Міністерство освіти і науки України
Одеський національний політехнічний університет

Інституту штучного інтелекту і робототехніки
Кафедра комп'ютерних систем

Рудніченко Артур Віталійович
студент групи АК-151

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Дослідження методів підвищення ефективності бортового графічного інтерфейсу автомобіля

Напрям підготовки: 123 – Комп'ютерна інженерія
Спеціалізація: Спеціалізовані комп'ютерні системи

Керівник:
Стрельцов Олег Васильович
к.т.н., доцент

Одеса – 2020

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної роботи магістра полягає у розв’язанні проблеми передачі хоча б частини обов'язків по керуванню та взаємодії з транспортним засобом елементам штучного інтелекту, що дозволить зробити дороги безпечнішими, уникнути численних аварійних ситуацій і зменшити кількість людських жертв. А саме в дослідженні методів роботи користувальницького інтерфейсу в бортовому комп’ютері автомобіля.

Робота пов’язана з концепцією “Розумних автомобілів” та направлена розв’язати загальну проблему –зменшити час на взаємодію користувача (водія) з автомобілем.

Мета роботи – підвищити ефективність користування бортовим комп’ютером автомобіля, шляхом зменшення часу на взаємодію користувача й автомобільного інтерфейсу.

Після аналізу поставленої мети було поставлено наступні задачі:

1. Провести аналіз вже існуючих загальних рішень;
2. Встановити як часто для рішення проблеми використовуються нейронні мережі.
3. Аналіз алгоритмів нейронних мереж
4. Проведення моделювання.

Об’єктом дослідження є процес збору, обробки даних, зібраних з автомобіля під час його використання, та прийняття рішень на основі зібраних даних.

Предметом дослідження є методи підвищення ефективності бортового графічного інтерфейсу автомобіля.

Кваліфікаційна робота магістра будувалась на порівняльному методі дослідження. Обираючи технології для реалізації певних функціональних потреб, висунутих критеріями проекту, зрівнювались існуючі технічні рішення,

та обирались певні на основі аналізу перспектив розвитку та розв’язання окремих проблем.

Інноваційне рішення полягає у впровадженні нового рішення для підвищення ефективності взаємодії користувача й бортового комп’ютера автомобіля, шляхом зменшення часу на взаємодію, завдяки використанню навченої нейронної мережі.

Отримані результати, потенційно, можуть стати напрямком дослідження у загальній галузі використання розумних автомобілів. Отримані результати підлягають перевірці шляхом моделювання на тестовому автомобільному стенді, що зможе наблизити умови користування до реальних показників авто. Потенційно, отриману модель можна буде інтегрувати в реальні автомобілі після успішного проходження всіх етапів тестування.

Робота побудована на персональних наукових положеннях та авторській ідеї. Результати роботи належать автору дипломної роботи та науковому керівнику дипломної роботи.

Отримані результати не були апробовані. Результати та висновки роботи перебувають на теоретичному етапі та підлягають досконалому дослідженню та моделюванню методом практичного втілення.

Результати були підготовлені для публікації в журналі та відтворені в науковій роботі магістра.

1. АНАЛІЗ ІНТЕРФЕЙСІВ КОРИСТУВАЧА З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

1.1. Постанова загальної проблеми

Загальна проблема наукової роботи полягає у знаходженні методів підвищення ефективності роботи бортового комп’ютера авто, шляхом зменшення часу на взаємодію користувача з системою.

Аналізуючи цілі проекту, та різні джерела даних, які описуюсь методи вирішення вузьких проблем, було вирішено об’єднати найліпші технології у єдине загальне теоретичне рішення.

1.2. Аналіз сучасного стану проблеми

Високотехнологічні автомобілі розробляються для того, щоб звільнити автомобіліста від необхідності вирішувати численні завдання в процесі руху, зробити поїздку більш комфортною і зменшити будь-які можливі ризики.

Багато експертів вважають, що передача хоча б частини обов'язків по управлінню транспортним засобом штучного інтелекту, дозволить зробити дороги безпечнішими, уникнути численних аварійних ситуацій і зменшити кількість людських жертв.

На сьогоднішній день розумні автомобілі можуть:

- Здійснювати контроль за станом водія
- Здійснювати аварійне оповіщення
- Здійснювати автоматичне освітлення
- Здійснювати автоматичне керування автомобілем

Було розглянуто найтехнологічніші автомобілі, існуючі на даний час, стислі дані наведені в таблиці 1 [1].

Таблиця 1 - Найтехнологічніші автомобілі

Марка авто	Наявність автопілоту	Особливості
Tesla	*Так*	Тільки електрокари. Легкові та грузові автомобілі. Авто розраховані на 4-6 пасажирів, залежно від комплектації. Набір сенсорів захищає автомобілі від зіткнень, а 360-градусна камера розпізнає дорожню розмітку, перехрестя, інші машини і транспортні засоби, пішоходів. Таким чином, автомобіль самостійно здійснює управління і регулює швидкість руху. У процесі використання електрокара автопілот самонавчається, а також передає дані на сервери компанії Tesla Motors. На основі зібраних даних співробітники проводять аналіз і вдосконалюють систему.
Безпілотний автомобіль від Google	*Так*	Повністю електрична машина розрахована на двох осіб, має два двигуни, розвиває швидкість до 25 миль / год (трохи більше 40 км / год), не має традиційних органів управління і управляється з кнопки пуску, не вимагаючи присутності людини, крім як в ролі пасажира. Для пересування автомобіля використовується 64-променевої лазерний світодальномір на даху, який швидко генерує об'ємну карту місцевості.
Nissan IDS	*Так*	Вбудований в бортовий комп'ютер штучний інтелект повністю відповідає за безпеку пересування, а управління можливо за допомогою голосу і жестів. Режим автономної їзди Piloted Drive запам'ятовує і повторює стиль водіння конкретної людини - саме це є відмінною рисою Nissan IDS від інших концептів
Toyota Concept-i	*Так*	Concept-i може вести діалог з людиною за допомогою голосу, світла і голограм.

BMW Vision Next 100	*Так*	Надає водієві будь-які потрібні дані, а також захищає його від зайвої інформації. В салоні встановлений екран для проекцій.

1.2.1. «Розумні» призначені для користувача інтерфейси

Більше індивідуальних можливостей і простота в управлінні - ось що в першу чергу демонструє цей концепт-кар. Водійські камера відразу розпізнає обличчя автомобіліста, а система сама регулює положення керма, дзеркал і температуру в салоні відповідно до індивідуальних переваг людини за кермом. Більш того, автомобіль змінює кольорову схему дисплея і автоматично завантажує календар зустрічей, улюблену музику, свіжі подкасти та маршрут до пункту призначення, який водій поставив ще в той час, поки сидів у себе вдома на кухні.

У шляху водійська камера постійно стежить за станом водія. Це особливо важливо в тому випадку, якщо очі водія починають закриватися. Вона розпізнає ступінь втоми людини і півсон - стан, яке особливо часто стає причиною серйозних ДТП. Його можна розпізнати по руху століття водія. Система оцінює здатність автомобіліста до концентрації і рівень втоми і при необхідності видає тривожний сигнал. Це робить дорожній рух безпечнішим. Крім того, система розпізнавання втоми водія постійно стежить за його манерою водіння, щоб при виникненні різких рухів негайно втрутитися.

Людино-машинний інтерфейс (HMI) перетворює автомобілі в справжніх персональних помічників. Управління концепт-каром здійснюється за допомогою жестів. В інтер'єрі автомобіля використовуються ультразвукові датчики, які спрацьовують, коли водій робить той чи інший рух в області їх видимості. Управління жестами легке і менше відволікає водія: автомобіліст може змінювати інформацію на дисплеї, приймати телефонні дзвінки або вибирати пісні зі списку відтворення, навіть не торкаючись до екрану.

Інноваційний дисплей дозволяє безпечно і вільно вибирати пункти меню за допомогою сенсорного управління зі зворотним зв'язком. Екран вібрує кожен раз, коли водій доторкається до нього. Більш того, автомобіліст відчуває подібність опуклих кнопок на звичайному плоскому екрані, і йому легко підібрати необхідну функцію - наприклад, регулювання гучності, - не відволікаючись від водіння [2].

1.2.2. Автомобіль розуміючий емоції водія

Сучасні технології дозволяють читати по обличчю, виявляючи сім універсальних для всіх людей емоцій, таких як страх, гнів, радість, сум, відраза, подив і презирство. Ці знання можуть знайти застосування в відеоіграх, медицині, рекламі і, можливо, в автомобілях. Як відомо, крім втоми іншим фактором ризику є емоційний стан водія.

Треба визнати, що вимірювати емоції нелегко, особливо безконтактним способом. Фахівці науково-дослідної лабораторії обробки сигналів адаптували систему розпізнавання осіб для використання в автомобілях. Пристрій покладається на інфрачервону камеру, розміщену позаду керма.

Основна проблема, з якою зіткнулися розробники, полягає в тому, щоб змусити пристрій розпізнавати дратівливість по мімічним змін особи. У кожного водія це виражається по-різному - у одного смикаються м'язи обличчя, другий лається, третій намагається приховати емоції за байдужим обличчям. Щоб спростити завдання на даному етапі проекту, дослідники вирішили відстежувати тільки два почуття: гнів і огиду.

Дослідження пройшло дві фази. Спочатку система «вчилася» виявляти емоції по серії фотографій людей з різними виразами обличчя, потім тренувалася на таких же емоційних відеороликах. Навчальні матеріали відображали реальні життєві ситуації, в тому числі в автомобілі. Швидкість виконання операції зіставлення прочитаних емоцій із зразками залежала від

використовуваних методів аналізу. Але в цілому система спрацювала добре і в більшості випадків з точністю визначала напади дратівливості.

Вчені планують провести додаткове дослідження, яке дозволить удосконалити систему, доповнивши її більш просунутими алгоритмами і людино-машинним інтерфейсом. Розпізнавання емоцій є лише малою частиною зусиль, спрямованих на підвищення безпеки і комфорту водіння [3].

1.3. Огляд сучасних технологій та використання нейронних мереж

На даний час в більшості, якщо не у всіх, розумних автомобілей присутній штучний інтелект, який допомагає облегшити процес використання авто.

