Розглянуто теоретичне обґрунтування основних методів машинного навчання, а саме методу опорних векторів, градієнтного бустингу, навчання нейронних мереж (з вчителем) та дерев прийняття рішень. Наведено означення ансамблів класифікаторів, дерев рішень, оптимальної гіперплощини. Подано алгоритми класифікації за згаданими вище методами. Виведено формули для знаходження оптимальної роздільної гіперплощини. Побудовано програмну реалізацію для практичного використання цих методів на прикладі моделювання кількості загиблих на “Титаніку”.

Theoretical foundation of basic machine learning methods, mainly, the method of support vector machine, gradient boosting, training neural networks (with teacher) and decision tree were considered. The definition of random forest, decision tree, optimal hyperplane and algorithms of classification are introduced. Formulas for finding optimal hyperplane were derived. The program implementation for practical use of these methods is constructed on the example of simulation of deaths number on “Titanic”.

[Вступ 5](#_Toc499764609)

[1. Random Forest (Випадковий ліс) 7](#_Toc499764610)

[1.1. Дерева рішень 7](#_Toc499764611)

[1.1.1. Розчеплення вершин 11](#_Toc499764612)

[1.1.2. Оптимальне розчеплення 12](#_Toc499764613)

[1.1.3. Критерій зупинки розчеплення 13](#_Toc499764614)

[1.1.4. Відсікання дерев 14](#_Toc499764615)

[1.2. Ансамблі класифікаторів 15](#_Toc499764616)

[1.3. Випадковий ліс 17](#_Toc499764617)

[2. Support Vector Machines (Метод опорних векторів) 21](#_Toc499764618)

[2.1. Оптимальна гіперплощина 21](#_Toc499764619)

[2.2. Алгоритм побудови оптимальної гіперплощини 26](#_Toc499764620)

[3. Метод градієнтного бустингу 30](#_Toc499764621)

[4. Нейронні мережі 36](#_Toc499764622)

[4.1. Біологічні нейронні мережі 36](#_Toc499764623)

[4.2. Штучний нейрон 37](#_Toc499764624)

[4.3. Активаційні функції 37](#_Toc499764625)

[4.4. Одношарові нейронні мережі 40](#_Toc499764626)

[4.5. Персептрони і зародження штучних нейронних мереж 41](#_Toc499764627)

[4.6. Багатошарові нейронні мережі 42](#_Toc499764628)

[4.7. Навчання нейронної мережі 43](#_Toc499764629)

[5. Моделювання кількості загиблих на “Титаніку” методами машинного навчання 47](#_Toc499764630)

[5.1. Аналіз історичних даних 47](#_Toc499764631)

[5.2. Програмна реалізація алгоритмів машинного навчання 58](#_Toc499764632)

[5.3. Результати 60](#_Toc499764633)

[Висновки 75](#_Toc499764634)

[Список використаних джерел і літератури 76](#_Toc499764635)

[Додаток 1. Текст програми 77](#_Toc499764636)

# **Вступ**

Стрімкий розвиток інформаційних технологій і відповідно, систем збору і зберігання даних – баз даних (databases), сховищ даних(data warehousing), хмарних репозиторіїв спричинив потребу у аналізі великих об’ємів даних, які аналітик не може обробити вручну. Отже, аналітику потрібно якимось чином представити вхідну інформацію в більш компактному вигляді, щоб прийняти рішення за прийнятний відрізок часу.

Для вирішення цієї проблеми до недавнього часу, програмістам доводилося писати складні і дуже точні інструкції, щоб комп’ютер міг перетворити вхідні дані в інформацію. Вхідні дані – це необроблені масиви даних, що отримані в результаті спостереження за деякою динамічною системою або об’єктом. Інформація – це оброблені дані, які мають інформаційну цінність для користувача. Зараз комп’ютер можна настроювати таким чином, щоб він навчався сам. Цей процес називається машинним навчанням.

Історія машинного навчання розпочалася в 1950 роки, коли інформатикам вдалося навчити комп’ютер грати в шашки. З розвитком комп’ютерної техніки зростала і складність обчислювальної потужності, а, отже, складність закономірностей і передбачень, які здатний розпізнати комп’ютер.

Машинне навчання – широка і динамічна область досліджень, яка використовує велику кількість теоретичних і практичних методів. Є три основні причини, що підтверджують необхідність машинного навчання, коли недостатньо просто запрограмувати комп’ютер. По-перше, розробники не можуть передбачити усі ситуації в яких опиниться машина. Наприклад, робот розроблений для проходження лабіринтів мусить дізнаватися план кожного нового лабіринту, який йому трапляється. По-друге, розробники не можуть передбачити усі зміни з часом: програма для прогнозування цін на фондовій біржі мусить навчатися для того, щоб могти адаптуватися, коли умови на ринку змінюються. По-третє, іноді люди не мають жодного уявлення про те, як запрограмувати розв’язок. Наприклад, більшість людей добре розпізнають обличчя своїх знайомих, але навіть найкращі програмісти не можуть створити програму для виконання цієї задачі, не використовуючи алгоритми навчання.

Основною метою даної роботи є теоретичний опис та практичне застосування найбільш поширених методів машинного навчання для подальшого їх застосування у практичних задачах з використанням програмування на мові R.

В першому розділі обґрунтований метод випадкових лісів. Розглянутий процес еволюції дерев рішень і ансамблів класифікаторів, що використовуються для побудови випадкового лісу. Наведена інформація про точність методу, його переваги та недоліки.

В другому розділі представлений метод класифікації, що ґрунтується на побудові оптимальних гіперплощин із застосуванням методу опорних векторів.

В третьому розділі розглянутий метод градієнтного бустингу, який будує множину не глибоких дерев рішень, які розглядаються по порядку, причому при розгляді кожного наступного дерева враховується антиградієнт втрат, тобто різниця між фактичними і прогнозованими результатами.

В четвертому розділі розібрано метод навчання нейронної мережі із вчителем та зворотнім врахуванням похибки, який дозволяє навчити мережу на підготовлених даних, які містять правильні вихідні дані.

В п’ятому розділі проаналізовані вхідні дані, обґрунтована важливість їх використання та розглянуті різні методи заповнення пропущених значень в вибірці. Також проаналізовані результати, які отримані за допомогою програмної реалізації методів. Розглянуто особливості, обмеження, переваги та недоліки методів.

# Random Forest (Випадковий ліс)

Методи машинного навчання можна розділити на три категорії:

1. Контрольоване навчання, коли кожен елемент даних позначається або прив’язується до певної категорії визначеного масиву даних (навчальної вибірки). До методів контрольованого навчання відносять: дерева прийняття рішень, наївний байєсівський класифікатор, метод найменших квадратів, логістичну регресію, метод опорних векторів, метод ансамблів і т.д.
2. Неконтрольоване навчання не присвоює категорії елементам даних. Замість цього, мета алгоритму неконтрольованого навчання – певне упорядкування даних або опис їх структури. Для неконтрольованого навчання використовують алгоритми кластеризації, метод головних компонент, сингулярний розклад.
3. Навчання з підкріпленням. Цей алгоритм вибирає дію у відповідь на отримані дані. Це найбільш поширений підхід в робототехніці.

Алгоритм машинного навчання, що використовує дерева прийняття рішень дуже поширений і універсальний для більшості прикладних задач. Різні комбінації дерев рішень можуть дати більш точне і якісне рішення, ніж інші алгоритми навчання.

В 2001 році Лео Брейман і Адель Катлер запропонували алгоритм Random Forest, що полягає у використанні дерев рішень.

Основним елементом випадкового лісу є дерево рішень, тому опис алгоритму варто розпочати з дерев рішень. Оскільки, від того як побудоване дерево залежить результат виконання алгоритму

## 1.1. Дерева рішень

Основна ідея дерев рішень полягає в розбитті множини можливих значень вектора ознак (незалежних змінних) на множини, що не перетинаються і підбору простої моделі для кожної такої множини. Для розуміння алгоритму дерев рішень варто знати означення дерева із теорії графів і означення навчальної вибірки.  
 **Означення 1.1.** Граф складається з скінченної, не порожньої множини , елементи якої називаються вершинами і множини пар вершин , що називаються ребрами.

**Означення 1.2.** Шляхом в графі називається послідовність . Якщо , то такий шлях називається циклом.

**Означення 1.3.** Якщо пара вершин що утворює ребро є впорядкованою, то таке ребро називають орієнтованим. Якщо всі ребра графа орієнтовані, то такий граф називається орієнтованим.

Дерево є зв’язаним графом без циклів. Кореневе дерево – це дерево в якому є одна вершина з якої виходять усі інші і називається ця вершина - коренем. Ми розглядатиме тільки орієнтовані кореневі дерева (рис. 1.1), в яких ребра напрямлені у напрямі від кореня. Такі дерева задовольняють наступним умовам:

* існує тільки одна вершина, яка називається коренем, в яку не веде жодна інша дуга;
* в кожну вершину (за винятком кореня) веде тільки одна дуга;
* існує єдиний шлях від кореня до будь-якої вершини.

Якщо – деяка дуга, то вершина називається батьківською для , а вершина – нащадком вершини .

**Означення 1.4.** Вершина, яка немає нащадків називається термінальною вершиною або листом.

**Означення 1.5.** Дерево називається бінарним, якщо кожна його вершина (за виключенням термінальних вершин) має рівно два нащадки.

Означення навчальної вибірки є ключовим в розпізнаванні образів. Під навчальною вибіркою розуміємо незалежну вибірку з деякого (невідомого) розподілу . Тут , – вектори ознак (прецедентів), координати яких представляють значення ознак (незалежних змінних), що виміряні в деякому експерименті. Множина всіх можливих векторів ознак називається простором образів. Ознаки можуть бути виміряні в різних шкалах – числовій, порядковій і номінальній. Відповідні значення представляють собою значення залежної змінної. Якщо може набувати тільки скінченне число значень, тобто , то ми маємо задачу класифікації. В цьому випадку називають міткою класу і це значення визначає приналежність об’єкта до одного із класів, а сама ознака називається класовою; якщо ж виміряні в числовій шкалі, то ми маємо задачу регресії; в цьому випадку ознака називається відповіддю.

**Означення 1.6.** Деревом рішень називається дерево, в якому з кожною вершиною пов’язані:

1. Деяка підмножина (з кореневою вершиною пов’язаний весь простір образів );
2. Підвибірка навчальної вибірки , така що (таким чином з кореневою вершиною зв’язана вся вибірка );
3. Деяка функція (тут – кількість нащадків вершини ), що визначає розбиття множини на підмножин, що не перетинаються. З термінальними вершинами не пов'язана ніяка функція.

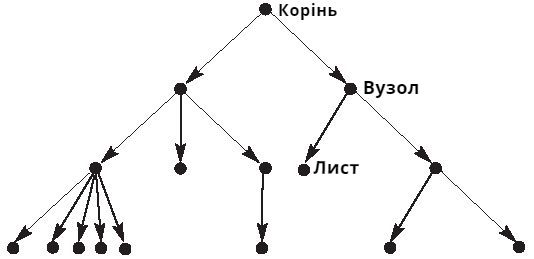


Рис. 1.1 – Орієнтоване кореневе дерево

Позначимо вершину, що є -им потомком вершини . Множина і правило визначають множину наступним чином:

Мета побудови дерева рішень полягає або в класифікації векторів з розподілу , або в оцінці умовного математичного сподівання при даному значенні . Процес прийняття рішення починається з кореневої вершини і полягає в послідовному застосуванні правил для кожної вершини дерева. Результатом цього процесу є визначення термінальної вершини такої, що .

Існує декілька причин, що зумовлюють широке застосування дерев рішень в різних областях, незважаючи на їх невисоку точність класифікації. Основними з них є:

1. Навчальна вибірка для побудови дерев рішень може містити ознаки, що виміряні як в числовій, так і в номінальній шкалі, що недопустимо для багатьох інших класифікаторів.
2. Ієрархічна структура дерев рішень зводить прийняття рішення (класифікацію) до послідовності більш простих і інтуїтивно зрозумілих рішень.
3. Дерева рішень є зручною моделлю представлення знань в експертних системах.
4. Алгоритми побудови дерев рішень незалежні від масштабування значень ознак.

Хоча алгоритм рішень як інструмент класифікації відомий достатньо давно, проте його широке застосування почалося з розробки алгоритму CART (Classification And Regression Trees). Алгоритм CART є базовим алгоритмом для якого може бути побудована множина конкретних алгоритмів, що приводять до побудови різних дерев рішень. За алгоритмом CART навчальна вибірка розбивається на дві більш однорідні підвибірки за допомогою однієї з ознак. Для реалізації цієї ідеї необхідно визначити поняття міри однорідності.

Зазвичай замість міри однорідності розглядається протилежна за значенням міра забрудненості (impurity). Нехай – деяка вершина дерева рішень, – підвибірка, що пов'язана з цією вершиною і – забрудненість вершини. Логічною вимагати, щоб забрудненість була рівна 0, якщо містить преценденти тільки одного класу і була би максимальною в тому випадку, якщо містить однакове число прецендентів кожного класу. Міра забрудненості вершини, що грунтується на означенні ентропії (entropy impurity) є найбільш поширеною:

де – частина об’єктів класу в підвибірці . Вважаємо, що

Дослідження показали, що вибір міри забруднення не дає суттєвого впливу на точність класифікації, більш важливим є вибір критеріїв зупинки дерева рішення.

### 1.1.1. Розчеплення вершин

Правило розбиття множини , тобто кожної вершини дерева рішень називається розчепленням (split). Кількість підмножин, на які розбивається , в принципі може бути різною для різних вершин, проте більшість алгоритмів, грунтуються на побудові бінарних дерев, в яких розчеплення відбувається на 2 підмножини. Це пов’язано з тим, що для будь-якого дерева рішень можна побудувати еквівалентне йому бінарне дерево. Крім цього, бінарні дерева значно простіше реалізувати програмно. Бінарне розчеплення вершини можна розглядати як функцію , де у випадку вектор відноситься до першого (лівого) нащадка, а у випадку – до другого (правого). Зазвичай ця функція має простий вигляд і залежить від значень тільки однієї ознаки. Якщо деяка ознака виміряна в числовій шкалі, то розчеплення полягає у виборі , що мінімізує міру забрудненості і визначення функції

де – довільна непуста множина , .

Розчеплення можна розглядати як лінійну комбінацію значень ознак:

така функція визначає, так звані, гострокутові дерева рішень (oblique trees). В цьому випадку мінімізація забруднень реалізується за параметрами і . Деякі алгоритми побудови дерев рішень використовують для розчеплення значення декількох ознак. Один із підходів розчеплення (рекомендований Брейманом), що використовує значення декількох ознак є метод головних компонент при якому з вхідної навчальної вибірки спочатку виділяються головні компоненти, а на другому етапі перетворення вибірки – будується дерево рішень.

### Оптимальне розчеплення

Як зазначалося раніше, розчеплення підвибірки здійснюється таким чином, щоб максимально зменшити забрудненість. Зменшення забрудненості вершини для бінарних дерев визначається як

де і частина об’єктів підвибірки , що відповідає лівому і правому нащадку ( і ). Найкращим розчепленням вершини є розбиття, яке максимізує величину .

В загальному випадку зменшення забрудненості визначається як

де – кількість нащадків вершини , – частина об’єктів підвибірки , що відповідає вершині і . Безпосереднє використання цієї формули призводить до того, що частіше для розчеплення будуть вибиратися ознаки, що мають більше рівнів у порівнянні з іншими. Тому зазвичай використовується наступна формула

і вибирається розчеплення, що максимізує величину .

### 1.1.3. Критерій зупинки розчеплення

Процес розчеплення вершин закінчується, коли стає неможливо зменшити значення забрудненості чергової вершини. Всі такі вершини оголошуються термінальними, а відповідне дерево повним. Зазвичай така ситуація характерна тим, що кожна термінальна вершина містить об’єкти тільки одного класу, а саме дерево містить велику кількість вершин. Загальновідомо, що повне дерево, як правило, має низьку точність класифікації. Це зв’язано з проблемою перенавчання (overfitting), яка полягає в тому, що статистична модель фактично описує тільки саму вибірку і вона не є придатною в якості моделі для всієї генеральної сукупності. Щоб запобігти побудові великих дерев існують критерії зупинки розчеплення.

