отже, говорячи про точність прогнозування

дослідженої системи, можна говорити не про конкретне значення, а середній діапазон значень точності передбачення, який склав 80-86,8%. Крім цього, слід зазначити наявність «брудних» даних, використовуваних в процесі навчання та тестування. Це пов’язано з тим фактом, що у датасеті було знайдено нереалістичні дані щодо кількості згенерованої електроенергії, коли за значної кількості і потужності вітру активна потужність ВС складала 0 кВт*год. Можливо це пов’язано з ремонтними роботами, або з навмисним відключенням вітряних турбін. Існує вірогідність що додаткова фільтрація даних може ще покращити результати прогнозування системи.

6 РОЗРОБКА СТАРТАП ПРОЕКТУ

Бізнес-ідея: система передбачення генерації електроенергії вітряною станцією на середньостроковий термін.

Метою стартапу є стабілізація енергопостачання для підприємств з частковою енергозалежністю від вітряних станцій, а також вітряних ферм що продають електроенергію на ринках електроенергії.

Тема: Система прогнозування генерації електроенергії вітряною станцією із застосуванням методів машинного навчання.

Назва: Система передбачення електрогенерації вітряною станцією у промисловій електромережі.

Суб’єкт замовлення: підприємства з частковою енергозалежністю від вітряних станцій, а також вітряні ферми що продають електроенергію на ринках електроенергії.

Об’єкт дослідження: багатомодульний програмний комплекс на базі машинного навчання для розрахунку генерації електроенергії вітряною станцією і, як наслідок, оптимізації використання «зеленої» енергії у виробництві та зменшення затрат на електроенергію вцілому.

Місце розробки у інноваційному ланцюжку цінностей: B2B модель. Так як виробництво спрямовано на взаємодію з юридичними особами.

Таблиця 6.1 – Плановий обсяг виконаних проектів по місяцям на перший рік.

Продукт (послуга) – розробка, підтримка та інтеграція програмного продукту для прогнозування генерації електроенергії вітряними станціями (ВС).

Ідея продукту полягає в тому, щоб використовувати технологію машинного навчання на історичних даних, зібраних з ВС, а також, даних прогнозу погоди в регіоні для високоточного прогнозування електрогенерації вітряною станцією на 6-8 годин вперед. Середня точність пронозу досягає 85% завдяки впровадженню технології штучних нейронних мереж (ШНМ) до програмного продукту. При використанні вітряних станцій з акумулюючим шаром у сукупності з даним продуктом досягається можливість отримання точної кількості електропостачання на майбутні 6-8 годин. Дана технологія є доцільною для використання на підприємствах з частковою енергозалежністю від вітряних станцій, а також на вітряних фермах що продають електроенергію на ринках електроенергії.

Бізнес-ідея полягає у наданні послуг з консультації, адаптації, інтеграції та підтримки програмного продукту у існуючу «зелену» електромережу. При зверненні клієнта (далі Замовник) до ТОВ «Oracle Solutions» (далі Виконавець) оговорюється завдання, необхідне до виконання. Замовником надається інформація для адаптації Виконавцем існуючого програмного засобу відповідно узгоджених вимог: кількість вітряних турбін, схема їх підключення між собою та у мережу, наявність акумулюючого шару необхідної ємності, статистичні дані зібрані з вітряного парку протягом досить тривалого терміну (близько одного року), бажана стабільна промислова потужність вітряного парку. Після цього Виконавець створює та надає план виконання проекту, можливі варіанти модернізації існуючої системи (за необхідністю), пропонує перелік та вартість послуг можливих до виконання (обов'язкових та опціональних, таких як подальша підтримка імплементованої системи, інтеграція нових функцій, тощо). Після двостороннього узгодження підписується акт проведення робіт, після чого Виконавець приступає до роботи та має завершити її в узгоджені з Замовником терміни.

Постачальники обладнання. Магазини роздрібної торгівлі електроніки та комп'ютерної техніки. Також для масштабних проектів розглядаються угоди з оптовими постачальниками обладнання. У разі необхідності збільшення акумулюючого шару вітропарку Замовнику пропонується розглянути пропозиції компаній, що спицифікуються у цій сфері (Redflow ZCell, Tesla Powerpack, Sonnen SonnenCommunity, Panasonic Smart Towns).

Кваліфікація персоналу. Спочатку персонал у ТОВ «Oracle Solutions» буде складатись з трьох інженерів, двох менеджерів, одного юриста, та одного економіста. Із ростом замовлень планується як горизонтальне так і вертикальне розширення персоналу.

Споживачами є компанії, які спеціалізуються на продажу вітряної електроенергії, а також підприємства, що мають часткове електропостачання від вітряних станцій.

Ринок збуту. 1 липня 2019 року в Україні запрацював ринок електроенергії. Згідно правил ринку, торги відбуваються «на добу наперед». Також, разом з цим запрацював «внутрішньодобовий ринок», що регулює продаж електроенергії на момент поточної доби. Так як критичним фактором є точна кількість можливої виробленої електроенергії, інтеграція програмного забезпечення представленого ТОВ «Oracle Solutions» у електросистему вітряних ферм матиме значний попит на ринку прогнозуючих систем. Індивідуальний підхід на всіх етапах реалізації та експлуатації продукту має забезпечити ТОВ «Oracle Solutions» постійним доходом на довгий період часу.

Так як критичним фактором є точна кількість можливої виробленої електроенергії, інтеграція програмного забезпечення представленого ТОВ «Oracle Solutions» у електросистему вітряних ферм матиме значний попит на ринку прогнозуючих систем. Індивідуальний підхід на всіх етапах реалізації та експлуатації продукту має забезпечити ТОВ «Oracle Solutions» постійним доходом на довгий період часу.

Ці ж переваги використання програмного продукту роблять енергію вітропарків більш надійною з точки зору стабільності та створюють попит і у підприємств з частковим енергозабезпеченням з вітряних станцій.

Конкурентні переваги. Прозора бізнес-модель. Допомога у впровадженні інноваційних технологій у досить молодій гілці енергетичної сфери в Україні. Спрощення шляху до енергетичної незалежності країни. Існуючі системи прогнозування видобутку вітряної енергії базуються або тільки на статистичних, або тільки на метеорологічних даних. Використання комбінованого підходу зі застосуванням штучних нейронних мереж пропонуються тільки компаніями-виробниками вітряних турбін. Так як українські компанії зазвичай не працюють на пряму з виробниками, або не мають змоги інтегрувати ПЗ у вже існуючі системи через використання застарілого (непідтримуваного) обладнання, послуги компанії ТОВ «Oracle Solutions» мають додаткову перевагу в порівняння з існуючими конкурентами.

Вартість готового продукту є сталою, а вартість впровадження ПЗ у конкретну мережу оговорюється індивідуально з Замовником в залежності від складності інтеграції та обраного пакету послуг.

Ринкова ціна складатиме близько 390 тис. грн/проект.

Період повернення капіталовкладень – 1,5 року.

6.3 АНАЛІЗ ЗОВНІШНЬОГО ТА ВНУТРІШНЬОГО СЕРЕДОВИЩА СТАРТАПУ

Таблиця 6.2 – Загрози і можливості зовнішнього середовища.

Таблиця 6.3 – Аналіз факторів зовнішнього оперативного середовища.

