НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ
КАФЕДРА АКУСТИКИ ТА АКУСТОЕЛЕКТРОНІКИ

«На правах рукопису»
УДК 004.934.2

«До захисту допущено»
Завідувач кафедри
_____Дідковський В.С.
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

зі спеціальності (спеціалізації) 171 електроніка

на тему: Алгоритми розпізнавання емоцій за мовними сигналами

Виконав: студент VI курсу, групи ДГ-72мп

Діденко Данііл Юрійович
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник проф. каф. А та АЕ, д.т.н. Продеус А.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет електроніки
(повна назва)

Кафедра акустики та акустоелектроніки
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) 171 електроніка
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Діденку Даніілу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Алгоритми розпізнавання емоцій за мовними сигналами
науковий керівник дисертації: Продеус Аркадій Миколайович д.т.н., проф.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07»лтистопада 2018р. №4114с

2. Строк подання студентом дисертації 7 грудня 2018

3. Об'єкт дослідження розпізнавання емоцій за мовними сигналами

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) алгоритми розпізнавання емоцій за мовними сигналами

5. Перелік завдань, які потрібно розробити: дослідити алгоритми розпізнавання та классифікації емоцій за мовним сигналом, змоделювати та порівняти роботу алгоритмів, обрати найперспективніший.

6. Перелік графічного (ілюстративного) матеріалу: презентація у Power Point.

8. Дата видачі завдання: 01.09.2018р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Збір та вивчення джерел інформації для написання дипломної роботи	01-30.09.2018	
2	Складання плану дипломної роботи	1-06.10.2018	
3	Написання першого розділу	07-20.10.2018	
4	Написання другого розділу	21.10-03.11.2018	
5	Написання третього розділу	04-10.11.2018	
6	Написання четвертого розділу	10-15.11.2018	
	Виправлення зауважень	15-17.11.2018	
7	Оформлення дипломної роботи	23-29.11.2018	
8	Здача дипломної роботи	30.11.2018	
9	Захист дипломної роботи	17.12.2018	

Студент _____ Д. Ю. Діденко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А. М. Продеус
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Дисертація містить основну частину на 38 аркушах, 24 ілюстрації.

Метою дисертації є аналіз та моделювання алгоритмів розпізнавання емоцій за мовленнєвими сигналами.

Об’єктом дослідження є алгоритми розпізнавання емоцій.

Предметом дослідження є розпізнавання емоцій за мовленнєвим сигналом.

Результатом роботи є:

- Дослідження принципів дії алгоритмів розпізнавання емоцій;
- Дослідження акустичних ознак мовленнєвого сигналу;
- Моделювання та порівняння різних алгоритмів розпізнавання емоцій за мовленнєвим сигналом.

Галузь застосування: цифрова обробка акустичних сигналів.

Ключові слова: розпізнавання, класифікація, емоції, мовленнєвий сигнал, алгоритми машинного навчання, нейронні мережі.

SUMMARY

The thesis contains the main part on 38 sheets, 24 illustrations.

The purpose of the dissertation is to analyze and simulate the algorithms for recognizing emotions by speech signals.

The object of research is the algorithms of emotion recognition.

The subject of the study is the recognition of emotions by the speech signal.

The result of the work is:

- Research of the principles of the algorithms of emotional recognition;
- Investigation of acoustic signs of a speech signal;
- Simulation and comparison of various algorithms for recognizing emotions by speech signal.

Field of application: digital processing of acoustic signals.

Key words: recognition, classification, emotions, speech signal, machine learning algorithms, neural networks.

ЗМІСТ

ВСТУП

В останні роки були досягнуті значні успіхи у розвитку систем автоматичного детектування та розпізнання мовленнєвих сигналів. Такі системи активно використовуються у комп'ютерних та мобільних програмах, системах голосового керування, системах ідентифікації дикторів по голосу та системах розпізнавання мови. Але мовленнєвий сигнал, окрім текстової інформації, містить емоційний окрас, що несе інформацію про емоційний стан мовця та його відношення до промовленого. Ця інформація може бути використана для виявлення психофізичного стану людини у системах контролю здоров'я або у системах виявлення брехні. Задача автоматичного розпізнавання емоцій менш досліджена в порівнянні з розпізнаванням мови. Причиною може бути менш широкий спектр задач де необхідна така інформація.

Створення системи розпізнавання емоцій, потребує вирішення наступних задач: виділення з мовленнєвого сигналу параметрів що відповідають за емоції, виділення інформаційних ознак та створення методів класифікації емоцій. Сучасні концепції систем розпізнавання емоцій, базуються на алгоритмах машинного навчання, які показують себе вельми ефективними у широкому спектрі задач розпізнавання та класифікації, у тому числі і в задачі розпізнавання мови.

РОЗДІЛ 1. АКУСТИЧНІ ОЗНАКИ МОВЛЕННЄВОГО СИГНАЛУ

Вступ

Першим етапом розробки системи класифікації емоцій є акустичний аналіз та обробка мовленнєвого сигналу, яка полягає у фільтрації завад та співставленні набору інформативних ознак до кожного фрагменту сигналу. Ці ознаки містять закодовану інформацію про емоції на даному фрагменті. Акустична обробка є одною з найважливіших частин створення системи розпізнавання емоцій, оскільки від її результатів сильно залежать результати роботи всієї системи. Виділення з сигналу акустичних ознак робиться з метою зменшити надлишковість сигналу та виділити найбільш корисну інформацію. Всі корисні ознаки, які описують сигнал з різних точок зору, комбінуються в один вектор ознак (супервектор), за допомогою якого буде відбуватися класифікація. Найбільш поширеними ознаками в задачі розпізнавання емоцій є: мел-частотні кепстральні коефіцієнти, хромаграма, короткочасна енергія сигалу, спектральний центроїд та zero crossing rate [1].

Розглянемо найбільш поширені ознаки, що виділяються з мовленнєвого сигналу.

1.1 Хромаграма

Хромаграма – це послідовність хроматичних векторів що являють собою розподілення енергії по частоті сигналу у 12 частотних смугах [5].

Розрахунок хромаграми виконується наступним чином:

1. Ноти у музиці логарифмічно співвідносяться за наступною формулою:

$$f_p = f_{ref} * 2^{\frac{p-69}{12}} \text{Гц}$$

   де p – номер ноти, f_{ref} – опорна частота, зазвичай дорівнює 440 Гц.

2. Для розрахунку тонового представлення у частотній області, елементи спектра мають бути присвоєні до найближчого тону:

$$M(i) = [12 * log_2\left(\frac{f_i}{f_{ref}}\right) + 69]$$

де f_i – частота, що відповідає i-му елементу спектра.

3. Тоновий спектр розраховується за формулою:

$$P(p) = \sum_{M(i)\ =p} |X(i)|^2$$

де X – спектр сигнала.

4. Хроматичний вектор з 12 коефіцієнтами розраховується за формулою:

$$Chroma(c) = \sum_{p:p=c(mod\ 12)} P(p)$$

Приклад хромаграми зображений на рис. 1.

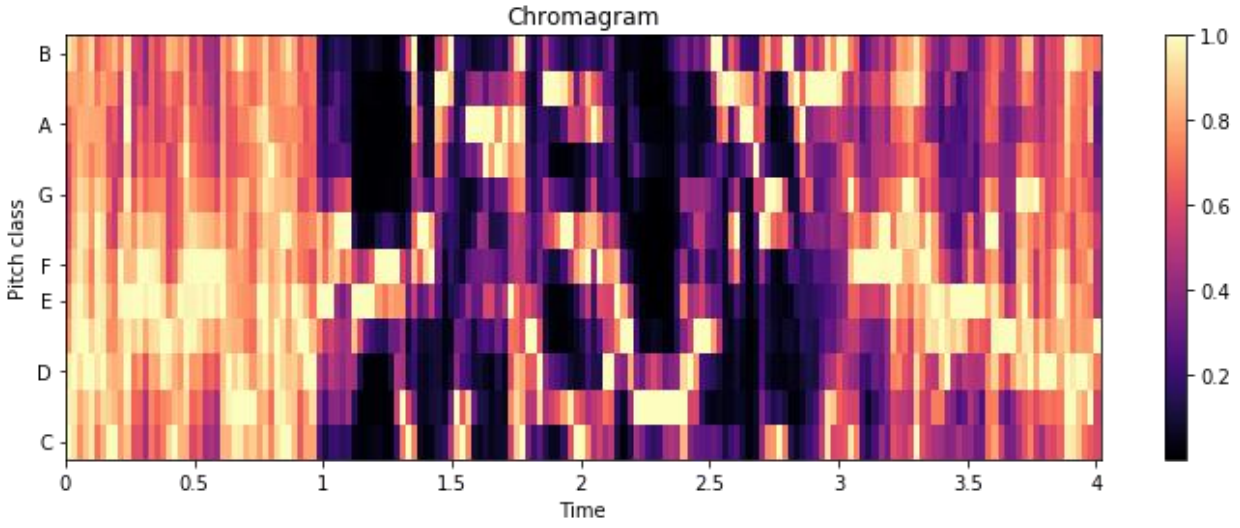

Рис. 1

1.2 Мел-частотні кепстральні коефіцієнти

Розглянемо процес розрахунок коефіцієнтів для одного фрейму.

Представимо фрейм у вигляді вектору x[k], $0 \leq k < N$, де N - розмір фрейму.

Спочатку розраховується спектр сигналу за допомогою дискретного перетворення Фур'є.

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{-2\pi ikn/N}, 0 \leq k < N$$

До отриманих значень застосовується віконна функція Хеммінга, щоб згладити значення на границях фреймів.

$$H[k] = X[k]H[k], 0 \leq k < N$$

Оскільки сприймана людиною висота звуку нелінійно залежить від його частоти, була введена психофізична одиниця висоти звуку, яка враховує цю нелінійність – мел. Графік залежності висоти звуку від частоти зображений на рис. 2, та описується формулою:

$$m = 1125\ln(1 + f/700)$$

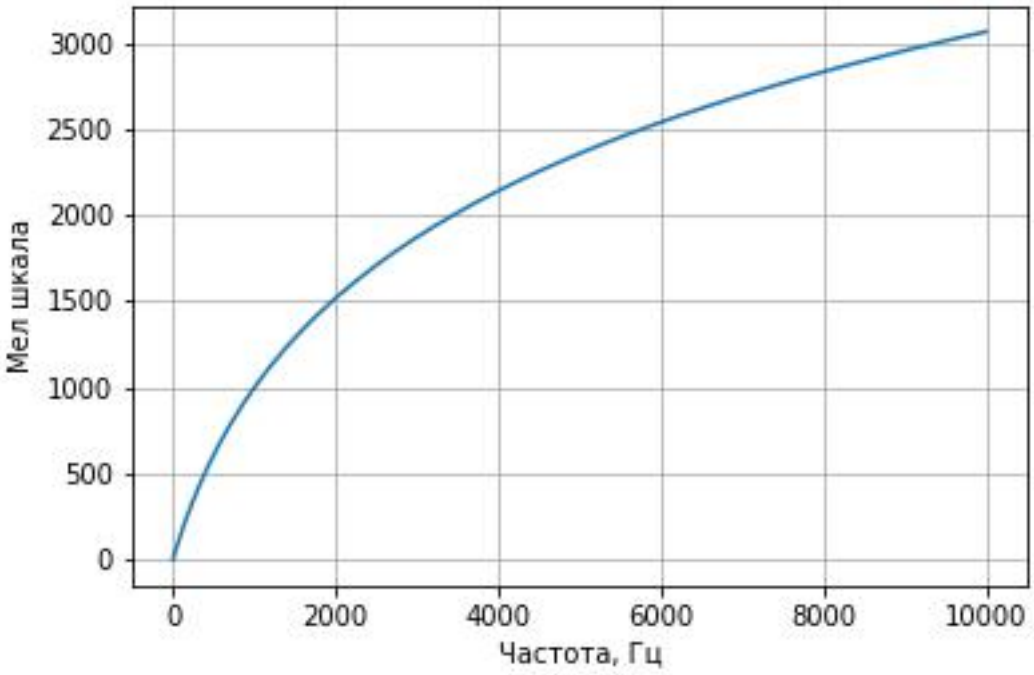

Рис. 2 Залежність висоти звуку від його частоти

Для того щоб розкласти спектр по мел-шкалі, використовується гребінка фільтрів, де кожний фільтр це трикутна віконна функція, яка сумує кількість

енергії на певному частотному діапазоні та отримує мел-коефіцієнт. Гребінка фільтрів зображена на рис. 3.

