**Міністерство освіти і науки України**

**Національний технічний університет**

**«Дніпровська політехніка»**

Інститут електроенергетики

(інститут)

Факультет інформаційних технологій

(факультет)

Кафедра Системного аналізу та управління

(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА

**кваліфікаційної роботи ступеню** магістра

(бакалавра, магістра)

**студента** Рибалка Ігоря Олександровича

(ПІБ)

**академічної групи** 124м-21-1

(шифр)

**спеціальності** 124 Системний аналіз

(код і назва спеціальності)

**на тему:** «Сучасні методи попередньої обробки та аналізу даних»

(назва за наказом ректора)

| Керівники | Прізвище, ініціали | Оцінка за шкалою | | Підпис |
|---|---|---|---|---|
| | | рейтинговою | інституційною | |
| **кваліфікаційної роботи** | д.ф.- м.н., проф. Купенко О.П. | | | |
| розділів: | | | | |
| Інформаційно-аналітичний | д.ф.- м.н., проф. Купенко О.П. | | | |
| Спеціальній | д.ф.- м.н., проф. Купенко О.П. | | | |

| | | | | |
|---|---|---|---|---|
| Рецензент | Доц. Гуліна І.Г. | | | |

| | | | | |
|---|---|---|---|---|
| Нормоконтролер | Доц.. Хом’як Т.В. | | | |

Дніпро

2022

**ЗАТВЕРДЖЕНО:**

завідувач кафедри

Системного аналізу та управління
(повна назва)

_____ Желдак Т.А.
(підпис) (прізвище, ініціали)

«_____»_____2022 р.

**Завдання**

**на кваліфікаційну роботу**

**ступеня** *магістра*
(бакалавра, магістра)

**студенту** Рибалка Ігоря Олександровича **академічної групи** 124м-21-1
(прізвище та ініціали) (шифр)

**спеціальності** 124 Системний аналіз

**на тему** «Сучасні методи попередньої обробки та аналізу даних»

Затверджена наказом ректора НТУ «Дніпровська політехніка» від 31.10.2022

№ 1200-с

| Розділ | Зміст | Термін виконання |
|---|---|---|
| Інформаційно-аналітичний | Дослідити методи сучасної попередньої обробки та аналізу даних. Визначити основні задачі дослідження даних. | 10.05.2021 |
| Спеціальний | Вирішити задачі, які є важливими для соціуму: пожежі у лісних масивах, виділення районів міста за злочинністю. Підготувати дані з цих областей для побудови моделей машинного навчання. Обрати та порівняти моделі машинного навчання. | 22.06.2021 |

**Завдання видано** _____ Купенко О.П.
(підпис керівника) (прізвище, ініціали)

**Дата видачі** _____

**Дата подання до екзаменаційної комісії** _____

**Прийнято до виконання** _____ Рибалко І.О
(підпис студента) (прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 75 с., 28 рис., 6 табл., 3 додатка, 12 джерел.

Проблема лісних пожеж набуває великих масштабів внаслідок змінення кліматичних умов.

Проблема злочинності у містах набуває все більших масштабів за рахунок перенаселення, та нерівномірного розподілу населення. Сучасні методи обробки та аналізу даних, а також моделі машинного навчання закликані вирішити поставлені проблеми.

Об’єктом дослідження є проблема лісних пожеж, та проблема підвищення злочинності у містах.

Предметом дослідження є алгоритми та методи, що застосовуються для підготовки та обробки даних, моделі машинного навчання.

Метою кваліфікаційної роботи є розв'язання проблеми лісних пожеж та злочинності у містах за допомогою сучасних методів попередньої обробки даних.

Методи дослідження: регресійний аналіз, дерево рішень, випадковий ліс, логістична регресія.

У інформаційно- аналітичному розділі наведено відомості про предмет дослідження, поставлені задачі дослідження та обрані методи їх розв’язання.

У спеціальному розділі виконано розв'язання задач контролю за лісними масивами, контролю за рівнем злочинності.

Практична цінність отриманих результатів – це в першу чергу можливість масштабувати та поширити отримані алгоритми попередньої обробки даних та моделі машинного навчання.

Ключові слова: РЕГРЕСІЯ, КЛАСТЕРІЗАЦІЯ, СТРАТИФІКАЦІЯ, ВИПАДКОВИЙ ЛІС, КРИТЕРІЇ ЗУПИНКИ, ФАКТОРНИЙ АНАЛІЗ.

# ABSTRACT

Explanatory note: 75 p., 28 pictures, 6 tables, 3 annexs, 12 sources.

The problem of forest fires is becoming larger due to changes in climate conditions. Modern methods of data collection using various sensors allow solving this problem by means of data analysis and machine learning.

The problem of crime in cities is becoming increasingly large due to overpopulation and uneven distribution of the population. Machine learning models are designed to solve this problem. Modern methods of data processing and analysis are the tools that will allow you to prepare data for machine learning models.

The object of the study is the problem of forest fires and the problem of increasing crime in cities.

The subject of research are algorithms and methods used for data preparation and processing, machine learning models.

The goal of the qualification work is to solve the problem of forest fires and crime in cities using modern data preprocessing methods.

Research methods: regression analysis, decision tree, random forest, logistic regression.

The informational and analytical section provides information on the subject of the study, set research tasks and selected methods of solving them.

In a special section, the tasks of controlling forest areas and controlling the level of crime have been solved.

The practical value of the obtained results is, first of all, the possibility to scale and spread the obtained data preprocessing algorithms and models.



Key words: REGRESSION, CLUSTERIZATION, STRATIFICATION, RANDOM FOREST, STOPPING CRITERIA, FACTOR ANALYSIS.

**ЗМІСТ**

# ВСТУП

У світі велика кількість даних зберігається у сирому вигляді, тобто вигляді, у якому їх не можна передавати в модель навчання.

Такі дані отримують з різних джерел, наприклад, датчики на полях, дані різних інтернет-операторів, дані великих продуктових мереж.

Так само слід згадати про те, що більшість даних придатна для дослідження тільки в контексті підходящих для цих даних завдань, тому так звана "фільтрація" даних, а також вибір фінальних та проміжних оцінок прямо залежить від структури та типу даних, що отримуються з джерел, вибір яких, у свою чергу залежить від поставлених завдань.

Головною метою даної дослідницької роботи є вивчення оптимальних, з точки зору математичного обґрунтування, підходів для підготовки даних до завдань навчання з учителем (навчання на розмічених даних) та навчання без вчителя. Основну увагу в роботи буде приділено розвідувальному аналізу даних. Буде обговорено низку підходів до вибору моделей, а також порівняння оцінювання та моделей, навчених реальних даних, наведених у роботі.

# 1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

## 1.1 Передобробка даних

### 1.1.1 Розмітка та очищення даних

Моделі навчання не можуть працювати із сирими даними. Після формування вибірки, наприклад, збору даних у файл формату csv і перетворення цих даних у структуру DataSet в Python за допомогою бібліотеки Pandas дані необхідно очистити (перетворити).

Очищення даних – це процес виявлення неузгоджених даних, нетипових даних (викидів), пропущених даних або зайвих даних у вибірці з подальшим перетворенням або видаленням цих даних залежно від того, як вони впливають на кінцеву модель.

Відповідно, аналітиком має бути проведена робота у зв'язку з якою буде прийнято рішення видаляти або перетворювати дані. Природними точками, на які має бути звернена увага є такі: розмір даних, які доведеться прибрати, вплив цих даних на оцінки, наприклад, викиди впливають на робастні оцінки, так само слід згадати, що дані завжди несуть у собі інформацію, звичайно, іноді ця інформація не принесе користі конкретній моделі. Наприклад, в даних до завдань з пошуку спалахів на полях часто існує інформація про орендаря землі, яка, природно ніяк не впливає на результат, що цікавить аналітика (буде пожежа чи ні).

Окремою темою можна назвати пропущені дані, робота з якими має безліч підходів.

### 1.1.2 Загальний огляд даних

Огляд даних відбувається у кілька етапів. Першим з яких можна виділити визначення коректного виду структури даних, у якому їх імпортували.

Якщо вони імпортовані коректно, можна перейти до більш детального огляду.

Які ознаки є у нашому наборі даних?

Ознаки можуть бути речовими, категоріальними, бінарними, номінальними, порядковими, також в складі неструктурованих та мета-даних. З усіма типами ознак при належному перетворенні ми можемо працювати як з речовими.

Як приклад подаю огляд даних з прокату мотоциклів.

Ознаки, подані у даних:

• season: 1 – весна, 2 – літо, 3 – осінь, 4 – зима

• yr: 0 - 2011, 1 - 2012

• mnth: від 1 до 12

• holiday: 0 – немає свята, 1 – є свято

• weekday: від 0 до 6

• workingday: 0 - неробочий день, 1 - робочий день

• weathersit: оцінка сприятливості погоди від 1 (чистий, ясний день) до 4 (злива, туман)

• temp: температура в Цельсіях

• atemp: температура відчуття в Цельсіях

• hum: вологість

• windspeed(mph): швидкість вітру в милях на годину

• windspeed(ms): швидкість вітру в метрах за секунду

• cnt: кількість орендованих велосипедів чи мотоциклів (це цільова ознака, яку ми передбачатимемо)

### 1.1.3 Візуальний огляд даних

У нашому наборі даних представлена цільова ознака, а значить це завдання навчання з учителем. Цільова ознака демонструє кількість орендованих мотоциклів. Візуальний огляд даних може бути представлений у різних видах на малюнку знизу «рис. 1» він представлений як точковий графік відношення цільової ознаки та всіх інших [8].

Який висновок ми можемо зробити?

Наприклад, висновок про наявність лінійної залежності між ознаками.

Очевидно, що залежність лінійна у таких ознак як температура, температура за відчуттями, швидкість вітру в милях на годину та за метри за секунду «рис. 1».

Також у огляді даних на наявність лінійних залежностей очевидною буде побудова кореляційної матриці, вона явно вкаже на сильні залежності між ознаками «рис. 2», а також може свідчити про наявність мультиколінеарності.

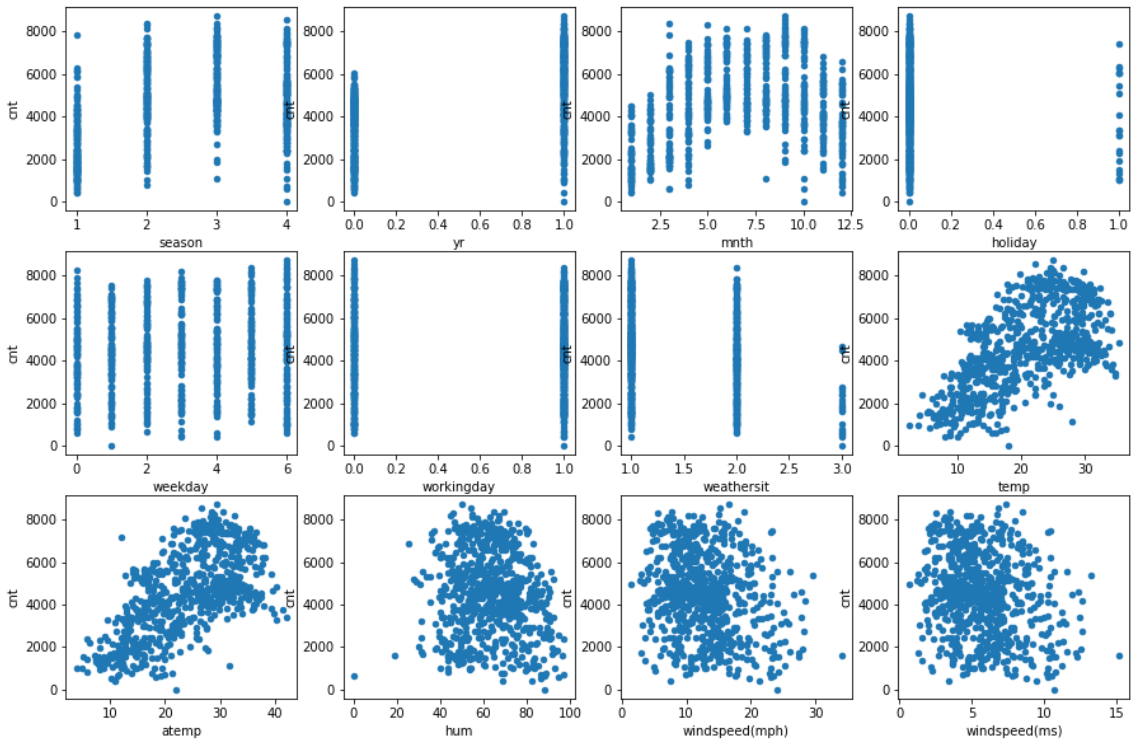


Рисунок 1.1 – графіки залежності факторів відносно цільового значення

]:

| | season | yr | mnth | holiday | weekday | workingday | weathersit | temp | atemp | hum | windspeed(mph) | windspeed(ms) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| season | 1.000000 | -0.001844 | 0.831440 | -0.010537 | -0.003080 | 0.012485 | 0.019211 | 0.334315 | 0.342876 | 0.205445 | -0.229046 | -0.229046 |
| yr | -0.001844 | 1.000000 | -0.001792 | 0.007954 | -0.005461 | -0.002013 | -0.048727 | 0.047604 | 0.046106 | -0.110651 | -0.011817 | -0.011817 |
| mnth | 0.831440 | -0.001792 | 1.000000 | 0.019191 | 0.009509 | -0.005901 | 0.043528 | 0.220205 | 0.227459 | 0.222204 | -0.207502 | -0.207502 |
| holiday | -0.010537 | 0.007954 | 0.019191 | 1.000000 | -0.101960 | -0.253023 | -0.034627 | -0.028556 | -0.032507 | -0.015937 | 0.006292 | 0.006292 |
| weekday | -0.003080 | -0.005461 | 0.009509 | -0.101960 | 1.000000 | 0.035790 | 0.031087 | -0.000170 | -0.007537 | -0.052232 | 0.014282 | 0.014282 |
| workingday | 0.012485 | -0.002013 | -0.005901 | -0.253023 | 0.035790 | 1.000000 | 0.061200 | 0.052660 | 0.052182 | 0.024327 | -0.018796 | -0.018796 |
| weathersit | 0.019211 | -0.048727 | 0.043528 | -0.034627 | 0.031087 | 0.061200 | 1.000000 | -0.120602 | -0.121583 | 0.591045 | 0.039511 | 0.039511 |
| temp | 0.334315 | 0.047604 | 0.220205 | -0.028556 | -0.000170 | 0.052660 | -0.120602 | 1.000000 | 0.991702 | 0.126963 | -0.157944 | -0.157944 |
| atemp | 0.342876 | 0.046106 | 0.227459 | -0.032507 | -0.007537 | 0.052182 | -0.121583 | 0.991702 | 1.000000 | 0.139988 | -0.183643 | -0.183643 |
| hum | 0.205445 | -0.110651 | 0.222204 | -0.015937 | -0.052232 | 0.024327 | 0.591045 | 0.126963 | 0.139988 | 1.000000 | -0.248489 | -0.248489 |
| windspeed(mph) | -0.229046 | -0.011817 | -0.207502 | 0.006292 | 0.014282 | -0.018796 | 0.039511 | -0.157944 | -0.183643 | -0.248489 | 1.000000 | 1.000000 |
| windspeed(ms) | -0.229046 | -0.011817 | -0.207502 | 0.006292 | 0.014282 | -0.018796 | 0.039511 | -0.157944 | -0.183643 | -0.248489 | 1.000000 | 1.000000 |

Рисунок 1.2 – кореляційна матриця

Для навчання без вчителя існують свої методи візуалізації як кожної конкретної задачі, так і загальні методи, що підходять і для навчання з вчителем, наприклад, ядерна оцінка щільності чи побудови гістограм, допомагають зрозуміти характер розподілу фактора (ознаки) «рис. 3».