Таблиця 6.4 – Переваги і недоліки внутрішнього середовища.

6.4 КЛЮЧОВІ ФАКТОРИ УСПІХУ СТАРТАПУ ЗА МЕТОДОМ ШОНФІЛДА

На підставі аналізу факторів зовнішнього і зовнішнього оперативного середовищ було визначено ключові фактори успіху стартап проекту. Під ключовими факторами успіху розглянемо ті, на які підприємство може самостійно впливати під час надання послуг. Ключові фактори успіху надано у вигляді діаграми Шонфільда.

Таблиця 6.5 – Оцінки характеристики за методом Шонфільда.

З урахуванням коефіцієнту вагомості визначається оцінка кожної характеристики для послуг ТОВ «Oracle Solutions» та конкурентів табл. 5.6.

Таблиця 6.6 – Оцінки характеристик з урахуванням коефіцієнту вагомості

На підставі отриманих бальних оцінок будується графік порівняння конкурентних переваг нашого підприємства з конкурентами.

.

Рисунок 6.1 – Порівняння конкурентних переваг підприємства з конкурентами

З графіку видно, що наші послуги мають перевагу або розділяють лідируючу позицію по всім показникам окрім ціни. Проте, зважаючи на низьку

якість продукту з найнижчою ціною, можна стверджувати, що наші послуги будуть користуватися попитом.

6.5 РОЗРАХУНОК ОСНОВНИХ ТЕХНІКО-ЕКОНОМІЧНИХ ПОКАЗНИКІВ ПРОЕКТУ

Для фінансування стартапу використовуються власні кошти. В собівартість продукціїї закладаються усі витрати, а з прибутку формується фонд розвитку підприємства. Така модель фінансування є найбільш вигідною, оскільки ми маємо достатньо вільних коштів.

Для початку роботи ТОВ «Oracle Solutions» достатньо трьох інженерів, двох менеджерів, одного юриста, та одного економіста. Інженери відповідальні за: створення проектного плану, який оговорюється з замовником під час платних консультацій; технічний контроль та керівництво процесом впровадження, та адаптації продукту. Один з менеджерів відповідальний за співпрацю з клієнтами. Його обов’язками є: заключення договорів про співпрацю, керівництво командою інженерів, заключення договорів на поставку обладнання, контроль виконання цих процесів. Інший менеджер відповідальний за: пошук клієнтів, розвиток проекту, рекламу, юридичний та фінансовий контроль, дослідження ринку. Юрист та економіст відповідальні за свої профільні задачі.

Таблиця 6.7 – Перелік послуг

Розглянемо усі витрати та підрахуємо собівартість продукту.

Таблиця 6.7 – Основні фонди.

Розрахуємо оборотні засоби при реалізації одного продукту.

Таблиця 6.8 – Оборотні засоби.

Річні оборотні засоби:

$$\text{ОбЗ} = \text{Загальні витрати} * \text{к} - \text{ть проектів} + \text{ФОП} * 12 =$$

$$= 47\,500 * 25 + 378\,840 * 12 = 1\,187\,500 + 4\,498\,080 = \mathbf{5\,685\,580} \text{ грн/рік}$$

Собівартість: $\text{С} = \text{ОбЗ} + \text{А} = 5\,685\,580 + 91\,000 = 5\,776\,580$ грн

Собівартість одиниці продукту:

$$\text{С}_{\text{од}} = \frac{\text{С}}{N} = \frac{5\,776\,580}{25} = \mathbf{231\,064} \text{ грн}$$

Розрахуємо ціни за основними методами ціноутворення:

1. Метод, орієнтований на витрати (витратний метод):

$$\text{Ц} = \frac{\text{С}}{N} + C \cdot 1\% = \frac{5\,776\,580}{25} + 5\,776\,580 \cdot 1\% = 288\,829 \text{ грн/проект}$$

2. Агрегатний метод – застосовується до товарів із складових елементів:

Ц = Ц1 + Ц2 + … + Ці = 378 840/2 + 47 500 + 10 000 = 246 920 грн/проект

3. Параметричний метод – враховує вагомість якісних параметрів товару і оцінку цих параметрів споживачем:

$$\text{Ц}_{\text{новоїмоделі}} = \text{Ц}_{\text{базовоїмоделі}} \cdot \frac{\text{Баловаоцінкановоїмоделі}}{\text{Баловаоцінкабазовоїмоделі}} =$$

$$= 288\,829 \cdot \frac{1}{0{,}8} = 361\,037 \text{ грн/проект}$$

4. Метод ціноутворення на основі поточних цін або конкурентний метод.

Оскільки на українському ринку немає явних конкурентів, то можна обирати будь яку ціну вище за 361 037 грн. Оберемо 390 000 грн та приймемо до уваги, що ціну можна знизити в разі малого попиту.

Ціна запланованих проектів на рік:

$$\text{В}_{\text{рік}}^{\text{грн}} = \text{Ц} \cdot B_{\text{рік}} = 390\,000 \cdot 25 = 9\,750\,000 \text{ грн}$$

Ціна продукту: $Ц_{од}$ = 390 000 грн.

Прибуток: $П_{од} = Ц_{од} - С_{од} = 390\,000 - 231\,064 = 158\,936$ грн

Річний прибуток: $П_р = П_{од} \cdot N = 3\,973\,400$ грн

Капіталовкладення:

$$К = ОФ + ОбЗ = 5\,776\,580 + 161\,000 = 5\,937\,580 \text{ грн}$$

Рентабельність: $Р = \frac{П}{С} = \frac{3\,973\,400}{5\,776\,580} = 68,8\%$

Економічна ефективність: $Еф = \frac{П}{К} = \frac{3\,973\,400}{5\,937\,580} = 0,67$

Термін повернення капіталовкладень: $Т = \frac{К}{П} = \frac{5\,937\,580}{3\,973\,400} = 1,5$ роки

Всі визначені техніко-економічні показники зведено у таблицю 5.11.

З отриманих даних можемо розрахувати точку беззбитковості. Позначимо необхідну кількість проектів через Х. Тоді можна стверджувати наступне:

$$К + А + С \cdot Х = Ц \cdot Х, \text{ де } Х = \frac{К+А}{Ц-С}$$

$$Х = \frac{5\,937\,580 + 91\,000}{390\,000 - 231\,064} = \frac{6\,028\,580}{158\,936} = 38$$

Точка беззбитковості - 38 проектів.

Таблиця 6.9 – Основні техніко-економічні показники підприємства.

За знайденими техніко-економічними показниками можна зробити висновок, що підприємство є прибутковим.

6.6 КАРТА БІЗНЕС-ПРОЦЕСІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

Метою розробки стартапу є створення нового, економічно-вигідного підприємства, яке б задовольняло попит, та відповідало всім вимогам нормативних документів. В майбутньому планується розширення лінійки продуктів та вихід на міжнародний ринок.

Карта бізнес-процесів виконання стартап проекту

Системний аналіз бізнес-процесів стартапу

Оцінка ризиків та страхування розробки

Таблиця 6.12 – Ризики інноваційної розробки та ймовірність їх настання.

Таблиця 6.13 – Методи управління ризиками.

ВИСНОВКИ

У ході розробки магістерської дисертації було досліджено системи прогнозування генерації електроенергії вітряними станціями. Було визначено типи таких систем, а також методи на основі яких вони працюють. Різні системи прогнозування працюють на різні проміжки часу, короткострокові, середньострокові або довгострокові.