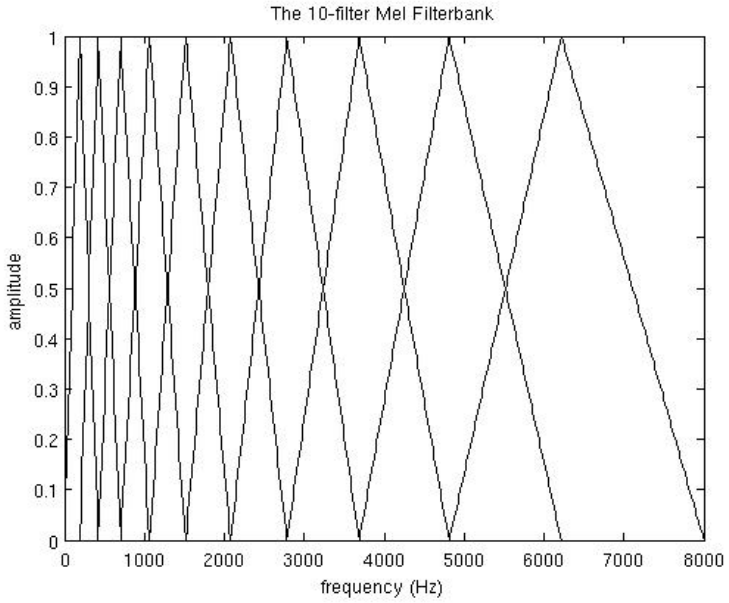

Рис. 3

Гребінка фільтрів будується за наступною формулою:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

Далі значення фільтрів перемножуються зі значеннями спектру. В результаті отримуються мел-коефіцієнти. Кількість коефіцієнтів дорівнює кількості фільтрів *M*.

$$S[m] = log\left(\sum_{k=0}^{N-1} |X[k]|^2 H_m[k]\right), 0 \le m < M$$

Далі отримуємо кепстральні коефіцієнти, виконавши дискретне косинусне перетворення.

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos\left(\pi * l * \frac{m + \frac{1}{2}}{M}\right), 0 \leq l < M$$

Таким чином отримується матриця розміром M*C, де M – кількість фреймів у сигналі, а C – кількість коефіцієнтів у фреймі.

Графік мел-частотних кепстральних коефіцієнтів зображений на рис. 4.

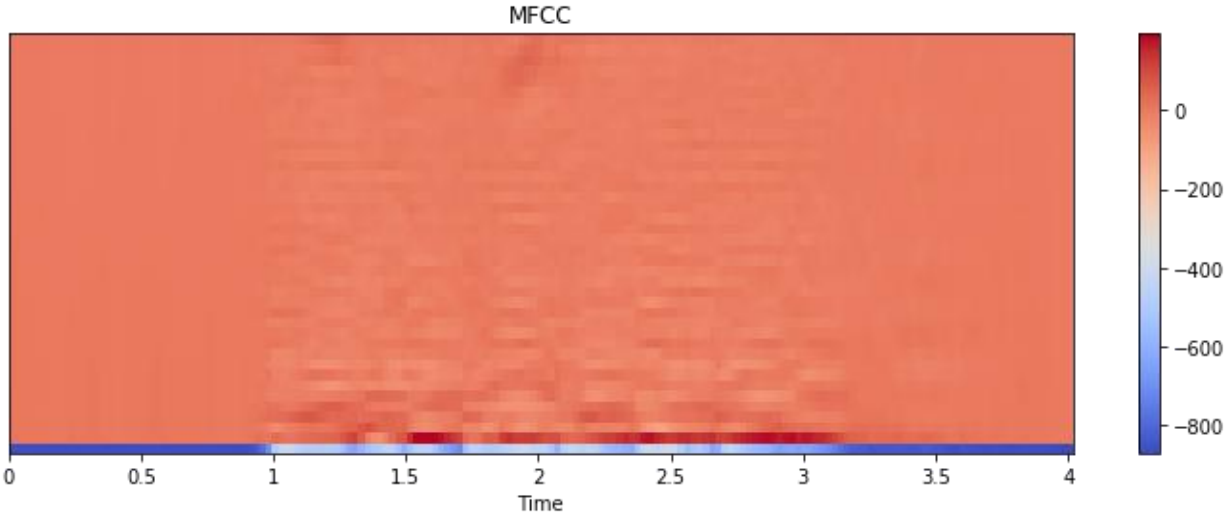

Рис. 4

1.3 Короткочасна енергія

Короткочасна енергія сигналу розраховується за наступною формулою:

$$E = \frac{1}{N} \sum_{k=1}^{N} s_k^2$$

де N – довжина фрейму, s_1,...,s_k,...,s_N - послідовність відліків.

1.4 Zero crossing rate

Zero crossing rate (ZCR) – вказує на кількість перетинань функцією осі OX. Це дає інформацію про наявність сигналу і його частотній складовій. Якщо знак функції часто міняє знак, то це вказує на наявність високочастотної складової. ZCR знаходиться за допомогою наступної формули:

$$Z = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{R<0}(s_t s_{t-1})$$

де s – це сигнал довжиною T, а $1_{R<0}$ – це індикаторна функція.

1.5 Спектральний центроїд

Спектральний центроїд (Spectral centroid) – являє собою інтерпретацію ‘центра мас’ спектра. Розраховується як сума частот, взважених відповідними амплітудами спектра, поділена на суму амплітуд.

$$SC = \frac{\sum_{k=1}^{N} kF[k]}{\sum_{k=1}^{N} F[k]}$$

де $F[k]$ – це амплітуда спектра, відповідна до k-го значення частоти у спектрі ДПФ. Графік спектрального центроїда зображений на рис. 5.

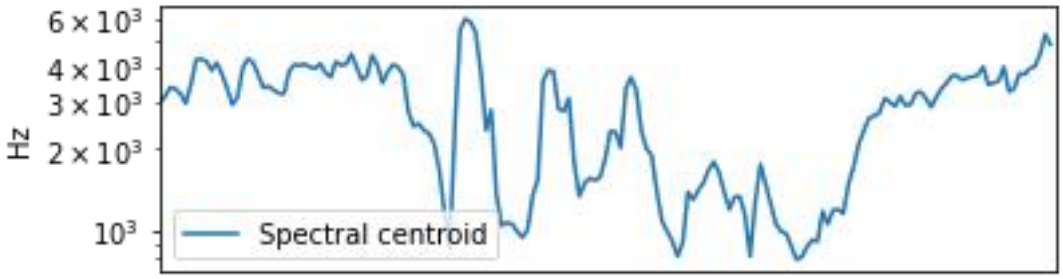

Рис. 5

РОЗДІЛ 2. АЛГОРИТМИ МАШИННОГО НАВЧАННЯ

Вступ

Алгоритми машинного навчання діляться на дві категорії: алгоритми навчання з учителем (supervised learning), та алгоритми навчання без учителя (unsupervised algorithms) [4].

У методах навчання з учителем, алгоритм отримує від користувача пари «об'єкт – відповідь», та знаходить способи отримання відповіді по об'єкту. Після навчання на великій кількості пар «об'єкт – відповідь», та узагальнивши отриману інформацію, алгоритм може видавати відповідь для об'єкту, якого він ніколи не отримував.

У методах навчання без учителя, алгоритм отримує об'єкти без відповідей. Такий алгоритм доцільно використовувати тоді, коли ми не знаємо наперед, які відповіді можуть бути отримані. Результати роботи таких алгоритмів значно важче інтерпретувати.

В даній роботі будуть розглянуті алгоритми навчання з учителем.

2.1 Класичні класифікатори

2.1.1 Метод k найближчих сусідів

Метод *k* найближчих сусідів є одним з найпростіших алгоритмів машинного навчання [4]. Будування моделі полягає у вивченні навчального набору даних. Для того, щоб зробити прогноз для нової точки даних, алгоритм знаходить найближчі до неї точки навчального набору, тобто знаходить «найближчих сусідів». На рис. 6 кружками та трикутниками відображені точки двох класів навчального набору даних, а зірочками нові точки які треба класифікувати. В даному випадку алгоритм працює по 3 найближчим сусідам. Кожна нова точка відноситься до того класу до якого відноситься більшість її сусідніх точок.

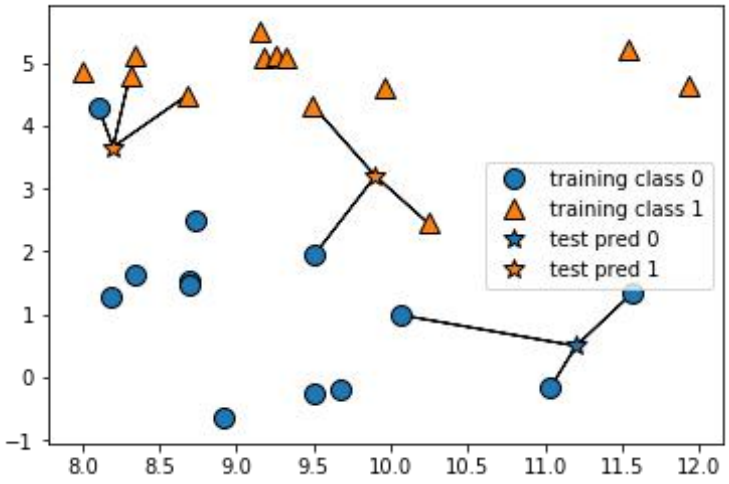

Рис. 6

Як можна побачити з рис. 7, кількість сусідніх точок по яким ведеться класифікація, значно впливає на границю прийняття рішень. У випадку моделі одного найближчого сусіда дає границю, яка дуже добре співпадає з навчальними даними. З більшою кількістю сусідів, границя згладжується, що відповідає більш простій моделі.

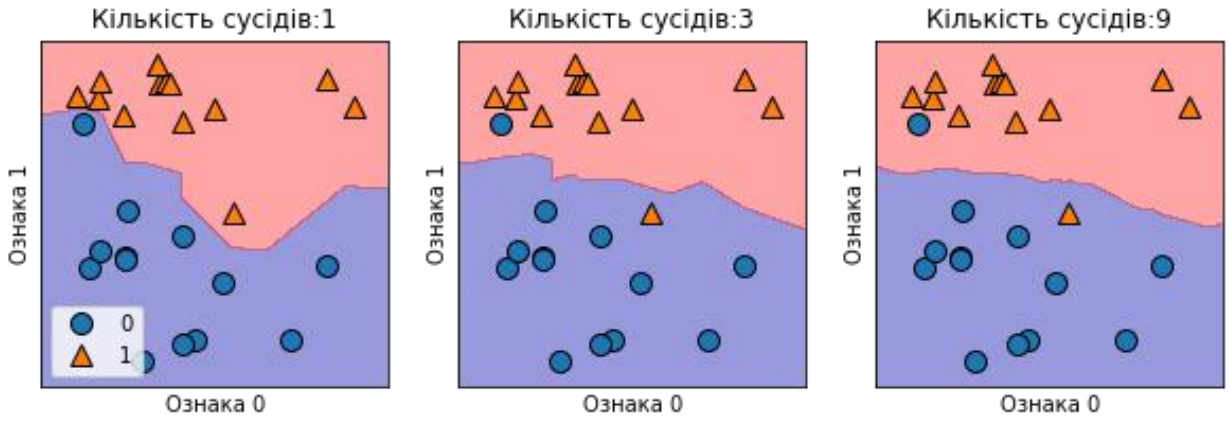

Рис. 7

Чим складніша модель, тобто, чим краще вона розділяє навчальні дані, тим гіршу здатність до узагальнення вона має. І навпаки, чим простіша модель, тим грубіше вона буде класифікувати нові дані. На рис. 8 зображений приклад залежності правильності класифікації від кількості сусідів для деякого набору даних.