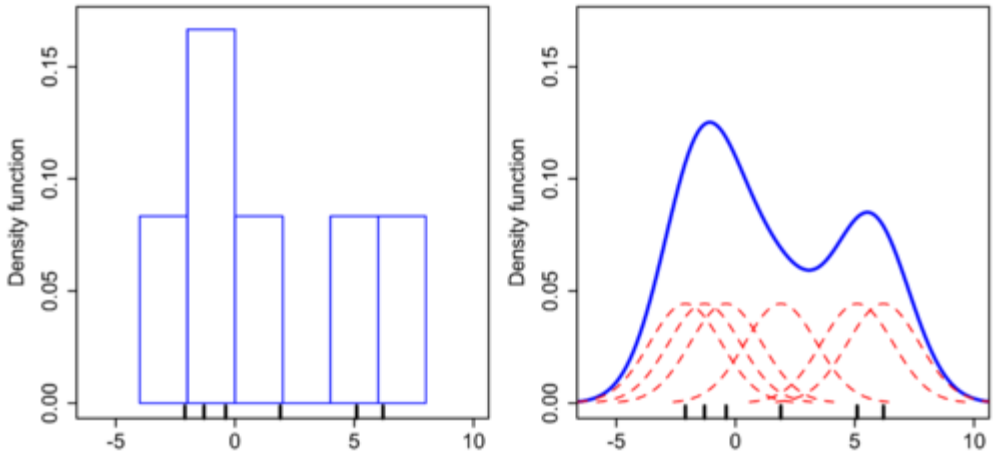


Рисунок 1.3 – Ядерна оцінка щільності та гістограма

### 1.1.4 Виявлення нетипових даних

Викиди - часта проблема для робастних оцінок, викиди погано впливають на точність моделей, а саме на функціонал, який ми мінімізуємо, наприклад, у задачі регресії. Нетипові дані будуть каменем спотикання для таких оцінок як середньоквадратична помилка або абсолютна помилка, тому

що наша модель буде на них працювати некоректно, а значить ми вносимо елемент неточності до самої моделі.

У цьому випадку є кілька походів, які застосовують для того, щоб виявити викиди на ранніх етапах, етапах огляду та підготовки даних, наприклад, найпривільнішим хоча і неточним буде використання гістограм визначення викидів (неточним з точки зору візуального огляду).

Найпопулярнішим же наочним з точки зору статистичних оцінок буде використання коробчастих діаграм [3].

Коробчасті діаграми вказують на максимальне та мінімальне значення у вибірці щодо медіальної оцінки, розділяючи область коробки щодо медіани на верхній та нижній квартиль, відповідно.

У разі використання подібних діаграм можна явно побачити нетипові точки «рис. 4».

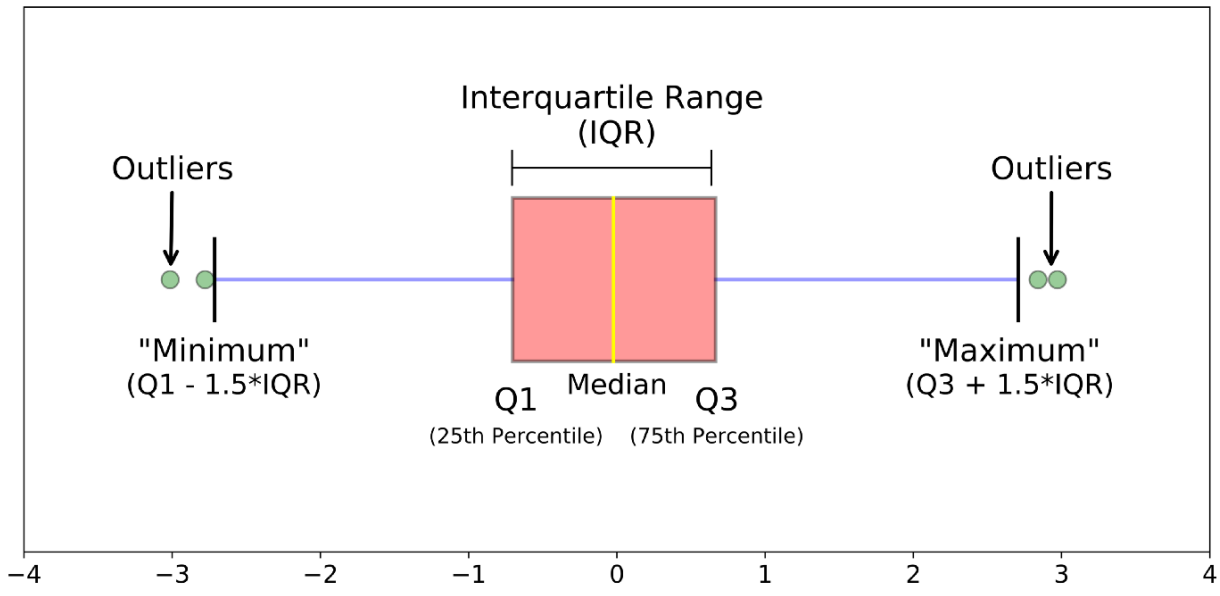


Рисунок 1.4 – Коробчаста діаграма

### 1.1.5 Пропущені дані

Пропущені дані - це дані відсутні в нашому наборі з різних причин, наприклад, це може бути некоректна робота датчиків або системи збору

даних, пошкодження даних, збої через різні кодування або первинної обробки даних на рівні заліза «рис. 5».

Існує чотири способи появи пропущених значень набору даних.

• Структурно відсутні дані,

• MCAR (відсутня зовсім випадково),

• MAR (відсутня випадково),

• NMAR (не випадково).

Насправді трапляється, що дані занадто високого порядку замінюються незначною величиною, це відбувається через системні налаштування робота, що виробляє формування набору.

У результаті ми отримуємо дані вже з пропусками і змушені вирішувати поставлене завдання одним із методів заповнення пропусків.

У подальших розділах буде розглянуто деякі популярні методи заповнення пропусків.

| | | | | | | |
|---|---|---|---|---|---|---|
| **0** | Douglas | Male | 97308.0 | 6.945 | True | Marketing |
| **1** | Thomas | Male | 61933.0 | NaN | True | NaN |
| **2** | Jerry | Male | NaN | 9.340 | True | Finance |
| **3** | Dennis | n.a. | 115163.0 | 10.125 | False | Legal |
| **4** | NaN | Female | 0.0 | 11.598 | NaN | Finance |
| **5** | Angela | NaN | NaN | 18.523 | True | Engineering |
| **6** | Shawn | Male | 111737.0 | 6.414 | False | na |
| **7** | Rachel | Female | 142032.0 | 12.599 | False | Business Development |
| **8** | Linda | Female | 57427.0 | 9.557 | True | Client Services |
| **9** | Stephanie | Female | 36844.0 | 5.574 | True | Business Development |
| **10** | NaN | NaN | NaN | NaN | NaN | NaN |

Рисунок 1.5 – Відсутні дані

1.1.4.1 Заповнення даних для часових рядів наступними або попередніми значеннями

Один із найчастіших способів заповнення пропущених даних у контексті часових рядів – це заповнення попереднім чи наступним значенням.

На графіку, наведеному нижче «рис 6» представлений графік залежності максимальної швидкості (MaxSpeed) від часу.

Рисунок 1.6 – графік залежності максимальної швидкості від часу з відсутніми даними

Дані заповнені наступним значенням «рис 7».

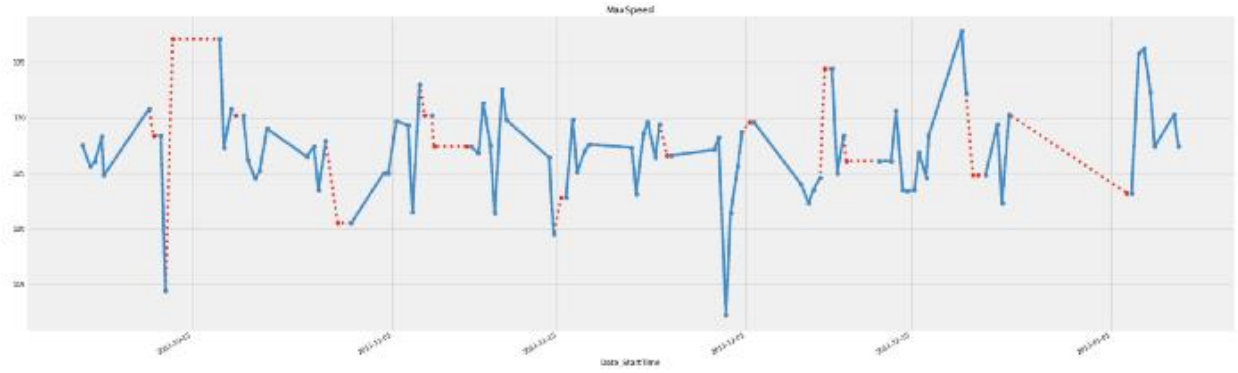

Рисунок 1.7 – Графік залежності максимальної швидкості від часу з заповненими за наступним значенням.

Дані заповнені попереднім значенням «рис. 8»

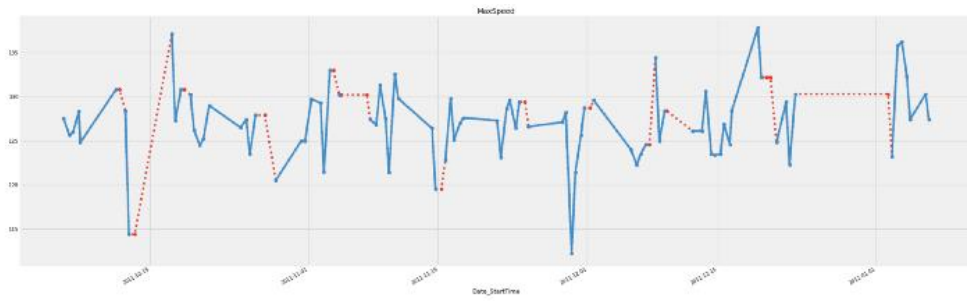

Рисунок 1.8 – Графік залежності максимальної швидкості від часу з заповненими за попереднім значенням.

Також можна заповнювати дані лінійно, поліноміально, квадратично, ігноруючи індекс.

1.1.4.2 Заповнення даних з допомогою середнього, модою, медіана.

Найпопулярнішими і тими, що найчастіше обираються дослідником є методи заповнення пропущених даних модою, медіаною, середнім [7].

Заповнення константними значеннями чи то мода, чи медіана робить дані придатними для подальших досліджень.

Однак, заповнення модою хороші, якщо не так багато пропусків у даних.

Уявімо, що даних відсутні більше сорока п'яти відсотків даних, заповнення модою у разі може призвести до значного зміщення розподілу ознаки.

Медіана та середня позбавлені такого недоліку, а вибір між методами заповнення проводиться на основі подальших оцінок у дослідженні.

### 1.1.4.3 Заповнення методами прогнозування

Метод k-найближчих сусідів – чудовий метод заповнення значень.

Метод k-найближчих сусідів використовується там, де пропущені значення ставляться заповнюються за допомогою середнього значення з K найближчих сусідів, знайдених у навчальному наборі.

Метод має кілька переваг, таких як простота реалізації та можливість працювати як з числовими, так і з категоріальними типами даних. Проте визначити кількість сусідів — k може бути непросто, оскільки це призводить до компромісу між надійністю та швидкістю. Якщо ми виберемо маленьке k, то ми матимемо швидкі обчислення, але менш надійні результати. Навпаки, якщо ми виберемо велике значення k, отримаємо більш надійне, але повільне обчислення.

Ми можемо використовувати метод k-найближчих сусідів, де пропущені значення ставляться за допомогою середнього значення з K найближчих сусідів, знайдених у навчальному наборі.

Нижче наведено графік візуалізації роботи методу, зеленим кольором показано відновлені дані.

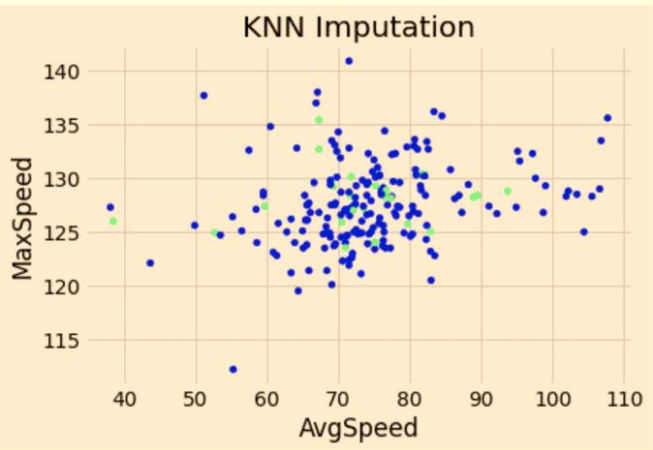


Рисунок 1.9 - Метод k-найближчих сусідів

1.1.4.4 Багатовимірне заповнення за допомогою ланцюгового рівняння

Інтерполяція за допомогою ланцюгових рівнянь (MICE) дозволяє передбачати значення для пропущених даних за пов'язаними рівняннями регресії, які побудовані на ознаках із заповненими даними.

MICE передбачає, що відсутні дані є випадковою втратою (MAR), що означає, що ймовірність втрати значення залежить тільки від значення, що спостерігається значення і може використовуватися для його прогнозування.

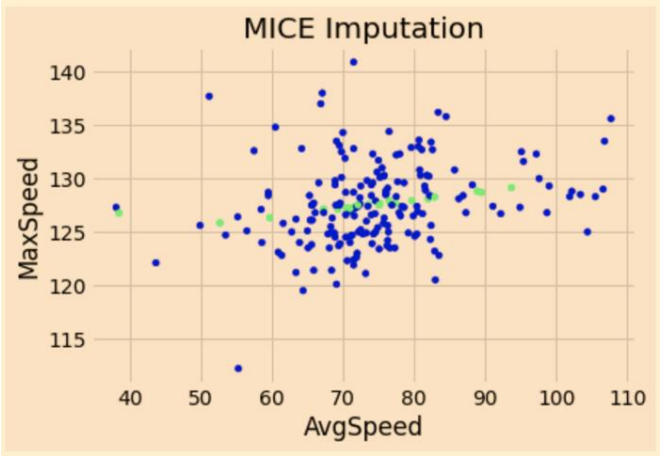


Рисунок 1.10 – Заповнення пропусків за допомогою MICE

1.1.4.5 Заповнення даних часу за допомогою випадкового лісу

Випадковий ліс для заповнювання пропущених даних – один із найефективніших алгоритмів відновлення пропущених даних. Існує безліч модифікацій стандартного методу. Не вдаючись до тонкощів технічних реалізацій, можна навести зразковий псевдо-алгоритм виконання [10].

Крок 1: Спочатку пропущені значення заповнюються середнім значенням відповідних стовпців для неперервних та найчастіших даних для категоріальних даних.

Крок 2: Набір даних розділений на дві частини: навчальні дані, що складаються з змінних, що спостерігаються, та інші відсутні дані, що використовуються для прогнозування. Потім ці набори для навчання та прогнозування передаються в Random Forest, а потім прогнозовані дані ставляться у відповідних місцях. Після визначення всіх значень завершується одна ітерація.

Крок 3: Описаний вище крок повторюється, доки не буде досягнуто умови зупинки. Процес ітерації гарантує, що алгоритм працює із даними вищої якості у наступних ітераціях. Процес триває доти, доки сума квадратів різниць між поточним і попереднім настановою не збільшиться або не буде досягнуто певної межі ітерацій. Зазвичай, потрібно до 10 ітерацій, щоб добре атрибутувати дані.