Чисельні методи передбачення ефективно працюють на довгострокових проміжках часу, так як у даній області дослідження використовують метеорологічні моделі атмосфери та океанів для прогнозування річного або десятирічного видобутку електроенергії.

Статистичні методи передбачення є основними конкурентами методів заснованих на використанні нейронних мереж. Сучасні нейромережеві методи передбачають кількість електрогенерації на короткострокові терміни з точністю до 85-90% і зазвичай базуються на історичних даних стану вітряної станції.

Застосування в даній магістерській дисертації ШНМ для прогнозування генерації електроенергії з використанням як історичних даних стану ВС так і даних прогнозу погоди в регіоні дало середню точність передбачення у 80-86,2%, а середньоквадратичне відхилення у 184,6 кВт, від середнього значення потужності у 1293,3 кВт, що є задовільним показником, особливо враховуючи фактор навчання на «брудних» даних. Фільтрація даних, використання

метеопрогнозу з декількох метеостанцій, вдосконалення моделі ШНМ та процесів навчання, на думку автора, може збільшити точність прогнозування ще на декілька відсотків.

Актуальність даної роботи є високою через світову тенденцію до поширення використання альтернативних джерел енергії, і вітряних станцій вцілому. Збільшення точності прогнозувань може полегшити інтеграцію вітряних парків в загальну елекромережу, збільшити частку гравців зі сфери «зеленої» енергетики на ринках електроенергії, зменшити собівартість вітряної електроенергії та збільшити довіру до вітряної енергетики вцілому.

Також, у ході розробки магістерської дистертації було опубліковано наступні статті:

1. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ЯК НЕОБХІДНИЙ ІНСТРУМЕНТ ДЛЯ ЕФЕКТИВНОГО ВИКОРИСТАННЯ БАЗ ДАНИХ СИСМЕТ SCADA.
   Topical issues of the development of modern science. Abstracts of the 3rd International scientific and practical conference. Publishing House “ACCENT”. Sofia, Bulgaria. 13-15 November 2019. Pp. 422-426.
2. Бугаєва Л. М., Сидоренко І. А. СИСТЕМА ПРОГНОЗУВАННЯ ГЕНЕРАЦІЇ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНОЮ СТАНЦІЄЮ ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ.
   Матеріали V Міжнародної науково-технічної конференції «Комп'ютерне моделювання та оптимізація складних систем», м. Дніпро, 6-8 листопада 2019 року. С. 111-113.
3. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ЗАСТОСУВАННЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДО БАЗ ДАНИХ SCADA СИСТЕМ.
   Всеукраїнська наукова Інтернет-конференція " Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення", 14 листопада 2019 р, Тернопіль. С.19-21

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ayodele T. Types of Machine Learning Algorithms / Taiwo Ayodele // New Advances in Machine Learning / Taiwo Ayodele., 2010.

2. Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning [Електронний ресурс] – Режим доступу до ресурсу: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervisedunsupervised-and-reinforcement-learning/.

3. Kamath U. Mastering Java Machine Learning / U. Kamath, K. Choppella., 2017.

4. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville., 2016. – (MIT Press).

5. Machine Learning Applications for Load and Price Forecasting and Wind Power Prediction in Power Systems [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/profile/Michael_Negnevitsky/publication/224089798_Machine_Learning_Applications_for_Load_Price_and_Wind_Power_Prediction_in_Power_Systems/links/540dacfc0cf2f2b29a39cf76.pdf

6. Chapelle O. Cluster kernels for semi-supervised learning / O. Chapelle, J. Weston, B. Schölkopf // Advances in Neural Information Processing Systems / O. Chapelle, J. Weston, B. Schölkopf., 2002.

7. Difference Between a Batch and an Epoch in a Neural Network [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

8. Світова енергетика в цифрах — що варто знати про 20в17-2040 роки [Електронний ресурс] – Режим доступу до ресурсу: https://nachasi.com/2017/06/22/energy-faq/

9. Krenker A. Introduction to the Artificial Neural Networks / A. Krenker, J. Bester, A. Kos // Artificial Neural Networks - Methodological Advances and Biomedical Applications / A. Krenker, J. Bester, A. Kos., 2011.

10. Cybenko G. Approximation by superpositions of a sigmoidal function / George Cybenko // Mathematics of Control, Signals and Systems / George Cybenko., 1989. 117

11. Snyman J. Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms / Jan Snyman., 2005. – (Springer US).

12. Looking inside neural nets [Електронний ресурс] – Режим доступу до ресурсу: https://ml4a.github.io/ml4a/looking_inside_neural_nets/.

13. Neural Network Zoo Prequel: Cells and Layers [Електронний ресурс] – Режим доступу до ресурсу: http://www.asimovinstitute.org/neural-network-zoo-prequelcells-layers/.

14. Neural Network Zoo [Електронний ресурс] – Режим доступу до ресурсу: http://www.asimovinstitute.org/neural-network-zoo/.

15. Quesada A. 5 algorithms to train a neural network [Електронний ресурс] / Alberto Quesada – Режим доступу до ресурсу: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network

16. Kukacka J. Regularization for Deep Learning: A Taxonomy / J. Kukacka, V. Golkov, D. Cremers. – 2017.

17. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ЯК НЕОБХІДНИЙ ІНСТРУМЕНТ ДЛЯ ЕФЕКТИВНОГО ВИКОРИСТАННЯ БАЗ ДАНИХ СИСМЕТ SCADA. Topical issues of the development of modern science. Abstracts of the 3rd International scientific and practical conference. Publishing House “ACCENT”. Sofia, Bulgaria. 13-15 November 2019. Pp. 422-426.

18. Бугаєва Л. М., Сидоренко І. А. СИСТЕМА ПРОГНОЗУВАННЯ ГЕНЕРАЦІЇ ЕЛЕКТРОЕНЕРГІЇ ВІТРЯНОЮ СТАНЦІЄЮ ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ. Матеріали V Міжнародної науково-технічної конференції «Комп'ютерне моделювання та оптимізація складних систем», м. Дніпро, 6-8 листопада 2019 року. С. 111-113.

19. Бугаєва Л.М., Безносик Ю.О., Сидоренко І.А. ЗАСТОСУВАННЯ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДО БАЗ ДАНИХ SCADA СИСТЕМ. Всеукраїнська наукова Інтернет-конференція " Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення", 14 листопада 2019 р, Тернопіль. С.19-21.