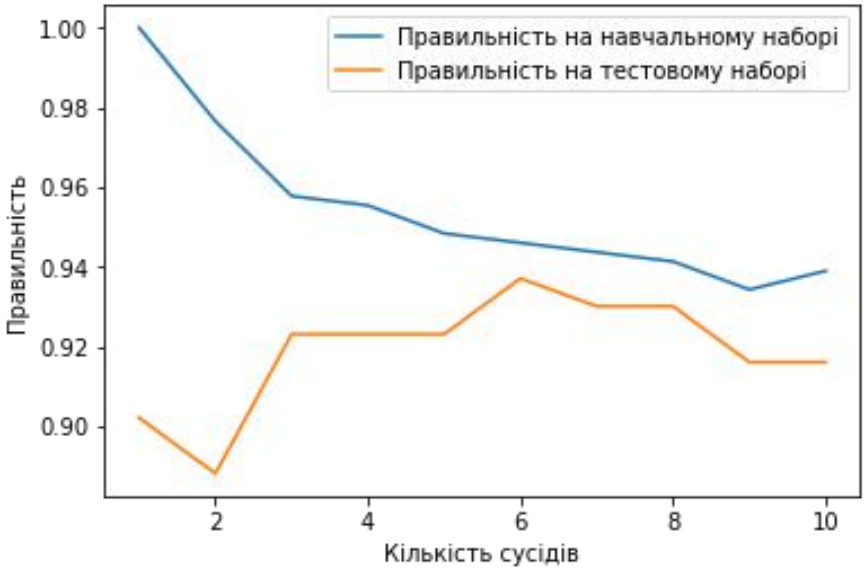

Рис. 8

Переваги методу:

Перевагами методу k найближчих сусідів є те, що цю модель дуже легко інтерпретувати і цей метод дає задовільну якість без великої кількості налаштувань.

Недоліки методу:

Недоліками цього методу є погана робота з великою кількістю ознак та якщо більшість ознак мають нульові значення.

2.1.2 Метод опорних векторів

Метод опорних векторів є одним з видів лінійних класифікаторів. Лінійні моделі дають прогноз, використовуючи лінійну функцію вхідних ознак. Для бінарної класифікації прогноз отримується за наступною формулою:

$$\widehat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b > 0$$

де x[0] до x[p] – ознаки для окремої точки даних, w і b – параметри моделі, що оцінюються у процесі навчання, а $\widehat{y}$ – прогноз, що видається моделлю.

Якщо функція менше нуля, то прогнозується клас -1, якщо вона більше нуля, прогнозується клас +1. Лінійний класифікатор розділяє класи за допомогою лінії, площини або гіперплощини.

Припустимо що є два набори даних, що належать двом класам (рис. 9).

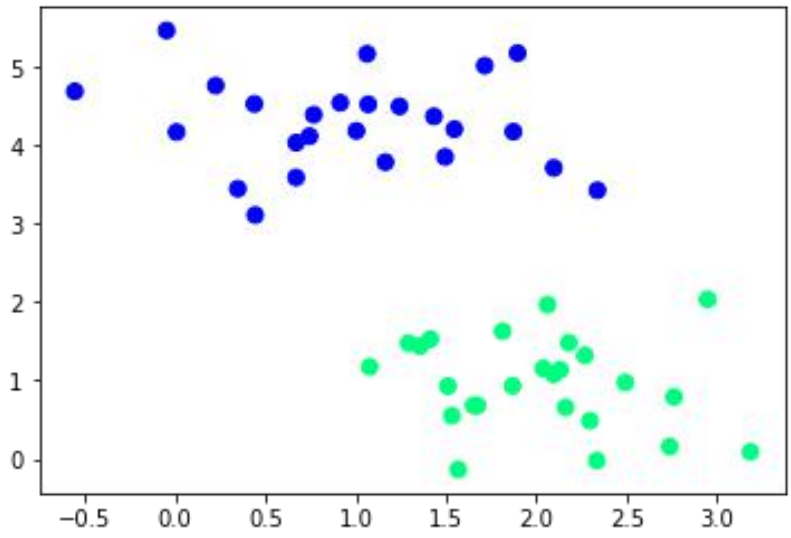

Рис. 9

Ці класи можна розділити лінією. Але цю лінію можна провести по різному. Як можна побачити з рис. 10, від способу проведення лінії залежить те, в який клас потрапить нова точка, яка позначена символом X.

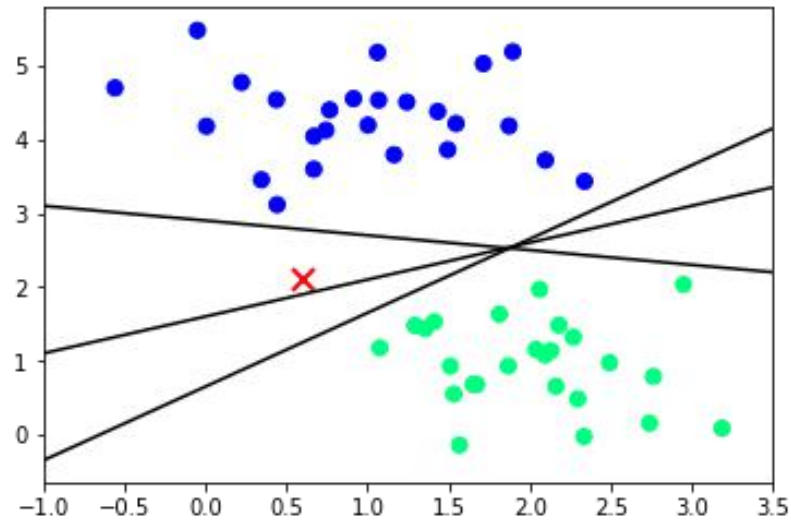

Рис. 10

Метод опорних векторів вирішує цю проблему. Розділяюча лінія проводиться таким чином, щоб відстань від неї до найближчих точок класів була максимальна (рис. 11).

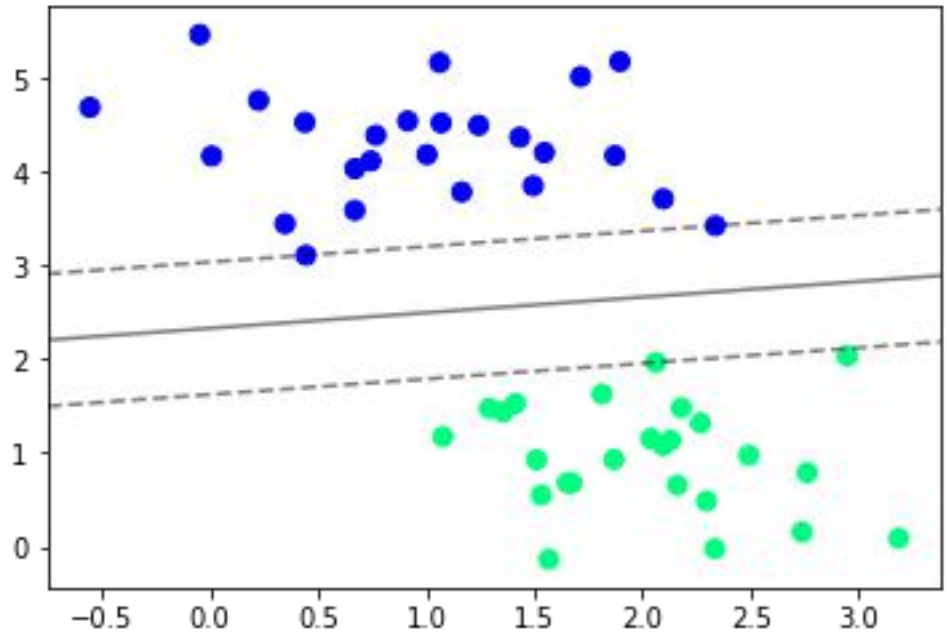

Рис. 11

Точки що лежать на відступі називаються опорними векторами. Контролюючим параметром моделі опорних векторів є параметр C. Більші значення параметру C відповідають більшій складності моделі, тобто модель обирає такі параметри, щоб кожна точка була класифікована правильно. На рис. 12 показано як змінюється розділяюча границя в залежності від параметру C.

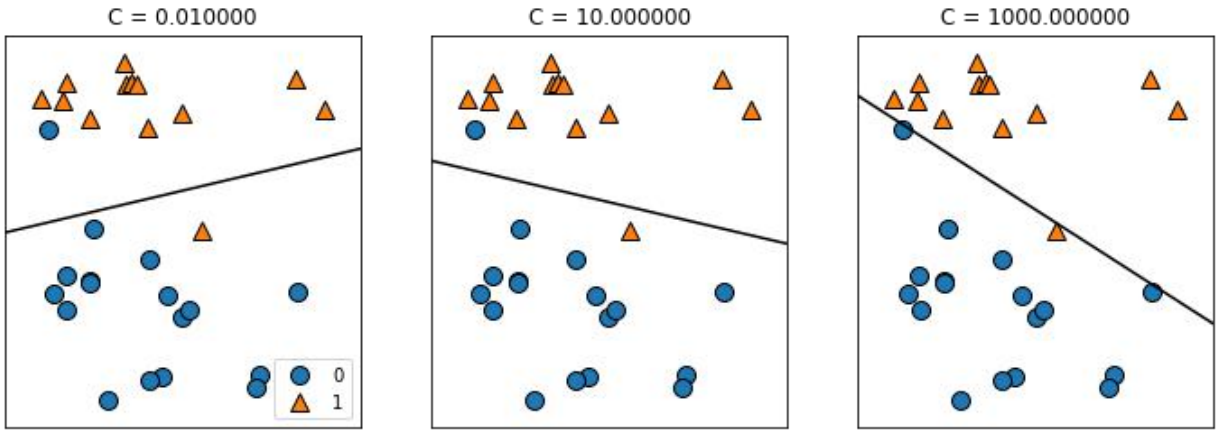

Рис. 12

У випадку мультикласової класифікації, лінійні методи працюють по принципу «один проти всіх». Цей принцип полягає у проведенні кількох ліній, кожна з яких відділяє один з класів від всіх інших.

Але досить часто бувають такі випадки, коли лінійне розділення класів неможливо. Такий випадок зображений на рис. 13.

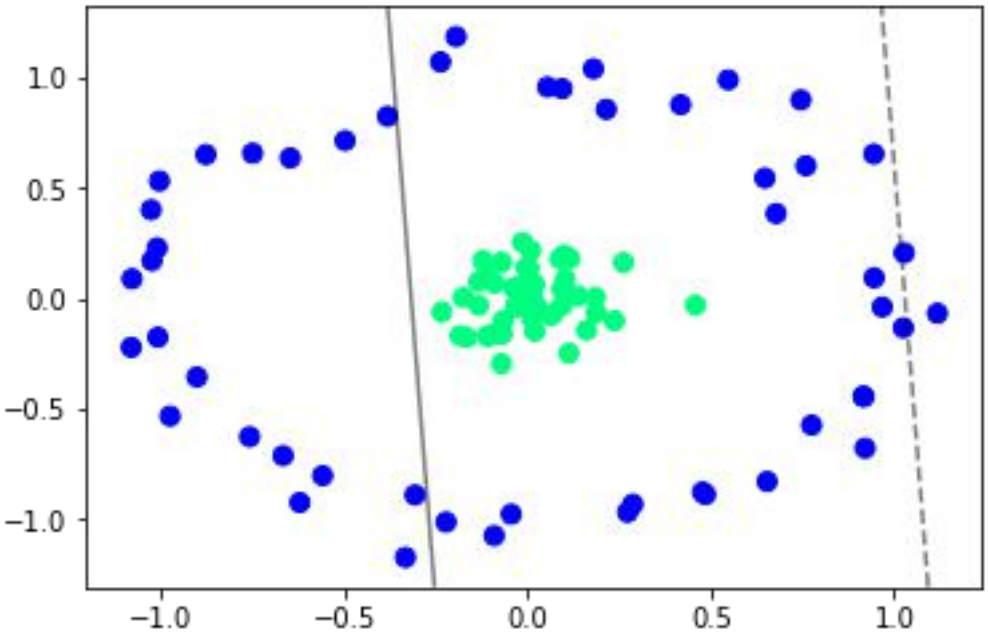

Рис. 13

Тоді, дані проектуються на простір більшої розмірності. Наприклад, одна з простих проекцій – розрахунок радіальної базисної функції (також ядро

Гауса), центрованої посередині даних. Відстань між точками даних вимірюється за наступною формулою:

$$k_{rbf}(x_1,x_2) = e^{-\gamma\|x_1 - x_2\|^2}$$

де x_1 та x_2 – точки даних, $\|x_1 - x_2\|^2$ – евклідова відстань, а γ (gamma) – параметр, який регулює ширину ядра Гауса. Тепер, виходячи з рис. 14, видно що класи легко розділити за допомогою площини.