Для прикладу, розглянемо декілька відомих концептів автівок в яких використовуються штучний інтелект та нейронні мережі.

1.3.1. NISSAN IDS

Безпілотне управління і нульові викиди в атмосферу - основа Nissan IDS. Повністю довіряючи бортовому комп'ютеру, сам стиль руху автомобіля буде нагадувати манеру водіння водія. Саме це є відмінною рисою Nissan IDS від інших концептів. Вбудований в бортовий комп'ютер штучний інтелект повністю відповідає за безпеку пересування, а управління можливо за допомогою голосу і жестів. Режим автономної їзди Piloted Drive запам'ятовує і повторює ваш стиль водіння [4].

1.3.2. NIO EVE

Є ручний і безпілотний режими. Автономне управління і поведінку машини засновані на системі NOMI - це самообуча~~єм~~ая система штучного інтелекту, яка "підлаштовується" під звички і завдання власників смарт кара і робить поїздку максимально комфортною.

NOMI, штучно інтелектуальний цифровий супутник, який дізнається про свого водія та пасажира для задоволення їхніх потреб. Варіанти виконання пристрою знаходяться спереду і ззаду. Полотно пристрою - це переднє лобове скло, щоб відображати будь-яку відповідну інформацію водієві та пасажирам. Полотно також використовується для покращення досвіду, будь то особиста інформація чи розвага. Це також відображає будь-які визначні пам'ятки як пропозицію. [5] .

1.3.3. LEXUS LF-30 ELECTRIFIED

Електромобіль нового покоління LF-30 використовує технологію бездротової зарядки, що спрощує повсякденну підзарядку батареї. Для оптимального перерозподілу електроенергії між автомобілем і домашньої електричної мережею використовується штучний інтелект, розклад підзарядки може бути також погоджено з повсякденним розкладом користувача.

Бортова система штучного інтелекту розпізнає голоси знаходяться в салоні людей і використовує персоналізовану інформацію, збережену в керуючому ключі водія, що дозволяє їй виступати в якості партнера і помічника. Ця система здатна підлаштовувати різні параметри навколишнього оточення в інтер'єрі - температуру повітря, роботу аудіосистеми, настройки маршрутів і пунктів призначення в навігації, а також запропонувати різні варіанти занять після прибуття в потрібне місце.

Штучний інтелект розуміє переваги водія і допомагає йому в реальному часі підлаштовувати роботу підвіски і силових агрегатів відповідно до цих уподобаннями і умовами руху. Також LF-30 підтримує технологію використання бортового дрона Lexus Airporter: наприклад, безпілотник Lexus Airporter в автономному режимі здатний доставити багаж від дверей будинку в багажний відсік автомобіля [6]

1.4. Висновок

В даному розділі було розглянуто вже існуючі варіанти рішення поставленої проблеми. Було представлено найпопулярніших виробників розумних автомобілів і зроблено поверхневий аналіз їх продукту. В ході якого було виявлено, що, в основному, для покращення власного продукту всі вони використовують штучний інтелект.

Всі провідні виробники використовують штучний інтелект для реалізації функції безпілотного керування автомобілем і лише декілька з них розширюють користувальницький функціонал, як, наприклад, Nio eve.

Отформатировано: украинский

2. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

2.1. Поняття нейронної мережі

Нейронна мережа [7] (також штучна нейронна мережа, ШНМ) - математична модель, а також її програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж - мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалок і У. Піттса [8]. Після розробки алгоритмів навчання одержувані моделі стали використовувати в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах управління та ін.

ШНМ є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів (Рисунок 2.1)). Такі процесори зазвичай досить прості (особливо в порівнянні з процесорами, використовуваними в персональних комп'ютерах). Кожен процесор подібної мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, тим не менше, будучи з'єднаними в досить велику мережу з керованим взаємодією, такі окремо прості процесори разом здатні виконувати досить складні завдання.

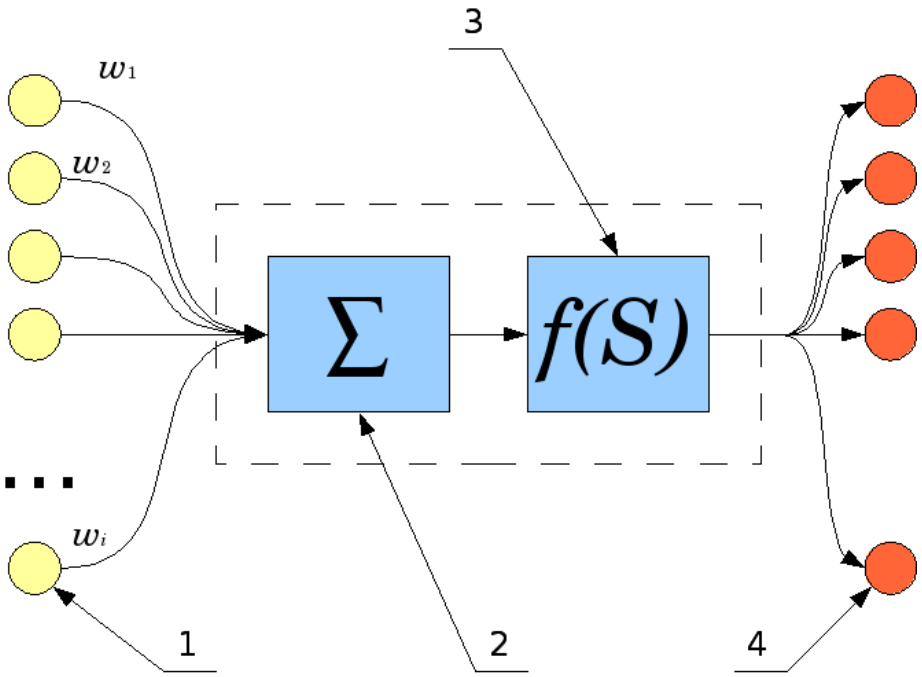

Рисунок 2.1 - Штучний нейрон

Схема штучного нейрона:

1. Нейрони, вихідні сигнали яких надходять на вхід даному
2. Суматор вхідних сигналів
3. Обчислювач передавальної функції
4. Нейрони, на входи яких подається вихідний сигнал даного
5. ω_i- ваги вхідних сигналів

Нейронні мережі не програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання - одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в

навчальній вибірці, а також неповних і / або «зашумлених», частково спотворених даних.

З'єднані між собою нейрони утворюють ШНМ. Таким чином, ШНМ — пара (M, V), де M — множина нейронів; V — множина зв'язків. Структура мережі задається у вигляді графа, у якому вершини є нейронами, а ребра являють собою зв'язки (з'єднання). Кожен нейрон мережі має вхідні ланцюги, причому їхня кількість є довільною для кожного нейрона. У загальному випадку ШНМ складається з декількох шарів, серед яких обов'язково є вхідний, що отримує зовнішні сигнали, вихідний, що відбиває реакцію нейронів на комбінації вхідних сигналів, і в багатошарових ШНМ — приховані шари (рис. 2.2). Така пошарова організація є аналогом шаруватих структур певних відділів мозку.

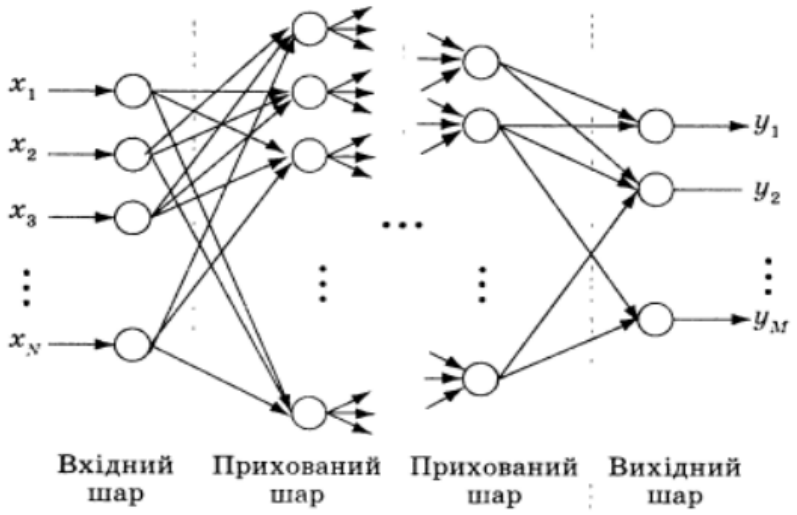

Рисунок 2.2- Структура ШНМ

Зв'язки між нейронами задаються у вигляді векторів і матриць. Ваги зручно подавати елементами матриці \mathcal{W} = [w_{ij}] розмірності N×M,

де N — кількість входів; M — кількість нейронів. Елемент w_{ij}\ відбиває зв'язок між і-м й *j*-м нейронами. При цьому, якщо:

- w_{ij} = 0 — зв'язок між і-м й *j* -м нейронами відсутній;
- w_{ij}12 < 0 — гальмуючий сигнал зв'язок;
- w_{ij}> 0 — прискорювальний сигнал (збуджувальний) зв'язок.

Залежно від того, чи містять ШНМ зворотні зв'язки, чи ні, розрізнюють такі їхні топології:

- ШНМ без зворотних зв'язків (прямого поширення, Feed forward)
- ШНМ зі зворотними зв'язками (зворотного поширення, рекурентні, Feedback):
  - з прямими зворотними зв'язками (direct feedback);
  - з непрямими зворотними зв'язками (indirect feedback);
  - з латеральними зв'язками (lateral feedback);
  - повнозв'язні.

У даній науковій роботі треба навчити нейронну мережу класифікувати вхідні дані і, спираючись на результат обробки, надавати користувачеві певну інформацію на екрані бортового комп'ютеру.

Це надає можливість реалізувати унікальний в своєму роді функціонал – направлення роботи нейронної мережі не на автоматичне пілотування автомобілем, а на взаємодію з користувачем (водієм). На даний момент подібний функціонал, як вже було сказано, існує лише, як концепт в єдиній компанії - «NIO», але ці розробки відсутні в відкритому доступі. Тому в даній кваліфікаційній роботі показаний відкритий аналог.

Далі розглянемо різновиди класифікаторів.