Є декілька способів прийняття рішення про зупинку розчеплення:

1. Найбільш простий спосіб полягає в заданні мінімального числа спостережень у підвибірках, що відповідають термінальним вершинам.
2. Фіксація верхньої межі забрудненості вершини, тобто задається деяке число і розчеплення вершини не відбувається, якщо забрудненість при розчепленні не перевищує .
3. Порівняння кількості помилково класифікованих спостережень до і після розчеплення вершини, якщо зменшення кількості помилок не відбувається, то вершина не розчеплюється.

У випадку бінарних дерев використовується статистичний підхід, що побудований на статистиці

де – кількість спостережень класу , що віднесені в результаті розчеплення в ліву підвибірку, і – очікувана кількість спостережень в лівій підвибірці при випадковому розчепленні вершини . Якщо максимальне значення (по всіх можливих розчепленнях) статистики не перевищує критичного значення, що відповідає вибраному рівню з важливості, то розчеплення не відбувається і вершина оголошується термінальною.

### 1.1.4. Відсікання дерев

Зупинка розчеплення дерев відбувається без врахування ситуації, яка могла би статися у випадку продовження розчеплення. Це основний недолік підходів для запобігання побудови великих дерев. Не виключено, що подальше розчеплення нащадків деякої вершини могло би помітно підвищити точність класифікації. У зв'язку з цим, ключове значення отримав альтернативний підхід, основна ідея, якого полягає у побудові та подальшому відсіканні (pruning) повних дерев рішень. Відсікання – означає процедуру заміни в побудованому повному дереві деякої вершини і зв'язаного з нею піддерева термінальної вершини. Загальним недоліком всіх методів відсікання є те, що вони не можуть гарантувати знаходження оптимального розв'язку, а також їх висока обчислювальна важкість.

Іншим популярним алгоритмом побудови дерев рішень є алгоритм С4.5, попередником якого є алгоритм ID3. Алгоритм ID3 припускає, що всі ознаки навчальної вибірки виміряні в номінальній шкалі. Якщо навчальна вибірка містить ознаки, що виміряні в числовій шкалі, то вони попередньо дискретизуються. Дискретизація означає розбиття області значень неперервної ознаки на сукупність інтервалів, що не перетинаються, кожний з яких трактується як рівень номінальної ознаки. Кількість підвибірок на які розчеплюється вибірка, що відповідає деякій вершині, дорівнює кількості рівнів ознак вибраних для розчеплення цієї вершини. В алгоритмі С4.5 номінальні ознаки використовуються аналогічно до ID3, а неперервні ознаки аналогічно до CART.

Побудова дерев рішень за навчальною вибіркою відноситься до методів рекурсивного розбиття. Поряд із класичними деревами рішень розробляється також підхід, що заснований на теорії нечітких множин, який приводить до побудови нечітких дерев рішень (fuzzy decision trees).

## 1.2. Ансамблі класифікаторів

**Означення 1.7.** Ансамбль класифікаторів – це множина класифікаторів, рішення яких комбінуються деяким чином для отримання остаточної класифікації спостереження.

Зазвичай, синтез рішень окремих класифікаторів, що складають ансамбль, реалізовується шляхом їх зважування. Основна причина, що зумовила інтерес до даної тематики полягає в тому, що при певних умовах ансамблі класифікаторів володіють точністю, яка значно перевищує точність окремих класифікаторів і ансамблі є стійкішими до «зашумлення» навчальної вибірки. Необхідною і достатньою умовою високої точності ансамблю класифікаторів є те, що класифікатори, що його складають мають бути достатньо точними і роздільними, тобто здійснювати помилки на різних прецендентах. Існує декілька методів створення ансамблів класифікаторів:

1. Маніпулювання об’єктами навчальної вибірки
2. Маніпулювання ознаками
3. Ін’єкція випадковості в індукційний алгоритм
4. Маніпулювання мітками класів
5. Байєсівське голосування

Випадкові ліси комбінують методи тільки перших трьох груп. Перша група методів полягає або в базового індуктивному алгоритмі на різних підвибірках вхідної навчальної вибірки, або в ітеративному переважуванні спостережень. Найбільш простий спосіб формування підвибірки запропонований Брейманом. Ідея методу полягає у формуванні навчальної вибірки для кожного класифікатора ансамблю за допомогою випадкової вибірки (того ж об’єму, що і вхідна навчальна вибірка) з поверненням з вхідної навчальної вибірки і використанні методу зважування для агрегування рішення окремих класифікаторів. Метод отримав назву баггінг (bagging). Багаточисельні експерименти показали, що використання баггінгу суттєво підвищує точність класифікації у випадку нестійкості базового класифікатора, коли невеликі відхилення навчальної вибірки призводять до суттєвих змін в класифікації. Дерева рішень є нестійким класифікатором. Теоретичний аналіз баггінгу дерев рішень показав, що баггінг приводить до скорочення середньоквадратичної похибки класифікації. Подальші дослідження показали, що дане теоретичне твердження не є вірним для всіх моделей класифікаторів.

Бустинг (boosting) – метод, що полягає у зважуванні спостережень навчальної вибірки. Ідея бустингу полягає в тому, що класифікатор ансамблю будується послідовно і на кожній ітерації відбувається корекція (переважування) спостережень навчальної вибірки (на першій ітерації вага всіх спостережень є рівною). Корекція здійснюється таким чином, щоб існуючий класифікатор робив менше помилок на тих спостереженнях на яких часто робили помилки класифікатори, що побудовані на попередніх ітераціях алгоритму. Крім цього, кожному класифікатору присвоюється певна вага, яка визначається з кількості помилок при класифікації.

До методів другої групи належить метод за яким вихідна множина розбивається на декілька підмножин, що не перетинаються. З цих підмножин будується ансамбль нейронних мереж, кожна з яких включає ознаки тільки однієї підмножини.

Методи третьої групи цікаві тим, що вводять поняття випадковість безпосередньо в базовий алгоритм. Наприклад, при побудові дерев рішень на кожному кроці розчеплення обчислюється найкращих розчеплень, далі випадковим чином обирається одне з них.

## 1.3. Випадковий ліс

В 1998 році Лео Брейман запропонував метод класифікації, який називається випадковий ліс. Фактично випадковий ліс Бреймана – це ансамбль дерев рішень (рис. 1.2), кожне з яких будується на основі випадкової вибірки з вхідної навчальної вибірки (баггінг), причому для розчеплення вершин використовується частина випадково обраних ознак. Крім цього, будується повне дерево. Класифікація дерев в ансамблі здійснюється більшістю голосів.

Алгоритм побудови і використання лісу вперше був розроблений Лео Брейманом і Адель Катлер.

Алгоритм індукції випадкового лісу може бути представлений в наступному вигляді:

1. Для (тут – кількість дерев в ансамблі) виконати наступні дії:

* Сформувати випадкову
* вибірку розміру за вхідною навчальною вибіркою ;
* За випадкову вибіркою індукувати дерево рішень без відсікання з мінімальною кількістю спостережень в термінальних вершинах , рекурсивно слідуючи наступному алгоритму:

1. з вхідного набору ознак випадково обираються ознак;
2. з ознак вибирається ознака, яка забезпечує найкраще розчеплення;
3. розчепити вибірку, що відповідає обробленій вершині на дві підвибірки;
4. В результаті виконання кроку 1 отримуємо ансамбль дерев рішень ;
5. Передбачення нових спостережень здійснюється наступним чином:

* для регресії
* для класифікації

Нехай – клас, передбачений деревом рішень , тобто ; тоді – клас елементів, що найбільш часто зустрічаються в множині .

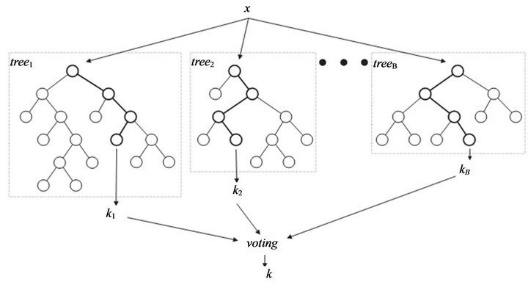


Рис. 1.2 – Ансамбль дерев рішень

Одною з переваг випадкового лісу є те, що для оцінки ймовірності помилкової класифікації немає необхідності використовувати тестову вибірку. Оцінка ймовірності помилкової класифікації випадкового лісу реалізовується методом «Out–Of–Bag» (OOB), який полягає в наступному. Відомо, що кожна випадкова вибірка не містить приблизно 37% спостережень вхідної навчальної вибірки (оскільки, вибірка з поверненням, то деякі спостереження в неї не потрапляють, а деякі потрапляють декілька раз). Класифікуємо деякий вектор . Для класифікації використовуються тільки ті дерева випадкового лісу, які будувалися за випадковими вибірками, що не містять і як зазвичай використовується метод голосування. Частота помилково класифікованих векторів навчальної вибірки і є оцінкою ймовірності помилкової класифікації випадкового лісу методом OOB. Практика застосування оцінки OOB показала, що у випадку доволі великої кількості дерев ця оцінка володіє високою точністю.

Випадкові дерева володіють цілим рядом привабливих якостей, що зумовило їх широке застосування, а саме:

1. Випадкові дерева забезпечують суттєве підвищення точності, бо дерева в ансамблі малокорельовані внаслідок випадковості в індуктивному алгоритмі.
2. Алгоритмічно складна задача відсікання повного дерева рішень знімається, бо дерева у випадковому лісі не відсікаються.
3. Простота застосування: єдиними параметрами алгоритму є кількість дерев у ансамблі і кількість ознак, що випадково відбираються для розчеплення в кожній вершині дерева.
4. Легкість організації паралельних обчислень.

Брейман дав наступне означення випадкового лісу.

**Означення 1.8**. Випадковим лісом називається класифікатор, що складається з набору дерев де – незалежні, однаково розподілені випадкові вектори і кожне дерево вносить один голос при визначенні класу .

Означення Бреймана є достатньо загальним. В залежності від того яким чином «вводиться випадковість» в алгоритм індукції дерев рішень можливі різні моделі випадкових лісів. Відповідно до цього означення, випадковий ліс є класифікатором, що складається з ансамблю дерев рішень, кожне з яких будується з використанням баггінгу.

Лео Брейман першим дослідив збіжність рішень випадкового лісу при необмеженому зростанні об’єму навчальної вибірки. Для випадкових лісів досліджується середньоквадратична збіжність. Позначимо через умовне математичне сподівання при відомому значенні вектора ознак . Середньоквадратична збіжність означає, що

У випадку класифікації ймовірність помилкової класифікації збігається до байєсівського ризику, тобто ризику оптимального байєсівського класифікатора.

# 2. Support Vector Machines (Метод опорних векторів)

Класифікації і регресії за допомогою методу опорних векторів використовують ефективні методи побудови оптимальної гіперплощини в просторі ознак великої розмірності. Оптимальність слід розуміти як мінімізацію верхніх оцінок ймовірності похибки.

## 2.1. Оптимальна гіперплощина

Для кращого розуміння розглянемо випадок повністю розподіленої вибірки, тобто випадок, коли навчання можна провести без помилок.

**Означення 2.1.** Вибірка , де і , називається роздільною за допомогою гіперплощини , якщо існує вектор одиничної довжини і число таке, що

(2.1)

У випадку, коли роздільна гіперплощина існує, визначимо

(2.2)

За означенням . Окрім цього, , якщо гіперплощина розділяє вибірку.

Введемо позначення

Тоді дорівнює половині суми відстаней від найближчих зверху і знизу точок до роздільної гіперплощини .

Припустимо, що вибірка роздільна, тобто існує таке, що умова (2.1) виконується. Максимум неперервної функції на компакті існує. Нехай максимум досягається при .

**Лема 2.1.** Нехай вказаний вище максимум досягається при . Тоді гіперплощина , де визначає вибірку і знаходиться посередині між найближчими зверху і знизу точками додатної і від’ємної частинами вибірки.

**Доведення**. Дійсно, при

При

Отже, гіперплощина знаходиться посередині між найближчими зверху і знизу точками додатної і від’ємної частинами вибірки.

Назвемо гіперплощину оптимальною. Для цієї гіперплощини сума відстаней від найближчих до неї зверху і знизу точок вибірки максимальна серед всіх розділяючих гіперплощин.

**Лема 2.2**. Оптимальна гіперплощина – єдина гіперплощина для якої сума відстаней від найближчої до неї зверху і знизу точок вибірки максимальна серед всіх розділяючих гіперплощин, які розташовані на рівних від неї відстанях.

**Доведення**. Максимум неперервної функції на компакті досягається на межі, оскільки, в протилежному випадку при було би і .

Цей максимум єдиний, оскільки, функція увігнута. Якщо би її максимум досягався в двох точках, що лежать на границі компакту, то він досягався би і у внутрішніх точці, що суперечить тільки, що доведеному.

Доведемо, що функція увігнута. Для цього потрібно перевірити, що

для всіх і , , що лежать в одиничній кулі.

Мають місце нерівності

для довільних функцій і та множини .

За означенням

З (2.6) при і маємо

Аналогічна нерівність має місце для максимумів.

Віднімаючи відповідні нерівності

отримуємо нерівність (2.5).

Розглянемо еквівалентне означення оптимальної гіперплощини. На базі цього означення буде розроблений алгоритмічно ефективний метод побудови оптимальної гіперплощини у вигляді задачі квадратичного програмування.

Знайдемо вектор і число так, щоб виконувалося

де і величина була би мінімальною при цих обмеженнях.

**Теорема 2.1**. Вектор , який задовольняє умови (2.7) і має мінімальну норму визначає оптимальну роздільну гіперплощину з ваговим вектором . При цьому

**Доведення**. Маємо

оскільки за (2.7):

Залишається довести, що неможливо. Допустимо протилежне. Розглянемо вектор . Для нього виконується нерівність

оскільки .

Доведемо, що вектор задовольняє умові (2.7) при . При :

У випадку :

Звідси, отримуємо суперечність. Оскільки, вектор має меншу норму, ніж . Отже, .

Після вибору величина максимальна

За теоремою 2.1 величина рівна відстані від найближчих точок від’ємної і додатної частини вибірки до оптимальної гіперплощини

яка розташована на рівних відстанях між гіперплощинами

що оптимально обмежують точки додатної і від’ємної частин вибірки.

Величину називають геометричним краєм помилки. Рівняння оптимальної гіперплощини можна записати також у такому вигляді

## 2.2. Алгоритм побудови оптимальної гіперплощини

Дві групи умов (2.7) запишемо у такому вигляді

при

Відповідно до попередньо отриманих результатів, для знаходження оптимальної гіперплощини ми повинні мінімізувати норму вагового вектора за обмежень (2.8).

Для розв’язування такої задачі оптимізації

за умов (2.7) (або еквівалентних їм умов (2.8)) складемо многочлен Лагранжа

де – множник Лагранжа.

Для того, щоб знайти сідлову точку лагранжіана (2.9), необхідно мінімізувати його за змінними і , а після цього, максимізувати за множниками Лагранжа за умов

Необхідна умова мінімуму многочлена Лагранжа має вигляд

З (2.10) – (2.11) випливають наступні співвідношення

Підставимо (2.12) в (2.9) і приймемо таке позначення . З врахуванням (2.13) отримуємо

Для знаходження оптимальної гіперплощини нам необхідно максимізувати функцію (2.14) за умов (2.13) і , де .

Нехай максимум досягається при Тоді розв’язок задачі пошуку оптимальної гіперплощини має вигляд

При цьому

Оптимальні розв’язки і повинні задовольняти умовам Каруша-Куна-Такера

при

Звідси випливає, що може бути тільки для тих , для яких , тобто для тих векторів, що лежать на гіперплощинах Такі вектори називаються опорними векторами (support vectors). Ваговий вектор є лінійною комбінацією опорних векторів де – число опорних векторів

Інші, не опорні вектори, можна не приймати до уваги, їх можна змінити, при цьому оптимальна гіперплощина (рис. 2.1) не зміниться.