Переваги Missing Forest.

Використання Missing Forest для поставлення має кілька переваг.

• Може працювати як числовими, так і категоріальними даними.

• Missing Forest може обробляти викиди, тому немає потреби в масштабуванні функцій, проте краще обробити їх заздалегідь.

• Випадкові ліси мають вбудований вибір ознак, що робить їх стійкими до зашумлених даних.

• Він може обробляти нелінійність даних.

Недоліки Missing Forest

• Основним недоліком є його технічна витратність

### 1.1.6 Проблема мультиколінеарності

Проблема мультиколлінеарності може виникати на різних етапах аналізу даних, тому боротьба із цією проблемою має починатися з раннього аналізу. Природно, що для регресійного аналізу даних нам потрібні

незалежні передикторні змінні, тобто стовпці нашої матриці даних мають бути лінійно незалежними.

В основному, мультиколлінеарність заважає правильній оцінці значущості факторів. І її прийнято позбавлятися. Неможливо повністю позбавитися залежності даних, однак, існують методи зменшення такої залежності.

Дослідники не бояться видалення деяких векторів стовпців з матриці і як слідство цього – втрату цінної інформації. Все це тому, що проблема мультиколлінеарності тісно пов'язана з поняттям лінійної залежності та інтерпретації правила лінійної залежності в алгебраїчному вигляді.

1.1.6.1 Виявлення та усунення мультиколінеарності

Найефективнішим з погляду візуального аналізу є кореляційна матриця. Матриця покаже, які фактори мають між собою тісний зв'язок, а які – ні. Варто пам'ятати, що кореляція між чинниками (ознаками) не завжди має справжнє значення залежності.

Іноді дані бувають помилково корельованими, і боротьба вже з цим явищем відбувається на етапах логічного огляду даних аналітиком.

Одним з найефективніших методів виявлення і оцінки даних після усунення мультиколінеарності в даних є знаходження коефіцієнта інфляції дисперсії (1.1):

$$VIF = \frac{1}{1-R^2} \tag{1.1}$$

де $R^2$ – коефіцієнт детермінації.

Розрахунок є рекурсивним, у тому плані що, ознака, навіть за відсутності явного лінійного зв'язку з будь-яким іншим чинником при візуальному огляді, вибирається як відгук системи. Він будується як рівняння регресії (навчається регресійна модель) по предикторным змінним, що залишилися.

Обчислене значення порівнюється з пороговим, для тому, щоб прийняти рішення про видалення змінної. Порогове значення - гіперпараметр, що налаштовується дослідником. Зазвичай вони налаштовуються за допомогою машинних методів по сітці.

Однак, оскільки ми використовуємо оцінку $R^{\wedge 2}$ в розрахунках, слід пам'ятати, що і вона може бути помилковою і не завжди відображати якість моделі.

Найефективнішим з погляду ресурсів є методи регуляризації. Найпопулярнішими методами регуляризації є L1 (lasso) (1.2) та L2 (ridge) (1.3) регуляризація.

$$min \, || Xw — y ||^2 + z|| w || \tag{1.2}$$

$$min \, || Xw — y ||^2 + z|| w ||^2 \tag{1.3}$$

де w – вектор ваг, X - вектор ознак, y – таргетна змінна.

Відмінності між L1 та L2 регуляризацією:

L1 є негладким – перевагою є внутрішній відбір ознак. Фактично, можна назвати, що регуляризация – це обмеження, накладені на ваги ознак у регресійному рівнянні. Через L1 регуляризації ваги при мало значних ознаках обнуляются. Це і дозволяє зменшити розмірність матриці даних і позбавиться мультиколлінеарності.

На формулі (1.3) видно квадрат при другому члені – функція гладка, отже, ваги при малозначимих ознаках – зменшаься, але не обтуляться, відбудеться коригування їх значимості.

### 1.1.7 Проблема перенавчання

Формулювання проблеми перенавчання проста – обрана та навчена модель добре передбачає значення на навчальних даних, але погано працює з новими даними.

Коротко проблему можна візуалізувати з прикладу простої задачі регресії.

У першому випадку модель ідеально описує дані та передбачає значення на них. Але помиляється на нових.

У другому випадку вжито заходів щодо уникнення перенавчання, а саме використання поділу вибірки у співвідношенні 80% на 20%, тобто на навчальну та тестову, відповідно «рис. 10».

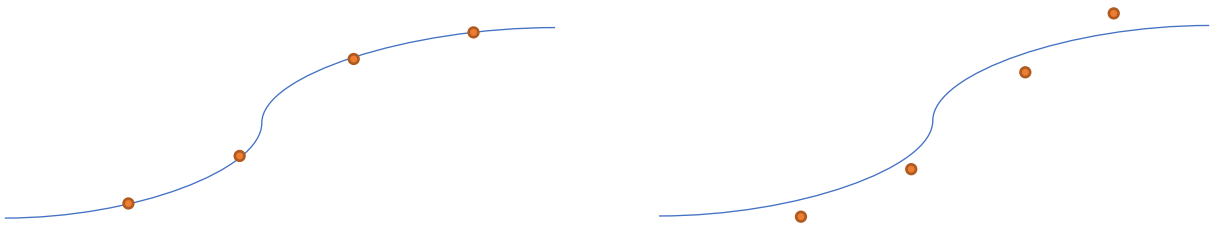

Рисунок 1.10 - Різниця між моделями

#### 1.1.7.1 Способи вирішення проблеми перенавчання та знаходження перенавчання в моделях

Як зрозуміти, що модель перенавчена?

Зазвичай на це вказують занадто великі ваги при змінних у рівнянні регресії.

Боротьба з перенавчанням починається ранніх етапах. Вибір методу боротьби починається з оцінки обсягу наданої вибірки даних, якщо обсяг даних досить великий на думку аналітика (оцінка проводиться з урахуванням поставленої задачі), можна розділити вибірку на навчальну і тестову.

Зазвичай співвідношення вибирається 80% на 20%, навчальну та тестову вибірку відповідно [4].

Цей метод добре працює для великої вибірки даних, але погано для середніх та малих вибірок.

Рішенням для подібних випадків є крос-валідація. Ідея крос-валідації проста:

1. Навчальна вибірка розбивається на k однакових за обсягом частин, що не перетинаються;

2. Присутній ітераційний підхід. На кожній ітерації відбувається таке:

2.1. Модель навчається на k-1 частини навчальної вибірки;

2.2. Модель тестується на частині навчальної вибірки, яка не брала участі у навчанні.

3. Кожна з частин k один раз використовується для тестування.

Крос-валідація – це потужний інструмент боротьби з перенавчанням.

Занадто маленькі вибірки не рекомендовано використовувати для навчання моделей, оскільки адекватність та точність таких моделей залишаться під питанням. І виникне проблема недонавчання.

Часто у технічних реалізаціях крос-валідація використовується разом із навчанням моделей. Після цього з моделей вибирається найкраща згідно з заданою заздалегідь оцінкою моделі.

### 1.1.8 Типи завдань машинного навчання, тонкощі налаштування моделей

Навчання з учителем – один із методів машинного навчання. Процес навчання є контрольованим, процес навчання виконується під видимою розміткою змінних спостереження чи відгуку, у навчанні без вчителя відгук недоступний.

Навчання з учителем зводиться до двох основних завдань. Завдання класифікації та завдання регресії, що визначаються щодо таргетних змінних.

Таргетними змінними здачі класифікації (як парної, так і множинної) є категоріальні змінні. А таргетні значення задачі регресії – безперервні змінні

1.1.8.1 Завдання регресії. Визначення точності задачі регресії

Найважливішим етапом визначення того, наскільки модель точна є вибір цільового функціоналу моделі. Зазвичай функціонал мінімізують, тому він має бути гладким (гладкою функцією) [2].

Звичайно, можна вибрати і не гладкий функціонал, наприклад порогова функція в задачі класифікації. Однак такий функціонал складно мінімізувати. Для цього зазвичай застосовуються методи апроксимації функції до того виду, з яким можуть працювати способи мінімізації функціоналу помилки, один з таких способів – стохастичний градієнтний спуск.

Рівняння регресії

$$a(x) = \langle w, x \rangle \tag{1.4}$$

де w – вектор ваг, x – вектор ознак.

Зовнішній вигляд функціоналу:

$$Q(a, X) = \frac{1}{l} \sum_{i=1}^{l} (a(x_i) - y_i)^2 \tag{1.5}$$

де a – модель, x – вектор ознак.

Мінімізація з використанням стохастичного градієнтного спуску:

$$w^t = w^{l-1} - \eta_t \nabla Q(w^{t-1} X) \tag{1.6}$$

де w – вектор ваг, Q – функціонал, $w^{t-1}$ – попередній вектор ваг.

1.1.8.2 Перехід у спрямовуючий простір. Налаштування моделі.

Перехід у спрямовуючий простір – по суті це налаштування моделі щодо нелінійних залежностей, помічених у даних.

Наприклад, поліноміальна регресія яскравий приклад переходу в простір, що спрямовує, за рахунок добудування полінома ступеня n.

Ще одним прикладом може бути логарифмізація моделі чи використання функції сигмоїди.

У такому підході налаштування моделі регресії важливо розуміти концепцію перенавчання моделей, а також чітко бачити зв'язки між даними, допомогою цього можуть бути методи розвідувального аналізу, розібрані раніше.

### 1.1.8.3 Завдання класифікації, гранична функція втрат. Визначення метрик у задачі класифікації

Завдання регресії зводиться до завдання класифікації, коли (у разі бінарної класифікації) точки поділяються гіперплощиною, таким чином, щоб у разі поділу з'явилося дві хмари точок. У такому разі відстань від точки (об'єкта) до гіперплощини – впевненість моделі у приналежності об'єкта тому чи іншому класу. Як згадувалося раніше - завдання класифікації представляє відгук системи як категоріальні дані, зазвичай це значення 0,1 або -1,1. Виходячи з категорій, позначених у даних, що надходять у модель даних - функції розподілу злегка змінюють свій вигляд.

Оціночний функціонал, метрики – оцінка якості моделі класифікації відбувається за граничною функцією втрат, така функція є негладкою, тому складно подається оцінці.

Спосіб вирішення цієї проблеми є заміна порогової функції втрат:

[M <0] => L(M), де L – заміна функції втрат.

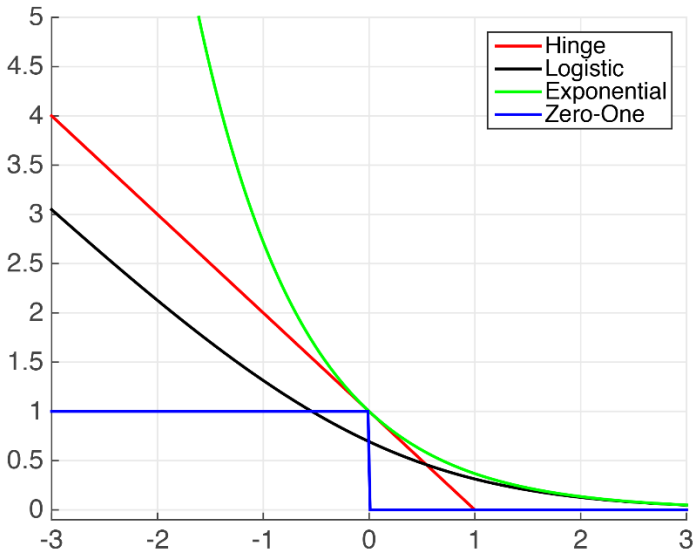


Рисунок 1.11 – Функція втрат

Після заміни функції втрат мінімізується не сам функціонал, а його верхня оцінка. Спільно з мінімізацією використовують і регуляризацію для штрафів ваг.

Для оцінки результатів моделі класифікації запроваджується поняття метрик, щоб пояснити реалізацію метрик запроваджується поняття матриці помилок.

FP - хибно-позитивний результат

FN - хибно-негативний

TP – правда-позитивний

TN – правда-негативний

Таблиця 1.1

**Таблиця до класифікатора**

| | y = 1 | y = -1 |
|---|---|---|
| a(x) = 1 | TP | FP |
| a(x) = -1 | FN | TN |

Перша і найпопулярніша метрика – точність (precision) (1.7), вона показує чи можна довіряти класифікатору, якщо він спрацьовує.

$$Precision\ (a, X) = TP/(TP+FP) \quad (1.7)$$

Друга метрика – точність (accuracy) (1.8), не дивлячись на однаковий переклад назви цієї метрики, показує вона інше. Вона демонструє кількість правильно проставлених міток класу (істинно позитивних та істинно негативних) від загальної кількості даних.

$$Accuracy = TP+TP/(TP+TN+FP+FN) \quad (1.8)$$

Наступна метрика – повнота (recall) (1.9), метрика визначає кількість істинно позитивних серед усіх міток класу, визначених як «позитивний».

$$Recall = TP/FP+FN \quad (1.9)$$

Можна використовувати змішану оцінку, таку як F-мера.

Розраховується вона так:

$$FB = (1 + B2) +((precision * recall)/(B2 * precision + recall)) \quad (1.10)$$

де B2 – коефіцієнт балансу.

Перевагою цієї метрики і те, що ми з допомогою гіперпараметра можемо проводити “важливість” оцінки, наприклад B = 2 – важливіше точність, B = 0.5 – важливіше повнота.

ROC та AUC криві

Графічні методи оцінки якості класифікації

ROC-крива це графік, що показує залежність правильно класифікованих об'єктів позитивного класу від хибно позитивно класифікованих об'єктів негативного класу. За допомогою ROC — кривої,

можна порівняти моделі, а також їх параметри для пошуку оптимальної комбінації.

Як чисельну оцінку кривої прийнято брати площу під нею AUC. Вона показує ймовірність того, що випадково обраний екземпляр негативного класу, матиме меншу ймовірність бути розпізнаним як екземпляр позитивного класу, ніж випадково обраний позитивний клас. Дуже схоже з метрикою accuracy, проте має менше переваг, ніж у accuracy, оскільки accuracy працює з ймовірностями.

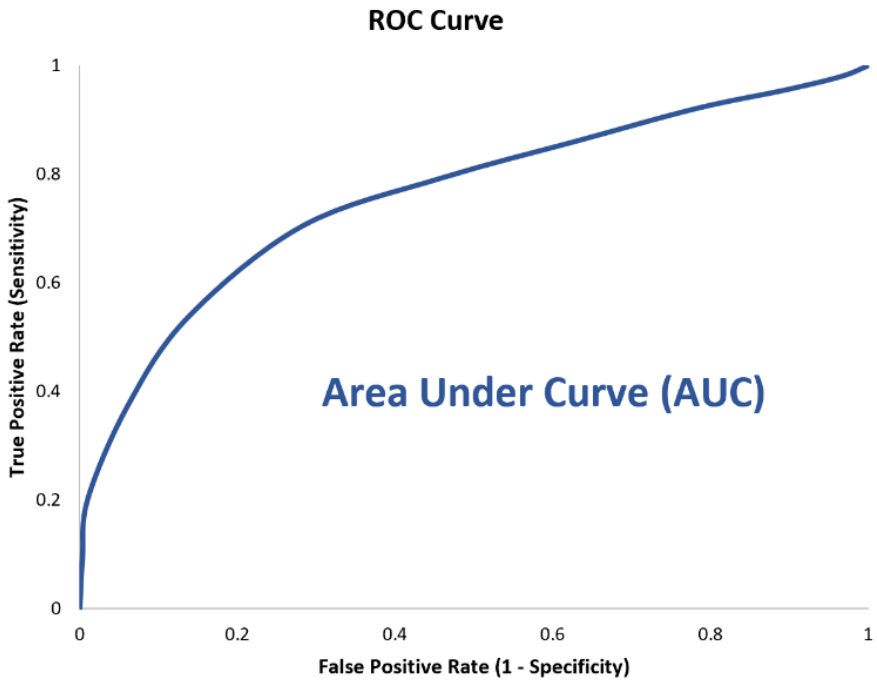


Рисунок 12 – ROC-крива

Множинна класифікація передбачає наявність міток класу більше двох, відповідно вже існуючі метрики для бінарної класифікації зазнають змін.

$$Average\ Accuracy = \frac{\sum_{i=1}^{K} Accuracy_i}{K}$$

$$Average\ Precision = \frac{\sum_{i=1}^{K} Precision_i}{K}$$

$$Average\ Recall = \frac{\sum_{i=1}^{K} Recall_i}{K}$$

$$Average\ F1-Score = \frac{\sum_{i=1}^{K} F1-Score_i}{K}$$

Рисунок 13 – метрики до бінарної класифікації,
K – кількість класів

1.1.8.4 Стратифікація та баланс класів

Стратифікація — це процес, який дозволяє підготувати дані до розбиття на тестову та навчальну вибірку із збереженням відношення класів.