2.2. Огляд методів класифікації в машинному навчання за допомогою Scikit-Learn

Бібліотека Scikit-learn - найпоширеніший вибір для вирішення завдань класичного машинного навчання. Вона надає широкий вибір алгоритмів навчання з учителем і без вчителя. Навчання з учителем передбачає наявність розміченого датасету, в якому відомо значення цільового показника. У той час як навчання без вчителі не передбачає наявності розмітки в датасету - потрібно навчитися отримувати корисну інформацію з довільних даних. Одне з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх один з одним. Ще однією перевагою є широка спільнота і докладна документація. Scikit-learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання, для досліджень, а так само для новачків, які тільки робить перші кроки в області машинного навчання.

Для своєї роботи, scikit-learn використовує такі популярні бібліотеки:

- NumPy: математичні операції і операції над тензорами
- SciPy: науково-технічні обчислення
- Matplotlib: візуалізація даних
- IPython: інтерактивна консоль для Python
- SymPy: символьна математика
- Pandas: обробка, маніпуляції і аналіз даних

2.2.1. Що містить Scikit-learn

До завдань бібліотеки не входить завантаження, обробка, маніпуляція даними і їх візуалізація. З цими завданнями відмінно справляються бібліотеки Pandas і NumPy. Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення завдань навчання з учителем - класифікації.

Бібліотека реалізує наступні основні методи:

- Лінійні: моделі, завдання яких побудувати розділяє (для класифікації) гіперплощину.
- Метричні: моделі, які обчислюють відстань по одній з метрик між об'єктами вибірки, і приймають рішення в залежності від цього відстані (K найближчих сусідів).
- Дерева рішень: навчання моделей, що базуються на безлічі умов, оптимально обраних для вирішення завдання.
- Ансамблеві методи: методи, засновані на деревах рішень, які комбінують міць безлічі дерев, і таким чином підвищують їх якість роботи, а також дозволяють проводити відбір ознак (бустінг, беггінг, випадковий ліс, мажоритарне голосування).
- Нейронні мережі: комплексний нелінійний метод для задач регресії і класифікації.
- SVM: нелінійний метод, який навчається визначати межі прийняття рішень.
- Наївний Байес: пряме розподіл усіх моделювання для задач класифікації.
- PCA: лінійний метод зниження розмірності і відбору ознак
- t-SNE: нелінійний метод зниження розмірності.
- K-середніх: найпоширеніший метод для кластеризації, требущіх на вхід число кластерів, за якими повинні бути розподілені дані.
- Крос-валідація: метод, при якому для навчання використовується весь датасета (на відміну від розбиття на вибірки train / test), проте навчання відбувається багаторазово, і в якості валідаційної вибірки на кожному кроці виступають різні частини датасету. Підсумковий результат уявляють собою усереднення отриманих результатів.
- Grid Search: метод для знаходження оптимальних гіперпараметрів моделі шляхом побудови сітки з значень гіперпараметрів і

послідовного навчання моделей з усіма можливими комбінаціями гіперпараметрів з сітки.

Це - лише базовий список. Крім цього, Scikit-learn містить функції для розрахунку значень метрик, вибору моделей, препроцесінгу даних та інші.

2.2.2. Основні терміни

У системах машинного навчання або ж системах нейронних мереж існують входи і виходи. Те, що подається на входи, прийнято називати ознаками (англ. Features).

Ознаки по суті є тим же, що і змінні в науковому експерименті - вони характеризують будь-якої спостережуваний феномен і їх можна якось кількісно виміряти.

Коли ознаки подаються на входи системи машинного навчання, ця система намагається знайти збіги, помітити закономірність між ознаками. На виході генерується результат цієї роботи.

Цей результат прийнято називати міткою (англ. Label), оскільки у виходів є якась позначка, видана їм системою, тобто припущення (прогноз) про те, в яку категорію потрапляє вихід після класифікації.

В контексті машинного навчання класифікація відноситься до *навчання з вчителем*. Такий тип навчання передбачає, що дані, що подаються на входи системи, вже помічені, а важлива частина ознак вже розділена на окремі категорії або класи. Тому мережа вже знає, яка частина входів важлива, а яку частину можна самостійно перевірити.

Процес навчання моделі - це подача даних для нейронної мережі, яка в результаті повинна вивести певні шаблони для даних. В процесі навчання моделі з вчителем, на вхід подаються ознаки і мітки, а при прогнозуванні на вхід класифікатора подаються тільки ознаки.

Прийняті мережею дані діляться на дві групи: набір даних для навчання і набір для тестування. Не варто перевіряти мережу на тому ж наборі даних, на яких вона навчалася, бо модель вже буде «заточена» під цей набір.

2.2.3. Типи класифікаторів

Scikit-Learn дає доступ до безлічі різних алгоритмів класифікації [9]. Ось основні з них:

- Метод k-найближчих сусідів (K-Nearest Neighbors);
- Метод опорних векторів (Support Vector Machines);
- Класифікатор дерева рішень (Decision Tree Classifier) / Випадковий ліс (Random Forests);
- Наївний байєсовський метод (Naive Bayes);
- Лінійний дискримінантний аналіз (Linear Discriminant Analysis);
- Лінійна регресія
- Логістична регресія (Logistic Regression);

2.3. Дослідження алгоритмів навчання нейронної мережі

2.3.1. Метод найближчих сусідів

Метод найближчих сусідів (k Nearest Neighbors, або kNN) - дуже популярний метод класифікації, також іноді використовується в задачах регресії. Це, нарівні з деревом рішень, один з найбільш зрозумілих підходів до класифікації. На рівні інтуїції суть методу така: подивися на сусідів, які переважають, такий і ти. Формально основою методу є гіпотеза компактності: якщо метрика відстані між прикладами введена досить вдало, то схожі приклади набагато частіше лежать в одному класі, ніж в різних (Рис. 2.3).

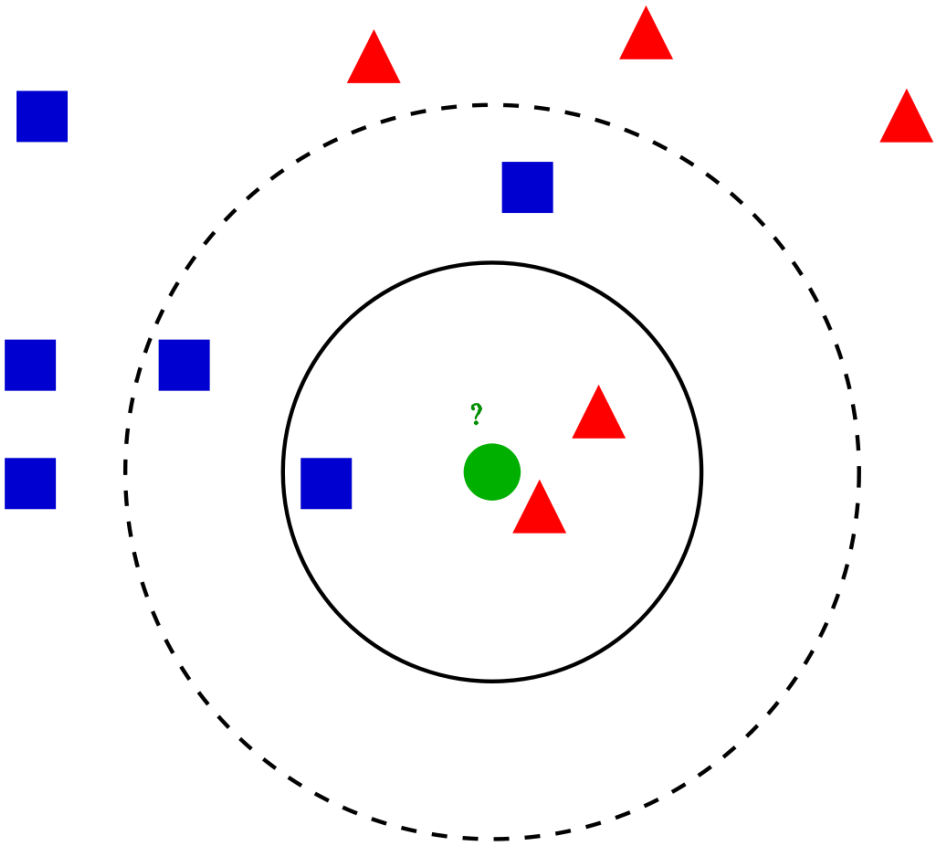

Рисунок 2.3- Схематичне зображення методу KNN

Для класифікації кожного з об'єктів тестової вибірки необхідно послідовно виконати наступні операції:

- Обчислити відстань до кожного з об'єктів навчальної вибірки
- Відібрати k об'єктів навчальної вибірки, відстань до яких мінімально
- Клас класифікованого об'єкту - це клас, який найчастіше трапляється серед k найближчих сусідів

Примітна властивість такого підходу - його «лінивість». Це означає, що обчислення починаються тільки в момент класифікації тестового прикладу, а

заздалегідь, тільки при наявності навчальних прикладів, ніяка модель не будується. У цьому відмінність, наприклад, від дерева рішень, де спочатку на основі навчальної вибірки будується дерево, а потім відносно швидко відбувається класифікація тестових прикладів.

Варто відзначити, що метод найближчих сусідів - добре вивчений підхід (в машинному навчанні, економетрики та статистики більше відомо, напевно, тільки про лінійну регресію). Для методу найближчих сусідів існує чимало важливих теорем, які стверджують, що на "нескінченних" вибірках це оптимальний метод класифікації. Автори класичної книги "The Elements of Statistical Learning" вважають kNN теоретично ідеальним алгоритмом, застосовність якого просто обмежена обчислювальними можливостями і розмірністю.

2.3.1.1. Плюси і мінуси методу найближчих сусідів

Плюси:

- Проста реалізація;
- Непогано вивчений теоретично;
- Як правило, метод хороший для першого рішення задачі, причому не тільки класифікації або регресії, а й, наприклад, рекомендації;
- Можна адаптувати під потрібне завдання вибором метрики або ядра (в двох словах: ядро: може задавати операцію подібності для складних об'єктів типу графів, а сам підхід kNN залишається тим же). До речі, професор ОМК МГУ і досвідчений учасник змагань з аналізу даних Олександр Дьяконов любить найпростіший kNN, але з налагодженою метрикою подібності об'єктів.
- Непогана інтерпретація, можна пояснити, чому тестовий приклад був класифікований саме так. Хоча цей аргумент можна атакувати: якщо число сусідів велике, то інтерпретація погіршується (умовно: "ми не дали

йому кредит, тому що він схожий на 350 клієнтів, у тому числі 70 - погані, що на 12% більше, ніж в середньому по вибірці").