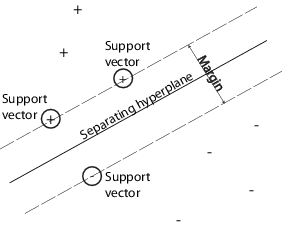


Рис. 2.1 – Опорні вектори, що розташовані на оптимальних гіперплощинах

Наведемо також деякі відношення з опорними векторами.

а також

Сумуючи (2.16) отримуємо

З (2.11) другий доданок цієї суми рівний 0. Звідси, використовуючи (2.18), отримуємо

Тому Звідси, також отримується таке співвідношення

# 3. Метод градієнтного бустингу

Нехай дана деяка множина об’єктів . Кожному об’єкту , що належить цій множині ставимо у відповідність величину , яка називається результатом і належить множині допустимих значень .

**Означення 3.1**. Впорядкована пара «об’єкт-результат» , де , називається прецедентом. Скінченний набір прецедентів утворює навчальну вибірку: , .

За допомогою навчальної вибірки необхідно побудувати таку модель (функцію) , яка отримавши на вхід прогнозує відповідь . Процес знаходження називається навчанням.

Основною вимогою до функції є висока узагальнююча здатність, тобто вона повинна показувати достатньо високі результати на нових даних, що не входять в навчальну вибірку. Таким чином оптимальний розв’язок задачі повинен задовольняти умові:

де – невід’ємна функція втрат(помилок), – множина допустимих розв’язків, , – прецеденти, що утворюють навчальну вибірку.

Один із підходів до розв’язування задач навчання полягає в комбінуванні моделей. Баггінг (bagging від Bootstrap Aggregation) і бустинг (boosting) – дві основні ідеї даного підходу. Першу з цих ідей було розглянуто в пункті 1.3. Бустинг на відміну від баггінгу навчає кожну наступну модель з використанням даних про помилки від попередніх моделей.

Основне завдання, яке необхідно розв’язати для навчання будь-якої моделі є завдання мінімізації сумарної помилки на прецедентах навчальної вибірки:

де

Одним з методів наближеного розв’язку є вибір мінімальної стратегії. Оскільки, сумарні помилки залежать не від самої функції , а лише від її значення в точках навчальної вибірки, тобто від . Звідси, задача зводиться до мінімізації функції змінних. Для розв’язання цієї задачі використовують ітераційні чисельні методи. Всі вони шукають оптимальну точку у вигляді суми послідовних наближень , де , а – початкова точка. В алгоритмі градієнтного бустингу використовується метод найшвидшого градієнтного спуску. Цей метод обчислює кожне в напрямку антиградієнта функції в поточній точці, тобто

де , а визначає довжину кроку і є розв’язком одновимірної задачі мінімізації за напрямом :

Далі відбувається перехід до наступної точки Метод градієнтного спуску дозволяє мінімізувати (3.2) при обробці навчальної вибірки для будь-якої диференційовної функції втрат. Проте у загальному випадку треба обчислювати функцію в нових точках , а вектор визначений тільки на прецедентах Дану проблему можна вирішити шляхом навчання базової моделі таким чином, щоб вона як можна точніше прогнозувала компоненти антиградієнта. При цьому можливе застосування будь-якої функції помилок та регресії в якості базової. В якості функції регресії часто вибирають дерева рішень, оскільки, вони дозволяють проводити навчання на вхідних даних без їх додаткової обробки, підтримують наявність пропущених значень, номінальних і числових змінних.

Дерево рішень розбиває множину допустимих вхідних векторів на підмножин , що не перетинаються, де – кількість листків на дереві. Кожній області присвоюється деяка константа . Отже, дерево можна описати наступним чином:

де – набір параметрів, що визначають конкретне дерево рішень: , де .

Як правило, є параметром, що задається до початку навчання, а підбирається в ході налаштування моделі в два етапи: на першому відбувається розбиття множини на , а на другому етапі обчислюються константи :

Алгоритм 1 є частковим випадком методу градієнтного бустингу, коли у якості базових моделей для навчання використовуються дерева рішень.

Алгоритм 1. Градієнтний бустинг дерев рішень для задачі регресії:

1. Взяти оптимальний константний розв’язок за початкове наближення
2. Для всіх :
3. Обчислити компоненти вектора антиградієнта
4. Побудувати регресійне дерево по вибірці
5. Знайти оптимальні константи для кожного листка дерева
6. Оновити модель
7. Остаточна модель

Обчислення довжини кроку для дерев рішень виконується незалежно для кожного листка, в той час як градієнтний спуск пропонує вибір лише одної константи для масштабування всього вектора антиградієнта. Алгоритм 1 має декілька параметрів: кількість бустинг операцій і число листків в кожному з дерев . Для навчання одного дерева рішень часто використовують великі дерева (з великою кількістю листків). Для покращення результату передбачення деякі піддерева виключаються з дерева рішень. Використання такого алгоритму в градієнтному бустингу призведе до збільшення обчислювальної складності алгоритму. Крім цього, використання великих дерев може викликати перенавчання, а відповідно збільшення похибки алгоритму. В зв’язку з цим рекомендується використовувати дерева однакового розміру, тобто , де кількість листків знаходиться в межах . Такі обмеження не є вичерпними для всіх задач, а лише рекомендацією для початкового вибору . Функція параметра полягає в зменшенні неправильних передбачень з ростом , проте вибір занадто великого значення параметра призводить до ефекту перенавчання. В алгоритмі 1 використовується ще один параметр – коефіцієнт масштабування. Використання дозволяє знизити вплив окремого дерева на всю модель. Параметри і є взаємозв’язаними: зменшення вимагає більшої кількості ітерацій алгоритму для досягнення меншої похибки. Додатково, для навчання можна використовувати не всю тестову вибірку, а лише її частину. Алгоритм, який використовує підвибірки для навчання базових моделей називається стохастичним градієнтним бустингом. Такий метод дозволяє зменшити обчислювальні затрати на навчання і при цьому може призвести до зменшення помилки.

Алгоритм 1 можна застосовувати лише для задачі регресії або бінарної класифікації. Застосування градієнтного бустингу для багатокласової задачі потребує побудови не однієї, а адитивних моделей.

Алгоритм 2. Градієнтний бустинг для задач класифікації:

1. Задаємо початкове наближення для кожної із адитивних моделей рівним нулю
2. Для всіх :
3. Для всіх :
4. Обчислити компоненти вектора антиградієнта
5. Побудувати регресійне дерево по вибірці
6. Знайти оптимальні константи для кожного листка дерева
7. Оновити модель
8. Остаточна модель

# 4. Нейронні мережі

## 4.1. Біологічні нейронні мережі

Нервова система людини є дуже складною і науковці ще й досі намагаються осягнути її природу. Відомо, що нервова система побудована з нейронів. Кожен нейрон має багато властивостей, які є спільними з іншими елементами тіла, але його унікальною здатністю є прийом, обробка і передача електрохімічних сигналів по нервових шляхах, які утворюють комунікаційну систему мозку.



Рис. 4.1 – Біологічний нейрон

На рисунку 4.1 показана структура двох типових біологічних нейронів.

**Означення 4.1.** Дендрити – це входи нейрона, що йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках з'єднання (синапсах).

Сигнали, які приймає синапс сумуються, причому одні сигнали стимулюють активізацію нейрона, а інші – зниження його активності. Коли сумарна активність (збудження) нейрона перевищує деякий поріг, нейрон переходить в активний стан, посилаючи по аксону (виходу нейрона) сигнал іншим нейронам. Описана схема є дуже спрощеною, проте більшість штучних нейронних мереж моделює лише ці прості властивості.

## 4.2. Штучний нейрон

Основними компонентами нейромережі є нейрони (елементи, вузли), які з’єднані зв’язками. Сигнали передаються через зважені зв’язки, які мають свої власні вагові коефіцієнти.

**Означення 4.2.** Штучний нейрон – це вузол штучної нейронної мережі, що імітує властивості біологічного нейрона.

На вхід штучного нейрона надходить множина сигналів, кожен з яких є вихідним сигналом іншого нейрона. Кожен вхідний параметр множиться на відповідну вагу, далі всі добутки сумуються і визначається рівень активації нейрона. На рисунку 4.2 представлена модель, що реалізує цю ідею. Множина вхідних сигналів, що надходить в штучний нейрон позначається вектором . Кожен сигнал почергово множиться на відповідний елемент вектора і надходить на суматор, який позначений . Позначимо суму через , тоді математично формула виглядатиме наступним чином

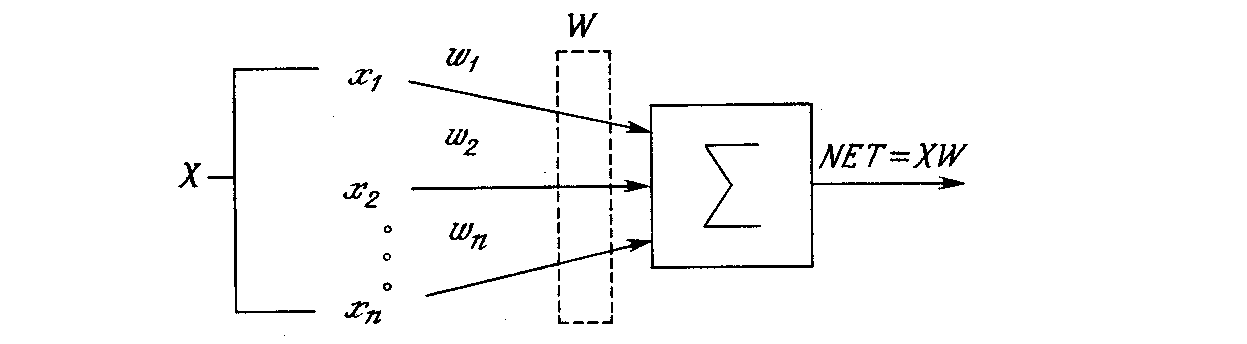


Рис. 4.2 – Штучний нейрон

## 4.3. Активаційні функції

Результуючий сигнал обчислений за допомогою формули (4.1) надходить у вигляді вхідного параметру у функцію , яка і дає вихідний нейронний сигнал . Активаційна функція може бути:

1. Бінарною пороговою функцією

де – деяка порогова величина, або ж функція, що точніше моделює нелінійну передавальну характеристику біологічного нейрона.

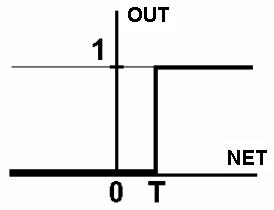


Рис. 4.3 – Порогова бінарна функція

1. Лінійною обмеженою функцією

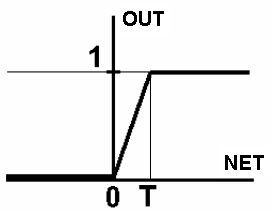


Рис. 4.4 – Лінійна обмежена функція

1. Функцією гіперболічного тангенса

де – коефіцієнт ширини сигмоїди по осі абсцис (за замовчуванням ).

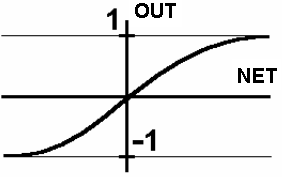


Рис. 4.5 – Функція гіперболічного тангенса

1. Сигмоїдною (S-подібною) або логістичною функцією

З (4.5) зрозуміло, що вихідне значення нейрона лежить в межах від 0 до 1. Сигмоїдна функція є найбільш популярною з ряду причин:

* здатна підсилювати слабкі сигнали більше, ніж сильні;
* монотонна і диференційовна по всій осі абсцис;
* має простий вираз для похідної

який дає можливість використовувати широкий спектр оптимізаційних алгоритмів.

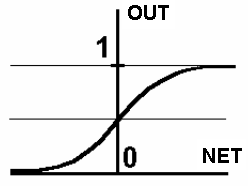


Рис. 4.6 – Сигмоїдна функція

**Означення 4.3.** Якщо блок , що приймає сигнал і видає сигнал звужує діапазон зміни величини так, що при будь-яких значеннях значення належать деякому скінченному інтервалу, то називається «стискаючою» функцією. Найбільш поширена стискаюча функція – сигмоїдна.

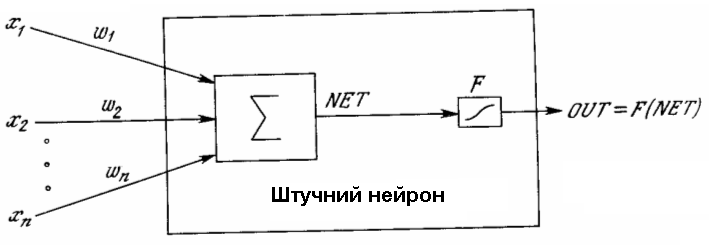


Рис. 4.7 – Штучний нейрон з активаційною функцією

## 4.4. Одношарові нейронні мережі

Для ефективної роботи нейрони об’єднують в мережі. Проста мережа зображена на рисунку 4.8 складається з групи нейронів, що утворюють шар. Кола зліва призначені лише для розподілу вхідних сигналів. Застосування для позначення кола потрібне для того, щоб відрізнити вхідні дані від обчислювальних нейронів, що позначені квадратами. Кожний елемент з множини входів сполучений з кожним штучним нейроном з врахуванням певної ваги. У загальному випадку багато з’єднань в мережі можуть бути відсутніми.



Рис. 4.8 – Одношарова НМ

Нехай всі ваги є елементами матриці . Матриця має рядків і стовпців, де – число входів, а – число нейронів. Наприклад, – це вага, що пов’язує -ий вхід з -им виходом. Отже, обчислення вихідного вектора , компонентами якого є виходи нейронів обчислені за формулою (4.1) зводяться до формули

де – вектори-рядки.

## 4.5. Персептрони і зародження штучних нейронних мереж

Перше систематичне вивчення штучних нейронних мереж було зроблене Мак-Каллоком і Піттсом в 1943 р.. В роботі вони досліджували мережеві парадигми для розпізнавання зображень, що піддаються зсувам і поворотам. Проста нейронна модель, показана на рисунку 4.9, використовувалася в більшій частині їх роботи. Елемент сумує добутки кожного входу на відповідну вагу . Якщо ця сума більше заданого порогового значення, то вихід рівний одиниці, інакше - нулю.

**Означення 4.4.** Персептрон – це система, що складається з одного шару штучних нейронів, сполучених за допомогою вагових коефіцієнтів з множиною входів (рис. 4.10).