ДОДАТОК А

Програмний код модуля завантаження даних

Main.java

```
package parsers.weather;

import com.fasterxml.jackson.databind.JsonNode;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Main {

    //
https://www.timeanddate.com/scripts/cityajax.php?n=@862469&mode=historic&hd=20180102&m
onth=1&year=2018&json=1
    public static final String ENDPOINT_URL_TEMPLATE =

"https://www.timeanddate.com/scripts/cityajax.php?n=@862469&mode=historic&hd=%s&month=
%d&year=%d&json=1";

    public static final short YEAR = 2018;


    public static void main(String[] args) throws IOException {

        LocalDate date = LocalDate.of(2018, 1, 1);
        PrintWriter fileWriter = new PrintWriter(new File("weather.csv"));
        ResponseHandler responseHandler = new WeatherResponseHandler(fileWriter);

        while (date.getYear() == YEAR) {
            JsonNode jsonResponse =
HttpRequestExecutor.sendRequest(resolveEndpoint(date));
            responseHandler.handleResponse(jsonResponse, date);
            date = date.plusDays(1);
        }

        fileWriter.flush();
        fileWriter.close();
        System.out.println("*****Data loading Completed*****");
    }

    private static String resolveEndpoint(LocalDate date) {
        return String.format(ENDPOINT_URL_TEMPLATE,
date.format(DateTimeFormatter.ofPattern("yyyyMMdd")), date.getMonthValue(),
date.getYear());
    }
}
```

ResponseHandler.java

```
package parsers.weather;
```

```java
import com.fasterxml.jackson.databind.JsonNode;

import java.time.LocalDate;

public interface ResponseHandler {

    void handleResponse(JsonNode response, LocalDate date);
}
```

WeatherResponseHandler.java

```java
package parsers.weather;


import com.fasterxml.jackson.databind.JsonNode;

import java.io.PrintWriter;
import java.time.LocalDate;
import java.util.Spliterator;
import java.util.stream.Stream;
import java.util.stream.StreamSupport;

public class WeatherResponseHandler implements ResponseHandler {

    private PrintWriter csvWriter;

    public WeatherResponseHandler(PrintWriter csvWriter) {
        this.csvWriter = csvWriter;
    }

    @Override
    public void handleResponse(JsonNode node, LocalDate date) {
        WeatherToCsvRecordAdapter toCsvAdapter = new WeatherToCsvRecordAdapter();
        toCsvAdapter.setDate(date);
        nodeToStream(node.spliterator())
                .map(n -> n.get("c"))
                .map(toCsvAdapter)
                .forEach(csvWriter::println);
    }

    private Stream<JsonNode> nodeToStream(Spliterator<JsonNode> spliterator) {
        return StreamSupport.stream(spliterator, false);
    }
}
```

HttpRequestExecutor.java

```java
package parsers.weather;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import javax.net.ssl.HttpsURLConnection;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;

public final class HttpRequestExecutor {

    private static final ObjectMapper MAPPER = new ObjectMapper();

    static {
        MAPPER.configure(JsonParser.Feature.ALLOW_SINGLE_QUOTES, true);
        MAPPER.configure(JsonParser.Feature.ALLOW_UNQUOTED_FIELD_NAMES, true);
    }
```

```
    public static JsonNode sendRequest(String url) throws IOException {
        return MAPPER.readTree(new URL(url));
    }

    private static HttpsURLConnection getConnection(String host) {
        try {
            URL url = new URL(host);
            return (HttpsURLConnection) url.openConnection();
        } catch (IOException e) {
            throw new RuntimeException("Unable to connect to " + host, e);
        }
    }

    private static String getContent(HttpsURLConnection con) {
        StringBuilder content = new StringBuilder();

        try (BufferedReader br = new BufferedReader(new
InputStreamReader(con.getInputStream()))) {

            System.out.println("****** Content of the URL ********");
            String input;
            while ((input = br.readLine()) != null) {
                content.append(input);
            }
        } catch (IOException e) {
            throw new RuntimeException("Exception during reading response from
server.", e);
        }
        System.out.println("****** End of content of the URL ********");
        return content.toString();
    }
}
```

WeatherResponseHandler.java

```
package parsers.weather;


import com.fasterxml.jackson.databind.JsonNode;

import java.io.PrintWriter;
import java.time.LocalDate;
import java.util.Spliterator;
import java.util.stream.Stream;
import java.util.stream.StreamSupport;

public class WeatherResponseHandler implements ResponseHandler {

    private PrintWriter csvWriter;

    public WeatherResponseHandler(PrintWriter csvWriter) {
        this.csvWriter = csvWriter;
    }

    @Override
    public void handleResponse(JsonNode node, LocalDate date) {
        WeatherToCsvRecordAdapter toCsvAdapter = new WeatherToCsvRecordAdapter();
        toCsvAdapter.setDate(date);
        nodeToStream(node.spliterator())
                .map(n -> n.get("c"))
                .map(toCsvAdapter)
                .forEach(csvWriter::println);
    }

    private Stream<JsonNode> nodeToStream(Spliterator<JsonNode> spliterator) {
        return StreamSupport.stream(spliterator, false);
```

```
    }
}
```

WeatherRecordParser.java

```
package parsers.weather;

import com.fasterxml.jackson.databind.JsonNode;

import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public final class WeatherRecordParser {

    public static LocalTime parseTime(JsonNode node) {
        String time = getRowData(node, 0).asText();
        time = time.contains("<") ? time.substring(0, time.indexOf("<")) : time;
        String pattern = time.length() < 5 ? "H:mm" : "HH:mm";
        return LocalTime.parse(time, DateTimeFormatter.ofPattern(pattern));
    }

    public static byte parseTemperature(JsonNode node) {
        return (byte) parseIntByFirstSymbolMeets(node, 2, '&');
    }

    public static short parseWindSpeed(JsonNode node) {
        return (short) parseIntByFirstSymbolMeetsIfValuePresent(node, 4, ' ', "No
wind", 0);
    }

    public static String parseWindDirection(JsonNode node) {
        Pattern pattern = Pattern.compile("<span .*title=\"(.+)\".*>.*</span>");
        String value = getRowData(node, 5).asText();
        Matcher matcher = pattern.matcher(value);
        if (matcher.find()) {
            return matcher.group(1);
        }
        throw new RuntimeException("Could not parse Wind Direction");
    }

    public static byte parseHumidity(JsonNode node) {
        return (byte) parseIntByFirstSymbolMeets(node, 6, '%');
    }

    public static short parseBarometer(JsonNode node) {
        return (short) parseIntByFirstSymbolMeets(node, 7, ' ');
    }

    public static JsonNode getRowData(JsonNode parent, int index) {
        return parent.get(index).get("h");
    }

    public static short parseVisibility(JsonNode node) {
        return (short) parseIntByFirstSymbolMeetsIfValuePresent(node, 8, '&', "N/A", -
1);
    }

    private static int parseIntByFirstSymbolMeets(JsonNode node, int index, char symb)
```

```
{
        return parseIntByFirstSymbolMeetsIfValuePresent(node, index, symb, null, -1);
    }

    private static int parseIntByFirstSymbolMeetsIfValuePresent(JsonNode node, int
index, char symb, String noNumberValue, int defVal) {
        String value = getRowData(node, index).asText();
        if (value.equals(noNumberValue)) {
            return defVal;
        }
        int indexOf = value.indexOf(symb);
        return Short.valueOf(value.substring(0, indexOf));
    }
}
```

WeatherToCsvRecordAdapter.java

```
package parsers.weather;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.StringJoiner;
import java.util.function.Function;

import static parsers.weather.WeatherRecordParser.*;

import com.fasterxml.jackson.databind.JsonNode;


public final class WeatherToCsvRecordAdapter implements Function<JsonNode, String> {

    private LocalDate date;

    public WeatherToCsvRecordAdapter(LocalDate date) {
        this.date = date;
    }

    public WeatherToCsvRecordAdapter() {

    }

    public LocalDate getDate() {
        return date;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }

    @Override
    public String apply(JsonNode node) {
        StringJoiner stringJoiner = new StringJoiner(",");
        LocalDateTime ldt = LocalDateTime.of(date, parseTime(node));
        stringJoiner
                .add(ldt.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")))
                .add(String.valueOf(parseTemperature(node)))
                .add(getRowData(node, 3).asText())
                .add(String.valueOf(parseWindSpeed(node)))
                .add(String.valueOf(parseHumidity(node)))
                .add(String.valueOf(parseBarometer(node)))
                .add(parseWindDirection(node))
                .add(String.valueOf(parseVisibility(node)));
        return stringJoiner.toString();
    }
}
```