Мінуси:

- Метод вважається найшвидшим у порівнянні, наприклад, з композиціями алгоритмів, але в реальних задачах, як правило, число сусідів, які використовуються для класифікації, буде великим (100-150), і в такому випадку алгоритм буде працювати не так швидко, як дерево рішень;
- Якщо в наборі даних багато ознак, то важко підібрати відповідні ваги і визначити, які ознаки не важливі для класифікації;
- Залежність від обраної метрики відстані між прикладами. Вибір за замовчуванням евклідової відстані найчастіше нічим не обґрунтований. Можна відшукати хороше рішення перебором параметрів, але для великого набору даних це забирає багато часу;
- Немає теоретичних підстав вибору певного числа сусідів - тільки перебір (втім, найчастіше це вірно для всіх гіперпараметрів всіх моделей). У разі малого числа сусідів метод чутливий до викидів, тобто схильний перенавчатися;
- Як правило, погано працює, коли ознак багато, через "прокляття розмірності". Про це добре розповідає відомий в ML-співтоваристві професор Pedro Domingos - в популярній статті [10] "A Few Useful Things to Know about Machine Learning", також "the curse of dimensionality" описується в книзі Deep Learning в розділі "Machine Learning basics" [11].

2.3.2. Метод опорних векторів (Support Vector Machines)

Метод опорних векторів (SVM) - це тип керованого алгоритму класифікації машинного навчання. SVM були запроваджені спочатку в 1960-х,

а пізніше були вдосконалені в 1990-х. Однак лише зараз вони стають надзвичайно популярними завдяки їх здатності досягати блискучих результатів. SVM реалізовані унікальним чином у порівнянні з іншими алгоритмами машинного навчання.

Метод опорних векторів - набір схожих алгоритмів навчання з учителем, що використовуються для задач класифікації та регресійного аналізу. Належить сімейству лінійних класифікаторів. Особливою властивістю методу опорних векторів є невпинне зменшення емпіричної помилки класифікації і збільшення зазору, тому метод також відомий як метод класифікатора з максимальним зазором.

Основна ідея методу - переклад вихідних векторів в простір більш високої розмірності і пошук розділяє гіперплощини з максимальним зазором в цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє класи. Розподільною гіперплощиною буде гіперплощина, максимізуюча відстань до двох паралельних гіперплощин. Алгоритм працює в припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим менше буде середня помилка класифікатора.

2.3.2.1. Простий SVM

У випадку лінійно відокремлюваних даних у двох вимірах, як показано на рис. 1, типовий алгоритм машинного навчання намагається знайти межу, яка розділяє дані таким чином, що помилку помилкової класифікації можна мінімізувати. Якщо уважно розглянути Рис. 2.4, може бути кілька меж, які правильно поділяють точки даних. Дві пунктирні лінії, а також одна суцільна лінія класифікують дані правильно.

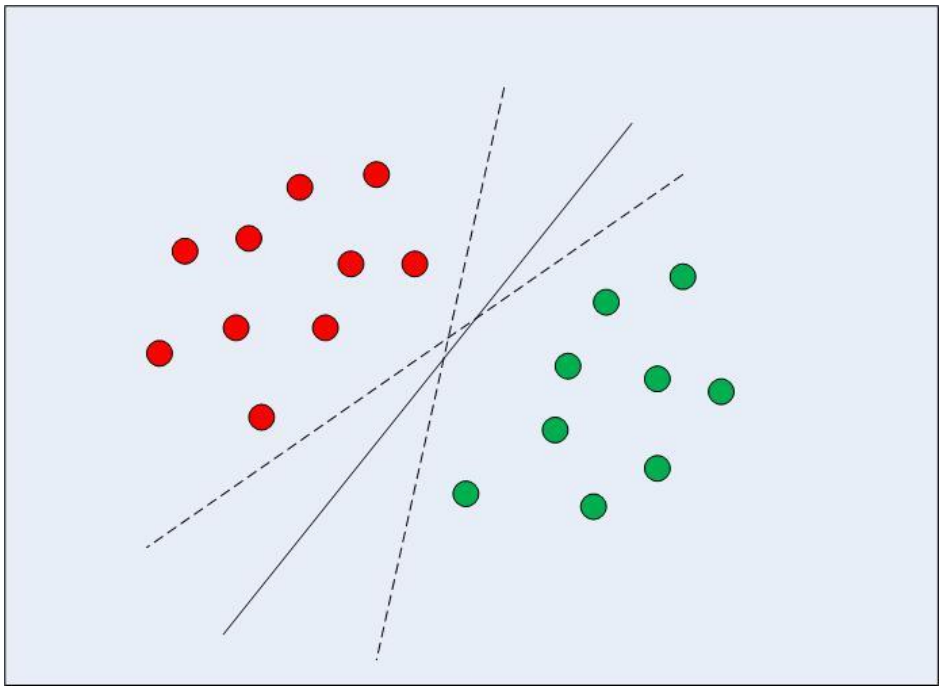

Рисунок 2.4- Кілька меж прийняття рішень

SVM відрізняється від інших алгоритмів класифікації тим, що вибирає межу рішення, яка максимізує відстань від найближчих точок даних усіх класів. SVM не просто знаходить межу рішення; він знаходить найбільш оптимальну межу прийняття рішення.

Найбільш оптимальною межею рішення є та, яка має максимальний запас від найближчих точок усіх класів. Найближчі точки від межі рішення, які максимізують відстань між межею рішення та точками, називаються опорними векторами, як показано на Рис. 2.5. Межа рішення у випадку машин з опорними векторами називається класифікатором максимального поля або гіперплощиною граничного поля .

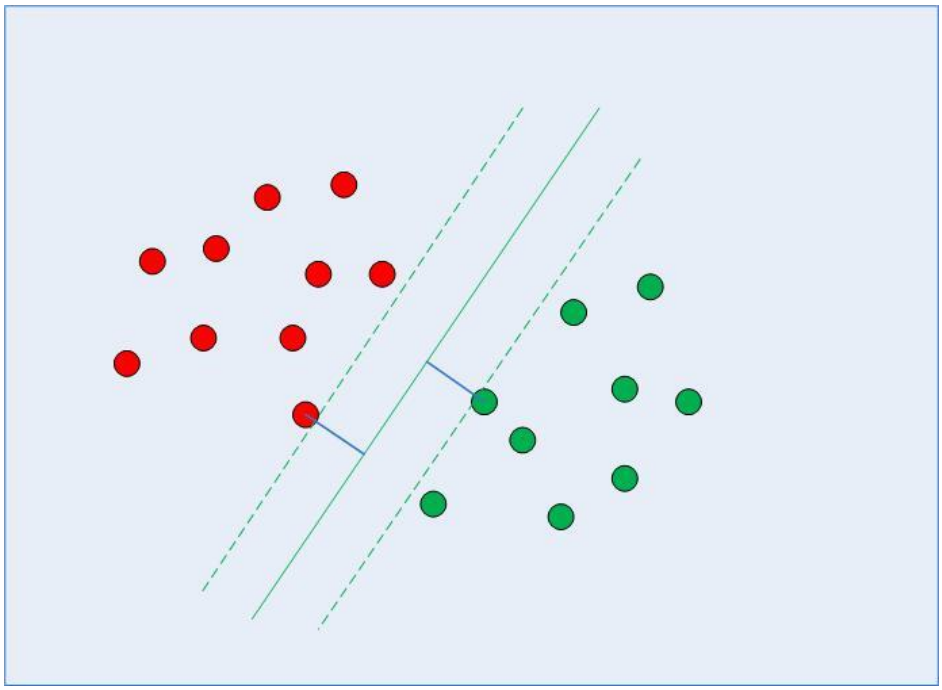

Рисунок 2.5- Межа рішення з векторами підтримки

2.3.2.2. Kernel SVM

У попередньому розділі ми побачили, як за допомогою простого алгоритму SVM можна знайти межу рішення для лінійно відокремлюваних даних. Однак у випадку нелінійно відокремлюваних даних, таких як дані, показані на Рис. 2.6, пряма лінія не може використовуватися як межа прийняття рішення.

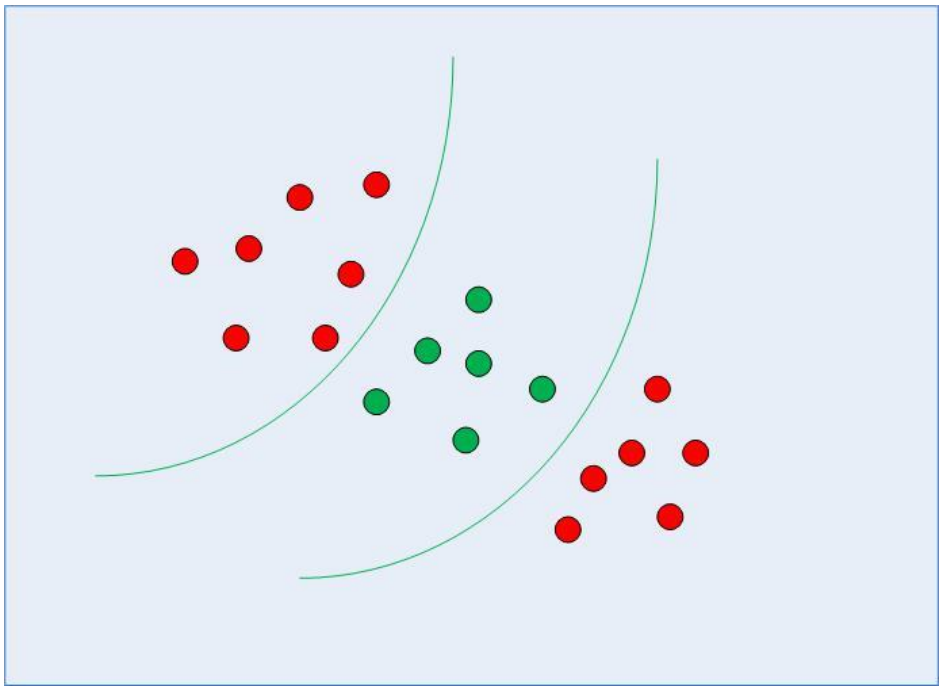

Рисунок 2.6 - Нелінійно відокремлені дані

У разі нелінійно відокремлюваних даних не можна використовувати простий алгоритм SVM. Натомість використовується модифікована версія SVM, яка називається Kernel SVM.