Баланс класів представляє серйозну проблему завдання класифікації, особливо, коли використовуються імовірнісні оцінки.

Однак, не завжди доводиться боротися з балансом класів, оскільки ряд емпіричних досліджень показав, що випадкові ліси справляються із завданням і при дисбалансі класів із досить високою точністю оцінки.

Існує безліч методів боротьби з дисбалансом класів, багато хто з них передбачає скорочення присутності домінуючого класу в даних, або доповнення менш широко представленого класу, за рахунок випадкового дублювання класів.

Є так само і машинні методи, такі як нейромережі, які дозволяють додавати екземпляри класу, вони представлені в бібліотеці keras.

Перелік популярних методів із загальними ідеями описаними вище:

• Random Under-Sampling

• Random Over-Sampling

• Under-sampling: Tomek links

• Synthetic Minority Oversampling Technique (SMOTE),

• NearMiss

• Change the performance metric

• Penalize Algorithms (Cost-Sensitive Training)

• Change the algorithm

### 1.1.9 Дерева та випадковий ліс

#### 1.1.9.1 Дерева

Дерева є ієрархічною структурою даних, побудовану у тих логічних операцій (правил), кожен окремий підсумок операції називається листом дерева.

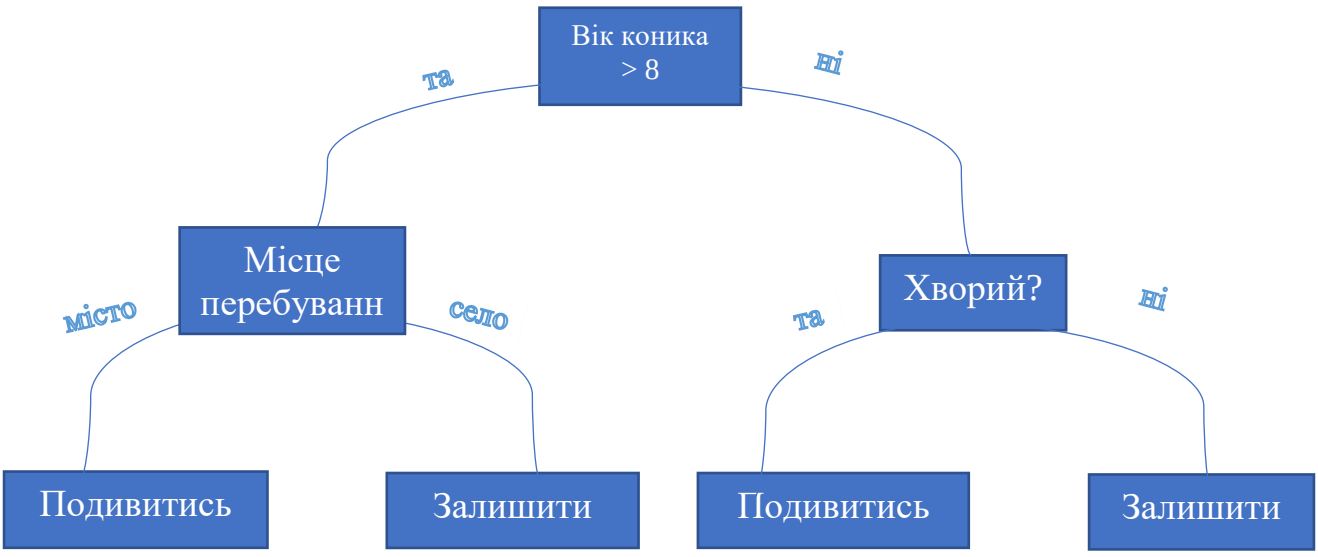


Рисунок 14 – Дерево рішень

Дерева - це та структура, яку можна використовувати з різними завданнями машинного навчання, а саме завдання з розміченими даними, такі як класифікація та регресія.

Слід сказати, що дерева легко перенавчаються, адже зрештою можна розподілити дані таким чином, щоб кожен окремий об'єкт був у своєму листі.

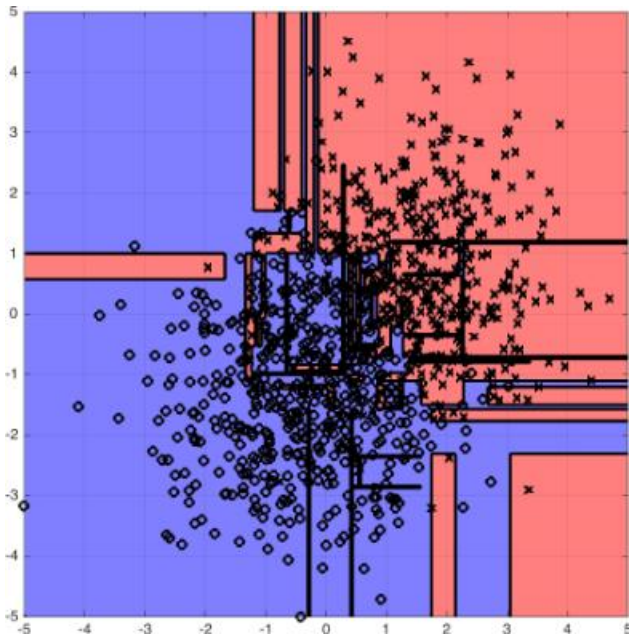

Рисунок 15 – перенавчання дерева

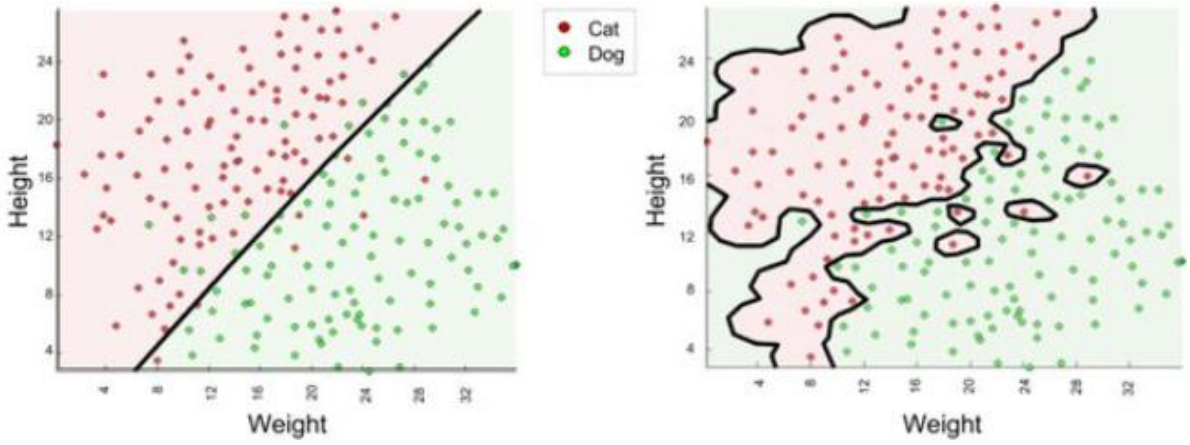


Рисунок 16 – порівняння адекватної моделе з перенавченою

Дерева мають підходи до роботи з категоріальними даними, і навіть підходи боротьби з перенавчанням, приклад - стрижка дерев, коли вибирається чіткий критерій зупинки.

У роботі не буде розібрано нюансів роботи безпосередньо з деревами, тому що сучасні дослідники даних переважно працюють з лісами – колекціями, зібраними з дерев. Однією з причин цього є зменшення можливості перенавчитися [5].

### 1.1.9.2 Випадковий ліс

Випадковий ліс є композицією з дерев. У задачі регресії їх відповіді усереднюють, у задачі класифікації приймається рішення голосуванням у більшості. Усі дерева будуються незалежно та за такою схемою[6]:

• Вибирається вибірка навчальної підвибірки розміру n – по ній будується дерево, кожне окреме дерево керується своєю вибіркою.

• Для побудови кожного нового поділу в дереві обираються свої випадкові ознаки.

• Вибираємо найкращі ознаки та розщеплення по ньому (за заздалегідь заданим критерієм). Дерево будується, як правило, до вичерпання вибірки (поки в листі не залишаться представники лише одного класу), але в сучасних реалізаціях є параметри, які обмежують висоту дерева, кількість об'єктів у листі та кількість об'єктів у підвибірці, при якому проводиться розщеплення.

Головна відмінність випадкового лісу від дерев, крім того, що перше це композиція другого, є відсутність перенавчання.

Для знаходження оптимального розбиття у вузлах вирішальних дерев використовують критерії інформативності.

Для регресії це MAE та MSE, середня абсолютна та середня квадратична помилки.

Для класифікації:

Помилка класифікації

$$\mathrm{I}(p_1 \dots p_C) = 1 - \max_i p_i \tag{1.11}$$

Частота помилок при класифікації найпотужнішим класом.

Ентропія

$$\mathrm{I}(p_1 \dots p_C) = -\sum_i p_i \ln p_i \tag{1.12}$$

Інтерпретація: міра невизначеності випадкової величини.

Критерій Джіні

$$\mathrm{I}(p_1 \dots p_C) = \sum_{k \neq k'} p_k p_{k'} = \sum_i p_i (1 - p_i) \tag{1.13}$$

Інтерпретація: вірогідність правильної класифікації, якщо передбачати класи з ймовірностями їхньої появи в цьому вузлі.

### 1.1.10 Завдання кластеризації

Завдання кластеризації – це завдання розбиття даних на групи (кластери) за схожістю даних, а об'єкти різних груп мають бути максимально різними. Кластеризація – це завдання навчання без вчителя, основними питаннями для формування завдання є: вибір міри схожості та кількості кластерів.

Загалом кластерний аналіз можна визначити такими етапами.

• Відбір об'єктів для кластеризації

• Відбір ознак та їх нормалізація, векторизація, кодування, створення нових ознак.

• Обчислення значення близькості або схожості між об'єктами щодо метрики, заданої в алгоритмі.

• Застосування методів кластерного аналізу для створення груп кластерів.

• Формування висновку, відображення результатів аналізу.

### 1.1.10.1 Заходи відстаней для задач кластеризації

Очевидним та універсальним вирішенням проблеми підготовки даних до застосування (обчислення) заходів відстаней є приведення всіх ознак у числовий вигляд, а також їх нормалізація для коректного обчислення відстаней між об'єктами. Так, зрештою, і виникає ідея збирати близькі об'єкти в кластери і ці кластери між собою відокремлювати.

Метрики для оцінки відстані мають широке застосування і дуже численними і поширеними.

Евклідова відстань, а також його квадрат:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2} \tag{1.14}$$

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2 \tag{1.15}$$

Мангеттенська метрика:

$$\rho(x,x') = \sum_{i}^{n} | x_i - x_i' | \tag{1.16}$$

Можна сказати, що вплив викидів на цю метрику значно менший, ніж у Евклідовій, оскільки відсутня операція зведення у квадрат.

Відстань Чебишева

$$\rho(x,x') = \max(| x_i - x_i' |) \tag{1.17}$$

Ступінна відстань

$$\rho(x,x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p} \tag{1.18}$$

де r та p – параметри, що визначаються користувачем. Параметр p відповідає за поступове зважування різниць за окремими координатами, параметр r відповідальний за прогресивне зважування великих відстаней між об'єктами. Якщо обидва параметри – r і p – дорівнюють двом, то ця відстань збігається з відстанню Евкліда.

Остаточний вибір метрики лежить на досліднику даних.

### 1.1.10.2 Алгоритми кластеризації

Ієрархічні алгоритми - це алгоритми, по суті, представлені у вигляді деревоподібних структур, саме дерево - дерево кластерів містить у своєму

корені вибірку, у листі найбільш дрібні кластери, після кластери групуються у великі групи.

Плоскі алгоритми, окремий випадок дерев'яних – будують лише одне розбиття об'єктів на кластери.

Чіткі алгоритми – відносять об'єкт до якогось конкретного кластера (кожен об'єкт належить одному кластеру)

Нечіткі алгоритми - на відміну від чітких ставлять у відповідність об'єкту не конкретний кластер, а ступінь ставлення об'єкта до якогось кластера (кожен об'єкт відноситься до кожного кластера з певною ймовірністю)

Виникає ароблема поєднання ієрархічних алгоритмів кластеризації, тому існує кілька метрик:

• Відстань найближчого сусіда. У цьому методі відстань між двома кластерами визначається відстанню між двома найближчими об'єктами (найближчими сусідами) у різних кластерах. У результати кластери можуть поєднуватися в ланцюжки.

• Відстань найвіддаленіших сусідів. Метод зазвичай працює дуже добре, коли об'єкти походять із окремих груп. Якщо ж кластери мають подовжену форму або їх природний тип є «ланцюжковим», цей метод непридатний.

• Незважений центроїдний метод. У цьому методі відстань між двома кластерами визначається як відстань між їхніми центрами тяжіння.

• Виважений центроїдний метод (медіана). Враховуються ваги для врахування різниці між розмірами кластерів.

Алгоритми квадратичної помилки

$$e^2(X,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i^{(j)} - c_j \|^2 \quad (1.19)$$

де $c_j$ - "Центр мас" кластера j (точка із середніми значеннями характеристик для даного кластера).

Завдання кластеризації можна як побудова оптимального розбиття об'єктів на групи. При цьому оптимальність може бути визначена як вимога до мінімізації середньоквадратичної помилки розбиття.

Алгоритми квадратичної помилки належать до типу плоских алгоритмів. Найпоширенішим алгоритмом цієї категорії є метод k-середніх. Цей алгоритм будує задану кількість кластерів, розташованих якнайдалі один від одного. Робота алгоритму поділяється на кілька етапів:

1. Випадково вибрати k точок, які є початковими центрами мас кластерів.

2. Віднести кожен об'єкт до кластера з найближчим центром мас.

3. Перерахувати «центри мас» кластерів відповідно до їхнього поточного складу.

4. Якщо критерій зупинення алгоритму не задоволений, повернутись до п. 2.

Алгоритм мінімального дерева, що покриває.

Алгоритм мінімального покриваючого дерева спочатку будує на графі мінімальне покриваюче дерево, а потім послідовно видаляє ребра з

найбільшою вагою. На малюнку зображено мінімальне дерево, що покриває, отримане для дев'яти об'єктів.

Завдання кластеризації можна як побудова оптимального розбиття об'єктів на групи. При цьому оптимальність може бути визначена як вимога до мінімізації середньоквадратичної помилки розбиття.