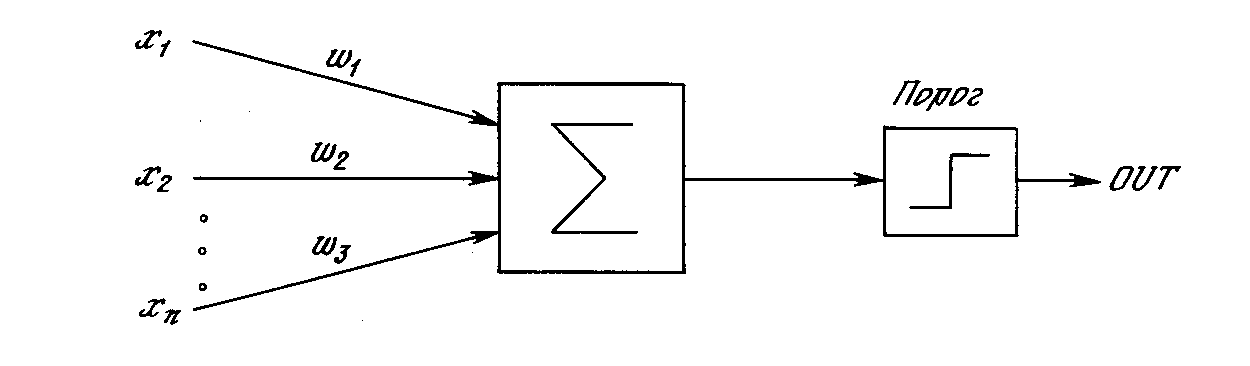


Рис. 4.9 – Нейрон персептрона

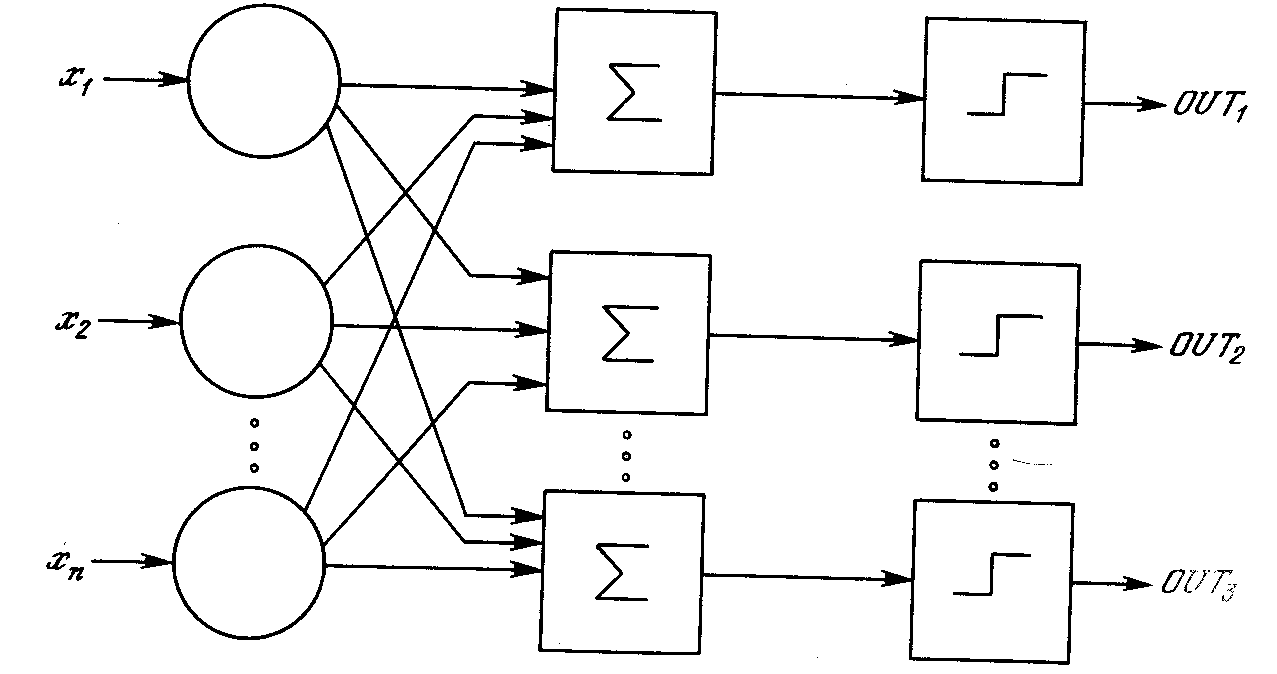


Рис 4.10 – Персептрон

Алгоритм навчання персептрона:

1. Ініціалізація вагової матриці випадковими значеннями
2. Подати на вхід сигнал і обчислити вихідне значення для порівняння з вектором (правильними результатами).
3. Якщо вихідне значення правильне – перейти на крок 4; якщо ні – обчислити різницю

та модифікувати ваги за формулою:

де – значення ваги до модифікації, – значення ваги після модифікації, - коефіцієнт швидкості навчання, – вхідний вектор, – номер ітерації

1. Виконувати цикл з кроку 2, допоки мережа не поверне очікуване вихідне значення або допоки відхилення не стане нижче деякого порогу.

Дослідник М. Мінський аналізував задачі з якими може справитися персептрон і показав, що є жорсткі обмеження на те, що можуть навчати персептрони. Наприклад таку просту функцію, як «Виключаюче АБО» персептрон відтворити не може.

## 4.6. Багатошарові нейронні мережі

Багатошарові нейронні мережі є значно кориснішими, ніж одношарові. До того ж багатошарові мережі можуть привести до збільшення обчислювальної потужності в порівнянні з одношаровими лише в тому випадку, якщо активаційна функція між шарами буде нелінійною. Обчислення вихідного значення шару полягає в обчисленні добутку вхідного вектора на першу вагову матрицю з подальшим обчисленням добутку результуючого вектора на наступну вагову матрицю . Оскільки, множення матриць асоціативне, то . Отже, двошарова лінійна мережа еквівалентна одношаровій з ваговою матрицею, яка рівна добутку двох вагових матриць. Звідси, для лінійної активаційної функції будь-яка багатошарова лінійна мережа може бути замінена еквівалентною одношаровою мережею.

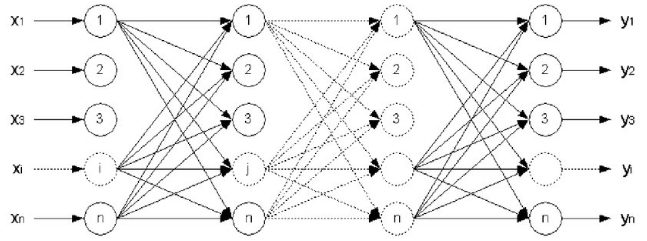


Рис.9. Багатошарова нейронна мережа

## 4.7. Навчання нейронної мережі

Суть навчання нейронної мережі – отримати бажану множину вихідних параметрів , подавши на вхід множину . Кожна вхідна або вихідна множина розглядається як вектор. Розрізняють алгоритми навчання з вчителем і без вчителя, детерміновані і стохастичні.

Навчання з вчителем. Навчання з вчителем припускає, що для кожного вхідного вектора існує цільовий вектор , що і є правильним виходом. Разом вони називаються навчальною парою. В ході навчання зчитується вхідний вектор , обчислюється вихід мережі і порівнюється з відповідним цільовим вектором , результат з (4.7) за допомогою зворотного зв'язку подається в мережу і ваги модифікуються відповідно до алгоритму, що мінімізує похибку . Зчитування векторів навчальної множини і налагодження ваг виконується до тих пір, поки сумарна помилка для всієї навчальної множини не досягне заданого мінімуму.

Навчання без вчителя. Не зважаючи на численні досягнення навчання з вчителем критикувалося за свою біологічну неправдоподібність. Важко уявити навчальний механізм в мозку, який би порівнював бажані і дійсні значення виходів, виконуючи корекцію за допомогою зворотного зв'язку. Навчання без вчителя є більш правдоподібною моделлю навчання в біологічній системі. Кохонен, Хебб та багато інших вчених доклали свої зусилля до розробки методу в якому не потрібний цільовий вектор для порівняння з ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налагоджує вагу мережі таким чином, щоб доволі близькі вхідні вектори давали однакові виходи.

Для навчання нейромережі в багатьох випадках використовують алгоритм зворотного розповсюдження помилки. Зворотне розповсюдження помилки означає, що помилка на виході враховується для корекції ваг попередніх шарів.

Навчання нейромережі з вчителем із врахуванням зворотного розповсюдження помилки відбувається за наступним алгоритмом:

1. Ініціалізуємо початкові вагові коефіцієнти матриці випадковими значеннями в діапазоні , де , .
2. Для приведення початкових даних до єдиної шкали їх потрібно нормалізувати в діапазоні . Одна з можливих функцій нормалізації
3. Застосовуємо алгоритм прямого розповсюдження похибки, який полягає у знаходженні вихідного вектора на основі вхідного за наступними формулами.

Шар 1:

де – асоціативна функція; – кількість елементів в першому шарі; – -ий елемент вхідного вектора; –вага для першого шару, яка пов’язує -ий вхід з -им нейроном.

Шар 2:

де – кількість елементів в другому шарі; – -ий елемент вектора ; –вага для другого шару, яка пов’язує -ий нейрон з -им нейроном.

Шар :

де – кількість елементів в шарі ; – -ий елемент вектора ; –вага для шару , яка пов’язує -ий нейрон з -им виходом.

1. В результаті прямого розповсюдження похибки обчислюємо помилку навчання мережі. Один з найпоширеніших методів визначення похибки, обчислення сумарної квадратичної помилки для всіх векторів

Якщо значення похибки менше заданого порогу, то процес навчання мережі завершується, якщо – ні, то переходимо до наступного кроку.

1. Зворотній хід полягає в корекції вагових коефіцієнтів через сигнал різниці

Шар :

– кількість елементів у вихідному векторі ; – номер ітерації; – -ий елемент вектора з ідеальними вихідними значеннями; – множник, що задає «швидкість руху»; – нове значення ваги для шару , яка пов’язує -ий нейрон з -им виходом.

Шар 2:

– кількість нейронів у другому шарі; – нове значення ваги для шару , яка пов’язує -ий нейрон з -им.

– кількість нейронів у першому шарі; – нове значення ваги для шару , яка пов’язує -ий вхід з -им нейроном.

Далі знову обчислюємо прямий хід методу, допоки похибка не буде перевищувати заданої межі.

# 5. Моделювання кількості загиблих на “Титаніку” методами машинного навчання

Розглянемо реальну задачу із використанням методів машинного навчання. Всім відомо, що великий пароплав “Титанік” затонув, це призвело до загибелі величезної кількості осіб. Для зменшення потенційних небезпек спробуємо розробити методи для передбачення кількості загиблих на кораблі.

## 5.1. Аналіз історичних даних

Спершу, проведемо аналіз статистичних даних для усвідомлення важливості використання певних змінних та їх впливу на передбачення. Всього на кораблі було 2223 особи, з них загинуло – 1517. Відомо, що капітан Титаніка, побачивши, що місць на всіх у шлюпках не вистачить наказав рятувати тільки дітей і жінок. Отже, розглянемо діаграму, яка показує залежності між статтю і числом врятованих.

З рисунку 5.1 видно, що:

* чоловіків на кораблі було більше, ніж жінок (чоловіків – 65%, жінок – 35%);
* більше людей померло, а ніж вижило (померло – 62%, вижило – 38%);
* судячи з кількості чоловіків, що вижили, наказ капітана рятувати тільки жінок і дітей виконувався не завжди.

Коротко підсумувати результати діаграми можна у вигляді таблиці.

|  |  |  |
| --- | --- | --- |
|  | Чоловіки | Жінки |
| Померли | 0.81 | 0.24 |
| Вижили | 0.19 | 0.74 |

Таблиця 5.1 – Статистичні дані, що показують кількість тих, хто вижив/помер за статтю

Ще одним важливим фактором є вік пасажира. Можна припустити, що пасажири, які є старшими рухаються повільніше і мають менше шансів на те, щоб врятуватися. Проте рисунок 5.2 показує, що розподіл за віком дає здебільшого однаковий шанс вижити, за винятком декількох випадків.

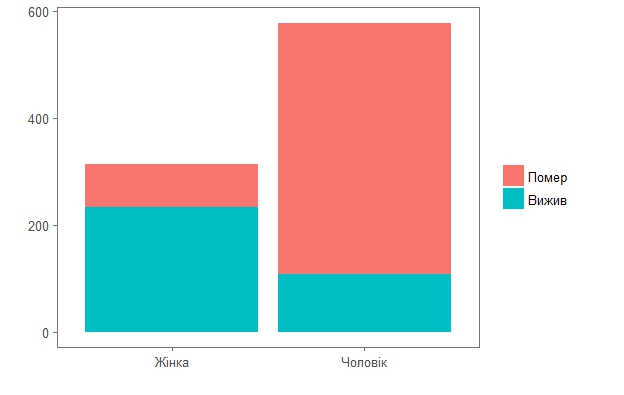


Рис. 5.1 – Діаграма, яка ілюструє залежність між статтю і числом врятованих

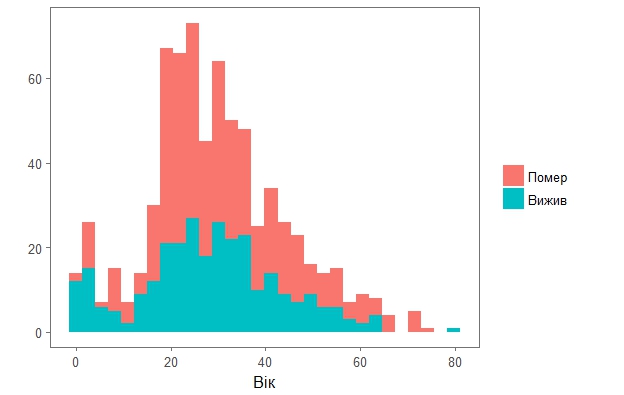


Рис. 5.2 – Діаграма, яка ілюструє залежність між віком та шансом вижити/померти

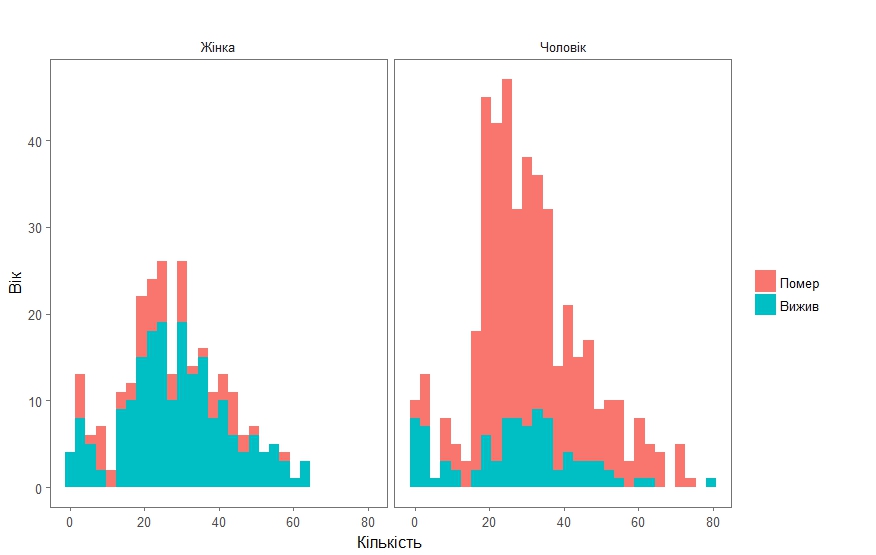


Рис. 5.3 – Діаграма, що ілюструє залежність між віком, статтю і кількістю тих, хто вижив/помер



Рис. 5.4 – Діаграма, що ілюструє залежність між віком, класом каюти, статтю і шансом вижити

Врахуємо до розподілу за віком фактор, який було розглянуто вище – стать. Як видно з рисунку 5.3 шанс вижити у жінок є високим і з віком тільки зростає, а у чоловіків шанс вижити є набагато нижчим і зростає зі зменшенням віку, тобто хлопчики до 16 років мають більші шанси вижити, ніж старші. Враховуючи розподіл пасажирів за класами можна більш детально та точно зрозуміти визначити шанс вижити.

З рисунку 5.4 можна отримати ряд важливих висновків:

* майже всі жінки, які жили в каютах першого і другого класу вижили;
* найбільший відсоток чоловіків вижив у першому класі;
* у третьому класі для жінок і в третьому та другому класі для чоловіків шанс вижити з віком помітно зменшується;
* у третьому класі для жінок і в третьому та другому класі для чоловіків шан вижити у дітей до п’яти років є найвищим.

В кожному класі є різні цінові категорії білетів. Наприклад, для першого класу ціна білетів змінюється від 10 до 512 доларів (рис. 5.5), для другого – від 5 до 75 доларів (рис. 5.6), для третього – від 5 до 70 доларів (рис. 5.7). Отже, траплялася така ситуація, що деякі люди, які перебували у другому та третьому класі платили за білет майже стільки, скільки ті, хто перебував у першому класі. Можна зробити припущення, що люди, які заплатили за квиток у своєму класі більше інших будуть довше залишатися на кораблі. В першому (для чоловіків) і третьому класі на рисунках 5.5 і 5.7, відповідно, ця закономірність є доволі помітною. В другому класі із-за того, що дуже мало чоловіків вижило і жінок померло відслідкувати якісь суттєві зміни за розподілом ціною неможливо.