В основному, Kernel SVM проектує нелінійно відокремлювані дані нижчих розмірів на лінійно відокремлювані дані у вищих вимірах таким чином, що точки даних, що належать до різних класів, розподіляються до різних вимірів. Знову ж таки, у цьому задіяна складна математика, але вам не доведеться турбуватися про це, щоб використовувати SVM.

2.3.3. Класифікатор дерева рішень (Decision Tree Classifier) / Випадковий ліс

Дерево рішень [12] як алгоритм машинного навчання - об'єднання логічних правил виду "Значення ознаки α менше \mathcal{x} I Значення ознаки b менше y…=> Клас 1" в структуру даних "Дерево". Величезна перевага дерев рішень в тому, що вони легко інтерпретованих, зрозумілі людині. Наприклад, за схемою на рисунку 2.7 можна пояснити позичальнику, чому йому було відмовлено в кредиті. Скажімо, тому, що у нього немає будинку і дохід менше 5000. Як ми побачимо далі, багато інших, хоч і більш точні, моделі не володіють цією властивістю і можуть розглядатися скоріше як "чорний ящик", в який завантажили дані і отримали відповідь. У зв'язку з цією "зрозумілістю" дерев рішень і їх схожістю з моделлю прийняття рішень людиною (можна легко пояснювати босові свою модель), дерева рішень отримали величезну популярність.

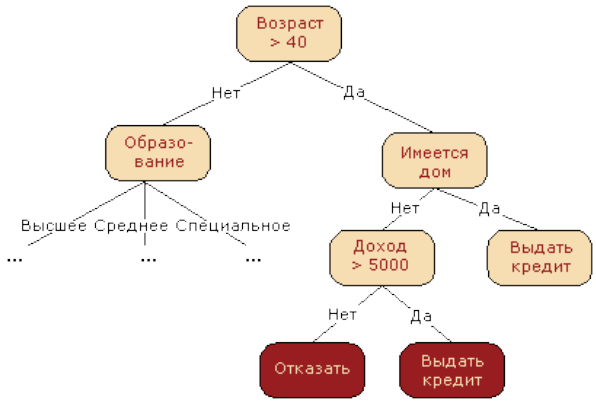

Рисунок 2.7- Задача бінарної класифікації

2.3.3.1. Ентропія

Ентропія Шеннона визначається для системи з можливими станами наступним чином:

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i,$$

де p_\mathcal{i} - ймовірність знаходження системи в -ом стані. Це дуже важливе поняття, яке використовується в фізиці, теорії інформації та інших областях. Опускаючи передумови запровадження (комбінаторні і теоретико-інформаційні) цього поняття, зазначимо, що, інтуїтивно, ентропія відповідає ступеню хаосу в системі. Чим вище ентропія, тим менше впорядкована система і навпаки. Це допоможе нам формалізувати "ефективний розподіл вибірки".

2.3.3.2. Плюси і мінуси дерев рішень

Плюси:

- Породження чітких правил класифікації, зрозумілих людині, наприклад. Це властивість називають інтерпретованістю моделі;
- Дерева рішень можуть легко візуалізувати, тобто може "інтерпретуватися" як сама модель (дерево), так і прогноз для окремого взятого тестового об'єкта (шлях в дереві);
- Швидкі процеси навчання і прогнозування;
- Мале число параметрів моделі;
- Підтримка і числових, і категоріальних ознак.

\

Мінуси:

- У породження чітких правил класифікації є й інша сторона: дерева дуже чутливі до шумів у вхідних даних, вся модель може кардинально змінитися, якщо трохи зміниться навчальна вибірка (наприклад, якщо прибрати один з ознак або додати кілька об'єктів), тому і правила класифікації можуть сильно змінюватися, що погіршує інтерпретованість моделі;
- Розділяюча межа, побудована деревом рішень, має свої обмеження (складається з гіперплоскостей, перпендикулярних якійсь координатній осі), і на практиці дерево рішень за якістю класифікації поступається деяким іншим методам;
- Необхідність відсікати гілки дерева (pruning) або встановлювати мінімальне число елементів в листі дерева або максимальну глибину дерева для боротьби з перенавчанням. Втім, перенавчання - проблема всіх методів машинного навчання;
- Нестабільність. Невеликі зміни в даних можуть суттєво змінювати побудоване дерево рішень. З цією проблемою борються за допомогою ансамблів дерев рішень;
- Проблема пошуку оптимального дерева рішень (мінімального за розміром і здатного без помилок класифікувати вибірку) NP-повна, тому на практиці використовуються евристики типу жадібного пошуку ознаки з максимальним приростом інформації, які не гарантують знаходження глобально оптимального дерева;
- Складно підтримуються пропуски в даних. Friedman оцінив, що на підтримку пропусків в даних пішло близько 50% коду CART (класичний алгоритм побудови дерев класифікації і регресії - Classification And Regression Trees, в sklearn реалізована поліпшена версія саме цього алгоритму);

- Модель вміє тільки інтерполювати, але не екстраполювати (це ж вірно і для лісу та бустінгу на деревах). Тобто дерево рішень робить константний прогноз для об'єктів, що перебувають у просторі ознак поза паралелепіпеда, що охоплює всі об'єкти навчальної вибірки.

2.3.4. Лінійна регресія

Лінійна регресія (Linear regression) - модель залежності змінної x від однієї або декількох інших змінних (факторів, регресорів, незалежних змінних) з лінійною функцією залежності.

Лінійна регресія відноситься до задачі визначення «лінії максимальної відповідності умовам» через набір точок даних і стала простим попередником нелінійних методів, які використовують для навчання нейронних мереж.

2.3.4.1. Метод найменших квадратів

В першу чергу, необхідно задати модель залежності пояснюється змінної y від пояснюють її чинників, функція залежності буде лінійною:

$$y = w_0 + \sum_{i=1}^{m} w_i x_i$$

Якщо ми додамо фіктивну розмірність $x_0 = 1$ для кожного спостереження, тоді лінійну форму можна переписати трохи більше компактно, записавши вільний член w_0 під суму: $y = \sum_{i=0}^{m} w_i x_i = \overrightarrow{w^T}\vec{x}$. Якщо розглядати матрицю спостереження-ознаки, у якій в рядках знаходяться приклади з набору даних, то нам необхідно додати одиничну колонку зліва. Задамо модель наступним чином:

$$\vec{y} = X\vec{w} + \epsilon,$$

де

- $\vec{y} \in R^n$ - пояснюється (або цільова) змінна;
- w - вектор параметрів моделі (в машинному навчанні ці параметри часто називають вагами);
- X- матриця спостережень і ознак розмірності n рядків на $m + 1$ стовпців (включаючи фіктивну одиничну колонку зліва) з повним рангом по стовпцям: $\text{rank}(X) = m + 1$;
- ϵ - випадкова змінна, відповідна випадкової, непрогнозованою помилку моделі.

Можемо виписати вираз для кожного конкретного спостереження

$$y_i = \sum_{j=0}^{m} w_j X_{ij} + \epsilon_i$$

Також на модель накладаються такі обмеження (інакше це буде якась інша регресія, але точно не лінійна):

- Математичне очікування випадкових помилок дорівнює нулю: $\forall i: E[\epsilon_i] = 0$
- дисперсія випадкових помилок однакова і кінцева, це властивість називається гомоскедастичністю: $\forall i$:$\text{Var}(\epsilon_i) = \sigma^2 <$
- випадкові помилки не скорельовані: $\forall i \neq j$:$\text{Cov}(\epsilon_i, \epsilon_j) = 0$

Оцінка $\widehat{w_i}$ ваг w_iназивається лінійної, якщо

$$\widehat{w_i} = \omega_{1i} y_1 + \omega_{2i} y_2 + \cdots + \omega_{ni} y_n,$$

де $\forall$ ω_{ki}залежить тільки від спостережуваних даних X і майже напевно нелінійно. Так як рішенням завдання пошуку оптимальних ваг буде саме лінійна оцінка, то і модель називається лінійної регресією. Введемо ще одне визначення. Оцінка $\widehat{w_i}$ називається не зсунутою тоді, коли математичне

очікування оцінки дорівнює реальному, але невідомому значення оцінюваного параметра:

$$E[\widehat{w_i}] = w_i$$

Один із способів обчислити значення параметрів моделі є **метод найменших квадратів** (МНК), який мінімізує середньоквадратичну помилку між реальним значенням залежної змінної і прогнозом, виданими моделлю:

$$\begin{aligned}\mathcal{L}(X, \vec{y}, \vec{w}) &= \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \vec{w}^T\vec{x}_i\right)^2 \\ &= \frac{1}{2n}\|\vec{y} - X\vec{w}\|_2^2 \\ &= \frac{1}{2n}(\vec{y} - X\vec{w})^T(\vec{y} - X\vec{w})\end{aligned}$$

Для вирішення даної оптимізаційної задачі необхідно обчислити похідні за параметрами моделі, прирівняти їх до нуля і вирішити отримані рівняння щодо $\vec{w}$:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\vec{w}} &= \frac{\partial}{\partial\vec{w}}\frac{1}{2n}\left(\vec{y}^T\vec{y} - 2\vec{y}^TX\vec{w} + \vec{w}^TX^TX\vec{w}\right) \\ &= \frac{1}{2n}\left(-2X^T\vec{y} + 2X^TX\vec{w}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\vec{w}} = 0 &\Leftrightarrow \frac{1}{2n}\left(-2X^T\vec{y} + 2X^TX\vec{w}\right) = 0 \\ &\Leftrightarrow -X^T\vec{y} + X^TX\vec{w} = 0 \\ &\Leftrightarrow X^TX\vec{w} = X^T\vec{y} \\ &\Leftrightarrow \vec{w} = \left(X^TX\right)^{-1}X^T\vec{y}\end{aligned}$$

Отже, маючи на увазі все визначення та умови описані вище, ми можемо стверджувати, спираючись на теорему Маркова-Гауса, що оцінка МНК є

найкращою оцінкою параметрів моделі, серед всіх лінійних і незміщене оцінок, тобто володіє найменшою дисперсією.