Алгоритми квадратичної помилки належать до типу плоских алгоритмів. Найпоширенішим алгоритмом цієї категорії є метод k-середніх. Цей алгоритм будує задану кількість кластерів, розташованих якнайдалі один від одного. Робота алгоритму поділяється на кілька етапів:

1. Випадково вибрати k точок, які є початковими центрами мас кластерів.

2. Віднести кожен об'єкт до кластера з найближчим центром мас.

3. Перерахувати «центри мас» кластерів відповідно до їхнього поточного складу.

4. Якщо критерій зупинення алгоритму не задоволений, повернутись до п. 2.

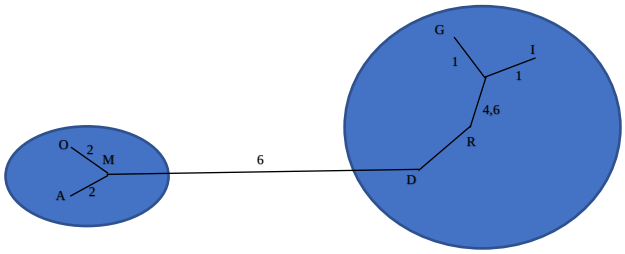


Рисунок 17 – Алгоритм мінімального дерева, що покриває.

Бустинги

Бустинг – це набір передбачень, які разом дають відповідь, вони будуються послідовно, тобто навчаються на помилках попередньої моделі.

Передбачення можуть бути різноманітними, від простої регресії до дерев (класифікаторів).

Бустинг – це техніка в якій можливо перенавчитись, тому треба дуже чутливо обирати критерій зупинки.

Одним з найпопулярніших бустингів є градієнтний бустинг.

Градієнтний бустинг – техніка для задач регресії та класифікації, яка будує слабкі передбачення, частіше за все це дерева рішень.

Мінімізація функції втрат може відбуватися з допомогою декількох функціоналів, частіше за все це середня квадратична помилка, тобто MSE – це функціонал. Мінімізація функціонала стандартна – градієнтний спуск. При використовуванні градієнтного спуску передбачення обновляються.

У цілому алгоритм бустингу можна уявити як набір таких пунктів:

- Спочатку будуються прості моделі та аналізуємо помилки;
- Визначаємо точки, які не вписуються у просту модель;
- Додаємо моделі, що обробляють складні випадки, які були виявлені на початковій моделі;
- Збираємо всі побудовані моделі, визначаючи вагу кожного предикатора.

Графічне уявлення перенавченого градієнтного бустінгу та адекватного.

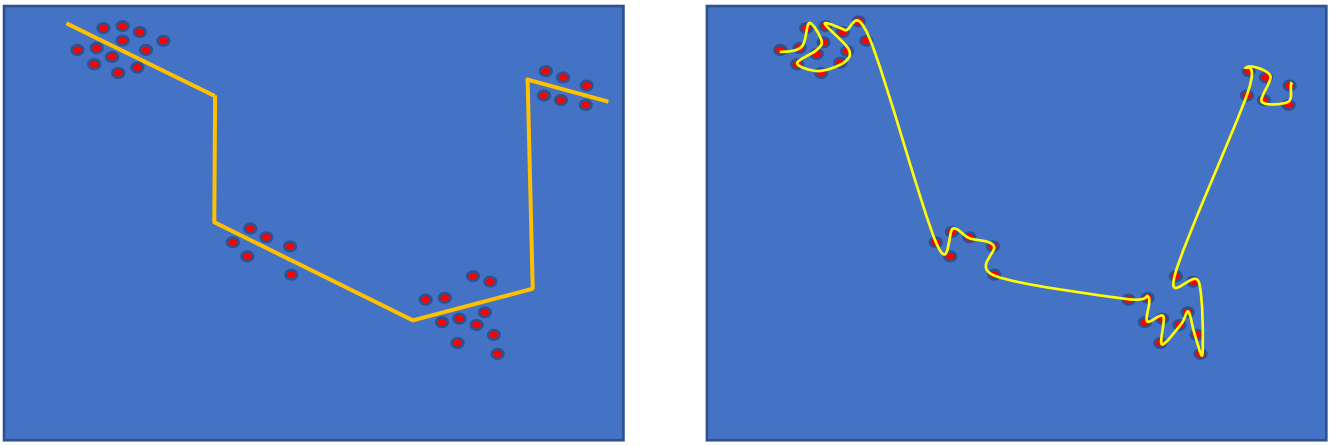

Рисунок 18 – Порівняння бустингів.

Використання алгоритмів бустингу:

- Завдання класифікації об'єктів:

Якщо дані зображення, що містять різні відомі у світі об'єкти, класифікатор може бути навчений на основі них для автоматичної класифікації об'єктів у майбутніх невідомих зображеннях. Прості класифікатори, побудовані з урахуванням деяких ознак зображення об'єкта, зазвичай виявляються малоефективними у класифікації. Використання методів бустингу для класифікації об'єктів — шлях поєднання слабких класифікаторів спеціальним чином поліпшення загальної можливості класифікації.

- Завдання ранжування видачі пошукових систем:

Завдання ранжирування видачі пошукових запитів розглянули з погляду функції втрат, яка штрафує за помилки порядку видачі, тому було зручно впровадити GBM в ранжування.

## 1.2 Висновок

У інформаційно-аналітичному розділі мною були опрацьовані підходи для попередньої обробки даних, та методи побудови простих предикатних моделей. Основними підходами, які зараз застосовують у провідному аналізу даних є ті, які опираються на обчислювальні потужності комп’ютерних систем.

# 2 СПЕЦІАЛЬНИЙ РОЗДІЛ

## 2.1 Постановка задачі за контролем лісних масивів

Природний парк Монтезіньо страждає від лісних пожеж. Задля запобігання пожеж було прийнято рішення створити модель машинного навчання, яка б вміла прогнозувати не тільки фактор виникнення пожежі, але й площу, яку пожежа займе.

Завдання: провести аналіз загальнодоступних методів аналізу даних щодо лісових пожеж. Зробити порівняння методів прогнозування в контексті прогнозування площі лісової пожежі.

### 2.1.2 Опис набору даних:

1. Title: Forest Fires
2. Sources
   Created by: Cortez and Morais (Univ. Minho) @ 2007
3. Past Usage:

   P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data.
   In Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence,
   December, 2007. (http://www.dsi.uminho.pt/~pcortez/fires.pdf)

   In the above reference, the output "area" was first transformed with a ln(x+1) function.
   Then, several Data Mining methods were applied. After fitting the models, the outputs were
   post-processed with the inverse of the ln(x+1) transform. Four different

input setups were
used. The experiments were conducted using a 10-fold (cross-validation) x 30 runs. Two
regression metrics were measured: MAD and RMSE. A Gaussian support vector machine (SVM) fed
with only 4 direct weather conditions (temp, RH, wind and rain) obtained the best MAD value:
12.71 +- 0.01 (mean and confidence interval within 95% using a t-student distribution). The
best RMSE was attained by the naive mean predictor. An analysis to the regression error curve
(REC) shows that the SVM model predicts more examples within a lower admitted error. In effect,
the SVM model predicts better small fires, which are the majority.

4. Relevant Information:

   This is a very difficult regression task. It can be used to test regression methods. Also,
   it could be used to test outlier detection methods, since it is not clear how many outliers
   are there. Yet, the number of examples of fires with a large burned area is very small.
5. Number of Instances: 517
6. Number of Attributes: 12 + output attribute

   Note: several of the attributes may be correlated, thus it makes sense to apply some sort of
   feature selection.
7. Attribute information:

   For more information, read [Cortez and Morais, 2007].

1) X - x-axis spatial coordinate within the Montesinho park map: 1 to 9
2) Y - y-axis spatial coordinate within the Montesinho park map: 2 to 9
3) month - month of the year: "jan" to "dec"
4) day - day of the week: "mon" to "sun"
5) FFMC - FFMC index from the FWI system: 18.7 to 96.20
6) DMC - DMC index from the FWI system: 1.1 to 291.3
7) DC - DC index from the FWI system: 7.9 to 860.6
8) ISI - ISI index from the FWI system: 0.0 to 56.10
9) temp - temperature in Celsius degrees: 2.2 to 33.30
10) RH - relative humidity in %: 15.0 to 100
11) wind - wind speed in km/h: 0.40 to 9.40
12) rain - outside rain in mm/m2 : 0.0 to 6.4
13) area - the burned area of the forest (in ha): 0.00 to 1090.84 (this output variable is very skewed towards 0.0, thus it may make sense to model with the logarithm transform).

### 2.1.3 Виконання попередньої обробки даних

1. Імпорт необхідних бібліотек.

```
import seaborn as sns
from scipy import stats
from sklearn.ensemble import IsolationForest
from sklearn import preprocessing
from scipy.special import logit, expit
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
```

2. За допомогою бібліотеки Pandas іпортуємо дані до структури даних типу “DataSet”

```
start_data = pd.read_csv("forestfires.csv")
```

Аналізуємо наявність пропущених даних:

```
start_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X       517 non-null    int64
 1   Y       517 non-null    int64
 2   month   517 non-null    object
 3   day     517 non-null    object
 4   FFMC    517 non-null    float64
 5   DMC     517 non-null    float64
 6   DC      517 non-null    float64
 7   ISI     517 non-null    float64
 8   temp    517 non-null    float64
 9   RH      517 non-null    int64
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(8), int64(3), object(2)
memory usage: 52.6+ KB
```

Висновок: пропущених даних немає. Як і було сказано в описі вище.

3. Перевірка наявності категоріальних даних.

Таблиця 2.1

**Перевірка наявності категоріальних даних**

| | X | Y | month | day | FFMC | DMC | DC | ISI | temp | RH | wind | rain | area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7 | 5 | mar | fri | 86.2 | 26.2 | 94.3 | 5.1 | 8.2 | 51 | 6.7 | 0.0 | 0.0 |
| **1** | 7 | 4 | oct | tue | 90.6 | 35.4 | 669.1 | 6.7 | 18.0 | 33 | 0.9 | 0.0 | 0.0 |
| **2** | 7 | 4 | oct | sat | 90.6 | 43.7 | 686.9 | 6.7 | 14.6 | 33 | 1.3 | 0.0 | 0.0 |
| **3** | 8 | 6 | mar | fri | 91.7 | 33.3 | 77.5 | 9.0 | 8.3 | 97 | 4.0 | 0.2 | 0.0 |
| **4** | 8 | 6 | mar | sun | 89.3 | 51.3 | 102.2 | 9.6 | 11.4 | 99 | 1.8 | 0.0 | 0.0 |
| **5** | 8 | 6 | aug | sun | 92.3 | 85.3 | 488.0 | 14.7 | 22.2 | 29 | 5.4 | 0.0 | 0.0 |
| **6** | 8 | 6 | aug | mon | 92.3 | 88.9 | 495.6 | 8.5 | 24.1 | 27 | 3.1 | 0.0 | 0.0 |
| **7** | 8 | 6 | aug | mon | 91.5 | 145.4 | 608.2 | 10.7 | 8.0 | 86 | 2.2 | 0.0 | 0.0 |
| **8** | 8 | 6 | sep | tue | 91.0 | 129.5 | 692.6 | 7.0 | 13.1 | 63 | 5.4 | 0.0 | 0.0 |
| **9** | 7 | 5 | sep | sat | 92.5 | 88.0 | 698.6 | 7.1 | 22.8 | 40 | 4.0 | 0.0 | 0.0 |

Висновок: Існує кілька категоріальних стовпців, а саме "month", "day".

4. Візуальний огляд даних, висунення гіпотез.

```
sorted_by_area_value = start_data.groupby(by="month").sum()
sorted_by_area_value.rename(columns= {"area": "dead_area"}, inplace=True)
sorted_by_area_value
```

Таблиця 2.2

**Площа, яка постраждала від пожеж у кожному місяці**

| | X | Y | FFMC | DMC | DC | ISI | temp | RH | wind | rain | dead_area |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **apr** | 52 | 38 | 772.1 | 143.2 | 437.0 | 48.4 | 108.4 | 422 | 42.0 | 0.0 | 80.02 |
| **aug** | 825 | 788 | 16990.0 | 28286.8 | 117958.3 | 2037.3 | 3980.2 | 8370 | 751.9 | 10.8 | 2297.99 |
| **dec** | 41 | 45 | 764.7 | 235.1 | 3161.2 | 31.2 | 40.7 | 346 | 68.8 | 0.0 | 119.97 |
| **feb** | 103 | 88 | 1658.1 | 189.5 | 1093.4 | 67.0 | 192.7 | 1114 | 75.1 | 0.0 | 125.50 |
| **jan** | 6 | 9 | 100.8 | 4.8 | 180.7 | 2.9 | 10.5 | 178 | 4.0 | 0.0 | 0.00 |
| **jul** | 167 | 147 | 2922.5 | 3532.4 | 14419.3 | 300.6 | 707.5 | 1444 | 119.5 | 0.2 | 459.83 |
| **jun** | 107 | 82 | 1520.3 | 1587.5 | 5061.0 | 200.2 | 348.4 | 767 | 70.3 | 0.0 | 99.30 |
| **mar** | 255 | 242 | 4830.0 | 1865.3 | 4100.9 | 383.8 | 706.5 | 2160 | 268.3 | 0.2 | 235.26 |
| **may** | 10 | 8 | 174.7 | 53.4 | 187.5 | 9.2 | 29.3 | 134 | 8.9 | 0.0 | 38.48 |
| **nov** | 6 | 3 | 79.5 | 3.0 | 106.7 | 1.1 | 11.8 | 31 | 4.5 | 0.0 | 0.00 |
| **oct** | 88 | 65 | 1356.8 | 621.3 | 10225.1 | 107.2 | 256.4 | 562 | 51.9 | 0.0 | 99.57 |
| **sep** | 754 | 708 | 15693.8 | 20798.7 | 126353.9 | 1475.3 | 3373.3 | 7369 | 611.9 | 0.0 | 3086.13 |

Таблиця показує кількість площі постраждалої від пожеж кожного місяця. Очевидно, що в Jan і Nov пожеж не було, це дає нам можливість припустити, що ці місяці не несуть корисної інформації для моделі, тому що протягом них не було зафіксовано жодного загоряння.