ДОДАТОК Б

Модуль формалізації та збереження даних

MainDatasetFormatAdapter.java

```
package processors;

public class MainDatasetFormatAdapter {

    private static final int WEATHER_RECORDS_PER_ROW_COUNT = 16;
    private static final int TURBINE_DATA_RECORDS_PER_ROW_COUNT = 24;

    public static void main(String[] args) {
        CsvFileModifier.adapt("weather.csv", "weather_1.csv",
                new CsvRowToColumnByDateTransposerCommand(30,
WEATHER_RECORDS_PER_ROW_COUNT, "yyyy-MM-dd HH:mm:ss"));
        CsvFileModifier.adapt("T1.csv", "T1_1.csv",
                new ActivePowerSumCommandByDate(10,
TURBINE_DATA_RECORDS_PER_ROW_COUNT, "dd MM yyyy HH:mm", 24));
    }
}
```

CsvFileModifier.java

```
package processors;

import java.io.*;
import java.util.Arrays;
import java.util.function.Function;

public final class CsvFileModifier {

    private static final CsvFileDataProcessor dataProcessor = new
CsvFileDataProcessor();

    public static void adapt(String oldFileName, String newFileName, Function<String,
?> adapterCommand) {
        try (FileWriter out = new FileWriter(newFileName)) {
            String data =
readStream(CsvFileModifier.class.getClassLoader().getResource(oldFileName).openStream(
));
            String result = dataProcessor.process(data, adapterCommand);
            String[] lines = result.split("\n");
            Arrays.stream(lines).forEach(l -> System.out.println("Col count: " +
l.split(",").length));
            System.out.println("Result file lines count: " + lines.length);
            out.write(result);
            out.flush();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    private static String readStream(InputStream inputStream) throws IOException {
```

```
        StringBuilder sb = new StringBuilder();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
        int linesReached = 0;
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line).append("\n");
            linesReached++;
        }
        reader.close();
        System.out.println("Lines found in file: " + linesReached);
        return sb.toString();
    }
}
```

DataProcessor.java

```
package processors;

import java.util.function.Function;

public interface DataProcessor<T, R> {

    T process(T data, Function<T, R> command);
}
```

CsvFileDataProcessor.java

```
package processors;

import java.util.function.Function;

public class CsvFileDataProcessor<R> implements DataProcessor<String, R> {

    @Override
    public String process(String data, Function<String, R> command) {
        Object res = null;

        String[] csvRows = data.split("\n");
        for (int i = 0; i < csvRows.length; i++) {
            res = command.apply(csvRows[i]);
        }
        return res.toString();
    }
}
```

CsvRowToColumnByDateTransposerCommand.java

```
package processors;

import utils.CsvDataExtractorHelper;

import java.util.Arrays;
import java.util.StringJoiner;
import java.util.function.Function;
import java.util.stream.Collectors;

public class CsvRowToColumnByDateTransposerCommand implements Function<String,
StringBuilder> {

    protected static final String DELIMITER = ",";

    protected final int recordsTimeInterval;
    protected final int rowsToTranformNumber;
    protected final StringBuilder sb = new StringBuilder();
    protected final CsvDataExtractorHelper csvUtils;

    protected StringJoiner localBuffer = new StringJoiner(DELIMITER);
```

```
    protected int rowsCounter = 0;
    protected String previousRow = null;

    private boolean isTriggered = false;

    public CsvRowToColumnByDateTransposerCommand(int recordsTimeInterval, int
rowsToTransformNumber ,String datePattern) {
        this.recordsTimeInterval = recordsTimeInterval;
        this.rowsToTranformNumber = rowsToTransformNumber;
        csvUtils = CsvDataExtractorHelper.build(datePattern, DELIMITER);
    }

    @Override
    public StringBuilder apply(String s) {
        if (rowsCounter >= rowsToTranformNumber) {
            flushLocalBuffer();
        }

        if (isRowMissedInInterval(s)) {
            previousRow = csvUtils.incrementRowDate(previousRow, recordsTimeInterval);
            localBuffer.add(previousRow);
            rowsCounter++;
            return apply(s);
        }

        localBuffer.add(s);
        rowsCounter++;
        previousRow = s;

        if (isStringIsFileHeader()) {
            extendFileHeaderForNewFields(s);
            flushLocalBuffer();
        }
        return sb;
    }

    protected void flushLocalBuffer() {
        sb.append(localBuffer.toString());
        sb.append("\n");
        localBuffer = new StringJoiner(DELIMITER);
        rowsCounter = 0;
    }

    protected final boolean isRowMissedInInterval(String currentRow) {
        return previousRow != null && csvUtils.getTimeDiffBetweenRows(currentRow,
previousRow) >= 1.5 * recordsTimeInterval;
    }

    protected final boolean isRowIntervalTooSmall(String currentRow) {
        if (previousRow != null) {
            long timeDiff = csvUtils.getTimeDiffBetweenRows(currentRow, previousRow);
            return timeDiff > 0 && timeDiff < recordsTimeInterval;
        }
        return false;
    }

    private boolean isStringIsFileHeader() {
        if (isTriggered) {
            return false;
        } else {
            isTriggered = true;
            return true;
        }
    }

    protected void extendFileHeaderForNewFields(String header) {
        String[] columns = header.split(DELIMITER);
        for (int i = 1; i < rowsToTranformNumber; i++) {
            final int columnNumber = i;
```

```
            localBuffer.add(Arrays.stream(columns).map(c -> c + "_" +
columnNumber).collect(Collectors.joining(DELIMITER)));
        }
    }
}
```

ActivePowerSumCommandByDate.java

```
package processors;

public class ActivePowerSumCommandByDate extends CsvRowToColumnByDateTransposerCommand
{

    private final int sumPowerRowsCount;

    private Double activePowerSum;

    public ActivePowerSumCommandByDate(int rowsTimeGapInterval, int rowsNumber, String
datePattern, int sumPowerRowsNumber) {
        super(rowsTimeGapInterval, rowsNumber, datePattern);
        this.sumPowerRowsCount = rowsNumber + sumPowerRowsNumber;
        activePowerSum = 0d;
    }

    @Override
    public StringBuilder apply(String s) {
        if (rowsCounter >= rowsToTranformNumber && rowsCounter < sumPowerRowsCount) {
            if (isRowMissedInInterval(s)) {
                activePowerSum +=
Double.valueOf(csvUtils.getRowColumnValue(previousRow, 1));
                rowsCounter++;
                previousRow = csvUtils.incrementRowDate(previousRow,
recordsTimeInterval);
                return apply(s);
            }
            activePowerSum += Double.valueOf(csvUtils.getRowColumnValue(s, 1));
            previousRow = s;
            rowsCounter++;

            return sb;
        } else if (rowsCounter == sumPowerRowsCount) {
            localBuffer.add(activePowerSum.toString());
            flushLocalBuffer();
        }
        return super.apply(s);
    }

    @Override
    protected void flushLocalBuffer() {
        super.flushLocalBuffer();
        activePowerSum = 0d;
    }

    @Override
    protected void extendFileHeaderForNewFields(String header) {
        super.extendFileHeaderForNewFields(header);
        localBuffer.add("ActivePowerSum");
    }
}
```