2.3.4.2. Плюси і мінуси лінійних моделей

Плюси:

- Добре вивчені
- Дуже швидкі, можуть працювати на дуже великих вибірках
- Практично поза конкуренцією, коли ознак дуже багато (від сотень тисяч і більше), і вони розріджені (хоча є ще факторизаційні машини)
- Коефіцієнти перед ознаками можуть інтерпретуватися (за умови що ознаки масштабовані) - в лінійної регресії як приватні похідні залежною змінною від ознак, в логістичній - як зміна шансів на віднесення до одного з класів $\exp^{\beta_i}$ в раз при зміні ознаки x_i на 1 од.
- Логістична регресія видає ймовірності віднесення до різних класів (це дуже цінується, наприклад, в кредитного скорингу)
- Модель може будувати і нелінійну кордон, якщо на вхід подати поліноміальні ознаки

Мінуси:

- Погано працюють в задачах, в яких залежність відповідей від ознак складна, нелінійна
- На практиці припущення теореми Маркова-Гауса майже ніколи не виконуються, тому частіше лінійні методи працюють гірше, ніж, наприклад, SVM і ансамблі (за якістю виконання завдання класифікації / регресії)

2.3.5. Логістична регресія

Логістична регресія [13] - це статистична модель, що використовує для прогнозування ймовірності повернення деякого події шляху його порівняння з логістичною кривою. Ця регресія видає відповідь у вірогідності бінарного події (1 або 0).

Логістична регресія застосовується для прогнозування вірогідності виникнення деякої події за значеннями багатьох визнань. Для цього вводиться так називається залежна перемінна y, приймаючи лише одне з двох значень - як правило, це цифри 0 (подія не відбувається) та 1 (подія, що виходить), і безліч незалежних змінних (також називаються ознаками, попередниками або регрессорами) – дробових $x_1, x_2, \ldots, x_n$, на основі значень яких потрібно вичислити ймовірність прийняття того чи іншого значення залежної змінної. Як і у випадку лінійної регресії, для простоти записів вводиться фіктивна ознака $x_0 = 1$

2.3.5.1. Лінійний класифікатор

Основна ідея лінійного класифікатора полягає в тому, що простір ознак (Рисунок2.8) може бути розділений гіперплощиною на два півпростору, в кожному з яких прогнозується одне з двох значень цільового класу.

Якщо це можна зробити без помилок, то навчальна вибірка називається *лінійно нероздільні*.

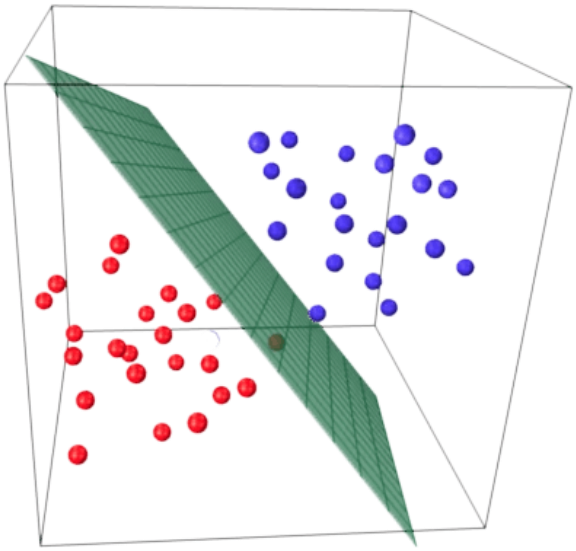

Рисунок 2.8 - Простір ознак

Ми вже знайомі з лінійної регресією і методом найменших квадратів. Розглянемо задачу бінарної класифікації, причому мітки цільового класу позначимо "+1" (позитивні приклади) і "-1" (негативні приклади).

Один з найпростіших лінійних класифікаторів виходить на основі регресії ось таким чином:

$$a(\vec{x}) = \text{sign}\left(\overrightarrow{w^T x}\right),$$

де

- $\vec{x}$ вектор ознак прикладу (разом з одиницею);
- $\vec{w}$ ваги в лінійної моделі (разом зі зміщенням);
- $\text{sign}(\cdot)$ функція "Сігнум", що повертає знак свого аргументу;
- $a(\vec{x})$ відповідь класифікатора на прикладі.

2.4. Висновок

У даному розділі було розглянуто поняття нейронної мережі в цілому та її ознаки. Встановлено, що цілком можливо реалізувати передачу керування частиною користувацького функціоналу автомобіля нейронній мережі, щоб зменшити час взаємодії водія з бортовим комп'ютером авто.

Проведено загальний огляд бібліотеки Scikit-learn для навчання нейронних мереж. Описано основні методи, якими володіє бібліотека, та терміни, які треба розуміти, при взаємодії з нейронною мережею.

Також в розділі йдеться про основні типи класифікаторів, які частіше за інших використовуються в навчанні нейронних мереж.

І найголовніше – в розділі описано принципи роботи кожного з розповсюджених типів класифікаторів (алгоритмів навчання нейронної мережі) також розглянуто їх плюси та мінуси.

Аналізуючи вище сказане та беручи до уваги завдання і мету наукового проекту було прийнято рішення використати декілька алгоритмів навчання нейронної мережі, а саме:

- Метод опорних векторів (Support Vector Machines);
- Логістична регресія
- Випадковий ліс

3. МОДЕЛЮВАННЯ НЕЙРОННОЇ МЕРЕЖІ

3.1. Опис середи моделювання

Для моделювання було обрано програмний пакет PyCharm 2020.3 - Professional Edition (Licensed to PyCharm Evaluator - License ID: L0YRAD7IT4). з використанням графічного фреймворку Qt з налаштованим розширення PyQt для створення графічного інтерфейсу користувача за допомогою програмної мови Python v 3.9.

3.2. Реалізація класифікатора

Перший крок в реалізації класифікатора - його імпорт в Python. Ось як це виглядає для логістичної регресії:

```
    from sklearn.linear_model import LogisticRegression
```

Ось огляди імпорту інших класифікаторів, розглянутих в попередньому розділі:

```
    from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.naive_bayes import GaussianNB
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.svm import SVC
```

Після цього потрібно створити екземпляр класифікатора. Зробити це можна створивши змінну і викликавши функцію, пов'язану з класифікатором.

```
    logreg_clf = LogisticRegression()
```

Тепер класифікатор потрібно навчити. Перед цим потрібно «підігнати» його під тренувальні дані.

Навчальні ознаки і мітки поміщаються в класифікатор через функцію fit:

```
logreg_clf.fit(features, labels)
```

Після навчання моделі дані вже можна подавати в класифікатор. Це можна зробити через функцію класифікатора predict, передавши їй параметр (ознака) для прогнозування:

```
logreg_clf.predict(test_features)
```

Ці етапи (створення екземпляра, навчання і класифікація) є основними при роботі з класифікаторами в Scikit-Learn. Але ця бібліотека може управляти не тільки класифікаторами, а й самими даними. Щоб розібратися в тому, як дані і класифікатор працюють разом над завданням класифікації, потрібно розібратися в процесах машинного навчання в цілому.

3.3. Процес машинного навчання

Процес містить у собі наступні етапи:

- підготовка даних,
- створення навчальних наборів,
- створення класифікатора,
- навчання класифікатора,
- складання прогнозів,
- оцінка продуктивності класифікатора
- настройка параметрів.

По-перше, потрібно підготувати набір даних для класифікатора - перетворити дані в коректну для класифікації форму і обробити будь-які аномалії в цих даних. Відсутність значень в даних або будь-які інші відхилення - всі їх потрібно обробити, інакше вони можуть негативно впливати на продуктивність класифікатора. Цей етап називається попередньою обробкою даних (англ. Data preprocessing).

Наступним кроком буде поділ даних на навчальні та тестові набори. Для цього в Scikit-Learn існує відмінна функція traintestsplit.

Як вже було сказано, класифікатор повинен бути створений і навчений на тренувальному наборі даних. Після цих кроків модель вже може робити прогнози. Порівнюючи свідчення класифікатора з фактично відомими даними, можна робити висновок про точність класифікатора.

Потрібно буде «коригувати» параметри класифікатора, поки ви не досягнете бажаної точності (т. к. малоймовірно, що класифікатор буде відповідати всім нашим вимогам з першого ж запуску).

3.4. Реалізація зразка класифікації

```
# Імпорт всіх потрібних бібліотек
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.svm import SVC
```

Далі імпортуємо набір даних (датасет), який збережений в форматі «*.csv*».

У бібліотеці Pandas є функція read_csv (), яка відмінно працює із завантаженням даних.

```
data = pd.read_csv('SampleDataset.csv')
```

Завдяки тому, що дані вже були підготовлені, довгої попередньої обробки вони не вимагають. Єдине, що може знадобитися - прибрати непотрібні стовпці таким чином:

```
data.drop('Media', axis=1, inplace=True)
data.drop('Radio', axis=1, inplace=True)
```

Тепер потрібно визначити ознаки і мітки. З бібліотекою Pandas можна легко «нарізати» таблицю і вибрати певні рядки / стовпці за допомогою функції iloc ():

```
X = data.iloc[:, 1:].values

# Тепер виділимо потрібний стовпець
y = data['Temperature_in']
```

Код вище вибирає кожен рядок і стовпець, обрізавши при цьому останній рядок.

Після того, як ви вибрали потрібні ознаки і мітки, їх можна розділити на тренувальні і тестові набори, використовуючи функцію train_test_split ():

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.02,
random_state=4)
```

Тепер можна створювати екземпляр класифікатора:

```
LR_model = LogisticRegression(random_state=0,
solver='lbfgs', multi_class='ovr')
RF_model = RandomForestClassifier(n_estimators=100,
max_depth=5, random_state=0)
SVC_model = SVC(kernel='linear', C=0.01)
```

Тепер потрібно навчити ці класифікатори:

```
LR_model.fit(X_train, y_train)
RF_model.fit(X_train, y_train)
SVC_model.fit(X_train, y_train)
```

Ці команди навчили моделі і тепер класифікатори можуть робити прогнози і зберігати результат в будь-яку змінну.