Видалення "jan", "nov".

```
forest_fires_data = pd.DataFrame()
forest_fires_data = start_data.loc[(start_data["month"] != "nov") &
                                   (start_data["month"] != "jan")].copy(deep=True)
forest_fires_data.groupby(by="month").sum()
forest_fires_data.reset_index(inplace=True)
```

Візуалізуємо місяці, що залишилися, і площа пожеж на точковому графіку.

```
scat = sns.scatterplot(forest_fires_data.month, forest_fires_data.area, s=15**2, c=["red"])
scat.set(title = "Scatter Plot of Area and Month",
        xlabel = "Month", ylabel = "Area");
```

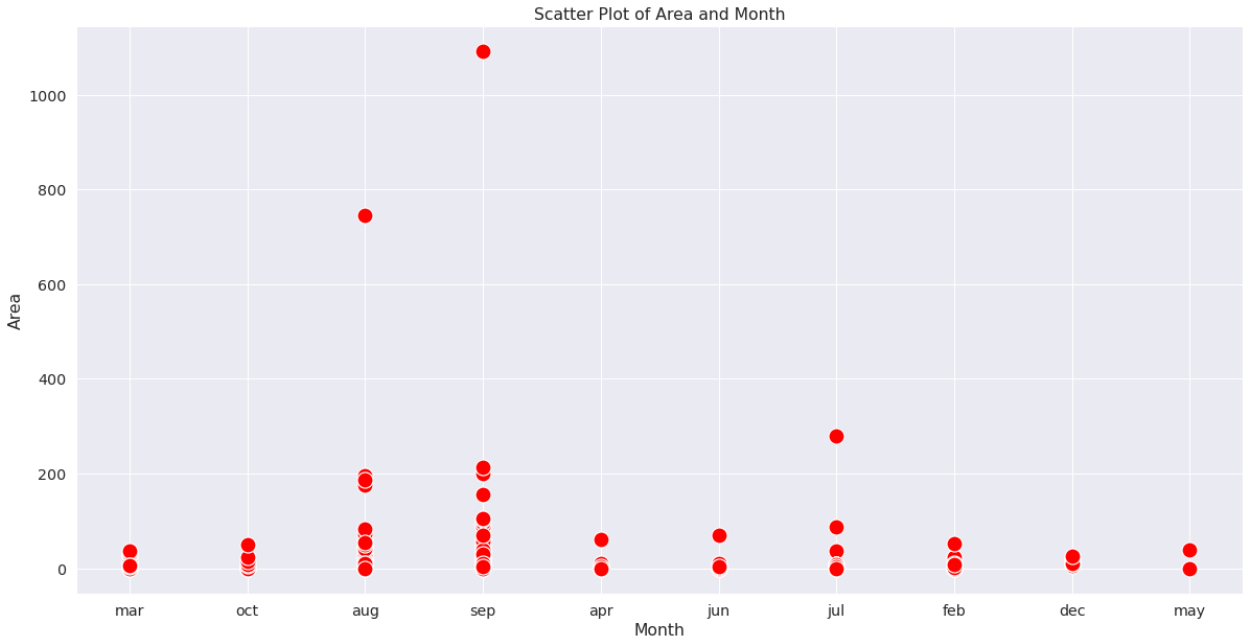


Рисунок 2.1 – Площа займання за кожним місяцем, кількість займань

Цей графік дозволить зробити одразу кілька припущень.

Видно, що дані про, а саме площу загоряння, знаходиться близько до нуля.

Також можна помітити явні викиди у aug, sep, jul. На графіку не зрозуміло, чи одна точка або кілька точок просто наклалися разом, тому буде зроблено обережне припущення про те, що викиди знаходяться в районі aug (700-800), sep (>1000), jun (320-350), оскільки точні значення нам невідомі.

Перевірити цю гіпотезу точно можна буде за допомогою гістограм.

Графік демонструє температуру повітря за місяцями.

```
plt.rcParams['figure.figsize'] = [20, 10]
sns.set(style = "darkgrid", font_scale = 1.3)
month_temp = sns.barplot(x = 'month', y = 'temp', data = forest_fires_data,
                         order = ['jan', 'feb', 'mar', 'apr', 'may', 'jun',
                                  'jul', 'aug', 'sep', 'oct', 'nov', 'dec'],
                         palette = "rocket",);
month_temp.set(title = "Month Vs Temp Barplot",
               xlabel = "Months", ylabel = "Temperature",);
```

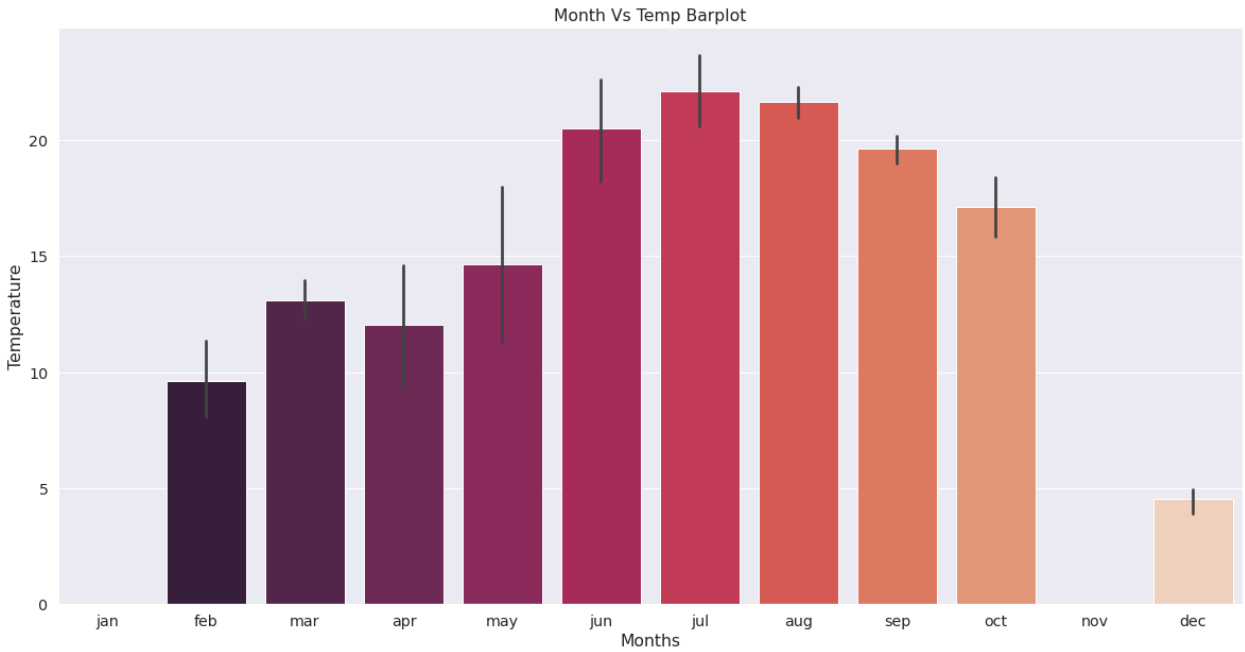


Рисунок 2.2 – Температура повітря за місяцями

Місяці, в які не було зафіксовано пожеж, виявилися найхолоднішими, температура в них в районі нуля або нижче, це додаткова причина виключити їх з робочої вибірки.

Так як це складна задача регресії з явним недоліком даних.

Питання про нестачу даних вводиться мною як припущення, підстави для цього припущення такі: безліч цільової змінної даних міститься в області нуля, що ускладнює роботу з ними, так як розподіли виходять занадто зміщеними це буде показано в подальшому на гістограмах.

5. Огляд даних про наявність лінійного зв'язку з цільовою ознакою.

Явного лінійного зв'язку між змінними та цільовою змінною не помічено.

Очевидно, що існують складніші зв'язки, тобто у майбутньому виборі моделі пріоритет буде відданий дерев'яним моделям, а саме регресору на основі випадкового лісу.

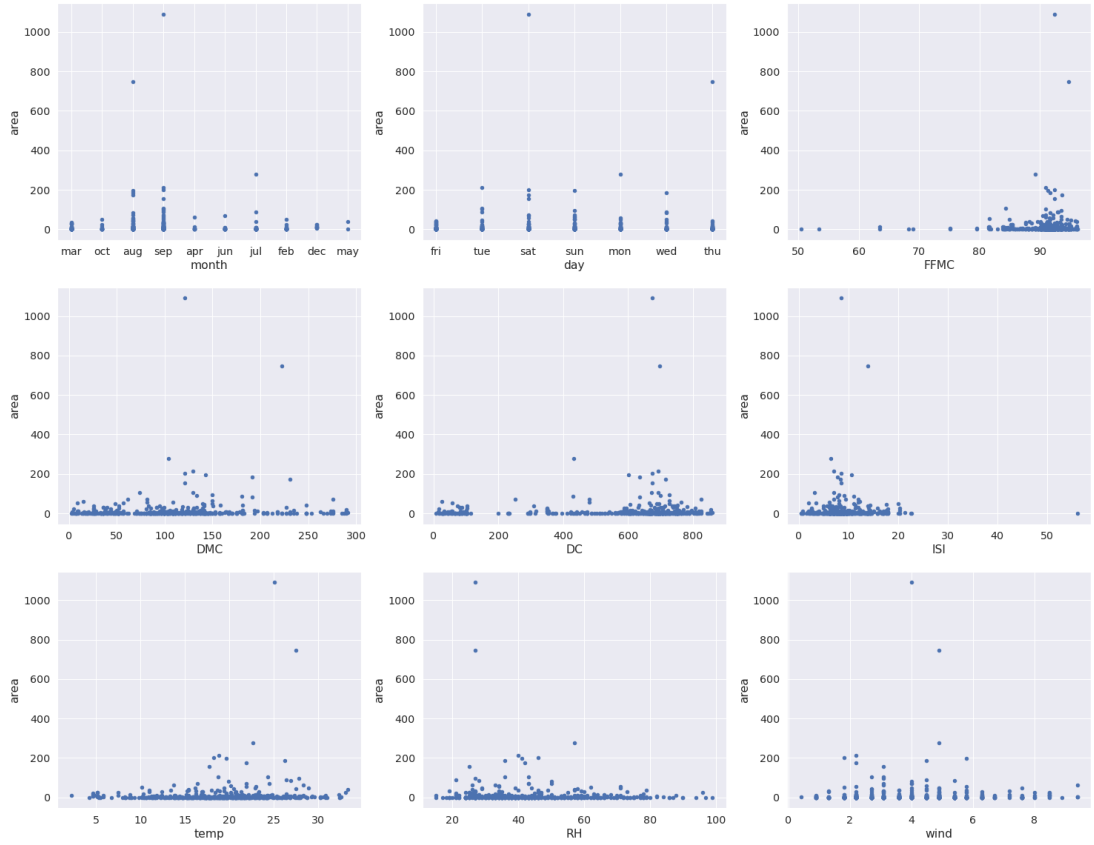


Рисунок 2.3 - Огляд даних про наявність лінійного зв'язку з цільовою ознакою

6. Візуальний огляд даних, перевірка на наявність викидів, нетипових значень, перевірка на зміщення даних.

```
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(30,10))
colors = ["DarkOrange", "RebeccaPurple", "BlueViolet", "DarkViolet",
          "DarkOrchid", "DarkMagenta", "Purple",
          "Indigo", "SlateBlue", "Black", "Blue", "Green", "Red"]
merged_axes = list(itertools.chain(*axes))
for ax, column_name, random_color in zip(merged_axes,
forest_fires_data.columns, colors):
    ax.grid(color='gray', fillstyle="bottom", linestyle='dashed')
    ax.hist(forest_fires_data[column_name], density=True,
        color=random_color, bins=10, alpha=0.8, linewidth=0.5,
edgecolor="white")
    ax.set_title(column_name)
fig.tight_layout()
```

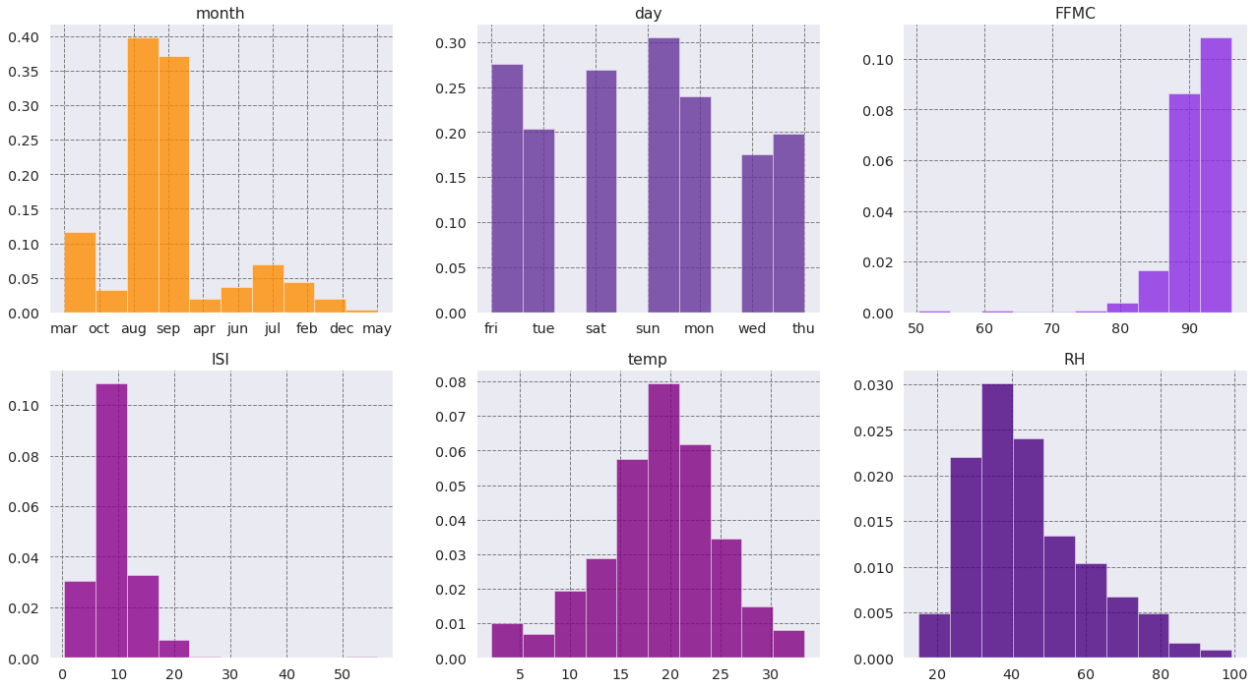


Рисунок 2.4 – Гістограми основних ознак

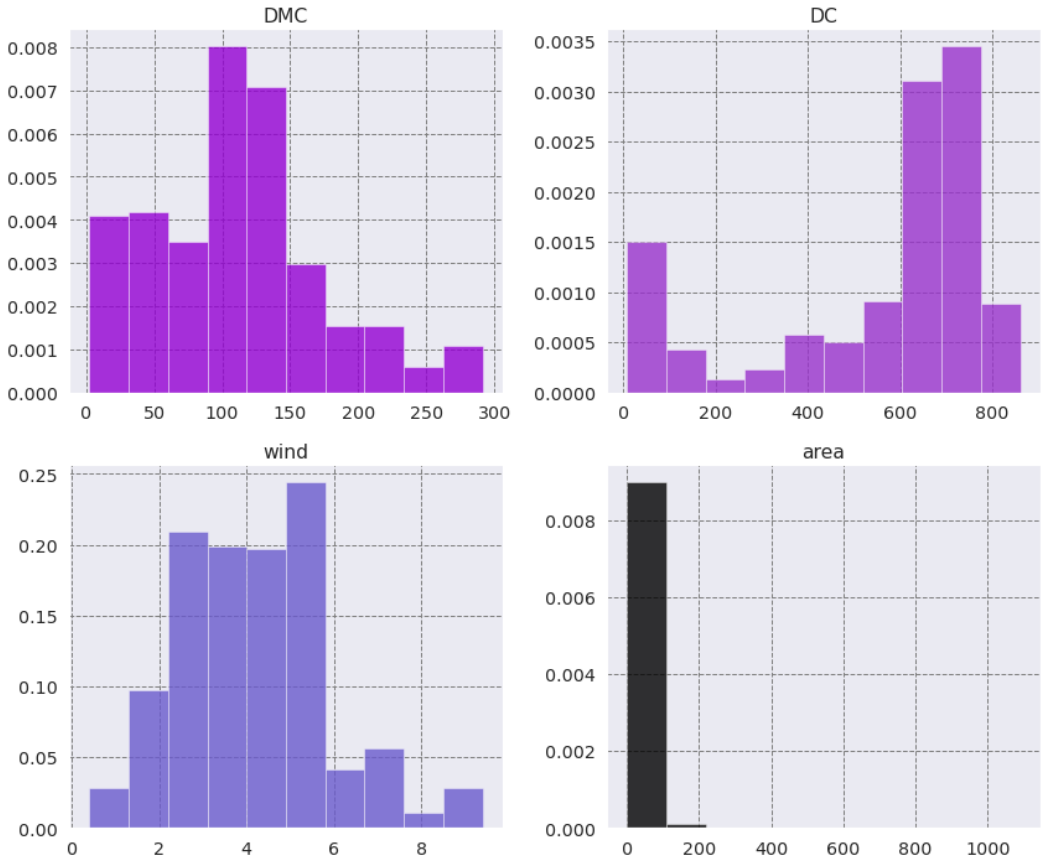


Рисунок 2.5 – Гістограми основних ознак

Видно сильну зміщеність вліво, у бік нуля, як було зазначено раніше.