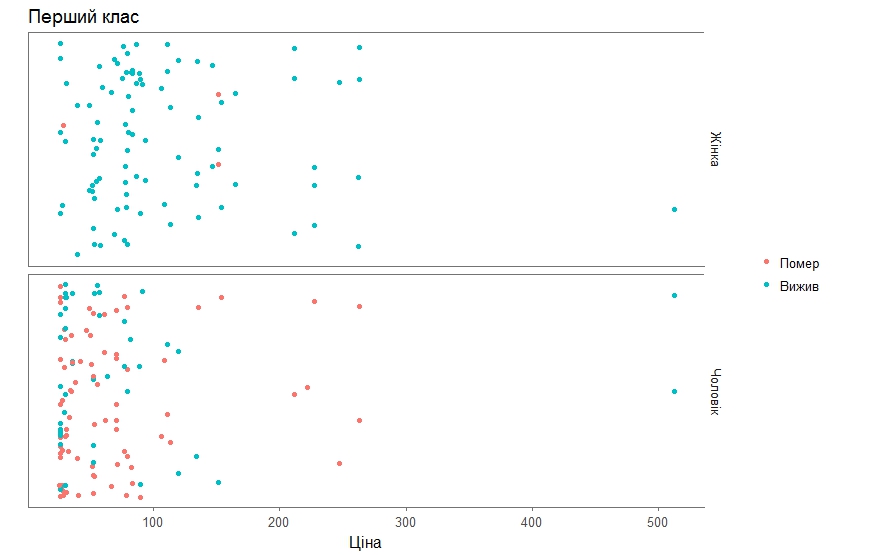


Рис. 5.5 – Діаграма з особами першого класу, яка ілюструє залежність між статтю, ціною за квиток і шансом вижити

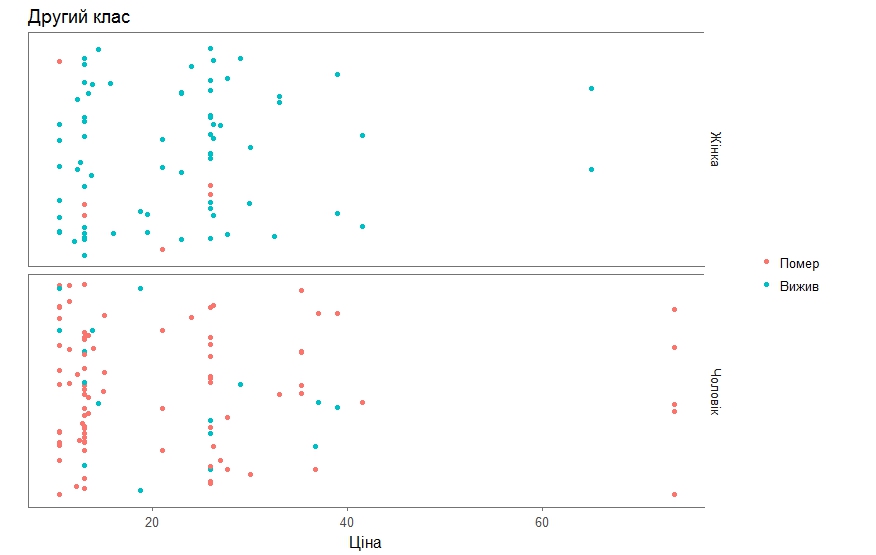


Рис. 5.6 – Діаграма з особами другого класу, яка ілюструє залежність між статтю, ціною за квиток і шансом вижити

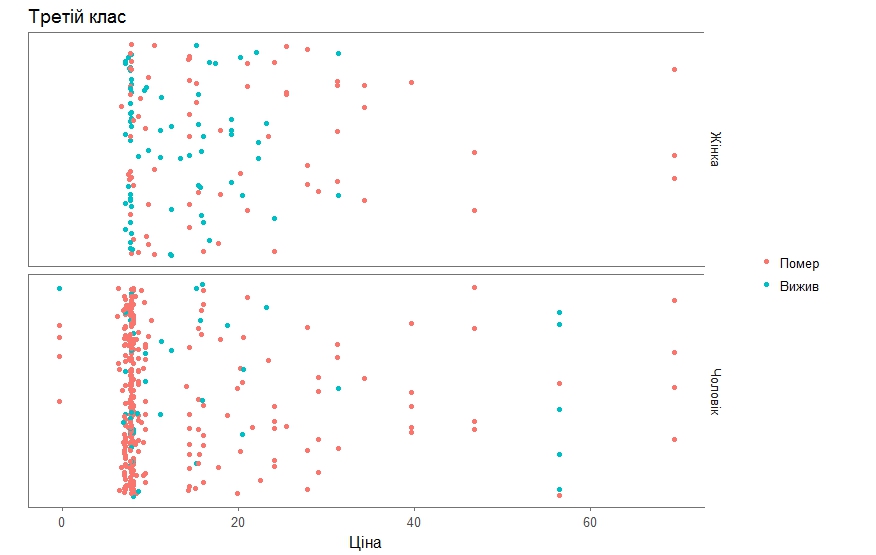


Рис. 5.7 – Діаграма з особами третього класу, яка ілюструє залежність між статтю, ціною за квиток і шансом вижити

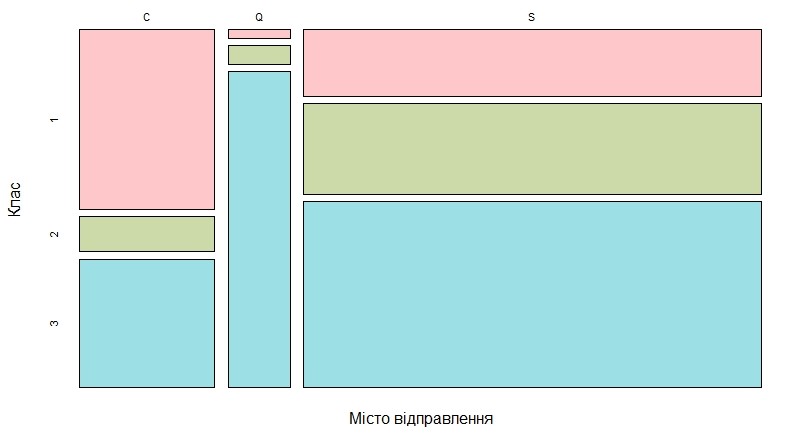


Рис. 5.8 – Розподіл за класами і містом відправлення

Відомо, що посадка пасажирів відбувалася в трьох містах: Шербург, Квінстаун, Саутгемптон. На рисунку 5.9 дані виглядають доволі добре розділеними, бо відповідають загальній статистиці смертності, але якщо подивитися на рисунок 5.8, то видно, що кількість людей, які сідали в кожному з портів і їх клас є дуже різними, тому для кращого результату треба використовувати більше факторів, ніж на рисунку 5.9.

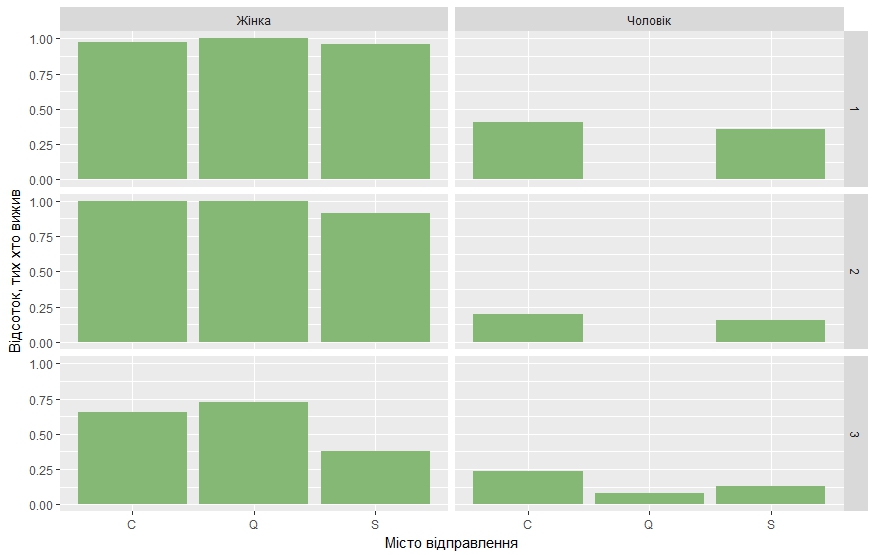


Рис. 5.9 – Діаграма, що ілюструє залежність між класом, статтю, місцем відправлення і шансом вижити

У даній вибірці ще є такі параметри як sibsp – рідні брати, сестри, законні чоловік чи жінка на борту Титаніка та parch – діти та батьки на борту Титаніка. Обидва параметри є кількісними. Як видно з рисунку 5.10 невелика кількість родичів на борту Титаніка для жінки суттєво збільшує її шанси на виживання. Проте в перерізі за класами з рисунку 5.11 бачимо, що для жінки в третьому класі, наявність родини зменшує шанси на те, щоб вижити, а для чоловіка в другому класі збільшує.

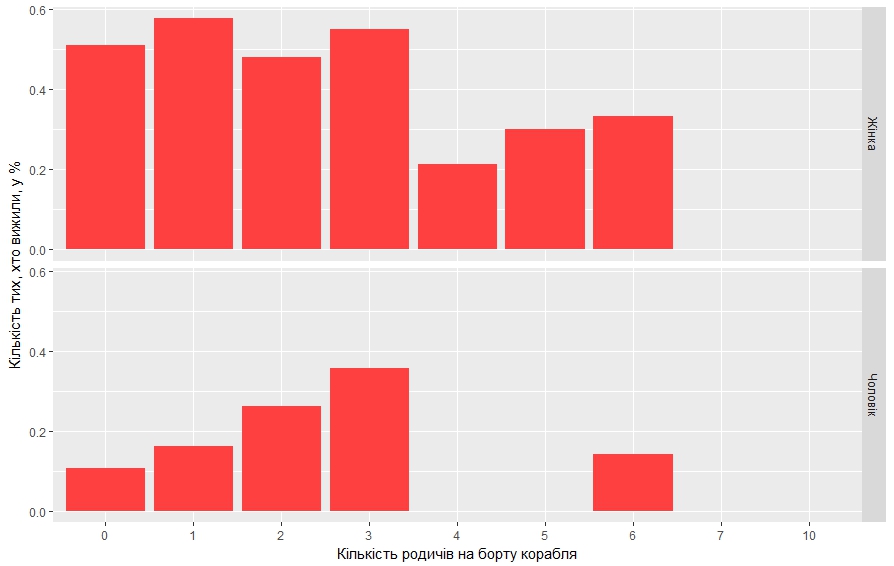


Рис. 5.10 – Розподіл тих, хто вижили за кількістю родичів на борту і статтю

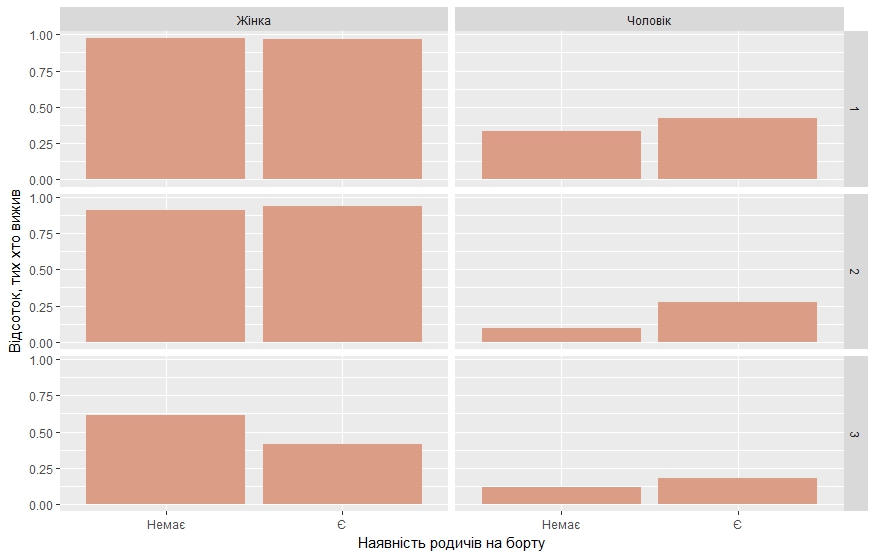


Рис. 5.11 – Розподіл тих, хто вижили за наявністю родичів на борту, класом і статтю

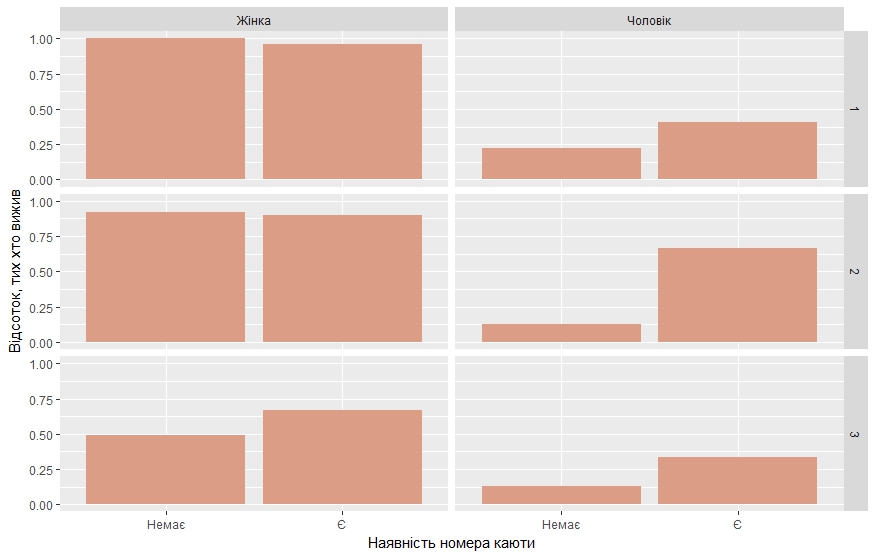


Рис. 5.12 – Розподіл, тих хто вижив за наявністю номера каюти, статтю і класом

Номер каюти вказаний у невеликого числа пасажирів. В номері є вказана палуба і на якому борту була каюта. Місцезнаходження каюти могло би бути корисним фактором, оскільки відомо в якому місці було зіткнення з айсбергом, але 20% наявних даних про каюту не вплинуть суттєво на точність моделі. Набагато цікавішою є інформація про наявність цього номера. Номери кают мешканців першого класу стали відомі зі списку, який був знайдений в одязі стюарта, що загинув, більше ніякої інформації не збереглося. Відповідно, можна зробити висновок, що, якщо відомий номер каюти пасажира другого або третього класу, то він вижив. З рисунку 5.12 видно, що припущення є справді інформативним, особливо для чоловіків.

Оскільки, є доволі велика кількість пропущених значень в колонці вік, нам треба якимось чином передбачити їх, тому введемо додаткову змінну Title. Кожне ім’я складається з прізвища, звання і ім’я. Якщо відділити звання, то ми отримаємо таку вибірку значень: Mr (містер – чоловік), Mrs (місіс – заміжня жінка або вдова), Miss (незаміжня жінка), Master (дитина чоловічої статі), Don (титул в деяких країнах Європи, визначає приналежність до знаті), Rev (звання, яке вживається перед іменем священнослужителя або представника влади), Dr (лікар або ж людина, яка має докторську степінь), Mme (те саме, що і Mrs, тільки французькою), Ms (заміжня жінка), Major (військовослужбовець), Lady (титул в Англії, який відносить жінку до вищого класу), Sir (титул, який надається чоловікові з вищого класу суспільства), Mlle (французький еквівалент Miss), Col (визначає високе військове звання), Capt (капітан корабля), Countess (графиня), Jonkheer (титул, який використовувався в Голландії та Бельгії для звертання до сина знатної особи).