```
LR_prediction = LR_model.predict(X_test)
RF_prediction = RF_model.predict(X_test)
SVC_prediction = SVC_model.predict(X_test)
```

Тепер необхідно оцінити точності класифікатора. Існує кілька способів це зробити.

Потрібно передати показання прогнозу щодо фактично вірних міток, значення яких були збережені раніше.

```
print("LR_model = ", accuracy_score(LR_prediction, y_test))+
print("RF_model = ", accuracy_score(RF_prediction, y_test))
print("SVC_model = ", accuracy_score(SVC_prediction, y_test))
```

Результат отриманих метрик має вигляд:

- SVC_model = 0.65
- LR_model = 0.7
- RF_model = 0.9

Далі розглянемо отримані результати в більш зручному вигляді. Зобразимо метрики класифікації у вигляді матриці. Для початку розглянемо результати для RF моделі (Рис. 3.1):

	precision	recall	f1-score	support
15	1.00	1.00	1.00	5
16	1.00	1.00	1.00	2
17	1.00	1.00	1.00	1
18	1.00	1.00	1.00	2
19	0.67	1.00	0.80	2
20	1.00	1.00	1.00	1
21	0.00	0.00	0.00	1
22	1.00	0.67	0.80	3
23	1.00	1.00	1.00	2
24	1.00	1.00	1.00	1
accuracy			0.90	20
macro avg	0.87	0.87	0.86	20
weighted avg	0.92	0.90	0.90	20

Рисунок 3.1- Звіт RF моделі

Отриманні результати можна побачити на матриці неточностей та нормалізованому її зображенні (Рис 3.2 та 3.3)

Кількість правильних прогнозів йде з верхнього лівого кута в нижній правий.

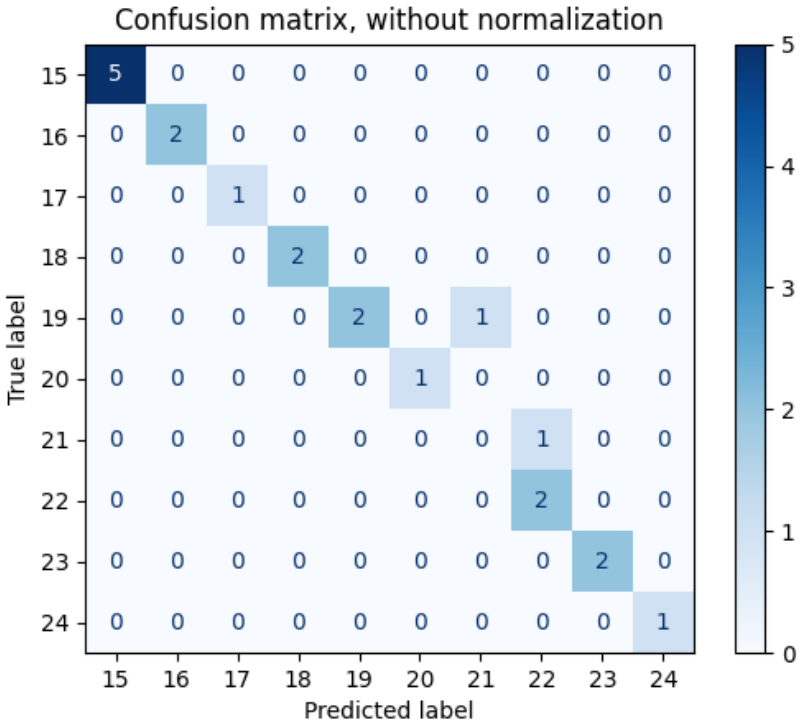

Рисунок 3.2- Матриця неточностей для RF моделі

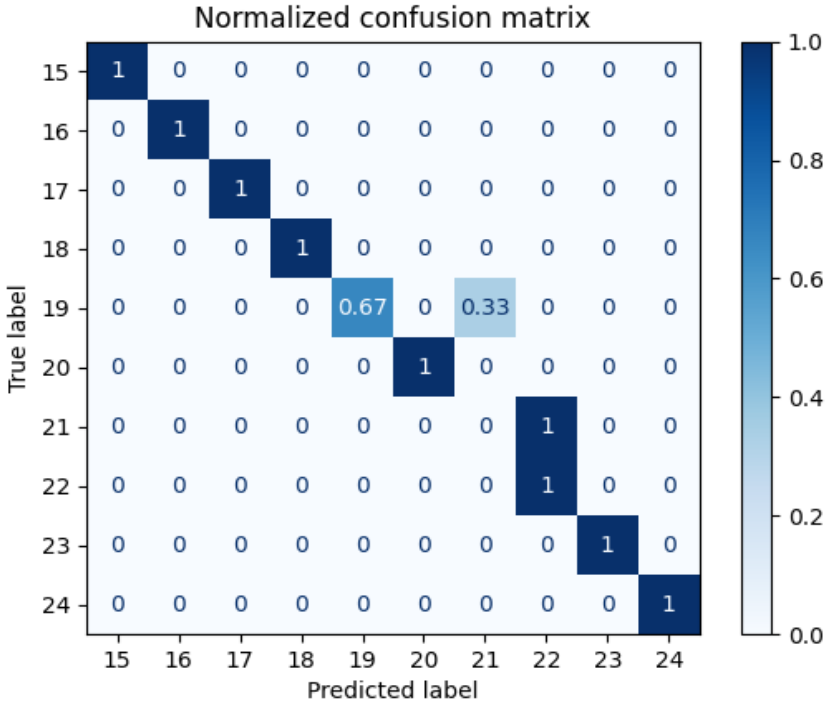

Рисунок 3.3-Нормалізована матриця неточностей для RF моделі

Те саме розглянемо й для SVC (Рис 3.4 – 3.6) та LR (Рис 3.7 – 3.9) моделей

	precision	recall	f1-score	support
15	0.80	1.00	0.89	4
16	1.00	0.67	0.80	3
17	1.00	1.00	1.00	1
18	1.00	0.40	0.57	5
19	0.00	0.00	0.00	0
20	0.00	0.00	0.00	0
21	0.00	0.00	0.00	1
22	1.00	0.67	0.80	3
23	0.50	1.00	0.67	1
24	1.00	0.50	0.67	2
accuracy			0.65	20
macro avg	0.63	0.52	0.54	20
weighted avg	0.89	0.65	0.71	20

Рисунок 3.4- - Звіт SVC моделі

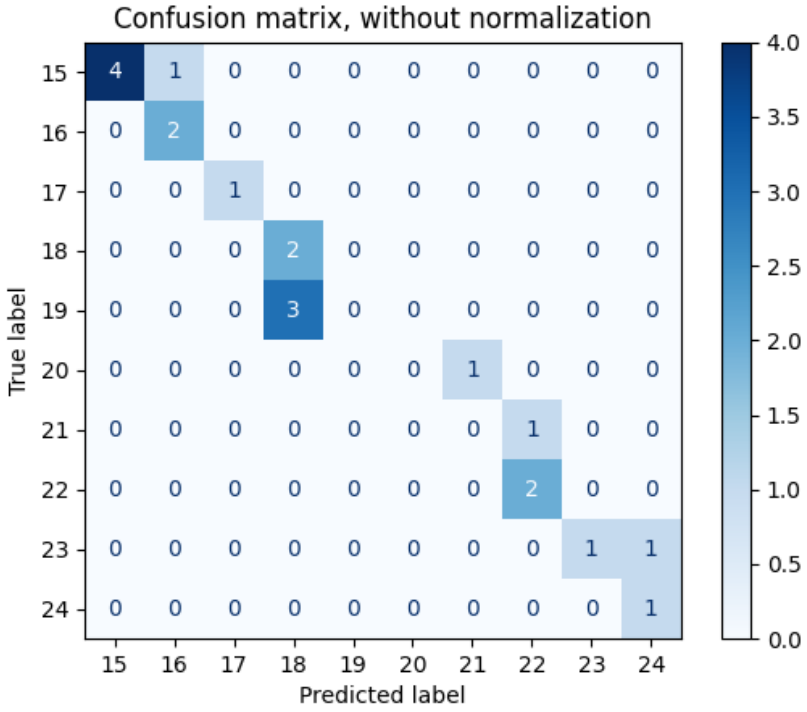

Рисунок 3.5- Матриця неточностей для SVC моделі

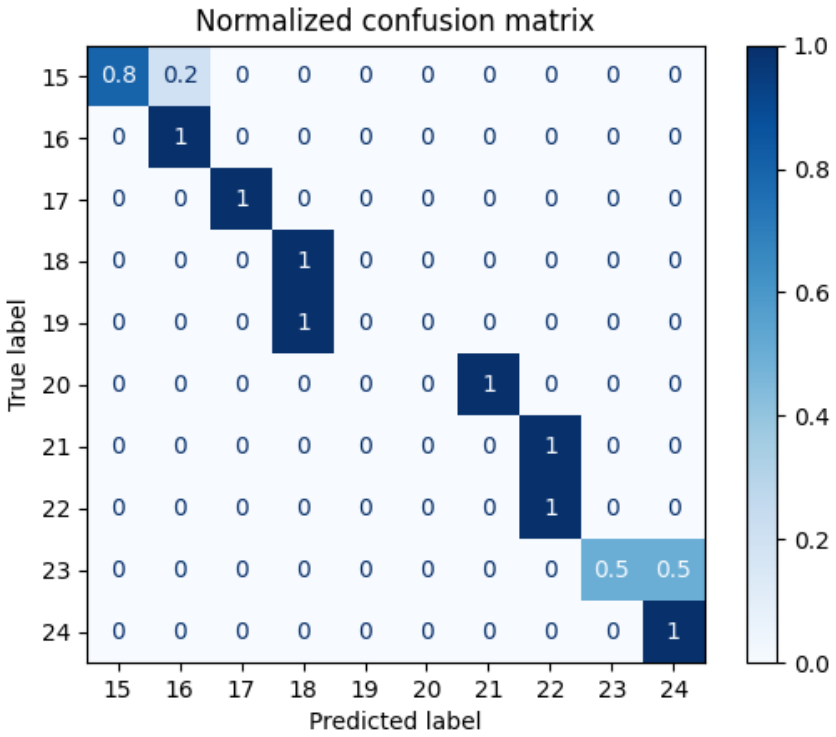

Рисунок 3.6- Нормалізована матриця неточностей для SVC моделі

	precision	recall	f1-score	support
15	1.00	1.00	1.00	5
16	0.50	1.00	0.67	1
17	1.00	0.50	0.67	2
18	1.00	0.33	0.50	6
19	0.00	0.00	0.00	0
20	0.00	0.00	0.00	0
21	0.00	0.00	0.00	0
22	1.00	0.67	0.80	3
23	1.00	1.00	1.00	2
24	1.00	1.00	1.00	1
accuracy			0.70	20
macro avg	0.65	0.55	0.56	20
weighted avg	0.97	0.70	0.77	20

Рисунок 3.7- Звіт LR моделі

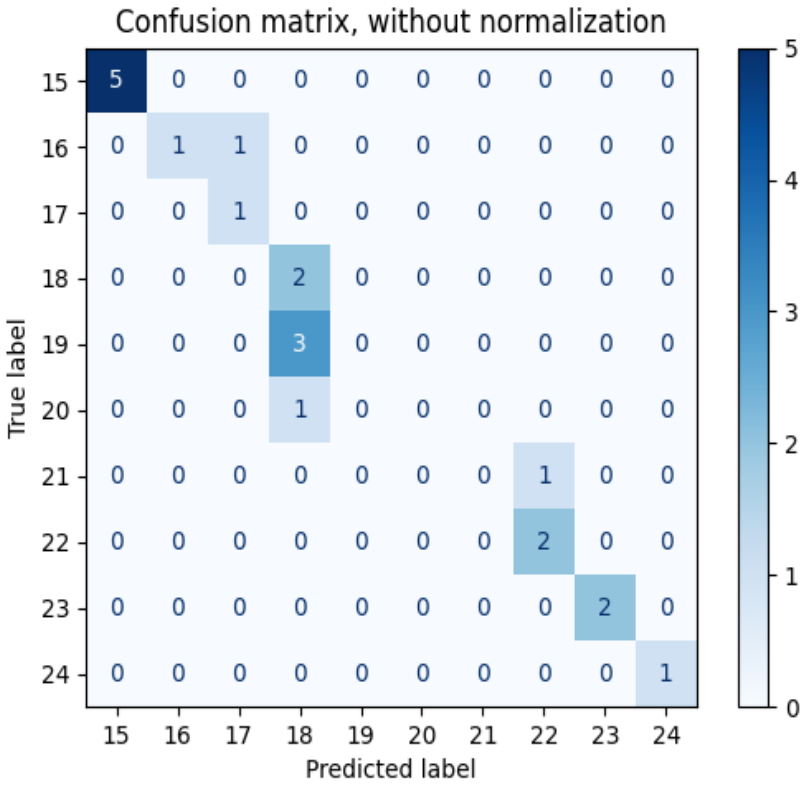

Рисунок 3.8 - Матриця неточностей для LR моделі

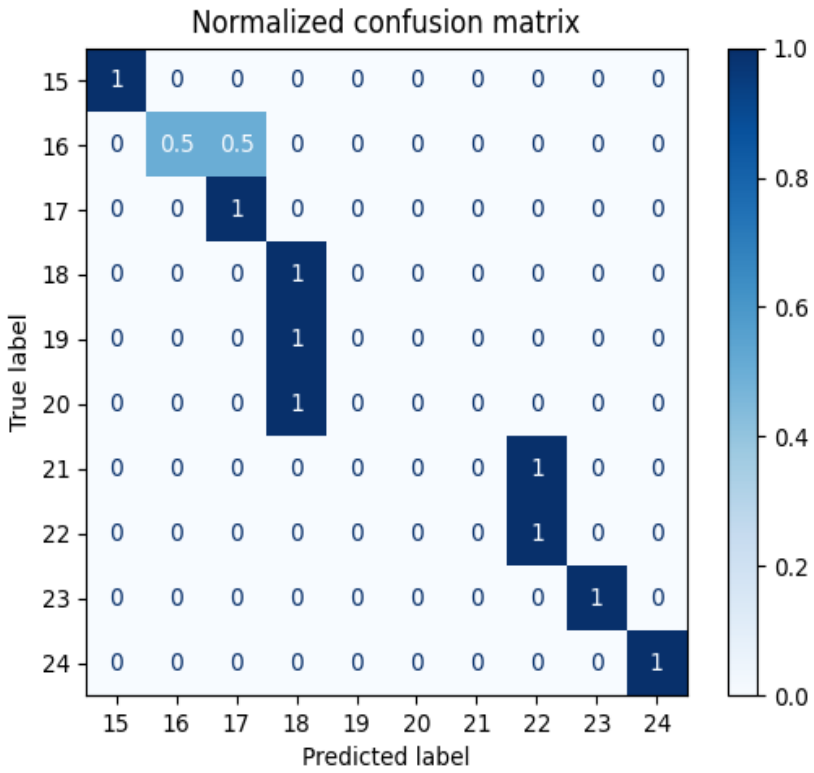

Рисунок 3.9 - Нормалізована матриця неточностей для LR моделі

3.5. Висновок

У цьому розділі було розглянуто процес реалізації класифікатора (в загальному вигляді) в програмному пакеті Scikit-Learn, було розглянуто основні етапи при роботі з класифікаторами - створення екземпляра, навчання і класифікація.

Було описано процес машинного навчання з усіма етапами, що до нього входять.

Також було докладно описано процес реалізації класифікатору, починаючи з оформлення даних для роботи і закінчуючи перевіркою працездатності навченої нейронної мережі, для вирішення поставленої проблеми. Задані ситуації були промодельовані, та були отримані позитивні результати і виявлено найкращий класифікатор для даної задачі.

- SVC_model = 0.65
- LR_model = 0.7
- RF_model = 0.9

Моделювання має певні недоліки:

- Моделювання не дає гарантії функціонування технологій у реальних умовах.
- Не представлений широкий масштабований проект.

Наукова робота у майбутньому буде вдосконалюватись, саме тому до ней можливо висунути рекомендації щодо створення великої моделі для дослідження працездатності.

ВИСНОВКИ

У науковій роботі була запропонована інженерна технічна ідея вирішення проблеми підвищення ефективності роботи бортового комп'ютера автомобіля, шляхом зменшення часу на взаємодію користувача з системою за допомогою передачі керування частиною функціоналу нейронній мережі.