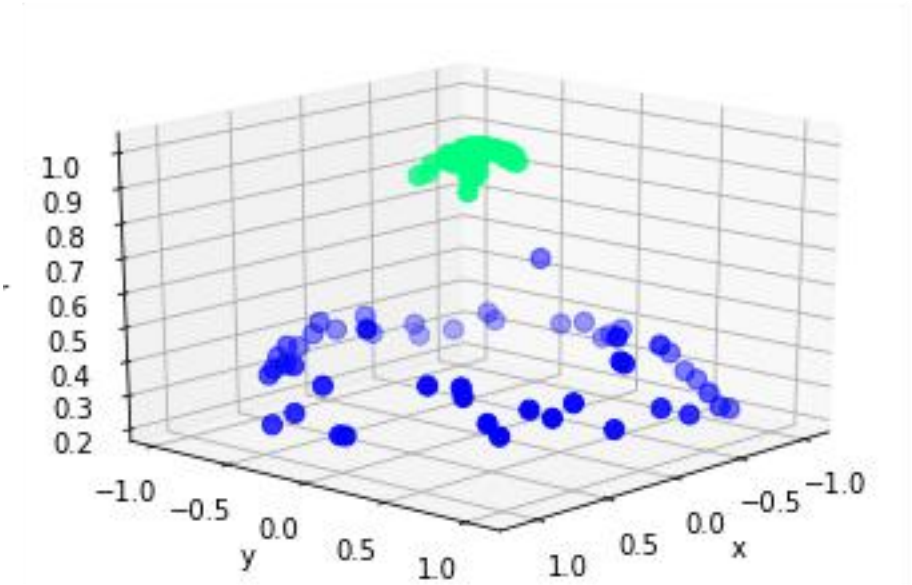

Рис. 14

Повертаючись до двомірного простору, бачимо перехід від лінійного розділення до нелінійного (рис. 15).

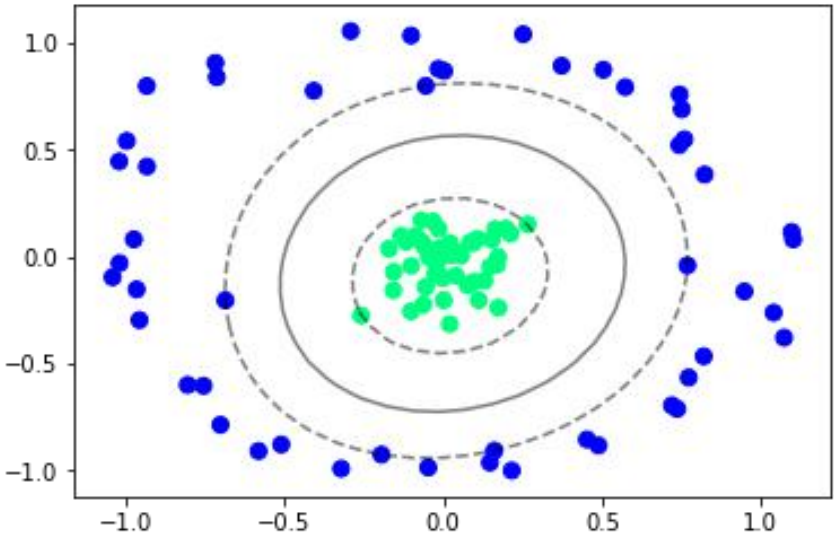

Рис. 15

Але при зростанні кількості точок зростає кількість вимірів, що призводить до величезного зростання кількості розрахунків. Ця проблема вирішується при використанні математичної процедури під назвою kernel trick, яка дозволяє навчання на перетворених за допомогою ядра даних у неявному вигляді, тобто, не будуючи повного N-мірного простору ядерної проекції.

Переваги методу:

Метод опорних векторів однаково добре працює як з великою так і з малою кількістю ознак. Більш того, він дозволяє отримати задовільні результати у випадку, коли ознак більше ніж вимірюваннь.

Недоліки методу:

Метод опорних векторів дуже чутливий до масштабування вхідних даних та потребує дуже точних налаштувань.

2.1.3 Ансамблі дерев рішень

Дерева рішень це моделі що досить широко використовуються для вирішення задач класифікації. Принцип їх роботи полягає у вибудовуванні ієрархії правил «якщо… то», яка приводить до рішення максимально коротким шляхом [4]. Ці правила називаються тестами. Якщо дані

представлені у вигляді неперервних ознак, то тести мають вигляд «Ознака *i* більше значення *a?*». Для того щоб побудувати дерево, алгоритм перебирає усі можливі тести, та обирає той, який є найбільш інформативним з точки зору прогнозування значень цільової змінної. Для набору значень, що зображений на рис. 16, перший тест повинен розділити два класи найбільш ефективним способом. Результат першого тесту зображений на рис. 17.

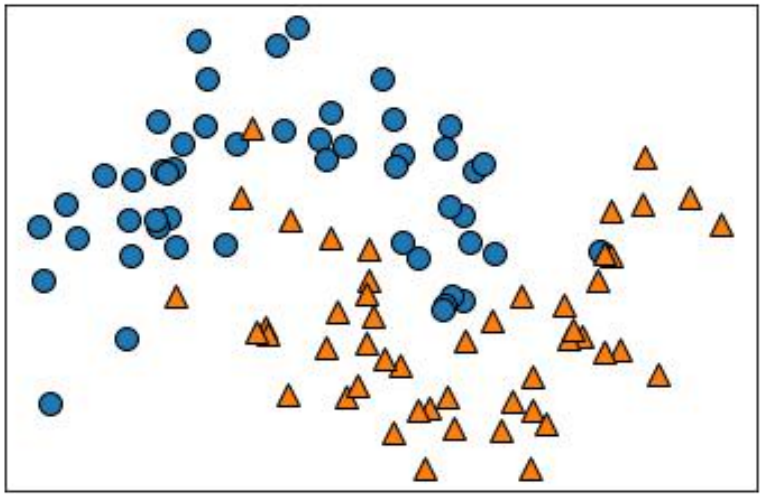

Рис. 26

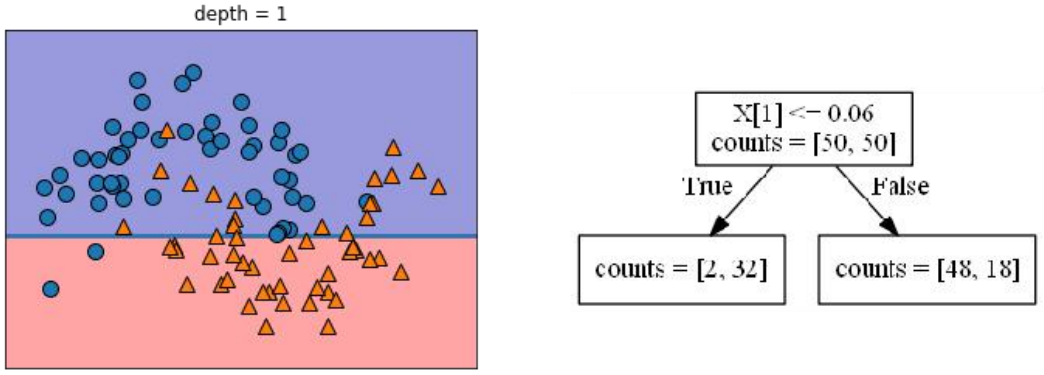

Рис. 17

Результат пошуку другого тесту зображений на рис. 18.

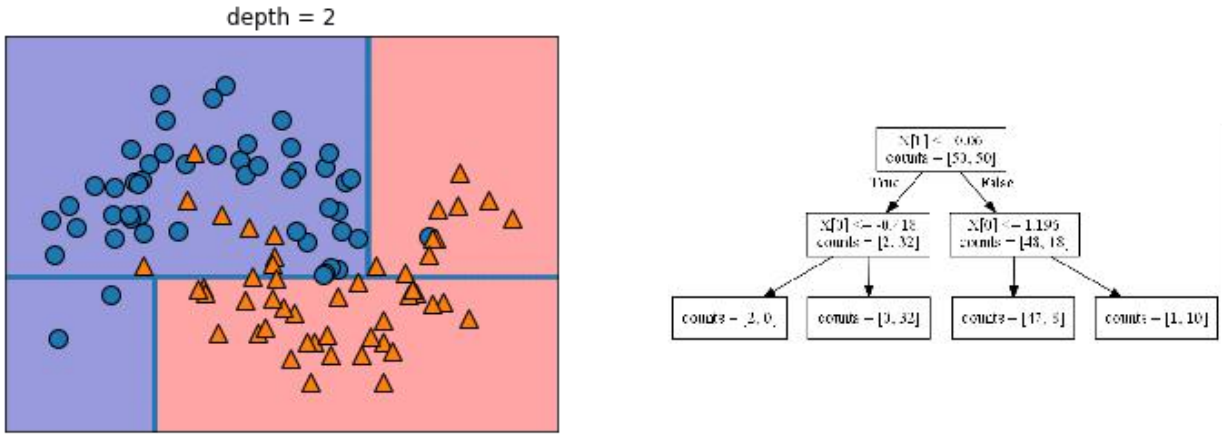

Рис. 18

Кожен тест розглядає одну ознаку, а області що отримуються, завжди мають границі паралельні осям. Рекурсивне розбиття даних повторюється до тих пір, поки всі точки у кожній області розбиття (у кожному листі дерева) не будуть належати до одного і того ж значення цільової змінної (класу). На рис. 19, зображено кінцеве розбиття набору даних.

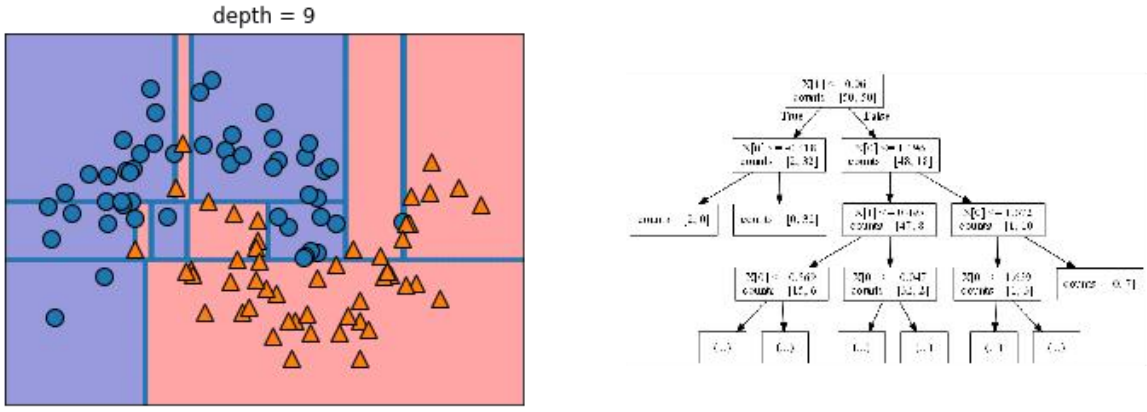

Рис. 39

Прогнозування для нових точок відбувається наступним чином: спочатку визначається у якій області розбиття простору ознак знаходиться дана точка, потім визначається клас, до якого класу відносяться більшість точок в цій області. Необхідна область знаходиться шляхом проходження дерева, починаючи з першої умови, а потім, у відповідності до виконання наступних умов.