CsvDataExtractorHelper.java

```java
package utils;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.time.temporal.ChronoUnit;

public final class CsvDataExtractorHelper {

    private final String DELIMITER;
    private final DateTimeFormatter DATE_FORMATTER;

    private CsvDataExtractorHelper(String dateTimePattern, String delimiter) {
        this.DELIMITER = delimiter;
        this.DATE_FORMATTER = DateTimeFormatter.ofPattern(dateTimePattern);
    }

    public static CsvDataExtractorHelper build(String dateTimePattern, String
delimiter) {
        return new CsvDataExtractorHelper(dateTimePattern, delimiter);
    }

    public String getRowColumnValue(String row, int columnNumber) {
        return row.split(DELIMITER)[columnNumber];
    }

    public LocalDateTime getRowDateValue(String row, int dateColNumber) {
        return LocalDateTime.parse(getRowColumnValue(row, dateColNumber),
DATE_FORMATTER);
    }

    public long getTimeDiffBetweenRows(String row, String previousRow) {
        try {
            LocalDateTime currentRowTime = getRowDateValue(row, 0);
            LocalDateTime prevRowTime = getRowDateValue(previousRow, 0);
            return Math.abs(ChronoUnit.MINUTES.between(currentRowTime, prevRowTime));
        } catch (DateTimeParseException e) {
            System.out.println("No pattern suitable Date column found");
            return -1;
        }
    }

    public String incrementRowDate(String row, int minutesToIncrease) {
        int dateColNumber = 0;
        String[] cols = row.split(DELIMITER);
        LocalDateTime rowDate = getRowDateValue(row, dateColNumber);
        rowDate = rowDate.plusMinutes(minutesToIncrease);
        cols[dateColNumber] = DATE_FORMATTER.format(rowDate);
        return String.join(DELIMITER, cols);
    }

}
```

ДОДАТОК В

Python скрипт побудови та навчання штучної нейронної мережі

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import math
from collections import Counter
%matplotlib inline
import openpyxl
import plotly.plotly as py
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import tensorflow as tf

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the
files in the input directory

import os

data_dir = '/home/wind-station/data/'
t1=pd.read_csv(data_dir + 'T1_1.csv')
weather=pd.read_csv(data_dir + 'weather_1.csv')
_weather = weather.copy()
_t1 = t1.copy()
_weather['Time'] = pd.to_datetime(_weather['Time'])
_weather['Time_1'] = pd.to_datetime(_weather['Time_1'])
_weather['Time_2'] = pd.to_datetime(_weather['Time_2'])
_weather['Time_3'] = pd.to_datetime(_weather['Time_3'])
_weather['Time_4'] = pd.to_datetime(_weather['Time_4'])
_weather['Time_5'] = pd.to_datetime(_weather['Time_5'])
_weather['Time_6'] = pd.to_datetime(_weather['Time_6'])
_weather['Time_7'] = pd.to_datetime(_weather['Time_7'])
_weather['Time_8'] = pd.to_datetime(_weather['Time_8'])
_weather['Time_9'] = pd.to_datetime(_weather['Time_9'])
_weather['Time_10'] = pd.to_datetime(_weather['Time_10'])
_weather['Time_11'] = pd.to_datetime(_weather['Time_11'])
_weather['Time_12'] = pd.to_datetime(_weather['Time_12'])
_weather['Time_13'] = pd.to_datetime(_weather['Time_13'])
_weather['Time_14'] = pd.to_datetime(_weather['Time_14'])
_weather['Time_15'] = pd.to_datetime(_weather['Time_15'])
_t1['Time'] = pd.to_datetime(_t1['Time'])
_t1['Time_1'] = pd.to_datetime(_t1['Time_1'])
_t1['Time_2'] = pd.to_datetime(_t1['Time_2'])
_t1['Time_3'] = pd.to_datetime(_t1['Time_3'])
_t1['Time_4'] = pd.to_datetime(_t1['Time_4'])
_t1['Time_5'] = pd.to_datetime(_t1['Time_5'])
_t1['Time_6'] = pd.to_datetime(_t1['Time_6'])
_t1['Time_7'] = pd.to_datetime(_t1['Time_7'])
```

```
_t1['Time_8'] = pd.to_datetime(_t1['Time_8'])
_t1['Time_9'] = pd.to_datetime(_t1['Time_9'])
_t1['Time_10'] = pd.to_datetime(_t1['Time_10'])
_t1['Time_11'] = pd.to_datetime(_t1['Time_11'])
_t1['Time_12'] = pd.to_datetime(_t1['Time_12'])
_t1['Time_13'] = pd.to_datetime(_t1['Time_13'])
_t1['Time_14'] = pd.to_datetime(_t1['Time_14'])
_t1['Time_15'] = pd.to_datetime(_t1['Time_15'])
_t1['Time_16'] = pd.to_datetime(_t1['Time_16'])
_t1['Time_17'] = pd.to_datetime(_t1['Time_17'])
_t1['Time_18'] = pd.to_datetime(_t1['Time_18'])
_t1['Time_19'] = pd.to_datetime(_t1['Time_19'])
_t1['Time_20'] = pd.to_datetime(_t1['Time_20'])
_t1['Time_21'] = pd.to_datetime(_t1['Time_21'])
_t1['Time_22'] = pd.to_datetime(_t1['Time_22'])
_t1['Time_23'] = pd.to_datetime(_t1['Time_23'])

#
# def check_no_wind(wind_val):
#     if 'No' in wind_val:
#         return 0
#     else:
#         return wind_val

def kmh_to_ms(km):
    return km * 0.27777778

def get_wind_direction_degrees(str):
     for s in str.split():
         s = s.replace('°','')
         if s.isdigit(): return int(s)