Деякі розподіли мають важкі хвости, так само видно і викиди, у випадку цільової змінної не так чітко, але ми вже зробили висновок, що ймовірність їх велика і надалі будуть використані методи визначення та видалення таких об'єктів.

Кореляційна матриця дозволить упевнитися у відсутності лінійного зв'язку між ознаками, а також зробити припущення про наявність мультиколінеарності.

```
corr_matrix = forest_fires_data.loc[0:, :].corr()
corr_matrix.style.background_gradient(cmap="coolwarm")
```

Таблиця 2.3

**Кореляційна матриця**

| | FFMC | DMC | DC | ISI | temp | RH | wind | area |
|---|---|---|---|---|---|---|---|---|
| **FFMC** | 1.000000 | 0.407270 | 0.346815 | 0.585603 | 0.449357 | -0.265140 | -0.090395 | 0.041440 |
| **DMC** | 0.407270 | 1.000000 | 0.676566 | 0.293383 | 0.459527 | 0.091509 | -0.113260 | 0.071608 |
| **DC** | 0.346815 | 0.676566 | 1.000000 | 0.215565 | 0.485846 | -0.023785 | -0.212662 | 0.047714 |
| **ISI** | 0.585603 | 0.293383 | 0.215565 | 1.000000 | 0.382868 | -0.119653 | 0.100879 | 0.006322 |
| **temp** | 0.449357 | 0.459527 | 0.485846 | 0.382868 | 1.000000 | -0.518943 | -0.240527 | 0.096714 |
| **RH** | -0.265140 | 0.091509 | -0.023785 | -0.119653 | -0.518943 | 1.000000 | 0.085090 | -0.074909 |
| **wind** | -0.090395 | -0.113260 | -0.212662 | 0.100879 | -0.240527 | 0.085090 | 1.000000 | 0.011576 |
| **area** | 0.041440 | 0.071608 | 0.047714 | 0.006322 | 0.096714 | -0.074909 | 0.011576 | 1.000000 |

Тісних лінійних зв'язків немає, оскільки відсутня сильна кореляція.

Припущення: мультиколінеарності немає.

У цьому аналізі не проводяться статистичні тести спрямовані на знаходження мультиколлінеарності, тому що вони трудомісткі і ми маємо можливість використовувати регуляризацію, яка дозволяє як і зберегти деякі змінні, сильно штрафуючи їх за ваги, так і обнулити деякі з них.

Відбір аномальних значень.

```
plt.figure(figsize = (15,8))
ax = sns.boxplot(data=forest_fires_data, order=list(forest_fires_data.loc[:, "FFMC":"wind"].columns),
                 orient = "h")
```

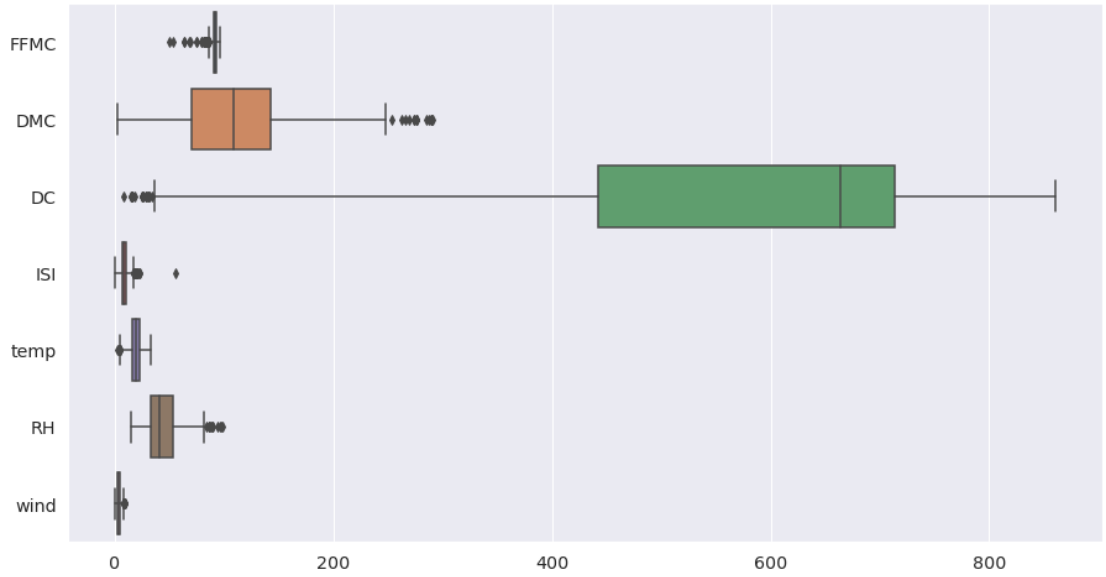


Рисунок 2.6 – Коробчасті діаграми

```
isolation_forest_model = IsolationForest(n_estimators=100,
max_samples='auto',  contamination=float(0.1),max_features=1.0)
names = forest_fires_data.columns[4:]
print(forest_fires_data.columns[4:])
outliers_fires_data_encoer = pd.DataFrame()
for name in names:
    isolation_forest_model.fit(forest_fires_data[[name]])
    outliers_fires_data_encoer[f'anomaly_score_{name}']
isolation_forest_model.predict(forest_fires_data[[name]])
```

Використовуємо модель ізолюючого лісу для знаходження та видалення викидів.

```
indexes = []
for index in list(forest_fires_data.index):
    if (int(outliers_fires_data_encoer["anomaly_score_area"].iloc[index]) == -1) &\
    (int(forest_fires_data["area"].iloc[index]) >= 200):
        indexes.append(index)
```

Видалятимуться значення вище відмітки 200 щодо цільової змінної, такий поріг був обраний виходячи з аналізу візуалізації.

Підсумок: Вилучено 6 значень. Нове верхнє значення у вибірці.

```
forest_fires_data["area"].max()
196.48
```

8. Кодування категоріальних даних та нормалізація.

```
encoder = ce.OneHotEncoder(cols=['day', "month"],return_df=True)
fires_data_encoder = encoder.fit_transform(forest_fires_data)
fires_data_encoder.head()
```

Нормалізація.

```
data = preprocessing.normalize(fires_data_encoder.iloc[:, :-1])
scaled_fires_data = pd.DataFrame(
data, columns=fires_data_encoder.columns[:-1])
scaled_fires_data["area"] = forest_fires_data["area"].replace(0, 1)
scaled_fires_data.dropna(inplace=True)
scaled_fires_data
```

```
plt.figure(figsize = (15,8))
ax = sns.boxplot(data=scaled_fires_data,
                 order=list(scaled_fires_data.loc[:, "FFMC":"wind"].columns),
                orient = "h")
```

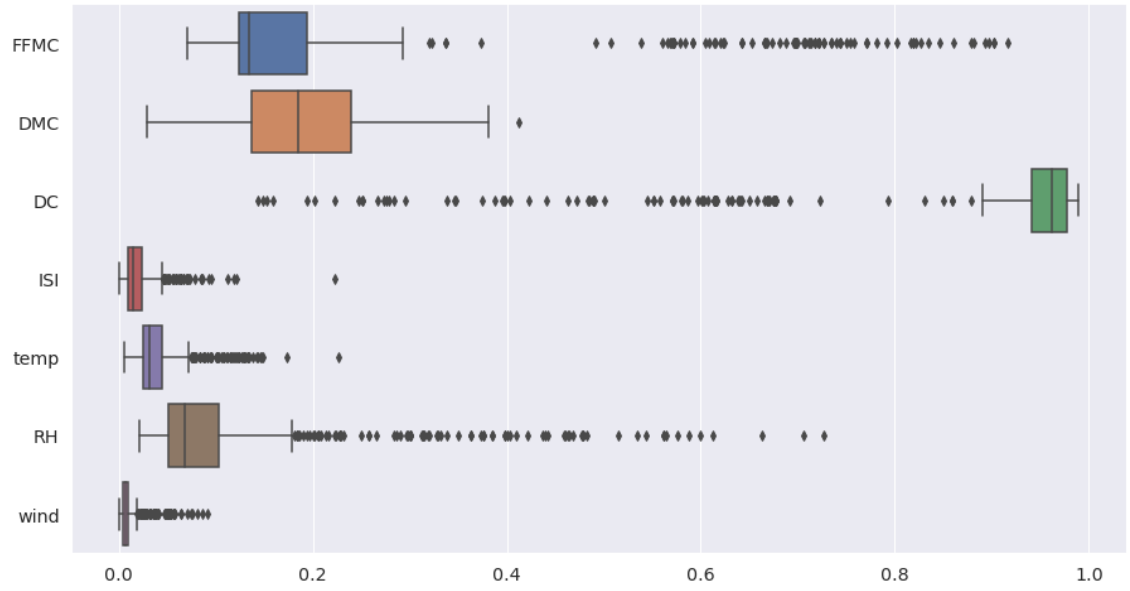


Рисунок 2.7 – Коробчасті діаграми після трансформації

Нормалізація даних за допомогою логарифму.

```
log_transform = np.log(scaled_fires_data.loc[:, "FFMC":"wind"].replace(0, 1))
simple_regression_data = pd.concat([scaled_fires_data.loc[:, :"day_7"],
                                    log_transform, np.log(scaled_fires_data["area"])], axis=1)
simple_regression_data.head()
```

Графіки гістограм після трансформації.

```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(30,10))
colors = ["DarkOrange", "RebeccaPurple", "BlueViolet", "DarkViolet",
          "DarkOrchid", "DarkMagenta", "Purple",
          "Indigo", "SlateBlue", "Black", "Blue", "Green", "Red"]
merged_axes = list(itertools.chain(*axes))
for ax, column_name, random_color in zip(merged_axes, log_transform.columns, colors):
    ax.set_axisbelow(True)
    ax.grid(color='gray', fillstyle="bottom", linestyle='dashed')
    ax.hist(log_transform[column_name], bins=10, alpha=0.8, density=True,
            color=random_color, linewidth=0.5, edgecolor="white")
    ax.set_title(column_name)
fig.tight_layout()
```

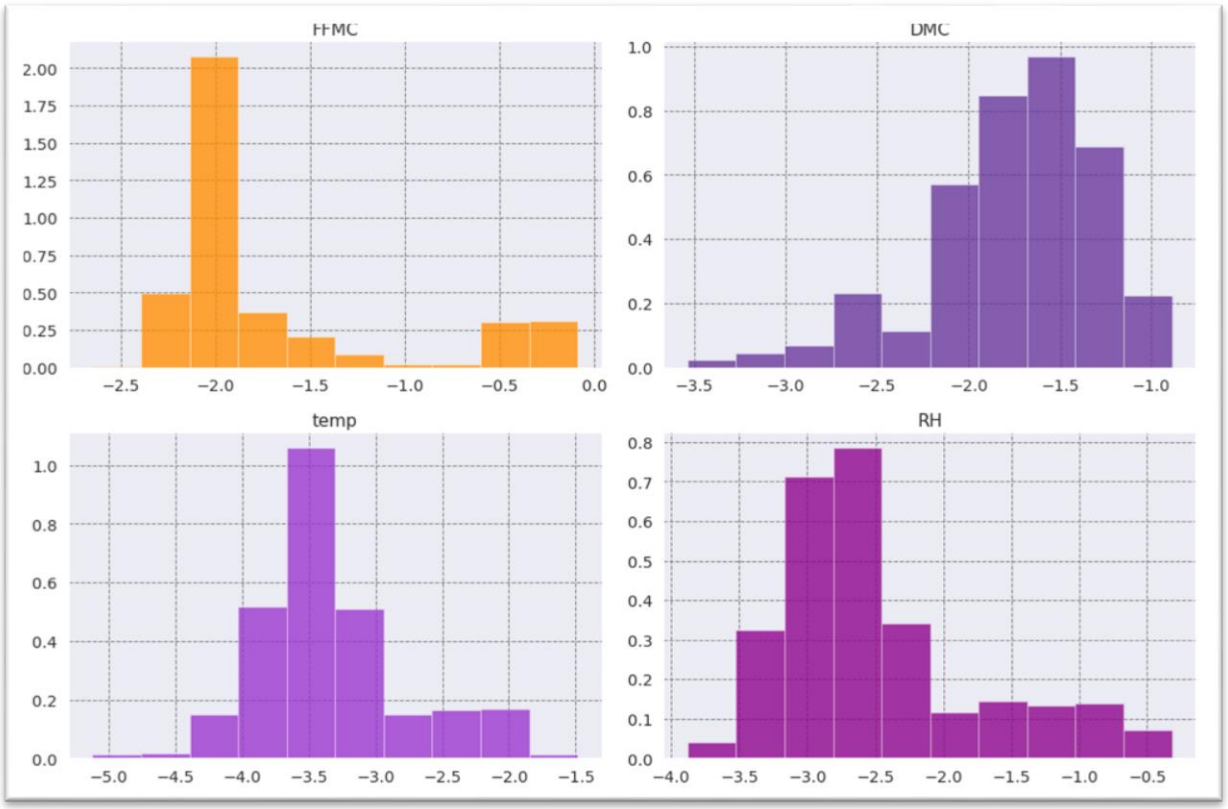


Рисунок 2.8 – Гістограми після трансформації

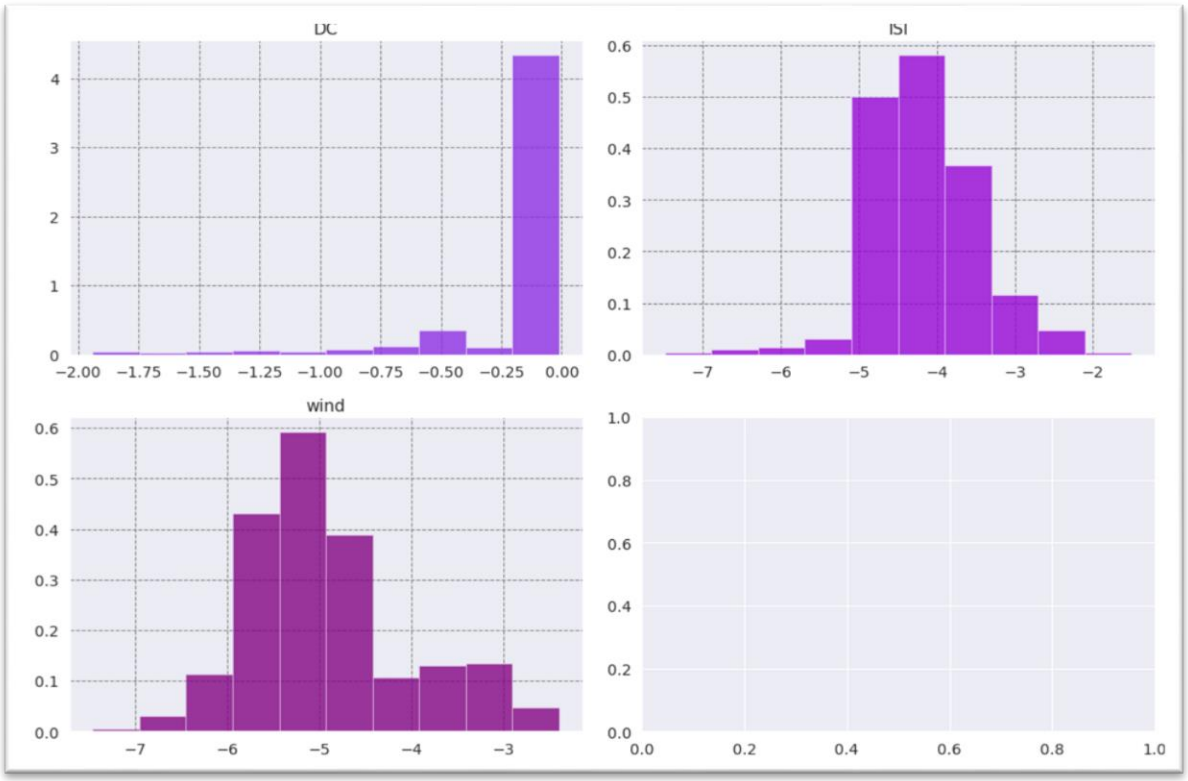


Рисунок 2.9 – Гістограми після трансформації

### 2.1.4 Побудова найпростіших регресійних моделей

#### 2.1.4.1 Побудова регресійної моделі без регулярізації але з застосуванням крос-валідації

```
target = simple_regression_data['area']
features = simple_regression_data.drop(columns = 'area')

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.15,
random_state = 196)

regressor = lr()
scores = cross_validate(regressor, X_train, y_train, cv=5,
                        scoring="neg_mean_squared_error",
                        return_train_score=True)
np.abs(scores["test_score"].max())
```

```
1.441080592713257
```