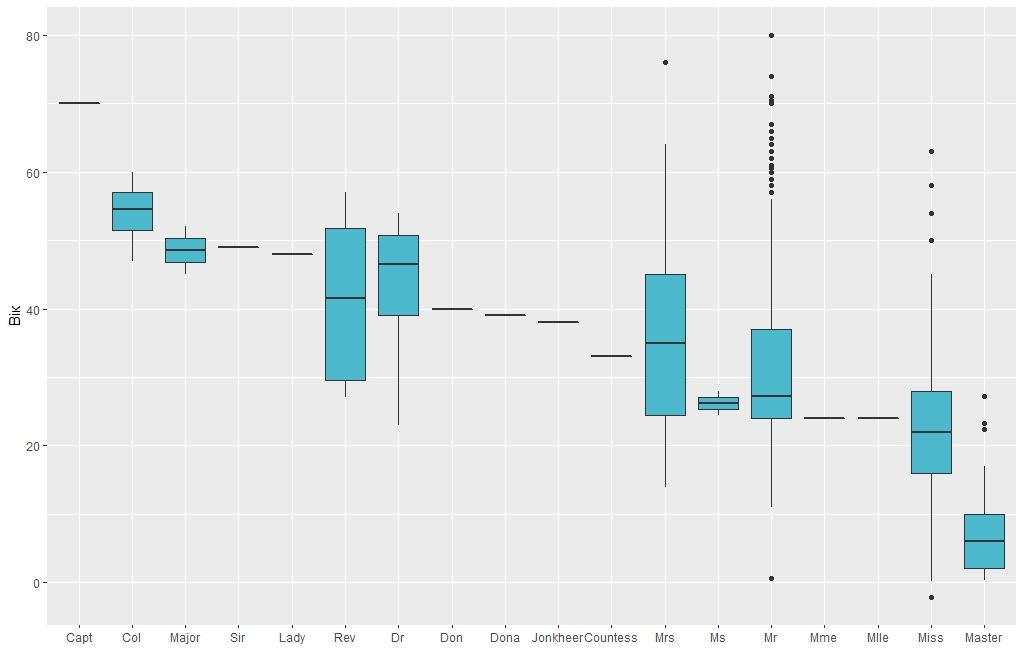


Рис. 5.13 – Розподіл за титул і віком особи

З рисунку 5.13 видно, що багато звань є лише в одному екземплярі і багато з них є близькими по віку, тому їх можна об’єднати в одну категорію. Тому звання Capt, Col, Dr, Major, Rev об’єднаємо в категорію – Officer; Dona, Don, Lady, Countess, Sir, Jonkheer – Aristocratic; Mlle, Ms – Miss; Mme – Mrs. На рисунку 5.14 зображено результат розподілу за віком після об’єднання. Очевидно, що така операція є корисною для реалізації алгоритмів передбачення, завелика кількість параметрів, які містять так мало значень може призвести до перенавчання.

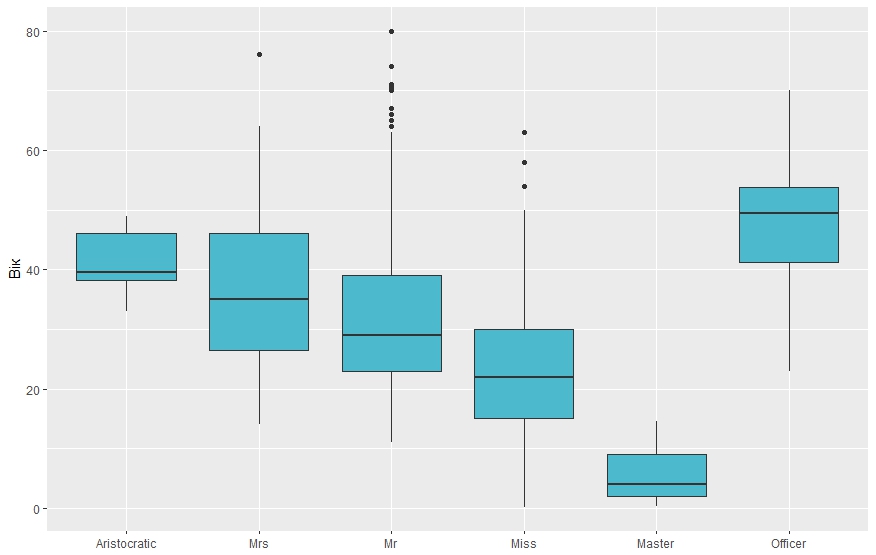


Рис. 5.14 – Розподіл за титулом і віком особи після об’єднання титулів

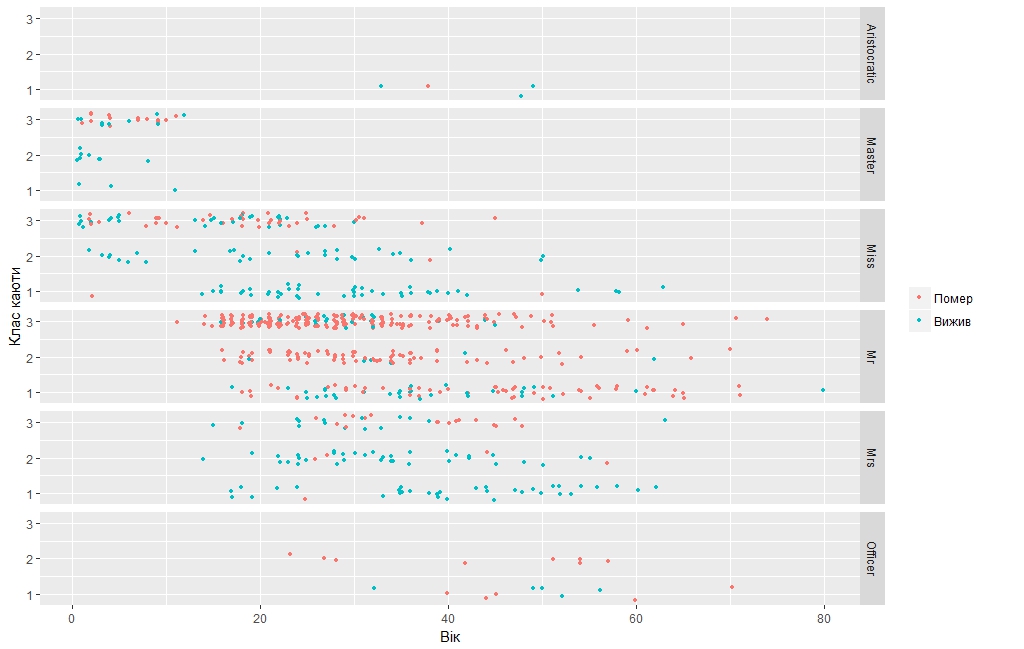


Рис. 5.15 – Розподіл за титулом віком і класом каюти

На рисунку 5.15 також можна помітити, що шанси вижити у людей зі званням офіцер є меншими, а у людей зі званням аристократи – більшими.

Як згадувалося вище, у загальній вибірці є пропущені значення, які унеможливлюють подальші застосування алгоритмів передбачення. Оскільки, всі алгоритми є чутливими до відсутності окремих даних. В таблиці 5.2 показано, що тільки в чотири фактори мають пропуски в значеннях.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Embarked | Age | Fare | Cabin |
| Відсутні значення | 2 | 263 | 21 | 1014 |

Таблиця 5.2 – Кількість пропущених значень

Фактор Cabin було розглянуто вище. Оскільки, ми будемо використовувати тільки інформацію чи є номер кабіни заповнювати пропущені значення немає сенсу. В стовпці Embarked є лише 2 пропущені значення, з огляду на рисунок 5.9, не зменшуючи загальності можна припустити, що обидва пасажири сідали на корабель в порту Саутгемптона. Значення Fare і Age не можна заповнювати таким простим усередненням як Embarked, оскільки є небезпека втратити важливі дані.

## 5.2. Програмна реалізація алгоритмів машинного навчання

Для передбачення пропущених значень ціни і віку використовуватимуться три різні методи:

1. Алгоритм Random Forest, який описаний в пункті 1

#Вибираю параметри для передбачення ціни квитка (клас, стать, к-сть родичів) та треную модель

Fare\_forest <- randomForest(Fare ~ Pclass + Sex + SibSp + Parch, data = train\_random\_fare, ntree = 5000)

#Застосування алгоритму передбачення для пропущених значень

full$Fare[is.na(full$Fare)] <- predict(Fare\_forest, full)[is.na(full$Fare)]

#Вибираю параметри для передбачення віку пасажира (клас, стать, к-сть родичів, ціну квитка, звання) та треную модель

Age\_forest <- randomForest(Age ~ Pclass + Sex +SibSp + Parch + Fare + Title, data = train\_random\_age, ntree = 5000)

#Застосування алгоритму передбачення для пропущених значень

full$Age[is.na(full$Age)] <- predict(Age\_forest, full)[is.na(full$Age)]

1. Лінійна класифікація, коли рішення приймається в результаті виконання лінійного оператора над вхідними даними.

Нехай – вектор вхідних даних. Класифікатор обчислює вихідне значення за формулою

де – дійсний вектор вагових коефіцієнтів, а – функція, яка перетворює скалярний добуток двох векторів на відповідне вихідне значення (лінійний функціонал, що відображає в ). Значення ваг обчислюються в процесі машинного навчання на навчальній множині.

# Створення лінійної моделі для передбачення ціни квитка (клас, стать, кількість родичів) та віку (клас, стать, кількість родичів, ціну квитка, звання)

Fare.mod<- lm(Fare ~ Pclass + Sex + SibSp + Parch, data = full)

Age.mod <- lm(Age ~ Pclass + Sex +SibSp + Parch + Fare + Title, data = full)

# Застосування алгоритму передбачення для пропущених значень ціни квитка та віку

full$Fare[is.na(full$Fare)] <- predict(Fare.mod, full)[is.na(full$Fare)]

full$Age[is.na(full$Age)] <- predict(Age.mod, full)[is.na(full$Age)]

1. Логічні припущення щодо віку і ціни.

Для того, щоб визначити вік скористаємося рисунком 5.14. З цього рисунку можна зробити висновок, що замість пропущених значень можна вставити середній вік пасажирів з таким самим званням, як і у того в кого вік відсутній. Так само, не зменшуючи загальності, можна вчинити і з ціною за квиток – замість пропущених значень можна вставити середню ціну, яку заплатили пасажири з того самого класу, що і пасажир в якого ціна квитка відсутня.

Програмна реалізація методів описаних в попередніх чотирьох розділах наведена в додатку 1.

## 5.3. Результати

* 1. SVM

Пропущені значення заповнені за допомогою:

* Лінійного класифікатора

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Функція, що є ядром в алгоритмі  Параметри | Гаусова | Поліноміальна | Лінійна | Гіперболічного тангенса | Лапласа | Бесселя | Сплайн |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.78947 | 0.73684 | 0.78468 | 0.61244 | 0.79904 | 0.77511 | 0.59330 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.78947 | 0.77511 | 0.58373 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.77990 | 0.78468 | 0.38755 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.77990 | 0.78468 | 0.58733 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.78947 | 0.78468 | 0.76076 | 0.61244 | 0.78468 | 0.77033 | 0.57894 |
| Pclass + Sex + Age + Fare + Embarked | 0.78468 | 0.76076 | 0.76076 | 0.58373 | 0.77511 | 0.77511 | 0.68421 |
| Pclass + Sex + Age + Fare | 0.76555 | 0.74002 | 0.76555 | 0.54002 | 0.77511 | 0.77033 | 0.70344 |

Таблиця 5.3 – Результат застосування алгоритму SVM (лінійний класифікатор для передбачення пропущених значень)

* Алгоритму Random Forest

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Функція, що є ядром в алгоритмі  Параметри | Гаусова | Поліноміальна | Лінійна | Гіперболічного тангенса | Лапласа | Бесселя | Сплайн |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.78947 | 0.78468 | 0.78468 | 0.62200 | 0.79425 | 0.77511 | 0.57894 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.78947 | 0.78468 | 0.78468 | 0.61722 | 0.78468 | 0.77033 | 0.57894 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.78947 | 0.78468 | 0.78468 | 0.62679 | 0.78468 | 0.77511 | 0.37320 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.77990 | 0.77033 | 0.59330 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.78947 | 0.76076 | 0.76076 | 0.61244 | 0.76555 | 0.75598 | 0.58851 |
| Pclass + Sex + Age + Fare + Embarked | 0.78468 | 0.76555 | 0.76555 | 0.53110 | 0.76555 | 0.77511 | 0.66507 |
| Pclass + Sex + Age + Fare | 0.76555 | 0.76555 | 0.76555 | 0.51216 | 0.76076 | 0.76555 | 0.27751 |

Таблиця 5.4 – Результат застосування алгоритму SVM (алгоритм Random Forest для передбачення пропущених значень)

* Логічних припущень

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Функція, що є ядром в алгоритмі  Параметри | Гаусова | Поліноміальна | Лінійна | Гіперболічного тангенса | Лапласа | Бесселя | Сплайн |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.79904 | 0.77511 | 0.58373 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.78947 | 0.78468 | 0.78468 | 0.60765 | 0.78468 | 0.77990 | 0.57894 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.78947 | 0.78468 | 0.78468 | 0.61244 | 0.78947 | 0.77990 | 0.38755 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.78947 | 0.78468 | 0.78468 | 0.60287 | 0.77511 | 0.78468 | 0.58373 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.78947 | 0.76076 | 0.76076 | 0.59808 | 0.77511 | 0.77033 | 0.57416 |
| Pclass + Sex + Age + Fare + Embarked | 0.77990 | 0.76555 | 0.76555 | 0.58373 | 0.77511 | 0.77033 | 0.68421 |
| Pclass + Sex + Age + Fare | 0.76555 | 0.76555 | 0.76555 | 0.55502 | 0.74641 | 0.76555 | 0.70334 |

Таблиця 5.5 – Результат застосування алгоритму SVM (логічні припущення для передбачення пропущених значень)

Попри використання різних методів для заповнення пропущених значень результати у таблицях 5.3, 5.4, 5.5 співпадають у абсолютній більшості випадків, крім тих, де використовується ядро гіперболічного тангенса та сплайн. З отриманих результатів легко зрозуміти, що згадані вище ядра не варто застосовувати до цієї моделі. Отже, в результаті перебору різних ядер і параметрів можна зробити висновок, що найкращий результат отримуємо при використанні гаусового, лінійного і лапласового ядра. Причому майже у всіх випадках такі параметри, як кількість родичів, наявність родичів та номера каюти не впливають на результат. Найкращий результат виходить у випадку використання ядра Лапласа та застосування всіх параметрів. До того ж, коли ми зменшуємо кількість параметрів, то зменшується кількість правильних передбачень.