def replace_weather_with_index(str):
    weather_states = ['Passing clouds.', 'Scattered clouds.', 'Sunny.', 'Clear.',
                      'Partly cloudy.', 'Thundershowers. Passing clouds.',
                      'Thunderstorms. Passing clouds.', 'Sprinkles. Partly cloudy.',
                      'Broken clouds.', 'Sprinkles. Broken clouds.',
                      'Sprinkles. Partly sunny.', 'Partly sunny.',
                      'Sprinkles. Passing clouds.', 'Fog.', 'More clouds than sun.',
                      'Sprinkles. Fog.', 'Light rain. More clouds than sun.',
                      'Light rain. Mostly cloudy.', 'Light rain. Partly cloudy.',
                      'Mostly cloudy.', 'Light rain. Overcast.', 'Overcast.',
                      'Light rain. Broken clouds.', 'Light rain. Cloudy.',
                      'Light rain. Partly sunny.', 'Sprinkles. More clouds than sun.',
                      'Rain showers. Passing clouds.', 'Sprinkles. Cloudy.',
                      'Snow flurries. Broken clouds.', 'Snow flurries. Partly sunny.',
                      'Snow showers. Broken clouds.', 'Snow flurries. Passing
clouds.',
                      'Sprinkles. Overcast.', 'Rain showers. Mostly cloudy.',
                      'Rain showers. Partly cloudy.', 'Rain showers. Broken clouds.',
                      'Drizzle. Fog.', 'Drizzle. Mostly cloudy.', 'Drizzle.
Overcast.',
                      'Low clouds.', 'Sprinkles. Mostly cloudy.',
                      'Drizzle. More clouds than sun.', 'Cloudy.',
                      'Drizzle. Broken clouds.', 'Snow flurries. Mostly cloudy.',
                      'Light mixture of precip. Mostly cloudy.',
                      'Thundershowers. Partly sunny.', 'Thunderstorms. Broken
clouds.',
                      'Rain showers. Partly sunny.', 'Quite cool.', 'Cool.',
                      'Rain. Cloudy.', 'Rain. Scattered clouds.', 'Rain. Overcast.',
                      'Snow. Overcast.', 'Thundershowers. Broken clouds.',
                      'Light rain. Fog.', 'Light rain. Passing clouds.',
                      'Rain. Mostly cloudy.', 'Thunderstorms. Partly sunny.',
                      'Rain. Fog.', 'Thundershowers. Partly cloudy.',
                      'Rain showers. Fog.', 'Thunderstorms. Partly cloudy.',
                      'Strong thunderstorms. Broken clouds.', 'Thunderstorms.
Overcast.',
                      'Thunderstorms. Cloudy.', 'Thunderstorms. Mostly cloudy.',
                      'Hail. Partly cloudy.', 'Scattered showers. Broken clouds.',
```

```
                    'Thundershowers. Scattered clouds.',
                    'Sprinkles. Scattered clouds.', 'Light fog.']
    return weather_states.index(str)


def replace_NaN_visibility(val):
    return -1 if math.isnan(val) else val


# _weather['WindSpeed'] = _weather['WindSpeed'].apply(check_no_wind)
_weather['WindSpeed'] = _weather['WindSpeed'].astype(float)
_weather['WindSpeed'] = _weather['WindSpeed'].apply(kmh_to_ms)
_weather['WindSpeed_1'] = _weather['WindSpeed_1'].astype(float)
_weather['WindSpeed_1'] = _weather['WindSpeed_1'].apply(kmh_to_ms)
_weather['WindSpeed_2'] = _weather['WindSpeed_2'].astype(float)
_weather['WindSpeed_2'] = _weather['WindSpeed_2'].apply(kmh_to_ms)
_weather['WindSpeed_3'] = _weather['WindSpeed_3'].astype(float)
_weather['WindSpeed_3'] = _weather['WindSpeed_3'].apply(kmh_to_ms)
_weather['WindSpeed_4'] = _weather['WindSpeed_4'].astype(float)
_weather['WindSpeed_4'] = _weather['WindSpeed_4'].apply(kmh_to_ms)
_weather['WindSpeed_5'] = _weather['WindSpeed_5'].astype(float)
_weather['WindSpeed_5'] = _weather['WindSpeed_5'].apply(kmh_to_ms)
_weather['WindSpeed_6'] = _weather['WindSpeed_6'].apply(kmh_to_ms)
_weather['WindSpeed_6'] = _weather['WindSpeed_6'].astype(float)
_weather['WindSpeed_7'] = _weather['WindSpeed_7'].apply(kmh_to_ms)
_weather['WindSpeed_7'] = _weather['WindSpeed_7'].apply(kmh_to_ms)
_weather['WindSpeed_8'] = _weather['WindSpeed_8'].apply(kmh_to_ms)
_weather['WindSpeed_8'] = _weather['WindSpeed_8'].apply(kmh_to_ms)
_weather['WindSpeed_9'] = _weather['WindSpeed_9'].apply(kmh_to_ms)
_weather['WindSpeed_9'] = _weather['WindSpeed_9'].apply(kmh_to_ms)
_weather['WindSpeed_10'] = _weather['WindSpeed_10'].apply(kmh_to_ms)
_weather['WindSpeed_10'] = _weather['WindSpeed_10'].apply(kmh_to_ms)
_weather['WindSpeed_11'] = _weather['WindSpeed_11'].apply(kmh_to_ms)
_weather['WindSpeed_11'] = _weather['WindSpeed_11'].apply(kmh_to_ms)
_weather['WindSpeed_12'] = _weather['WindSpeed_12'].apply(kmh_to_ms)
_weather['WindSpeed_12'] = _weather['WindSpeed_12'].apply(kmh_to_ms)
_weather['WindSpeed_13'] = _weather['WindSpeed_13'].apply(kmh_to_ms)
_weather['WindSpeed_13'] = _weather['WindSpeed_13'].apply(kmh_to_ms)
_weather['WindSpeed_14'] = _weather['WindSpeed_14'].apply(kmh_to_ms)
_weather['WindSpeed_14'] = _weather['WindSpeed_14'].apply(kmh_to_ms)
_weather['WindSpeed_15'] = _weather['WindSpeed_15'].apply(kmh_to_ms)
_weather['WindSpeed_15'] = _weather['WindSpeed_15'].apply(kmh_to_ms)

res = _t1.copy()
res = res.merge(_weather, left_index=True, right_index=True)
res.info()

# res.rename(columns={' Weather': 'Weather'}, inplace=True)
res['WindDir_y'] = res['WindDir_y'].apply(get_wind_direction_degrees)
res['WindDir_1_y'] = res['WindDir_1_y'].apply(get_wind_direction_degrees)
res['WindDir_2_y'] = res['WindDir_2_y'].apply(get_wind_direction_degrees)
res['WindDir_3_y'] = res['WindDir_3_y'].apply(get_wind_direction_degrees)
res['WindDir_4_y'] = res['WindDir_4_y'].apply(get_wind_direction_degrees)
res['WindDir_5_y'] = res['WindDir_5_y'].apply(get_wind_direction_degrees)
res['WindDir_6_y'] = res['WindDir_6_y'].apply(get_wind_direction_degrees)
res['WindDir_7_y'] = res['WindDir_7_y'].apply(get_wind_direction_degrees)
res['WindDir_8_y'] = res['WindDir_8_y'].apply(get_wind_direction_degrees)
res['WindDir_9_y'] = res['WindDir_9_y'].apply(get_wind_direction_degrees)
res['WindDir_10_y'] = res['WindDir_10_y'].apply(get_wind_direction_degrees)
res['WindDir_11_y'] = res['WindDir_11_y'].apply(get_wind_direction_degrees)
res['WindDir_12_y'] = res['WindDir_12_y'].apply(get_wind_direction_degrees)
res['WindDir_13_y'] = res['WindDir_13_y'].apply(get_wind_direction_degrees)
res['WindDir_14_y'] = res['WindDir_14_y'].apply(get_wind_direction_degrees)
res['WindDir_15_y'] = res['WindDir_15_y'].apply(get_wind_direction_degrees)
res_learn = res.copy()

res_learn['Day_x'] = res_learn['Time_x'].dt.day
res_learn['Day_y'] = res_learn['Time_y'].dt.day
res_learn['Month_x'] = res_learn['Time_x'].dt.month
```