Для контролю складності моделі дерева рішень є дві стратегії. Перша стратегія – рання зупинка будування дерева, що називається попередньою обрізкою. Друга стратегія – побудова дерева з наступним видаленням малоінформативних вузлів, що називається пост-обрізкою. Критеріями попередньої обрізки є обмеження максимальної глибини дерева, обмеження максимальної кількості віток та мінімальна кількість спостережень у вузлі, необхідна для розбиття.

Випадковий ліс – це набір дерев рішень, де кожне дерево трохи відрізняється від інших. Ідея полягає в тому, що якщо побудувати багато дерев, які добре працюють та перенавчаються з різним ступенем, можна зменшити перенавчання шляхом усереднення їх результатів.

Переваги методу:

Випадковий ліс є досить простим у налаштуваннях методом, має високу прогнозну силу та добре працює з великими наборами даних.

Недоліки методу:

Гірше працює з на даних високої розмірності ніж лінійні моделі, а також потребує більшої кількості пам’яті та навчається з меншою швидкістю ніж лінійні моделі.

2.2 Штучні нейронні мережі

Штучні нейронні мережі – це математична модель, що побудована за принципом організації та функціювання біологічних нейронних мереж. Вона являє собою послідовність нейронів, що зв’язані між собою синапсами.

Нейрон являє собою одиницю обробки інформації у нейронній мережі. На рис. 20 зображена модель нейрона, що лежить в основі штучних нейронних мереж. В цій моделі можна виділити три елемента [8].

1. Набір синапсів або зв’язків, кожен з яких характеризується своєю вагою або силою. В даному випадку, сигнал x_n на вході синапса n, зв’язаного з нейроном, помножується на вагу w_n. Синаптична вага

може мати як штучного нейрону може приймати як додатні, так і від'ємні значення.

2. Суматор сумує вхідні сигнали, що зважені відносно відповідних синапсів нейрона.
3. Функція активації обмежує амплітуду вихідного сигналу нейрона. Ця функція також називається функцією стиснення. Зазвичай нормований діапазон амплітуд виходу нейрона лежить у інтервалі [-1, 1] або [0, 1].

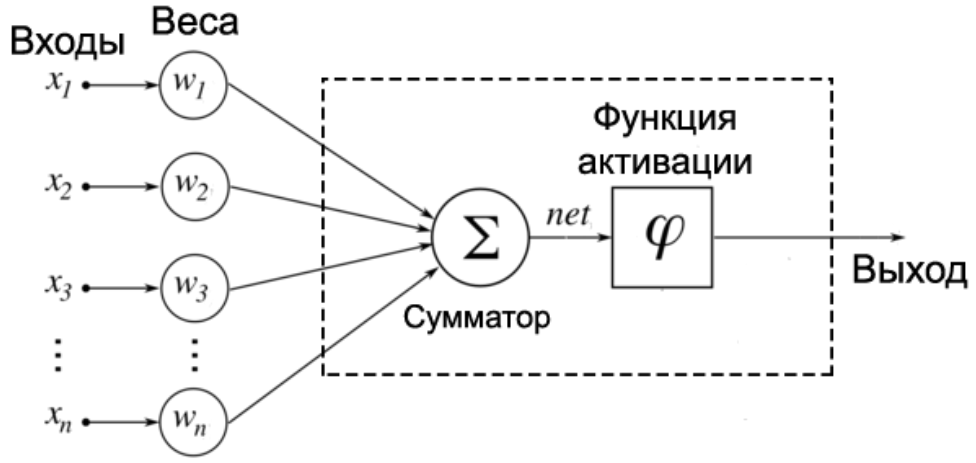

Рис. 20

2.2.1 Перцептрон

Перцептрон – це простий алгоритм, який отримує вхідний вектор x, що мітить n значень $(x_1,x_2,...,x_n)$, які називаються вхідними ознаками, та видає на виході 1 (так) або 0 (ні) [7]. Таким чином визначається функція:

$$f(x) = \begin{cases} 1, & wx + b > 0 \\ 0, & wx + b \leq 0 \end{cases}$$

де w – ваговий вектор, wx – скалярний добуток $\sum_{j=1}^{m} w_j x_j$, а b – зміщення. Оскільки рівняння $wx + b = 0$ описує граничну гіперплощину, можна сказати що перцептрон має схожий принцип роботи з лінійними класифікаторами.

Навчання перцептрона являє собою зміну вагової матриці. Існують 4 виду перцептронів:

1. Перцептрон з одним прихованим шаром;

2. Одношаровий перцептрон, у якому вхідні елементи з’єднані з вихідними напряму за допомогою вагової системи. Ця мережа є найпростішою мережею прямого розповсюдження.;
3. Багатошаровий перцептрон, який має додаткові приховані шари;
4. Багатошаровий перцептрон, який має приховані шари, та навчається за методом зворотнього розповсюдження похибки.

Проблемою нейронних мереж, що складені з перцептронів є те, що через стрибкоподібну функцію активації, на виході кожного нейрону може бути лише 1 або 0 і не може бути ніяких проміжних значень. Це означає, що навіть невеликі зміни у вхідному сигналі можуть викликати кардинальні зміни у роботі мережі. Через це стає неможливим поступове навчання. Для того щоб позбавитися цього недоліку були створені інші функції активації.

Найпопулярнішими функціями активації є сигмоїдальна функція та блок лінійної ректифікації.

Сигмоїдальна функція

Сигмоїдальна функція визначається за наступною формулою:

$$\sigma(x) = \frac{1}{1 + e^{-ax}}$$

Сигмоїдальна функція є неперервною та змінюється від 0 до 1, коли аргумент проходить область визначення $(-\infty,\infty)$. Це дозволяє отримувати на виході нейрону проміжні значення між 0 та 1. Змінюючи коефіцієнт a, можна змінювати крутизну функції. Вплив коефіцієнту a можна побачити на рис. 21.

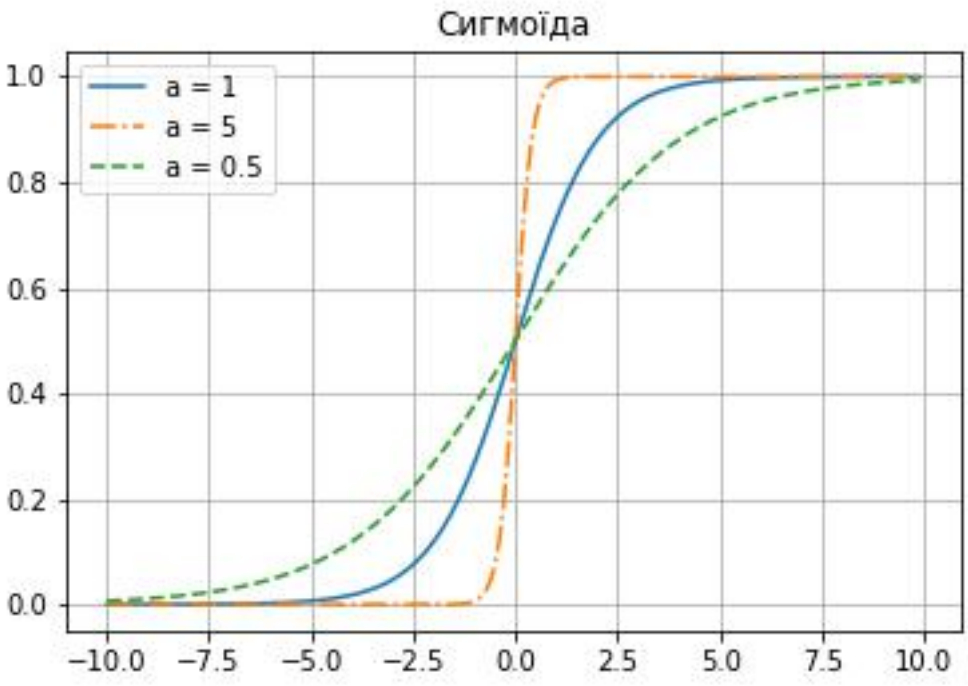

Рис. 21

Блок лінійної ректифікації

Блок лінійної ректифікації (rectified linear unit, ReLU) – це дуже проста функція, яка дає чудові результаті та стала однією з найпопулярніших. ReLU визначається формулою:

$$f(x) = \max(0,x)$$

Графік функції ReLU зображений на рис. 22.

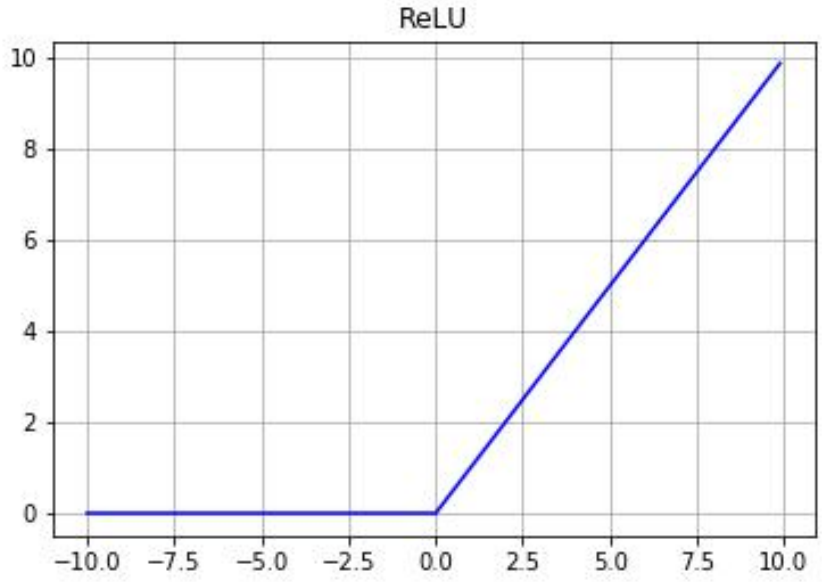

Рис. 22

Нейрони у нейронних мережах, можна поділити на групи в залежності від їх розташування у мережі:

1. Вхідні нейрони, на які поступає вектор вхідних ознак;
2. Приховані нейрони, у яких відбуваються основні операції навчання;
3. Вихідні нейрони, які виводять результат роботи нейронної мережі.

Вихідні значення багатошарового перцептрону розраховуються наступним чином:

1. Візьмемо нейронну мережу з одним прихованим шаром нейронів, що зображена на рис. 23. Дана модель містить три вхідних нейрони, три прихованих нейрони, та один вихідний.

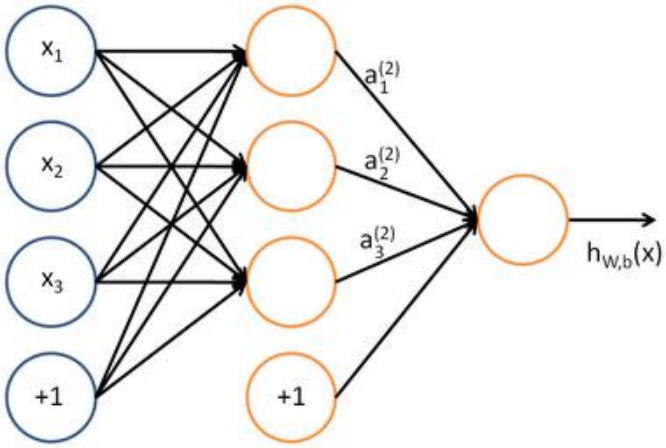

Рис. 23

2. Виходи прихованих нейронів, що позначені як a_n , будуть розраховуватися наступним чином:

$$a_1^{(2)} = f\left(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_1^{(1)}\right)$$
$$a_2^{(2)} = f\left(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_1^{(1)}\right)$$
$$a_3^{(2)} = f\left(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_1^{(1)}\right)$$

3. Тоді отримуємо наступну систему, що описуватиме мережу:

$$\begin{cases} z^{(2)} = w^{(1)}x + b^{(1)} \\ a^{(2)} = f(z^{(2)}) \\ z^{(3)} = w^{(2)}a^{(2)} + b^{(2)} \\ h = f(z^{(3)}) \end{cases}$$

де z – функція сумування.

Алгоритм зворотного поширення помилки

Багатошаровий перцептрон навчається на даних за допомогою процесу, що називається зворотнім поширенням. Його можна описати як постійне виправлення помилок по мірі їх знаходження.