* 1. Random Forest

Пропущені значення заповнені за допомогою:

* Лінійного класифікатора

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість побудованих дерев для виконання алгоритму  Параметри | 7000 | 10000 | 13000 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.77990 | 0.80861 | 0.77990 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.80861 | 0.80861 | 0.80861 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.80382 | 0.80382 | 0.79904 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.79904 | 0.79904 | 0.80382 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.78468 | 0.79947 | 0.78947 |
| Pclass + Sex + Age + Fare + Embarked | 0.78468 | 0.77990 | 0.77990 |
| Pclass + Sex + Age + Fare | 0.78468 | 0.78468 | 0.78468 |

Таблиця 5.6 – Результат застосування алгоритму Random Forest (лінійний класифікатор для передбачення пропущених значень)

* Алгоритму Random Forest

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість побудованих дерев для виконання алгоритму  Параметри | 7000 | 10000 | 13000 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.78468 | 0.78947 | 0.78947 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.81339 | 0.80861 | 0.80861 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.79425 | 0.79904 | 0.79904 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.79904 | 0.79425 | 0.80382 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.77990 | 0.77990 | 0.77990 |
| Pclass + Sex + Age + Fare + Embarked | 0.77990 | 0.77990 | 0.77990 |
| Pclass + Sex + Age + Fare | 0.77033 | 0.77033 | 0.77033 |

Таблиця 5.7 – Результат застосування алгоритму Random Forest (алгоритм Random Forest для передбачення пропущених значень)

* Логічних припущень

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість побудованих дерев для виконання алгоритму  Параметри | 7000 | 10000 | 13000 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin | 0.77990 | 0.77990 | 0.78468 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isCabin | 0.80861 | 0.80861 | 0.81339 |
| Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily | 0.80382 | 0.79904 | 0.79904 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.80382 | 0.80382 | 0.79904 |
| Pclass + Sex + Age + Fare + Embarked+ Title | 0.78468 | 0.78947 | 0.78468 |
| Pclass + Sex + Age + Fare + Embarked | 0.78468 | 0.78947 | 0.78468 |
| Pclass + Sex + Age + Fare | 0.78947 | 0.78947 | 0.78947 |

Таблиця 5.8 – Результат застосування алгоритму Random Forest (логічні припущення для передбачення пропущених значень)

Використання різних методів для заповнення пропущених значень результатів у таблицях 5.6, 5.7, 5.8 певним чином впливає на результат. При застосуванні лінійного класифікатора результати при всіх значеннях виходять доволі рівними, але не досягається такий максимум, який є в інших методах. Заповнення віку і ціни білету за допомогою Random Forest дозволяє отримати максимальний результат для експерименту, але зі зменшенням кількості параметрів відчутно погіршуються результати. Результати, що представлені в таблиці 5.8 є найкращими для цієї моделі, оскільки досягається максимальний результат і середній результат по всіх параметрах є найвищим. В алгоритмі Random Forest змінювалась кількість дерев, які використовуються для навчання. Для випадкових дерев характерно, що для кращого результату необхідна більша кількість параметрів за якими можна розділяти вибірку. В даному випадку оптимально використовувати 10000 дерев, але кількість дерев залежить від кількості параметрів і величини навчальної вибірки. Тому необхідно підбирати кількість дерев з обережністю, бо завелика кількість веде до перенавчання. Тобто отриманий алгоритм показує кращі результати на навчальній вибірці, але зовсім невдалі на тестовій. При використанні більше 15000 дерев для даної навчальної вибірки результат на тестовій вибірці відчутно погіршується. Отже, відбувається перенавчання.

* 1. Градієнтний бустинг

Пропущені значення заповнені за допомогою:

* Лінійного класифікатора

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Коефіцієнти алгоритму  Параметри | (𝜂, ) | | | | | |
| (0.02,10) | (0.3,10) | (1,10) | (0.02,3) | (0.3,3) | (1,3) |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin+isFamily | 0.76076 | 0.76555 | 0.74641 | 0.79425 | 0.78468 | 0.7711 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin | 0.74641 | 0.73684 | 0.75119 | 0.79425 | 0.78468 | 0.7711 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isFamily | 0.75119 | 0.75119 | 0.77033 | 0.79425 | 0.7468 | 0.77033 |
| Pclass+Sex+ Age + Fare +Embarked +Title | 0.75119 | 0.75119 | 0.77033 | 0.79425 | 0.78468 | 0.77033 |
| Pclass+Sex+ Age + Fare | 0.74641 | 0.76076 | 0.76076 | 0.77033 | 0.78947 | 0.77511 |

Таблиця 5.9 – Результат застосування алгоритму градієнтного бустингу (лінійний класифікатор для передбачення пропущених значень)

* Алгоритму Random Forest

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Коефіцієнти алгоритму  Параметри | (𝜂, ) | | | | | |
| (0.02,10) | (0.3,10) | (1,10) | (0.02,3) | (0.3,3) | (1,3) |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin+isFamily | 0.75598 | 0.78468 | 0.73684 | 0.78947 | 0.77990 | 0.78468 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin | 0.75598 | 0.74641 | 0.75119 | 0.78947 | 0.77990 | 0.78468 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isFamily | 0.74641 | 0.74162 | 0.76076 | 0.77033 | 0.77990 | 0.76555 |
| Pclass+Sex+ Age + Fare +Embarked +Title | 0.75119 | 0.74641 | 0.75598 | 0.77033 | 0.77990 | 0.76555 |
| Pclass+Sex+ Age + Fare | 0.74641 | 0.75598 | 0.74641 | 0.79425 | 0.78947 | 0.78947 |

Таблиця 5.10 – Результат застосування алгоритму градієнтного бустингу (алгоритм Random Forest для передбачення пропущених значень)

* Логічних припущень

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Коефіцієнти алгоритму  Параметри | (𝜂, ) | | | | | |
| (0.02,10) | (0.3,10) | (1,10) | (0.02,3) | (0.3,3) | (1,3) |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin+isFamily | 0.76555 | 0.75598 | 0.76555 | 0.79425 | 0.78425 | 0.78947 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isCabin | 0.76555 | 0.75598 | 0.76555 | 0.79425 | 0.78468 | 0.78947 |
| Pclass+Sex+ Age + Fare +Embarked+Title+isFamily | 0.77033 | 0.77033 | 0.77511 | 0.79245 | 0.79425 | 0.76555 |
| Pclass+Sex+ Age + Fare +Embarked +Title | 0.77033 | 0.77033 | 0.77511 | 0.79425 | 0.79425 | 0.76555 |
| Pclass+Sex+ Age + Fare | 0.77511 | 0.77033 | 0.76555 | 0.79425 | 0.78468 | 0.79425 |

Таблиця 5.11 – Результат застосування алгоритму градієнтного бустингу (логічні припущення для передбачення пропущених значень)

– коефіцієнт стиску для запобігання перенавчанню. Тобто кожна вага, яка враховує попередні помилки множиться на коефіцієнт стиску, щоб зменшити вплив кожного попереднього дерева на наступне. – максимально можлива глибина дерева. Використання різних методів для заповнення пропущених значень результатів у таблицях 5.9, 5.10, 5.11 впливає на результат. Найкращі результати отримані в таблиці з логічними припущеннями, а найгірші – з використанням алгоритму Random Forest. Як видно із згаданих вище таблиць, найкращий результат забезпечує малий коефіцієнт стиску в неглибокому дереві.

* 1. Нейронні мережі

Пропущені значення заповнені за допомогою:

* Лінійного класифікатора

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість нейронів в шарі  Параметри | Перший шар з трьома нейронами | Перший шар з п’ятьма | Перший шар 3 нейрони, другий – 2 |
| Pclass +Sex+ Age + Embarked + Title FsizeD + isFamily+isCabin | 0.78947 | 0.69377 | 0.69856 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isCabin | 0.79425 | 0.75598 | 0.73205 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isFamily | 0.79425 | 0.71770 | 0.66028 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.78947 | 0.77511 | 0.63636 |
| Pclass + Sex + Age + Embarked+ Title+ FsizeD | 0.79425 | 0.49760 | 0.70813 |
| Pclass + Sex + Age + Fare + Embarked | 0.73205 | 0.70813 | 0.64114 |
| Pclass + Sex + Age + Fare | 0.70334 | 0.66028 | 0.66028 |

Таблиця 5.12 – Результат застосування алгоритму навчання нейронної мережі (лінійний класифікатор для передбачення пропущених значень)

* Алгоритму Random Forest

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість нейронів в шарі  Параметри | Перший шар з трьома нейронами | Перший шар з п’ятьма | Перший шар 3 нейрони, другий – 2 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isFamily+isCabin | 0.69377 | 0.66507 | 0.75119 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isCabin | 0.75598 | 0.74162 | 0.75598 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isFamily | 0.71291 | 0.72248 | 0.73684 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.53110 | 0.55023 | 0.66985 |
| Pclass + Sex + Age + Embarked+ Title+ FsizeD | 0.73684 | 0.70334 | 0.72727 |
| Pclass + Sex + Age + Fare + Embarked | 0.63157 | 0.54545 | 0.65550 |
| Pclass + Sex + Age + Fare | 0.72248 | 0.65550 | 0.71770 |

Таблиця 5.13 – Результат застосування алгоритму навчання нейронної мережі (алгоритм Random Forest для передбачення пропущених значень)

* Логічних припущень

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість нейронів в шарі  Параметри | Перший шар з трьома нейронами | Перший шар з п’ятьма | Перший шар 3 нейрони, другий – 2 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isFamily+isCabin | 0.37320 | 0.37320 | 0.37320 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isCabin | 0.37320 | 0.37320 | 0.37320 |
| Pclass +Sex+ Age + Embarked + Title + FsizeD + isFamily | 0.37320 | 0.36842 | 0.37320 |
| Pclass + Sex + Age + Fare + Embarked + Title + FsizeD | 0.37320 | 0.37799 | 0.37320 |
| Pclass + Sex + Age + Embarked+ Title+ FsizeD | 0.37320 | 0.37320 | 0.37320 |
| Pclass + Sex + Age + Fare + Embarked | 0.37320 | 0.61722 | 0.37799 |
| Pclass + Sex + Age + Fare | 0.37320 | 0.37320 | 0.37320 |

Таблиця 5.14 – Результат застосування алгоритму навчання нейронної мережі (логічні припущення для передбачення пропущених значень)

Результат застосування нейронної мережі представлений у таблицях 5.12, 5.13, 5.14. Як видно, найкращі результати виходять у результаті застосування лінійного класифікатора. Проте велика різниця в результатах при невеликій зміні параметрів і нейронів показує те, що нейронна мережа не є придатною для застосування на даній вибірці. Можна припустити, що суть проблеми полягає в тому, що наша вибірка є занадто малою для навчання нейронної мережі.

Результати всіх використаних методів можна підсумувати кількома пунктами:

1. Алгоритми SVM, Random Forest, XGBoost показують високі результати. Після усереднення кожного з них (таблиця 5.15) видно, що всі значення перевищують 0.76, що є дуже високим результатом, в таблиці 5.16 наведені максимальні значення для кожного з алгоритмів. З таблиці 5.16 видно, що найкращий результат показує алгоритм Random Forest (кількість вірних значень – 0.81339).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод знаходження загиблих  Метод  знаходження віку | SVM | Random Forest | Градієнтний бустинг | Нейронні мережі |
| Лінійний класифікатор | 0.77766 | 0.79405 | 0.76720 | 0.71155 |
| Random Forest | 0.77662 | 0.78924 | 0.76650 | 0.68489 |
| Логічні припущення | 0.777848 | 0.79380 | 0.77775 | 0.38505 |

Таблиця 5.15 – Усереднені результати в залежності від методів передбачень

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод знаходження загиблих  Метод  знаходження віку | SVM | Random Forest | Градієнтний бустинг | Нейронні мережі |
| Лінійний класифікатор | 0.79904 | 0.80861 | 0.79425 | 0.79425 |
| Random Forest | 0.79425 | 0.81339 | 0.79425 | 0.75598 |
| Логічні припущення | 0.79904 | 0.81339 | 0.79425 | 0.61722 |

Таблиця 5.16 – Максимальні результати в залежності від методів передбачень

1. Графік з результатами алгоритму SVM зображений на рисунку 5.16. В результаті перебору різних ядер і параметрів можна зробити висновок, що найкращий результат отримуємо при використанні гаусового, лінійного і лапласового ядра. Також видно, що результат покращується із збільшенням кількості параметрів. Аналогічно для алгоритму Random Forest, з рисунку 5.17 зрозуміло, що кількість ознак за якими відбувається розчеплення позитивно впливає на результат виконання алгоритму.

Рис. 5.16 – Графік, який відображає вплив ядра, що використовує алгоритм SVM та кількості вхідних параметрів на якість передбачення

Рис. 5.17 – Графік, який відображає вплив кількості дерев, що використовує алгоритм Random Forest та кількості вхідних параметрів на якість передбачення

1. Алгоритм градієнтного бустингу використовує ряд важливих параметрів, проте найсуттєвішими є максимальна глибина дерева і коефіцієнт стиску. Графік залежності між якими зображений на рисунку 5.18. З графіка видно, що вибравши більшу глибину дерева варто обирати великий коефіцієнт стиску і в протилежному випадку при не великій глибині дерева необхідний малий коефіцієнт стиску.

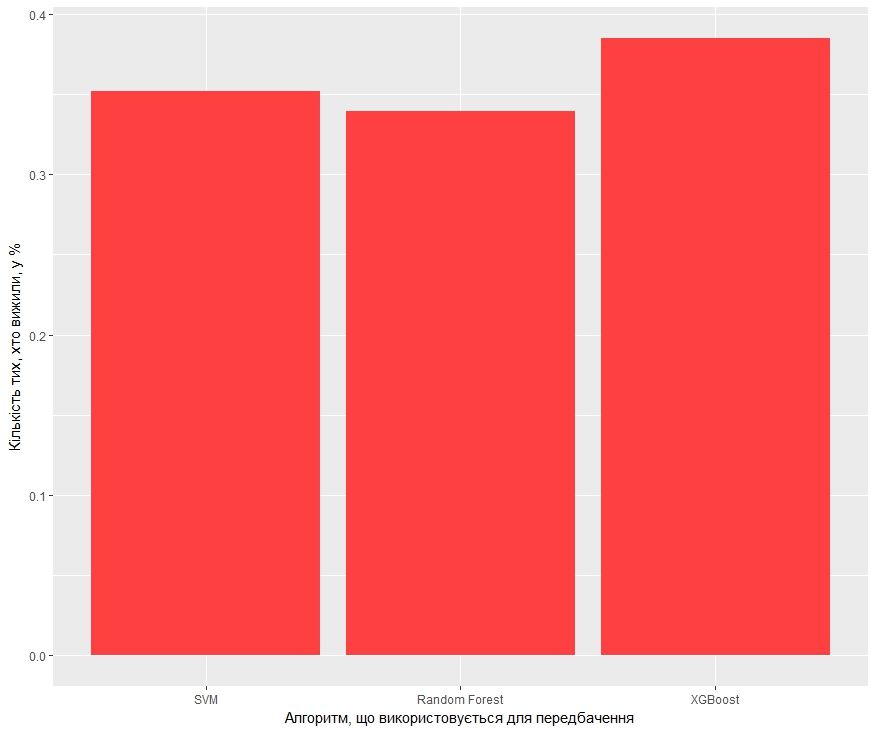
Рис. 5.18 – Графік, який відображає плив максимальної глибини і коефіцієнту стиску на якість передбачення

Рис. 5.19 – Діаграма, що показує відмінність у кількості тих, хто вижив в залежності від алгоритму

1. Для побудови діаграми, що показує відмінність між кількістю людей, що вижили і алгоритмом, що використовується беруться результати з найвищою точністю правильного передбачення, для Random Forest – 0.81339, SVM – 0.79904, XGBoost – 0.79425. З побудованої за наведеними вище умовами діаграми 5.19 видно, що незважаючи на те, що результати передбачень у цих трьох випадках є дуже близькими, кількість загиблих в кожному з алгоритмів відрізняється. Найбільше людей, що вижили показує алгоритм XGBoost, найменше – Random Forest.
2. Вибір алгоритму залежить від об’єму, якості, природи даних, програмних інструкцій, що реалізовують алгоритм, часу за який необхідно отримати результат. Точно вказати, який алгоритм покаже найкращий результат не зможе навіть найдосвідченіший спеціаліст з аналізу даних, допоки не спробує реалізовувати різні алгоритм. Для більш ефективної роботи і збільшення прибутків практично всі великі компанії використовуються методи машинного навчання, зокрема описані вище методи. Наприклад, машинне навчання використовується для фільтрації спаму, підбору найточніших результатів при введенні пошукових запитів, привертання уваги покупців (коли на базі куплених товарів і заповненої форми в інтернет-магазині пропонуються товари, що можуть зацікавити покупця), передбачення вартості акцій і т.д. Рекомендації до застосування:
3. Якщо вибірка містить велику кількість ознак. Бувають випадки, коли кількість ознак перевищує кількість об’єктів. Зазвичай це трапляється в області генетики. Велика кількість ознак буде заважати ефективній роботі деяких алгоритмів, із-за чого час навчання стає неприпустимо великим. Для подібних випадків добре підходить метод опорних векторів. Одне з важливих обмежень застосування цього методу полягає у можливості поділу лише на два класи.
4. Швидкість роботи алгоритму Random Forest відчутно залежить від кількості параметрів, проте він дає кращі результати, ніж SVM на вибірках з середньою чи не великою кількістю параметрів. Також він вимагає малі затрати пам’яті навіть при великих об’ємах даних.
5. Алгоритм XGBoost повільніший від Random Forest, проте вважається точнішим. Оскільки, він враховує помилки кожного попереднього алгоритму, то для його успішної реалізації потрібні значно більші об’єми пам’яті, ніж в попередніх алгоритмах. Нейронні мережі є одним з найцікавіших методів машинного навчання, оскільки, вони здатні давати результати при невідомих закономірностях розвитку ситуацій і залежностях між вхідними та вихідними даними. Проте вони часто ведуть до неоднозначного розв’язку, або ж взагалі до тупикової ситуації. Також ще один недолік нейронної мережі – для підготовки навчальної вибірки необхідно багато даних, що не завжди можливо забезпечити.