Був проведений аналіз джерел інформацій у галузі використання нейронних мереж в «розумних автомобілях» та розглянуто приклади вже існуючих варіантів вирішення поставленої задачі. На основі порівняння технологій між собою, були відібрані найкращі за признаками перспективності розвитку, простоти втілення, ергономічності, та інших показників.

Було створено принципово новий підхід до використання нейронних мереж в автомобілях.

Проект було розглянуто та виділено умови, проблеми, та переваги. До проекту висунуті умови реалізації. Також були виділені функції, які повинен втілювати проект.

Обрані технічні рішення для реалізації окремих функцій та вузлів проекту були досліджені більше детально, та був розібраний механізм функціонування, були запропоновані певні ідеї для поліпшення, чи створення нововведень.

Актуальність проекту полягає в його унікальності, тому що з наведених аналогів існує лише один, який має щось спільне з темою кваліфікаційної роботи, але вся існуюча інформація є закритою, тому не можна з точністю сказати, наскільки вона схожа з розглядуваною темою.

При моделюванні було розглянуто можливість навчання нейронної мережі на зібраних з автомобіля даних. На цих даних було навчено нейронну мережу й було отримано певні, позитивні, результати, а також дали підтвердження необхідності використання подібної технології.

Моделювання має свої недоліки та переваги.

З переваг слід відмітити підтвердження певних наукових ідей, та висунених критеріїв. Була підтверджена відносна простота реалізації обраних технологій.

Серед недоліків є неможливість перевірки та створення великого макету проекту з причин недоступності, реального автомобіля на якому можна було б протестувати подібний функціонал.

Автор науково-дослідницької роботи має подальші рекомендації по розвитку дослідницької ідеї. А саме відтворення тестової моделі у реальному автомобілі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розумні автомобілі [Електронний ресурс] – Режим доступу до ресурсу: https://center2m.ru/ymnie-avtomobili
2. Який він, розумний автомобіль? [Електронний ресурс] – Режим доступу до ресурсу: https://www.nalin.ru/kakoj-on-umnyj-avtomobil-5679
3. Автомобіль розуміючий емоції водія [Електронний ресурс] – Режим доступу до ресурсу: https://hi-news.ru/auto/avtomobili-nauchat-ponimat-emocii-voditelej.html
4. «NISSAN IDS CONCEPT» [Електронний ресурс] – Режим доступу до ресурсу: https://www.nissan.co.uk/experience-nissan/concept-cars/ids-concept.html
5. «NIO» [Електронний ресурс] – Режим доступу до ресурсу: https://www.nio.com/de_DE?noredirect=
6. «ЕЛЕКТРИЧНИЙ КОНЦЕПТ-КАР LF-30 ELECTRIFIED» [Електронний ресурс] – Режим доступу до ресурсу: https://www.lexus.ru/discover-lexus/lexus-news/news/2019/electric-future-lf-30#hero
7. Нейронна мережа // Велика російська енциклопедія: [в 35 т.] / Гл. ред. Ю. С. Осипов. - М.: Велика російська енциклопедія, 2004-2017.
8. Мак-Каллок У. С., Піттс В. Логічне числення ідей, що відносяться до нервової активності Архівна копія від 27 листопада 2007 року на Wayback Machine // Автомати / Под ред. К. Е. Шеннона і Дж. Маккарті. - М .: Изд-во іноз. лит., 1956. - С. 363-384. (Переклад англійської статті 1943 г.)
9. Алгоритми класифікаторів. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/user_guide.html

10.«A Few Useful Things to Know About Machine Learning» [Електронний ресурс] – Режим доступу до ресурсу: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
11.Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.deeplearningbook.org/contents/ml.html
12.Ентропія і дерева прийняття рішень "на Хабре [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/171759/
13.«Logistic Regression - Interpreting Parameters» [Електронний ресурс] – Режим доступу до ресурсу: https://www.unm.edu/~schrader/biostat/bio2/Spr06/lec11.pdf