Спершу усім ваговим коефіцієнтам призначаються випадкові значення. Потім мережа активується для кожного вхідного значення з навчального набору: значення розповсюджуються у прямій направленості від вхідного шару через приховані шари до вихідного, який видає прогноз. Оскільки

істинне значення для навчального набору відоме, то можна розрахувати похибку прогнозу. Ідея полягає в тому, щоб виконати зворотне розповсюдження похибки та за допомогою алгоритму оптимізації (наприклад, градієнтного спуску), скорегувати вагові коефіцієнти нейронної мережі з метою зменшення похибки. Цей процес повторюється кілька разів, поки похибка не стане нижче деякого наперед заданого значення.

Переваги методу:

Нейронні мережі здатні обробляти інформацію, що міститься у великих об’ємах даних, та будувати неймовірно складні моделі. При правильних та точних налаштуваннях, нейронні мережі досить часто перевершують інші методи машинного навчання.

Недоліки методу:

Нейронні мережі потребують багато часу на навчання. Вони потребують ретельної попередньої обробки вхідних даних так як і метод опорних векторів. Також, налаштування нейронних мереж є дуже складною задачею.

2.3 Зниження розмірності даних за допомогою аналізу головних компонент

Аналіз головних компонент являє собою метод, який виконує поворот даних з тим, щоб перетворені ознаки не корелювали між собою. Часто цей поворот супроводжується вибором нових ознак в залежності від їх важливості з точки зору інтерпретації даних. Наступний приклад ілюструє результат використання PCA до синтетичного двомірного масиву даних:

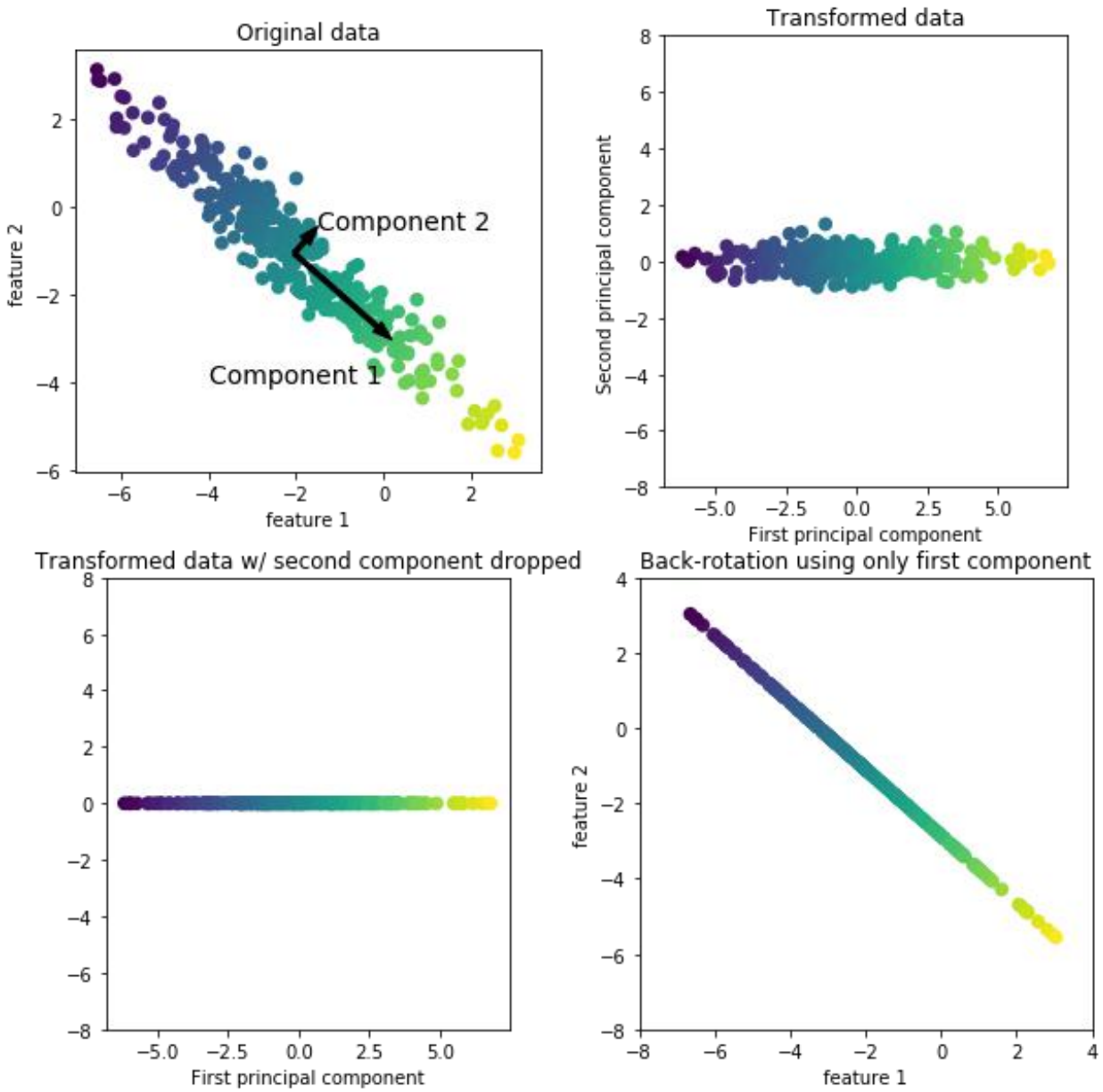

Рис. 24

Верхній правий графік зображує ті ж самі дані, але тепер вони повернуті таким чином, що перша головна компонента співпадає з віссю x, а друга – з віссю y. Перед повертанням від кожного значення віднімається середнє, таким чином, перетворені дані центровані біля нуля. В новому представленні даних, дві осі стають некорельованими. Це означає, що у новому представленні усі елементи кореляційної матриці даних, крім діагональних, будуть дорівнювати нулю.

Таким чином аналіз алгоритм PCA можна використовувати для зменшення розмірності, зберігаючи тільки декілька головних компонент.

Висновки

В цьому розділі були розглянуті найбільш широко застосовані у задачі класифікації емоцій методи машинного навчання. Було з’ясовано в яких випадках краще використовувати той чи інший алгоритм. Були виявлені основні переваги та недоліки розглянутих методів машинного навчання.

Короткий огляд методів:

Метод k найближчих сусідів – є одним з найпростіших, підходить для невеликих масивів даних, має дуже простий принцип роботи.

Метод опорних векторів – добре працює з набором даних середнього розміру та великою розмірністю, потребує попередньої обробки вхідних даних, зокрема, масштабування, чутливий до зміни параметрів.

Випадковий ліс – легко налаштовується, є дуже стійким, не потрібно масштабувати вхідні дані, погано працює з даними високої розмірності та розрідженими даними.

Нейронні мережі – дозволяють будувати дуже складні моделі для великих масивів даних. Чутливі до масштабування вхідних даних та вибору параметрів. Потребують багато часу для навчання.

РОЗДІЛ 3. КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ЕМОЦІЙ

Комп'ютерне моделювання проводилося на мові програмування Python 3.6.6 з використанням бібліотеки машинного навчання Scikit-learn.

У дослідженні були використані дві англомовні бази аудіо даних, розмічених відповідно до емоційного стану мовця. Перша база The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) містить 1440 записів емоційної мови 24 акторів, 1012 записів емоціонального співу 23 акторів. Записи розмічені по 8 емоціям: нейтральність, спокій, радість, сум, злість, страх, відраза, здивування. Кожна емоція була записана з малою та великою інтенсивністю. Довжина одного запису становить від 3-4 секунд.

Друга база Toronto emotional speech set (TESS) містить 2800 записів емоційної мови 2 акторів (26 років і 64 роки). Записи розмічені по 7 емоціям: нейтральність, радість, сум, злість, страх, відраза, здивування. Довжина одного запису становить від 3-4 секунд.

Дослідження проводилося для 7 емоцій, що були спільними для обох баз даних: нейтральність, радість, сум, злість, страх, відраза, здивування.

Всі 4874 записи були перемішані випадковим чином та розділені на навчальну частину та тестову частину з відношенням 6 до 4 відповідно.

Підготовка даних

Для обробки мовленнєвого сигналу, є необхідним використання короткочасного аналізу. Сигнал має бути розбитий на часові вікна фіксованого розміру. У роботі [3] показано, що для задачі розпізнавання емоцій є достатнім розмір фрейму довжиною 0.25 с та кроком 0.1 с.

Далі отримуються мел-частотні кепстральні коефіцієнти та перша і друга похідна від них (MFCC, Δ, 2Δ), коефіцієнти логарифму енергії спектру (LOGFBANK), хромаграма у 12 частотних смугах (Chroma), коефіцієнти

спектрального центроїда (SSC). Для кожного двомірного масиву мел-частотних кепстральних коефіцієнтів та коефіцієнти логарифму енергії спектру, знаходяться середні значення кожного коефіцієнту по всім фреймам. Таким чином матриці перетворюються на вектори коефіцієнтів. Оскільки на вхід класифікатора потрібно подати двомірний масив даних розміром M*N, де M – кількість записів, а N – довжина вектору ознак, за допомогою конкатенації векторів ознак, отримується супервектор.

Код програми обробки сигналів та формування бази даних приведений у додатку А.

В даній роботі було проведено декілька експериментів, у яких використовувалися різні комбінації ознак на різних алгоритмах класифікації.

Були побудовані 8 векторів ознак:

1. Мел-частотні кепстральні коефіцієнти та перші дві похідні (MFCC, Δ, 2Δ), довжина вектору ознак – 156 значень;
2. Логарифмічні коефіцієнти енергії спектру (LOGFBANK) , довжина вектору ознак – 52 значень;
3. Коефіцієнти спектрального центроїда (SSC) , довжина вектору ознак – 52 значень;
4. LOGFBANK + SSC, довжина вектору ознак – 104 значень;
5. MFCC, Δ, 2Δ + LOGFBANK + SSC, довжина вектору ознак – 260 значень;

6. LOGFBANK + SSC + хромаграма (Chroma) , довжина вектору ознак – 128 значень;
7. ALL = MFCC, Δ, 2Δ + LOGFBANK + SSC + Chroma, довжина вектору ознак – 498 значень;
8. ALL_PCA = MFCC, Δ, 2Δ + LOGFBANK + SSC + Chroma;
   За допомогою методу головних компонент (PCA) кількість елементів у векторі ознак (ALL_PCA) зменшено до 300. Таким чином ми хочемо

знизити кореляцію даних, та можливо, підвищити точність класифікації.

У ході експерименту були протестовані чотири класифікатори:

1. Метод k найближчих сусідів (k-NN). При кожному новому навчанні, параметр кількості сусідів корегувався у діапазоні (1, 5) для знаходження найбільш якісного налаштування моделі.
2. Метод опорних векторів (SVM) також налаштовувався за двома параметрами: «C» у діапазоні (0.1, 100), та «gamma» у діапазоні (0.1, 100). Також, для підвищення ефективності, перед навчанням моделі, вхідні дані масштабувалися таким чином, щоб всі ознаки знаходились строго у діапазоні від 0 до 1.
3. Випадковий ліс (RndForest) будував від 10 до 150 дерев з максимальною глибиною від 10 до 30 рівнів.
4. Нейронна мережа (DNN) будувалася на основі багатошарового перцептрону з функцією активації ReLU, з 10 шарами по 15 нейронів. Максимальна кількість навчальних ітерацій становила 1500.

Обмеження накладені на роботу алгоритмів обумовлені дуже великим часом за який моделі вчились. На навчання нейронної мережі іноді витрачалося до 3-4 годин, при умові 100% навантаження комп’ютерного процесору. Навчання методу опорних векторів та випадкового лісу займало приблизно 15-20 хвилин. Дуже добре себе показав метод k найближчих сусідів на навчання якого витрачалося приблизно 2-3 хвилини.

Код програми розмежування наборів даних та моделювання алгоритмів наведений у додатку Б.

У таблиці 1 зображені результати комп’ютерного моделювання. Числові значення означають точність з якою алгоритм розпізнав тестові дані.