```
res_learn['Month_y'] = res_learn['Time_y'].dt.month
res_learn = res_learn.drop('Time_x', axis=1)
res_learn = res_learn.drop('Time_y', axis=1)

for i in range(1, 16):
    # res_learn['Day_' + str(i) + '_x'] = res_learn['Time_' + str(i) + '_x'].dt.day
    # res_learn['Day_' + str(i) + '_y'] = res_learn['Time_' + str(i) + '_y'].dt.day
    # res_learn['Month_' + str(i) + '_x'] = res_learn['Time_' + str(i) +
'_x'].dt.month
    # res_learn['Month_' + str(i) + '_y'] = res_learn['Time_' + str(i) +
'_y'].dt.month
    res_learn = res_learn.drop('Time_' + str(i) + '_x', axis=1)
    res_learn = res_learn.drop('Time_' + str(i) + '_y', axis=1)

for i in range(16, 24):
    res_learn = res_learn.drop('Time_' + str(i), axis=1)
    res_learn = res_learn.drop('Time_' + str(i), axis=1)

res_learn = res_learn.drop('Time_17', axis=1)
res_learn = res_learn.drop('Time_18', axis=1)
res_learn = res_learn.drop('Time_19', axis=1)
res_learn = res_learn.drop('Time_20', axis=1)
res_learn = res_learn.drop('Time_21', axis=1)
res_learn = res_learn.drop('Time_22', axis=1)
res_learn = res_learn.drop('Time_23', axis=1)
rl = res_learn

rl.Conditions = rl.Conditions.apply(replace_weather_with_index)
rl.Conditions_1 = rl.Conditions_1.apply(replace_weather_with_index)
rl.Conditions_2 = rl.Conditions_2.apply(replace_weather_with_index)
rl.Conditions_3 = rl.Conditions_3.apply(replace_weather_with_index)
rl.Conditions_4 = rl.Conditions_4.apply(replace_weather_with_index)
rl.Conditions_5 = rl.Conditions_5.apply(replace_weather_with_index)
rl.Conditions_6 = rl.Conditions_6.apply(replace_weather_with_index)
rl.Conditions_7 = rl.Conditions_7.apply(replace_weather_with_index)
rl.Conditions_8 = rl.Conditions_8.apply(replace_weather_with_index)
rl.Conditions_9 = rl.Conditions_9.apply(replace_weather_with_index)
rl.Conditions_10 = rl.Conditions_10.apply(replace_weather_with_index)
rl.Conditions_11 = rl.Conditions_11.apply(replace_weather_with_index)
rl.Conditions_12 = rl.Conditions_12.apply(replace_weather_with_index)
rl.Conditions_13 = rl.Conditions_13.apply(replace_weather_with_index)
rl.Conditions_14 = rl.Conditions_14.apply(replace_weather_with_index)
rl.Conditions_15 = rl.Conditions_15.apply(replace_weather_with_index)
# rl.Visibility = rl.Visibility.apply(replace_NaN_visibility)

indexOfTarget = rl.columns.get_loc('ActivePowerSum')
# get a list of the columns
col_list = list(rl)
# use this handy way to swap the elements
col_list[0], col_list[indexOfTarget] = col_list[indexOfTarget], col_list[0]
# assign back, the order will now be swapped
rl.columns = col_list

n = rl.shape[0]
p = rl.shape[1]
data = rl.values
train_start = 0
train_end = int(np.floor(0.8*n))
test_start = train_end
test_end = n
data_train = data[np.arange(train_start, train_end), :]
data_test = data[np.arange(test_start, test_end), :]
# Масштабирование данных
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)
```

```
X_train = data_train[:, 1:]
y_train = data_train[:, 0]
X_test = data_test[:, 1:]
y_test = data_test[:, 0]


records = 212
n_neurons_1 = 32
n_neurons_2 = 16
n_neurons_3 = 6
n_neurons_4 = 2
n_target = 1

X = tf.placeholder(dtype=tf.float32, shape=[None, records])
Y = tf.placeholder(dtype=tf.float32, shape=[None])

sigma = 1
weight_initializer = tf.variance_scaling_initializer(mode="fan_avg",
distribution="uniform", scale=sigma)
bias_initializer = tf.zeros_initializer()

W_hidden_1 = tf.Variable(weight_initializer([records, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
W_hidden_2 = tf.Variable(weight_initializer([n_neurons_1, n_neurons_2]))
bias_hidden_2 = tf.Variable(bias_initializer([n_neurons_2]))
W_hidden_3 = tf.Variable(weight_initializer([n_neurons_2, n_neurons_3]))
bias_hidden_3 = tf.Variable(bias_initializer([n_neurons_3]))
W_hidden_4 = tf.Variable(weight_initializer([n_neurons_3, n_neurons_4]))
bias_hidden_4 = tf.Variable(bias_initializer([n_neurons_4]))


W_out = tf.Variable(weight_initializer([n_neurons_4, n_target]))
bias_out = tf.Variable(bias_initializer([n_target]))

hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
hidden_2 = tf.nn.relu(tf.add(tf.matmul(hidden_1, W_hidden_2), bias_hidden_2))
hidden_3 = tf.nn.relu(tf.add(tf.matmul(hidden_2, W_hidden_3), bias_hidden_3))
hidden_4 = tf.nn.relu(tf.add(tf.matmul(hidden_3, W_hidden_4), bias_hidden_4))

out = tf.transpose(tf.add(tf.matmul(hidden_4, W_out), bias_out))

mse = tf.reduce_mean(tf.squared_difference(out, Y))

opt = tf.train.AdamOptimizer().minimize(mse)

net = tf.Session()
writer = tf.summary.FileWriter('logs', net.graph)
net.run(tf.global_variables_initializer())

plt.ion()
fig = plt.figure()
ax1 = fig.add_subplot(111)
line1, = ax1.plot(y_test)
line2, = ax1.plot(y_test*0.5)
plt.show()

epochs = 30
batch_size = 5
out_res = data_test.copy()
for e in range(epochs):
    shuffle_indices = np.random.permutation(np.arange(len(y_train)))
    X_train = X_train[shuffle_indices]
    y_train = y_train[shuffle_indices]
    for i in range(0, len(y_train) // batch_size):
        start = i * batch_size
        batch_x = X_train[start:start + batch_size]
        batch_y = y_train[start:start + batch_size]
        # Run optimizer with batch
        net.run(opt, feed_dict={X: batch_x, Y: batch_y})
```

```
        if np.mod(i, 5) == 0:
            # Prediction
            pred = net.run(out, feed_dict={X: X_test})
            out_res[:, 0] = pred
            line2.set_ydata(pred)
            plt.title('Epoch ' + str(e) + ', Batch ' + str(i))
            file_name = '/home/wind-station/img/epoch_' + str(e) + '_batch_' + str(i)
+ '.jpg'
            plt.savefig(file_name)
            plt.pause(0.01)



writer.close()
mse_final = net.run(mse, feed_dict={X: X_test, Y: y_test})
print(mse_final)
out_res = scaler.inverse_transform(out_res)
# print(out_res[:, 0])

def mean_square(arr1, arr2):
    n = arr1[:, 0].size
    square_diff = 0
    if arr1[:, 0].size == arr2[:,0].size:
        for i in range(n):
            square_diff += (arr1[i, 0] - arr2[i, 0])**2
    return math.sqrt(square_diff/n)

expected_res = rl.values[np.arange(test_start, test_end), :]
mean_square(out_res, expected_res)
rl.ActivePowerSum.mean()
rl.ActivePowerSum.max()
```