# 

# Висновки

В магістерській кваліфікаційній роботі розглянуті чотири підходи контрольованого навчання – випадкові дерева, градієнтний бустинг, нейронні мережі (навчання з вчителем) та метод опорних векторів. Алгоритм випадкових дерев використовує деревоподібний граф, який допомагає структуровано і систематично розв’язати задачу. Метод опорних векторів будує оптимальну роздільну гіперплощину, яка розділяє об’єкти на два класи. Градієнтний бустинг передбачає побудову ансамблю елементарних моделей, які уточняють одна одну, тобто кожна наступна модель навчається враховуючи помилки попередніх. Алгоритм контрольованого навчання нейронної мережі натреновує мережу на вибірці, в якій є готові правильні вихідні дані.

Для дослідження було використано дані про пасажирів пароплаву “Титанік”. В якості ознак для навчання алгоритму передбачення виступали: стать, вік, ціна квитка, клас каюти, кількість родичів на борту, порт відправлення, звання та наявність номера кабіни у пасажира. Передбачення здійснювалося на базі згаданих вище методів машинного навчання. Загалом всі методи, окрім навчання нейронної мережі, показали високі результати вірних передбачень, понад 76% правильних відповідей у середньому по алгоритму. Метод випадкових дерев показав найбільший відсоток вірних відповідей, а саме – 81.34%.

Машинне навчання застосовується практично у всіх галузях сучасного життя, зокрема: захист даних (передбачення шкідливості того чи іншого файлу на основі даних про вже відомі шкідливі файли), торгівля цінними паперами (передбачення поведінки ринку в конкретний день), медицина (виявлення схильності захворіти на певну хворобу), рекомендації (аналіз активності покупця для передбачення, що він захоче купити чи переглянути наступним), інтелектуальні автомобілі (автоматичне регулювання параметрів руху в режимі автопілота у справжньому дорожньому трафіку) тощо.

# Список використаних джерел і літератури:

1. Вьюгин В.В. Математические основы машинного обучения и прогнозирования. М.: МЦНМО, 2014. С. 88-155.
2. Чистяков С. Случайные леса: обзор // Труды Карельского научного центра РАН (1). 2013. С. 117-136.
3. Behlmann P., Yu B. Analyzing bagging // Annals of Statistics. 2002. Vol. 30. P. 927–961
4. Breiman L. Random forests // Machine learning. 2001. Р. 5-32.
5. Breiman L., Friedman R., Olshen R., Stone C. Classiﬁcation and Regression Trees. Belmont, California: Wadsworth International, 1984. Р. 246-312.
6. Freund Y., Schapire R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting // Journal of Computer and System Sciences – 1997. – V. 55. – P. 119–139.
7. Friedman J.H. Greedy Function Approximation: a Gradient Boosting Machine. Technical Report. Dept. of Statistics, Stanford University, 1999. Р. 1189-1232.
8. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2008. Р. 337-455, 587-603.
9. Ho T. K. C4.5 Decision Forests // Proceedings of the 14rd International Conference on Pattern Recognition. (Brisbane, Australia, 1998). 1998. P. 17–20.
10. Scholkopf B. and Smola A. Learning with Kernels. MIT Press, Cambridge, MA, 2002. Р. 10-19.
11. Vapnik V.N. Statistical Learning Theory. – New York: Wiley, 1998. Р. 119-155.

# Додаток 1. Текст програми

#Підключення бібліотек

require(readr)

library(plyr)

library(foreign)

library('ggplot2')

library('ggthemes')

library('scales')

library('dplyr')

library('randomForest')

library('corrplot')

library('plyr')

library(data.table)

library(rpart)

library(neuralnet)

#Задання робочої директорії

setwd("C:/Users/Oksana/Desktop/Titanic\_habr")

data\_train <- read\_csv("train.csv")

data\_test <- read\_csv("test.csv")

data\_test$Survived <- 0

#Об'єднання тестової і навчальної вибірки

full <- join(data\_train, data\_test, type = "full")

#Введення нових змінних кількості родичів, наявності родичів і кабіни в пасажира

full$Fsize <- full$SibSp+full$Parch

full$isFamily <- as.factor(as.numeric(full$Fsize > 0))

full$isCabin <- factor(ifelse(is.na(full$Cabin),0,1))

#Оскільки, тільки в двох пасажирів не заповнений порт відпралення, то заповнюємо його значенням порту в якому сіло найбільше пасажирів

full$Embarked[c(62, 830)] <- 'S'

# Заповнення пропущених значень віку і ціни квитка за допомогою алгоритму Random Forest

# library(randomForest)

# full$Fare[full$Fare < 6] <- NA

# full$Sex <- factor(full$Sex)

# train\_random\_fare <- full[!is.na(full$Fare), ]

# train\_random\_fare %<>% select(Pclass, Sex, SibSp, Parch,Fare)

train\_random\_fare[is.na(train\_random\_fare$Pclass),]

# Fare\_forest <- randomForest(Fare ~ Pclass + Sex + SibSp + Parch, data = train\_random\_fare, ntree = 5000)

# full$Fare[is.na(full$Fare)] <- predict(Fare\_forest, full)[is.na(full$Fare)]

# train\_random\_age <- full[!is.na(full$Age), ]

# Age\_forest <- randomForest(Age ~ Pclass + Sex +SibSp + Parch + Fare, data = train\_random\_age, ntree = 5000)

# full$Age[is.na(full$Age)] <- predict(Age\_forest, full)[is.na(full$Age)]

# Заповнення пропущених значень віку і ціни квитка за допомогою побудови лінійної моделі

# full$Fare[is.na(full$Fare)] <- 0

# Fare.mod<- lm(Fare ~ Pclass + Sex +

# SibSp + Parch + Age, data = full)

# Age.mod <- lm(Age ~ Pclass + Sex +

# SibSp + Parch + Fare, data = full)

# full$Fare[full$Fare < 6] <- predict(Fare.mod, full)[full$Fare[full$Fare < 6]]

# full$Age[is.na(full$Age)] <- predict(Age.mod, full)[is.na(full$Age)]

# В лінійній моделі можуть бути випадки від'ємних значень, тому міняємо їх на нулі

# full$Fare[full$Fare < 0] <- 0

# full$Age[full$Age<0] <- 0

# Введення нової змінної титул, яка виокремлюється з імені

full$Title <- full$Name %>% str\_extract(., "\\w+\\.") %>% str\_sub(.,1, -2)

unique(full$Title)

# Функція, що змінює масив старих титулів на один новий

change.titles <- function(data, old\_title, new\_title) {

for (title in old\_title) {

data$Title[data$Title == title] <- new\_title

}

return (data$Title)

}

full$Title <- change.titles(full,

c("Capt", "Col", "Dr", "Major", "Rev"),

"Officer")

full$Title <- change.titles(full, c("Don", "Dona", "Jonkheer", "Lady", "Sir", "Countess"), "Aristocratic")

full$Title <- change.titles(full, c("Ms"), "Mrs")

full$Title <- change.titles(full, c("Mlle", "Mme"), "Miss")

full$Title <- as.factor(full$Title)

# Визначення пропущених значень віку і ціни за допомогою логічних припущень

# Функція, що усереднює числове значення за відповідним значенням фактору множини полів

impute.mean <- function (impute\_col, filter\_var, var\_levels) {

for (lev in var\_levels) {

impute\_col[(filter\_var == lev) & is.na(impute\_col)] <-

mean(impute\_col[filter\_var == lev], na.rm = T)

}

return (impute\_col)

}

# Заповнення пропущеного віку за відповідними усередненими значеннями по званню пасажирів

full$Age <- impute.mean(full$Age, full$Title, c("Aristocratic", "Officer" "Master", "Mrs", "Miss", "Mr"))

full$Fare[full$Fare < 6] <- NA

# Заповнення пропущеної вартості квитка за відповідними усередненими значеннями класу пасажирів

full$Fare <- impute.mean(full$Fare, full$Pclass, as.numeric(levels(full$Pclass)))

# Приведення даних до факторів

data\_train$Pclass <- factor(data\_train$Pclass)

data\_train$Sex <- factor(data\_train$Sex)

data\_train$isFamily <- factor(data\_train$isFamily)

data\_train$Embarked <- factor(data\_train$Embarked)

data\_train$Title <- factor(data\_train$Title)

data\_train$isCabin <- factor(data\_train$isCabin)

data\_train$Survived <- factor(data\_train$Survived)

data\_train$FsizeD <- factor(data\_train$FsizeD)

data\_test$Pclass <- factor(data\_test$Pclass)

data\_test$Sex <- factor(data\_test$Sex)

data\_test$isFamily <- factor(data\_test$isFamily)

data\_test$Embarked <- factor(data\_test$Embarked)

data\_test$Title <- factor(data\_test$Title)

data\_test$isCabin <- factor(data\_test$isCabin)

data\_test$Survived <- factor(data\_test$Survived)

data\_test$FsizeD <- factor(data\_test$FsizeD)

# Навчання алгоритму Random Forest на тренувальній вибірці при зміні кількості дерев і перевірка точності на тестовій вибірці

forest <- randomForest(Survived ~Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train, ntree = 7000)

data\_test$Survived <- predict(forest, data\_test)

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/rf\_model\_sol\_log\_7\_1.csv', row.names = F)

forest <- randomForest(Survived ~Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train, ntree = 10000)

data\_test$Survived <- predict(forest, data\_test)

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/rf\_model\_sol\_log\_10\_1.csv', row.names = F)

forest <- randomForest(Survived ~Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train, ntree = 12000)

data\_test$Survived <- predict(forest, data\_test)

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/rf\_model\_sol\_log\_13\_1.csv', row.names = F)

# Навчання алгоритму SVM на тренувальній вибірці при зміні ядра і перевірка точності на тестовій вибірці

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

kernel="besseldot",

kpar=list(sigma = 2, order = 2, degree = 2),C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_bessel\_1.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

kernel="splinedot",

C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_splinedot\_1.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

kernel="laplacedot",

kpar=list(sigma = 0.1),C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_laplacedot\_1.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass + Sex + Age + Fare,

data = data\_train,

kernel="laplacedot",

kpar=list(sigma = 0.1),C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_laplacedot\_7.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass + Sex + Age + Fare,

data = data\_train,

kernel="tanhdot",

kpar=list(scale = 1, offset = 1),C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_tanhdot\_7.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

type="C-svc",

kernel="vanilladot", degree=5,

C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_vanilladot\_1.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

type="C-svc",

kernel="polydot", degree=5,

C=5,cross=3)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_polydot\_1.csv', row.names = F)

svm.model <- ksvm(Survived ~ Pclass +Sex+ Age + Fare + Embarked + Title + FsizeD + isFamily+isCabin,

data = data\_train,

kernel="rbfdot",

kpar=list(sigma = 0.015),C=50,cross=4)

data\_test$Survived <- predict(svm.model, data\_test, type = "response")

solution <- data.frame(Survived = data\_test$Survived, PassengerID = data\_test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/svm\_model\_sol\_log\_rbfdot\_1.csv', row.names = F)

## Передбачення за допомогою алгоритму XGBoost

train = full[0:891,]

test = full[892:1309,]

data = rbind(train,test)

data$Survived = as.factor(data$Survived)

data$Pclass = as.factor(data$Pclass)

# Запуск XG Boost Model

names(train)

data.xg = train[,c(3,5,6,10,12,14,15,17)]

label.xg = train[,"Survived"]

names(test)

test.data.xg = test[,c(3,5,6,10,12,14,15,17)]

# Використання dummyVars функції з бібліотеки caret для кодування змінних (тільки у такому випадку можна застосувати градієнтний бустинг)

dummy.train = dummyVars(~Pclass+Sex+Embarked+Title+isCabin+isFamily, data = data.xg)

dummy.train

data.train <- predict(dummy.train,data.xg)

train.set = cbind(data.xg,data.train)

names(train.set)

dummy.test = dummyVars(~Pclass+Sex+Embarked+Title+isCabin+isFamily, data = test.data.xg)

dummy.test

data.test <- predict(dummy.test,test.data.xg)

test.set = cbind(test.data.xg,data.test)

names(test.set)

# Вибір змінних, що будуть брати участь у передбаченні

test.set <- test.set[,c(3,4,9:13)]

test.set <- as.matrix(test.set)

train.set <- train.set[,c(3,4,9:13)]

train.set <- as.matrix(train.set)

# Запуск алгоритму градієнтного бустингу для навчання на тренувальній вибірці і перевірки точності на тестовій

xgb.model = xgboost(data = train.set, label = as.matrix(label.xg),max\_depth = 3, eta = 1, nthread = 2, nrounds = 2, objective = "binary:logistic", verbose = 0 )

xgb.model

survival\_pred = predict(xgb.model,newdata = test.set)

test$Survived = as.numeric(survival\_pred>0.5)

solution <- data.frame(Survived = test$Survived, PassengerID = test$PassengerId)

write.csv(solution, file = 'C:/Users/Oksana/Desktop/Titanic\_habr/Test/xgboost\_model\_log\_sol5\_3\_1.csv', row.names = F)

## Передбачення алгоритму за допомогою алгоритму навчання нейронної мережі з вчителем

train = full[0:891,]

test = full[892:1309,]

# Об'єднання двох таблиць для підготовки до застосування алгоритму

all<-rbind(train,test)

all[,Title:= as.factor(Title)]

all[,Embarked:= as.factor(Embarked)]

all[,Sex:= as.factor(Sex)]

all[,isFamily := as.numeric(factor(isFamily))]

all[,isCabin := as.numeric(factor(isCabin))]

all[,Sex := as.numeric(factor(Sex))]

all[,Title := as.numeric(factor(Title))]

all[,Embarked := as.numeric(factor(Embarked))]

all[,Pclass := as.numeric(factor(Pclass))]

# Нормалізація значень в проміжку від 0 до 1

maxs <- apply(all[,.(Title,Sex,Age,Fare,Embarked,Pclass,FamSize,Survived,isCabin, isFamily)], 2, max)

mins <- apply(all[,.(Title,Sex,Age,Fare,Embarked,Pclass,FamSize,Survived,isCabin, isFamily)], 2, min)

scaled <- as.data.frame(scale(all[,.(Title,Sex,Age,Fare,Embarked,Pclass,FamSize,Survived,isCabin, isFamily)], center = mins, scale = maxs - mins))

train\_scaled<-scaled[1:891,]

test\_scaled<-scaled[892:1309,]

# Запуск алгоритму навчання нейронної мережі з вчителем для навчання на тренувальній вибірці і перевірки точності на тестовій

f<-as.formula(Survived~Pclass +Sex+ Age + Embarked + Title + FamSize + isFamily+isCabin)

nn <- neuralnet(f,data=train\_scaled,hidden=3,linear.output=T)

pr.nn <- compute(nn,test\_scaled[colnames(test\_scaled) %in% c("Pclass","Sex", "Age","FamSize","Embarked","Title","isCabin","isFamily")])

test[,Survived := (pr.nn$net.result > -0.5)\*1]

my\_solution <- test[,.(PassengerId,Survived)]

write.csv(my\_solution, row.names = FALSE, file = "C:/Users/Oksana/Desktop/Titanic\_habr/Test/my\_solution\_nn\_log\_3\_1.csv")