Таблиця 1

	k-NN	SVM	RndForest	DNN
MFCC, Δ, 2Δ	0.89	0.9	0.85	0.81
LOGFBANK	0.76	0.86	0.75	0.76
SSC	0.84	0.91	0.84	0.76
LOGFBANK SSC	0.84	0.92	0.84	0.81
MFCC, Δ, 2Δ LOGFBANK SSC	0.86	0.92	0.87	0.92
LOGFBANK SSC Chroma	0.84	0.92	0.85	0.78
ALL	0.86	0.93	0.87	0.83
ALL_PCA	0.92	0.91	0.83	0.81

Висновки

З таблиці 1 видно що найкраще себе проявив метод опорних векторів. Найгірші результати виявилися у нейронної мережі, що є досить неочікуваним. Але такий результат можна пояснити тим, що у нейронних мереж дуже багато чутливих налаштувань, які необхідно встановлювати вручну, на що просто не вистачило часу. Також, низька ефективність нейронної мережі можна пояснити порівняно невеликою кількістю даних для навчання.

Алгоритм k найближчих сусідів, незважаючи на свою простоту, також показав непоганий результат. Цікавим є той факт, що застосування методу головних компонент призвело до падіння точності класифікації на всіх алгоритмах крім k найближчих сусідів, точність якого навпаки, зросла.

Найбільш інформативним набором ознак виявився набір мел-частотних кепстральних коефіцієнтів та його похідних, та його комбінація з логарифмічними коефіцієнтами енергії спектру і коефіцієнтами спектрального центроїду.

Також не можна не побачити досить високу точність класифікації загалом. Точність алгоритму опорних векторів наближається до точності такого ж алгоритму описаного у роботі [6].

Для того щоб перевірити результат на практиці, було зроблено 35 аудіозаписів з короткими емоційними реченнями. По п’ять записів на кожну емоцію. Речення були ті ж самі що і в базі даних RAVDESS.

Записи надавалися вже навченим алгоритмам. Найвищу точність продемонстрував метод опорних векторів – 0.33. Найнижчу точність продемонстрував алгоритм k найближчих сусідів – 0.19. Це дуже різке падіння точності, але потрібно зауважити, що на відміну від сторонніх баз даних, в цих записах брали участь не актори, а звичайні люди. Через це емоції виражалися не настільки яскраво.

РОЗДІЛ 4. СТАРТАП-ПРОЕКТ

“Акустичний детектор емоцій”

Таблиця 2. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Комп’ютерна програма “Акустичний детектор емоцій” являє собою цифрову систему розпізнавання емоцій у мовленнєвому сигналі. Програма призначена для відстеження психофізичного стану людини та отримання інформації про її емоціональний стан.	Пристрої автоматичного відслідковування емоційного стану робітників та клієнтів кол-центрів.	Зниження кількості конфліктів робітників з клієнтами за рахунок відстежування та можливості вчасного втручання.
	Використання замість поліграфів або разом з ними у якості детектора брехні.	Спосіб безконтактного виявлення психофізичного стану людини.
	Використання у системах голосового управління пристроями.	Покращення зворотного зв’язку та реакція на емоційний стан.

Ефективних систем розпізнавання емоцій по мовленнєвому сигналу досі не існує.

Таблиця 3 Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні показники	Потенційні товари-конкуренти	Слабка сторона	Нейтральна сторона	Сильна сторона
1	Вартість	немає	Мала сфера застосування	доступність	точність

Таблиця 4. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Алгоритм класифікації емоцій у мовленнєвому сигналі	Методи машинного навчання	Технології наявні але потребують вдосконалення	Доступні

Висновок: технології необхідні для реалізації проекту існують та є в наявності але вони потребують значного вдосконалення. Оскільки використання алгоритмів машинного навчання не потребує значних витрат, то розробку проекту можна вважати доцільною.

Попередня характеристика потенційного ринку стартап-проекту

Ринок вільний оскільки досі не існує подібних проектів.

Таблиця 5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

1	Необхідність контролювати психофізичний стан людей	Лікарі, Поліція	Направленість на результат	Коректна робота

Таблиця 6. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява конкурентів	Втрата ринку	Прискорення розробки

Таблиця 7. Фактори можливостей

№	Зміст можливості	Можлива реакція компанії
1	Взяття участі у виставках	Приймає участь
2	Співпраця з органами правоохорони	Згода

Таблиця 8. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти	Потенційні конкуренті	Постачальники	Клієнти	Товари-замінники
	Конкурентів не має	Google	Немає	Ринку немає	немає
Висновки		Будь-коли			

Таблиця 9. Обгрунтування факторів конкурентноспроможності

№	Фактор конкурентноспроможності	Обгрунтування
1	Конкуренти відсутні	Конкурувати нема з ким

Таблиця 10. SWOT-аналіз стартап-проекту

Сильні сторони: відсутність конкурентів	Слабкі сторони: невеликий попит на товар
Можливості: стати лідером ринку	Загрози: можливість появи конкурентів

Таблиця 11. Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розширення команди	1	місяць

Таблиця 12. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
1	Органи правоохорони	середня	50-100	Відсутня	висока
2	Лікарі	Середня	70-100	Відсутня	висока

Таблиця 13. Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охплення ринку	Ключові конкурентноспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Зосередження на одній цільовій групі	Стратегія концентрованого розвитку	Схожі товари	Стратегія спеціалізації

Таблиця 17. Визначення базової стратегії конкурентної поведінки

№	Чи є проект першопроходцем на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента і які?	Стратегія поведінки
1	так	Забирати існуючих	Ні	Стратегія виклику лідера
2	Так	Забирати існуючих	Ні	Стратегія наслідування лідера

Таблиця 18. Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку	Ключові

		пропонує товар	переваги перед конкурентами
1	Точна класифікація	Точна класифікація	доступність

Таблиця 19. Визначення меж ціни

№	Рівень цін та товарів замінників	Рівень цін на товари аналоги	Рівень доходів цільової групи споживачів	Межі ціни
1	16$	-	500$	10-16$

Таблиця 20. Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікації клієнтів	Ключові позиції	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Недовіра до всіх	Віч на віч	Потреба у товарі	Показати іноваційність	Контроль емоцій

Висновки

На даний момент, оскільки подібних пристроїв ще нема, є чудова можливість вийти на ринок без конкурентів. Необхідно якомога швидше вийти на ринок, щоб створити свою міцну цільову аудиторію та не дати можливим конкурентам закріпитися на ринку.

ВИСНОВКИ ПО РОБОТІ

В даній роботі були досліджені алгоритми розпізнавання емоцій за мовленнєвим сигналом. Були розглянуті найпоширеніші ознаки які можуть бути вилучені з мовленнєвого сигналу, класичні алгоритми машинного навчання та нейронні мережі, а також їх переваги та недоліки. Була зачеплена тема попередньої обробки даних на прикладі масштабування та зменшення кількості вимірів простору ознак.

Розглянутий випадок використання масштабування показав підвищення ефективності розпізнавання, а використання зниження розмірності даних погіршило результати для методу опорних векторів, випадкового лісу та нейронної мережі але покращило результати методу найближчих сусідів.

Проведений експеримент показав що найефективнішим, серед розглянутих, алгоритмом класифікації емоцій є метод опорних векторів. З іншої сторони, метод k найближчих сусідів виявився найшвидшим та найпростішим алгоритмом, а його результати не дуже сильно відставали від інших методів при даних умовах. Також було показано, що ефективність алгоритмів, навчених на записах емоцій професійних акторів, сильно знижується в задачах розпізнавання емоцій звичайних людей.

Використання нейронних мереж в задачі розпізнавання емоцій потребує подальшого дослідження. Необхідно дослідити роботу нейронних мереж з більшою кількістю навчальних даних, та точнішим налаштуванням.

В подальшому, планується дослідження багатоетапних способів розпізнавання емоцій, з використанням комбінацій класичних методів та

нейронних мереж. Також, бажано розглянути більш складні види нейронних мереж для аналізу послідовних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Saikat Basu, Jaybrata Chakraborty, Arnab Bag, Md. Aftabuddin, *A Review on Emotion Recognition using Speech*, International Conference on Inventive Communication and Computational Technologies ICICCT 2017.
2. Кузнецов В. *Автоматический синтез речи*, Таллин: Валгус, 1989 – 56с.
3. E. Mower Provost*, Identifying salient sub-utterance emotion dynamics using flexible* units and estimates of affective flow, in Proceedings of IEEE ICASSP 2013.
4. Андреас Мюллер, Сара Гвидо *Введение в машинное обучение с помощью Python: руководство для специалистов по работе с данными*. - Москва 2016-2017, 393 стр.
5. S. Steidl, *Automatic Classification of Emotion-Related User States in Spontaneous Children's Speech*, Logos Verlag, Berlin, 2009.
6. Yixiong Pan, Peipei Shen, Liping Shen, *Speech Emotion Recognition Using Support Vector Machine*, International Journal of Smart Home Vol. 6, No. 2, April, 2012
7. Дж. Вандер Плас, *Python для сложных задач: наука о данных и машинное обучение*. – СПб.: Питер, 2018. – 576с.
8. Хайкин, Саймон. *Нейронные сети: полный курс, 2-е изд*.: Пер. с англ. – М.:ООО «И.Д. Вильямс», 2016.-1104 с.
9. Vladimir Chernykh, Pavel Prikhodko. *Emotion Recognition From Speech With Recurrent Neural Networks.* 2018 – 18 с.
10. Jinkyu Lee, Ivan Tashev. *High-level Feature Representation using Recurrent Neural Network for Speech Emotion Recognition.* - 4 с.

11. Антонио Джулли, Суджит Пал. *Библиотек Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью Theano и TensorFlow.* Пер. с англ. Слинкин А. А. – М.:ДМК Пресс, 2018 -294 с.
12. Стерлинг Г. Приходько П. *Глубокое обучение в задаче распознавания эмоций из речи.*

ДОДАТОК А. ФОРМУВАННЯ ВЕКТОРІВ ОЗНАК ТА НАБОРУ ДАНИХ

Обробка звукових файлів:

audio_utils.py

```
from pydub import AudioSegment
from pydub.silence import split_on_silence
import numpy as np
import librosa

class Sample:

    def __init__(self, filename):
        self.sig = AudioSegment.from_file(filename)
        self.y, self.sr = librosa.load(filename)
    def get_bit_rate(self):
        return self.sig.sample_width

    def get_sample_rate(self):
        return self.sig.frame_rate

    def get_seconds(self):
        return self.sig.duration_seconds

    def max(self):
        return self.sig.max_possible_amplitude()

    def get_data(self):
        return np.array(self.sig.get_array_of_samples())

    def get_librosa(self):
        return self.y, self.sr

    def resample(self, new_bit_rate = 2, new_frame_rate = 22050):
        self.sig = self.sig.set_sample_width(new_bit_rate)
        self.sig = self.sig.set_frame_rate(new_frame_rate)

    def remove_silence(self, min_silence_len=10, silence_thresh=-100,
keep_silence=10):
        parts = split_on_silence(self.sig, min_silence_len, silence_thresh,
keep_silence)
        res = AudioSegment.empty()
        for part in parts:
            res += part
        self.sig = res
```

Здобуття векторів ознак з файлу.

features_utils.py

```
from python_speech_features import mfcc, logfbank, ssc
import librosa
import python_speech_features
import numpy as np


def features_extraction(sig, rate, y=None, sr=None, numcep=26, nfft=4500,
**kwargs):
    """Compute mean and standard deviation of each feachures from an audio
"""
    mfcc_feat = mfcc(sig, rate, winlen=0.2, winstep=0.1, numcep=numcep,
                     nfilt=26, nfft=nfft, lowfreq=50, highfreq=8000,
winfunc=np.hamming, **kwargs)
    mfcc_mean = mfcc_feat.mean(axis=0)
    mfcc_std = mfcc_feat.std(axis=0)
    mfcc_delta = librosa.feature.delta(mfcc_feat)
    mfcc_delta2 = librosa.feature.delta(mfcc_feat, order=2)
    mfcc_delta_mean = mfcc_delta.mean(axis=0)
    mfcc_delta2_mean = mfcc_delta2.mean(axis=0)
    mfcc_delta_std = mfcc_delta.std(axis=0)
    mfcc_delta2_std = mfcc_delta2.std(axis=0)

    logfbank_feat = logfbank(sig, rate, nfft=nfft, **kwargs)
    logfbank_mean = logfbank_feat.mean(axis=0)
    logfbank_std = logfbank_feat.std(axis=0)

    ssc_feat = ssc(sig, rate, nfft=nfft, **kwargs)
    ssc_mean = ssc_feat.mean(axis=0)
    ssc_std = ssc_feat.std(axis=0)

    chroma = librosa.feature.chroma_stft(y=y, sr=sr, n_fft=nfft)
    chroma_mean = chroma.mean(axis=1)
    chroma_std = chroma.std(axis=1)

    return mfcc_mean, mfcc_std, mfcc_delta_mean, mfcc_delta_std,
mfcc_delta2_mean,mfcc_delta2_std, logfbank_mean, logfbank_std, ssc_mean,
ssc_std,chroma_mean, chroma_std,
```

Формування датасету розмічених по емоціям супервекторів ознак.

dataset_utils.py

```
import numpy as np
import os
from audio_utils import Sample

class Dataset:

    def __init__(self, directory, feature_extraction_function):
        self.directory = directory
        self.feature_extraction_function = feature_extraction_function
        self.__create_categories(directory)
        self.__create_dataset()

    def __create_categories(self, directory):
        categories_folders = [name for name in os.listdir(directory) if
```

```
os.path.isdir(directory+name)]
        self.categories = {}
        category_N = 0
        for category in categories_folders:
            self.categories[category_N] = category
            category_N += 1

    def __create_dataset(self):
        self.data_1 = []
        self.data_2 = []
        self.data_3 = []
        self.data_4 = []
        self.data_7 = []
        self.target = []

        for category_N, category in self.categories.items():
            waves = [f for f in os.listdir(self.directory + category) if
f.endswith('.wav')]
            for wav in waves:
                sample = Sample(self.directory + category + '/' + wav)
                sample.resample()
                sample.remove_silence()
                rate, sig = sample.get_sample_rate(), sample.get_data()
                y, sr = sample.get_librosa()
                # try:

                #     rate, sig = wavfile.read(self.directory + category +
'/' + wav)
                # except ValueError:
                #     print('Bad sample: ' + self.directory + category + '/'
+ wav)
                #     continue

                mfcc_mean, mfcc_std, mfcc_delta_mean, mfcc_delta_std,
mfcc_delta2_mean, \
                mfcc_delta2_std, logfbank_mean, logfbank_std, ssc_mean,
ssc_std, \
                chroma_mean, chroma_std =
self.feature_extraction_function(sig, rate, y, sr)

                '''mfcc_mean, mfcc_std, mfcc_delta_mean, mfcc_delta_std,
mfcc_delta2_mean, mfcc_delta2_std'''
                self.feachures_1 = np.concatenate((mfcc_mean, mfcc_std,
mfcc_delta_mean, mfcc_delta_std,

mfcc_delta2_mean, mfcc_delta2_std))
                '''logfbank_mean, logfbank_std'''
                self.feachures_2 = np.concatenate((logfbank_mean,
logfbank_std))
                '''ssc_mean, ssc_std'''
                self.feachures_3 = np.concatenate((ssc_mean, ssc_std))
                '''chroma_mean, chroma_std'''
                self.feachures_4 = np.concatenate((chroma_mean, chroma_std))
                '''mfcc_mean, mfcc_std, mfcc_delta_mean, mfcc_delta_std,
mfcc_delta2_mean, mfcc_delta2_std, logfbank_mean, ssc_mean'''
                self.feachures_7 = np.concatenate((mfcc_mean, mfcc_std,
mfcc_delta_mean, mfcc_delta_std,mfcc_delta2_mean, mfcc_delta2_std,
logfbank_mean, ssc_mean))


                self.data_1.append(self.feachures_1)
                self.data_2.append(self.feachures_2)
                self.data_3.append(self.feachures_3)
                self.data_4.append(self.feachures_4)
```

```
            self.data_7.append(self.feachures_7)
            self.target.append(category_N)


    self.target = np.asarray(self.target)
    self.data_1 = np.asarray(self.data_1)
    self.data_2 = np.asarray(self.data_2)
    self.data_3 = np.asarray(self.data_3)
    self.data_4 = np.asarray(self.data_4)
    self.data_7 = np.asarray(self.data_7)
```

ДОДАТОК Б. РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ЕМОЦІЙ

Наступний код програми був написаний у програмі Jupyter notebook.

```
import numpy as np
from features_utils import features_extraction
from dataset_utils import Dataset

DIR = './datasets/data/'
d = Dataset(DIR, features_extraction)
categories = d.get_categories()

"""LOGFBANK+SSC"""
data_2_3 = np.concatenate((d.data_2,d.data_3), axis=1)

"""MFCC,Δ,2Δ+LOGFBANK+SSC"""
data_1_2_3 = np.concatenate((d.data_1, data_2_3), axis=1)

"""LOGFBANK+SSC+Chroma"""
data_2_3_4 = np.concatenate((data_2_3, d.data_4), axis=1)

"""ALL"""
data = np.concatenate((d.data_1, d.data_2, d.data_3, d.data_4,
                       d.data_5, d.data_6, d.data_7), axis=1)

"""ALL_PCA"""
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(data)
data_scaled = scaler.transform(data)
pca = PCA(n_components=300)
pca.fit(data_scaled)
data_pca = pca.transform(data_scaled)

"""Разделение на обучающие и тестовые выборки"""
from sklearn.model_selection import train_test_split

d_train_1, d_test_1,
```

```
t_train_1, t_test_1 = train_test_split(d.data_1, d.target,
random_state=10)
d_train_2, d_test_2,
t_train_2, t_test_2 = train_test_split(d.data_2, d.target,
random_state=10)
d_train_3, d_test_3,
t_train_3, t_test_3 = train_test_split(d.data_3, d.target,
random_state=10)
d_train_7, d_test_7,
t_train_7, t_test_7 = train_test_split(d.data_7, d.target,
random_state=10)
d_train_2_3, d_test_2_3,
t_train_2_3, t_test_2_3 = train_test_split(data_2_3,
                                           d.target, random_state=10)
d_train_1_2_3, d_test_1_2_3,
t_train_1_2_3, t_test_1_2_3 = train_test_split(data_1_2_3,
                                               d.target,
random_state=10)
d_train_2_3_4, d_test_2_3_4,
t_train_2_3_4, t_test_2_3_4 = train_test_split(data_2_3_4,
                                                 d.target,
random_state=10)
d_train, d_test,
t_train, t_test = train_test_split(data, d.target, random_state=10)
d_train_pca, d_test_pca,
t_train_pca, t_test_pca = train_test_split(data_pca,
                                           d.target, random_state=10)


"""Метод k ближайших соседей"""
from sklearn.neighbors import KNeighborsClassifier

def kNN(X_train,X_test, Y_train, Y_test, name=''):
    parameters = {'n_neighbors': [1,2,3,5]}
    clf = GridSearchCV(KNeighborsClassifier(), parameters)
    clf.fit(X_train, Y_train)
    print("Правильность на тестовом наборе: {:.2f}"
          .format(clf.score(X_test, Y_test)))
    print("Наилучшие параметры: {}".format(clf.best_params_))
    joblib.dump(clf, 'kNN_'+name+'.pkl')


"""Метод опорных векторов"""
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.externals import joblib
import numpy as np
import matplotlib.pyplot as plt

def SVC1(X_train,X_test, Y_train, Y_test, name=''):
    pipe_svc = Pipeline([("scaler", MinMaxScaler()), ("svm", SVC())])
    pipe_svc.fit(X_train, Y_train)
    print("Правильность на тестовом наборе: {:.2f}"
          .format(pipe_svc.score(X_test, Y_test)))

    parameters = {'svm__C': [0.1, 1, 10, 100],
```

```
                'svm__gamma': [0.1, 1, 10, 100]}
    svc = GridSearchCV(pipe_svc, param_grid=parameters, cv=5)
    svc.fit(X_train, Y_train)

    print("Наилучшее значение правильности перекр проверки: {:.2f}"
          .format(svc.best_score_))
    print("Правильность на обучающем наборе: {:.2f}"
          .format(svc.score(X_train, Y_train)))
    print("Правильность на тестовом наборе: {:.2f}"
          .format(svc.score(X_test, Y_test)))
    print("Наилучшие параметры: {}".format(svc.best_params_))
    joblib.dump(svc, 'SVC_'+name+'.pkl')


"""Случайный лес"""
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.externals import joblib

def RND_FRST(X_train,X_test, Y_train, Y_test, name=''):
    parameters = {'n_estimators': [10, 50, 100, 150],
                  'random_state': [0, 5, 7, 9],
                'max_depth': [10, 20, 30]}
    forest = GridSearchCV(RandomForestClassifier(),
                          param_grid=parameters, cv=5)
    forest.fit(X_train, Y_train)

    print("Наилучшее значение правильности перекр проверки: {:.2f}"
          .format(forest.best_score_))
    print("Правильность на обучающем наборе: {:.2f}"
          .format(forest.score(X_train, Y_train)))
    print("Правильность на тестовом наборе: {:.2f}"
          .format(forest.score(X_test, Y_test)))
    print("Наилучшие параметры: {}".format(forest.best_params_))
    joblib.dump(forest, 'RND_FRST_'+name+'.pkl')


"""Многослойный персептрон"""
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV

def DNN(X_train,X_test, Y_train, Y_test, name=''):
    pipe = Pipeline([("scaler", MinMaxScaler()), ("dnn",
MLPClassifier())])
    pipe.fit(X_train, Y_train)
    print("Правильность на тестовом наборе: {:.2f}"
          .format(pipe.score(X_test, Y_test)))

    parameters = {'dnn__solver': ['adam', 'lbfgs'],
                  'dnn__max_iter': [900, 1000,1500],
                  'dnn__alpha': 10.0 ** -np.arange(1, 2),
                  'dnn__hidden_layer_sizes':np.arange(13, 15),
                  'dnn__random_state':[9,10]}
    mlp = GridSearchCV(pipe, parameters, n_jobs = 4)
    mlp.fit(X_train, Y_train)
    print("Наил значение правильности перекр проверки: {:.2f}"
```

```
            .format(mlp.best_score_))
        print("Правильность на обучающем наборе: {:.2f}"
            .format(mlp.score(X_train, Y_train)))
        print("Правильность на тестовом наборе: {:.2f}"
            .format(mlp.score(X_test, Y_test)))
        print("Наилучшие параметры: {}".format(mlp.best_params_))
        joblib.dump(mlp, 'DNN_'+name+'.pkl')


"""Моделирование работы алгоритмов"""

kNN(d_train_pca, d_test_pca, t_train_pca, t_test_pca, 'all_pca')
Правильность на тестовом наборе: 0.92
Наилучшие параметры: {'n_neighbors': 1}

SVC1(d_train_pca, d_test_pca, t_train_pca, t_test_pca, 'all_pca')
Правильность на тестовом наборе: 0.65
Наилучшее значение правильности перекр проверки: 0.90
Правильность на обучающем наборе: 1.00
Правильность на тестовом наборе: 0.91
Наилучшие параметры: {'svm__C': 10, 'svm__gamma': 0.1}

RND_FRST(d_train_pca, d_test_pca, t_train_pca, t_test_pca, 'all_pca')
Наил значение правильности перекр проверки: 0.82
Правильность на обучающем наборе: 1.00
Правильность на тестовом наборе: 0.83
Наилучшие параметры: {'max_depth': 30, 'n_estimators': 150,
'random_state': 9}

DNN(d_train_pca, d_test_pca, t_train_pca, t_test_pca, 'all_pca')
Правильность на тестовом наборе: 0.84
Наил значение правильности перекр проверки: 0.85
Правильность на обучающем наборе: 0.94
Правильность на тестовом наборе: 0.85
Наилучшие параметры: {'dnn__alpha': 0.1, 'dnn__hidden_layer_sizes': 14,
'dnn__max_iter': 900, 'dnn__random_state': 9, 'dnn__solver': 'adam'}
```