#### 2.1.4.2 Побудова моделі з застосуванням регулярізації Lasso

```
regressor = LassoCV(cv=7, random_state=0).fit(X_train, y_train)
print(regressor.score(X_train, y_train))
mean_squared_error(regressor.predict(X_test), y_test)
```

```
0.0
1.9438163213182733
```

```
poly = PolynomialFeatures(4)
poly_X_train = poly.fit_transform(X_train)
poly_X_test = poly.fit_transform(X_test)
```

```
model_2 = LinearRegression()
model_2.fit(poly_X_train, y_train)
predictions_poly = model_2.predict(poly_X_test)
print ("Mean Squared Error : ", mean_squared_error(y_test, predictions_poly))
print ("r2 Score : ", r2_score(y_test, predictions_poly))
```

```
Mean Squared Error :  1151206.9928188494
r2 Score :  -601376.9346801438
```

2.1.4.3 Побудова моделі з застосуванням регулярізації Ridge

```
regressor = RidgeCV(cv=7).fit(training_X, training_Y)
print(regressor.score(training_X, training_Y))
mean_squared_error(tasting_Y, regressor.predict(tasting_X))
```

```
r2 Score: 0.015827096459779755
Mean Squared Error : 2.1171549383915496
```

В якості скорінгу (оцінкаи моделі) використовувався NMSE та MSE.

2.1.4.4 Побудова регрессору вибадковий ліс.

```
regr = RandomForestRegressor(max_depth=10, random_state=0,
                             min_samples_split=2, min_samples_leaf=1, bootstrap=True)
model = regr.fit(training_X, training_Y)
print(mean_squared_error(tasting_Y, regr.predict(tasting_X)))
print(regr.score(training_X, training_Y))
model.get_params
```

```
2.1947063479523163
0.5718321801785462
```

2.1.5 Зведення задачі лінійної регресії до задачі класифікації.

Зчитування даних.

```
fires_data = pd.read_csv("forestfires.csv")
```

Кодування категоріальних даних.

```
encoder = ce.OneHotEncoder(cols=['day', "month"],return_df=True)
fires_data_encoder = encoder.fit_transform(fires_data)
fires_data_encoder.head()
```

Встановлення міток для класифікації, тобто синтез нової змінної.

```
fires_data_encoder["area"].replace(0, 1)
replace_val = {
            0:(fires_data_encoder["area"] >= 0) & (fires_data_encoder["area"] <= 2),
            1:(fires_data_encoder["area"] > 2) & (fires_data_encoder["area"] <= 400),
            3:(fires_data_encoder["area"] > 400),
           }
for key in replace_val.keys():
   fires_data_encoder["area"].loc[replace_val[key]] = key

fiers_data_simple_cla = fires_data_encoder.loc[fires_data_encoder["area"] != 3]
```

Демонстрація балансу класів.

```
fiers_data_simple_cla["area"].hist()
```

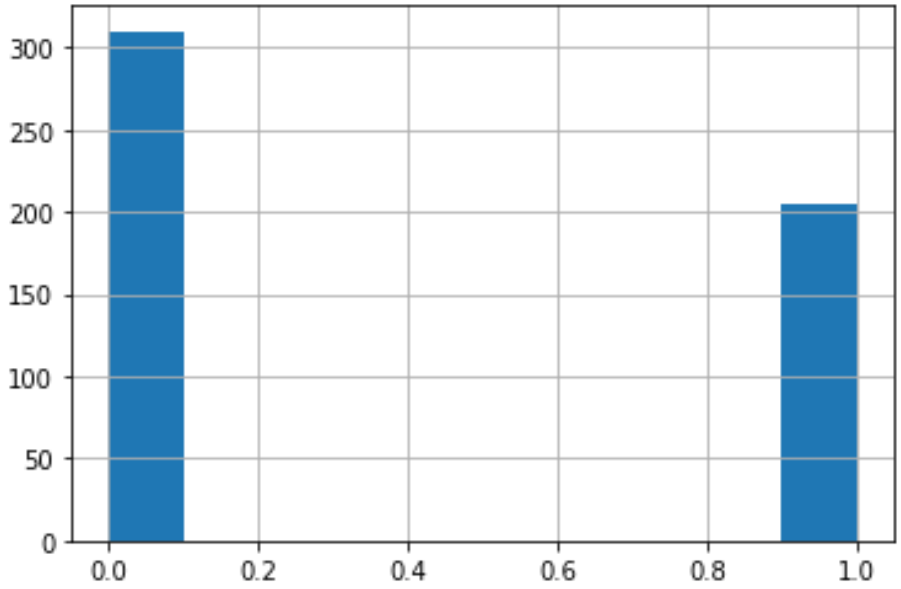


Рисунок 2.10 - Демонстрація балансу класів

Класи мають деякий дисбаланс, але коли ми звели задачу регресії, яка була складною, то ми отримали просту задачу бінарної класифікації, так як баланс класів незначний, при роботі класифікаційного лісу він не буде грати великої ролі.

Стратифікація.

```
X = data_for_tree_classifier_model.drop("area", axis=1)
y = data_for_tree_classifier_model["area"]
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=0)
sss.get_n_splits(X, y)
sss
```

```
StratifiedShuffleSplit(n_splits=1, random_state=0, test_size=0.2,
        train_size=None)
```

Розбиття на тестову та тренувальну вибірку.

```
for train_index, test_index in sss.split(X, y):
    X_train, X_test = X.loc[train_index], X.loc[test_index]
    y_train, y_test = y.loc[train_index], y.loc[test_index]
```

2.1.5.1 Побудова регресійного лісу для задачі бінарної класифікації.

```
model = DecisionTreeClassifier(max_depth = 10)
dtree_model = model.fit(X_train, y_train)
dtree_predictions = dtree_model.predict(X_test)
# creating a confusion matrix
cm = confusion_matrix(y_test, dtree_predictions)
model.score(X_train, y_train)
model.score(X_test, y_test)
```

```
Точність на тесті 0.6019417475728155
```

Малюнок композиції.

```
dotfile = open("tree.dot", "w")
tree.export_graphviz(model, out_file = dotfile,
                     feature_names = data_for_tree_classifier_model.columns[:-1])
dotfile.close()
dot_data = StringIO()
tree.export_graphviz(model, out_file=dot_data,  filled=True, rounded=True,
                     special_characters=True,
                     feature_names = data_for_tree_classifier_model.columns[:-1])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())


graph.write_png("tree.png")
graph.write_pdf("tree.pdf")
img = mpimg.imread("tree.png")
plt.figure(figsize=(15,15))
plt.imshow(img)
```

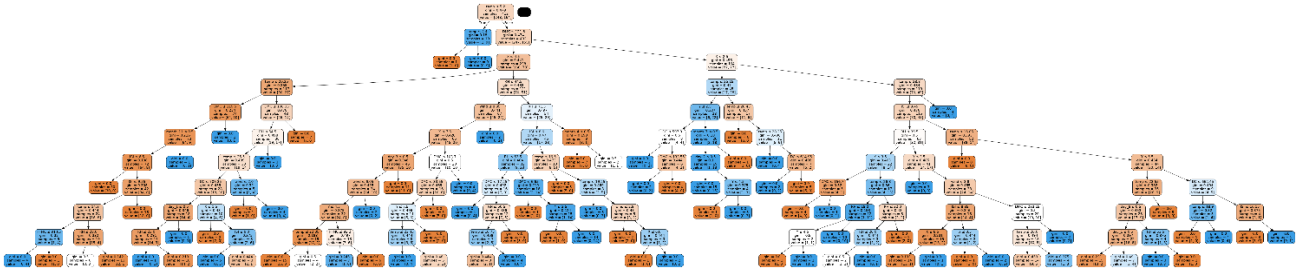

Рисунок 2.11 – регресійній ліс

## 2.2 Задача багатокласової класифікації

### 2.2.1 Опис датасету

Цей набір даних відображає зареєстровані випадки злочинів (за винятком вбивств, за якими існують дані по кожній жертві), що відбулися в місті Чикаго з 2001 року до теперішнього часу, за винятком останніх семи днів. Дані витягуються із системи CLEAR Департаменту поліції Чикаго (аналіз та звітність щодо правозастосування громадян). З метою захисту конфіденційності жертв злочинів адреси відображаються лише на рівні блоків, а конкретні розташування не вказуються.

Ціль: підготувати данні до задачі мультикласифікації. Проробити різні моделі, обрати найкращу.

Проблема задачі:

Передбачити в який **Community Area** відбудеться кримінальне правопорушення.

Підготовка даних та розробка моделей.

Імпорт бібліотек

```
import pandas as pd
import category_encoders as ce
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import itertools
import matplotlib
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedShuffleSplit
from matplotlib.pyplot import figure
```

Зчитування підготовленого файлу.

```
sity_crimes_data = pd.read_csv("start_data")
```

Таргетне значення перенесемо у кінець датафрейм.

```
columns_names = list(sity_crimes_data.columns)

columns_names.append(columns_names.pop(columns_names.index("Community Area")))

columns_names
```

```
['Primary Type',
 'Description',
 'Location Description',
 'Arrest',
 'Domestic',
 'District',
 'Year',
 'Community Area']
```

```
sity_crimes_data = sity_crimes_data[columns_names]
```

]:

| | Primary Type | Description | Location Description | Arrest | Domestic | District | Year | Community Area |
|---|---|---|---|---|---|---|---|---|
| 0 | BATTERY | DOMESTIC BATTERY SIMPLE | RESIDENCE | False | True | 9.0 | 2015 | 61.0 |
| 1 | THEFT | POCKET-PICKING | CTA BUS | False | False | 15.0 | 2015 | 25.0 |
| 2 | THEFT | OVER $500 | RESIDENCE | False | True | 6.0 | 2018 | 44.0 |
| 3 | NARCOTICS | POSS: HEROIN(BRN/TAN) | SIDEWALK | True | False | 14.0 | 2015 | 21.0 |
| 4 | ASSAULT | SIMPLE | APARTMENT | False | True | 15.0 | 2015 | 25.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 7580880 | BATTERY | SIMPLE | APARTMENT | False | False | 25.0 | 2022 | 18.0 |
| 7580881 | ASSAULT | SIMPLE | APARTMENT | False | True | 18.0 | 2022 | 7.0 |
| 7580882 | MOTOR VEHICLE THEFT | AUTOMOBILE | STREET | False | False | 3.0 | 2022 | 69.0 |
| 7580883 | WEAPONS VIOLATION | UNLAWFUL POSSESSION - HANDGUN | STREET | True | False | 11.0 | 2022 | 28.0 |
| 7580884 | BATTERY | SIMPLE | SIDEWALK | False | False | 19.0 | 2022 | 3.0 |

Рисунок 2.12 – дані задачі

Видалення незаповнених значень

```
crimes_data = sity_crimes_data[(pd.isnull(sity_crimes_data['Description']) == False)&\
            (pd.isnull(sity_crimes_data['District']) == False)&\
            (pd.isnull(sity_crimes_data['Primary Type']) == False)&\
            (pd.isnull(sity_crimes_data['Location Description']) == False)&\
            (pd.isnull(sity_crimes_data['Arrest']) == False)&\
            (pd.isnull(sity_crimes_data['Domestic'])== False)&\
            (pd.isnull(sity_crimes_data['Community Area']) == False)&\
            (pd.isnull(sity_crimes_data['Year']) == False)]

crimes_data.reset_index(inplace=True)

crimes_data = crimes_data.loc[:, "Primary Type":]

crimes_data.loc[:,["Arrest", "Domestic"]] = crimes_data[["Arrest", "Domestic"]].astype(int)
```

Розрахунок кількості заарештованих людей за різноманітними правопорушеннями

```
pd.set_option("display.max_rows", None)
print(crimes_data.loc[:,["Community Area", "Location Description", "Arrest", "Domestic"]].groupby(by=[
   "Community Area", "Location Description"]).sum().loc[25, "Arrest"])
pd.set_option("display.max_rows", 20)
```

Найпопулярніші місця в яких було скорено правопорушення.

```
STREET                  49337
SIDEWALK                51470
APARTMENT               10936
RESIDENCE               10229
ALLEY                   8610
DEPARTMENT STORE        3539
```

```
must_popular_comm_area = (crimes_data.groupby("Year")["Arrest", "Community
Area"].value_counts()
                    .rename('counts_Arrest').reset_index()
                    .drop_duplicates("Year"))
must_popular_comm_area
```

Таблиця 2.4

**Кількість скорених правопорушень за роками**

| | Year | Arrest | Community Area | counts_Arrest |
|---|---|---|---|---|
| **0** | 2001 | 0 | 76.0 | 1373 |
| **147** | 2002 | 0 | 25.0 | 12402 |
| **302** | 2003 | 0 | 25.0 | 17342 |
| **457** | 2004 | 0 | 25.0 | 16908 |
| **613** | 2005 | 0 | 25.0 | 15971 |
| **...** | ... | ... | ... | ... |
| **2781** | 2019 | 0 | 25.0 | 11091 |
| **2935** | 2020 | 0 | 25.0 | 10369 |
| **3243** | 2022 | 0 | 25.0 | 4706 |

Кількість заарештованих людей у різноманітних **Community Area.**

```
count_arrest_in_comm_area = pd.DataFrame(crimes_data.loc[:,["Community Area", "Arrest"]].groupby(by=[
    "Community Area"]).sum().sort_values(by="Arrest", na_position='first'))
count_arrest_in_comm_area
```

Таблиця 2.5

**Кількість заарештованих людей у різноманітних Community Area**

| | **Arrest** |
|---|---|
| **0.0** | 8 |
| **9.0** | 789 |
| **12.0** | 1373 |
| **47.0** | 2199 |
| **18.0** | 2390 |
| **...** | ... |
| **67.0** | 58726 |
| **8.0** | 62562 |
| **29.0** | 76463 |
| **23.0** | 82997 |
| **25.0** | 166685 |

Розподілення ознак

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,20))
# n_bins = 10
colors = ["DarkOrange", "RebeccaPurple", "BlueViolet", "DarkViolet",
          "DarkOrchid", "DarkMagenta", "Purple",
          "Indigo", "SlateBlue", "Black", "Blue", "Green", "Red"]
merged_axes = list(itertools.chain(*axes))
for ax, column_name, random_color in zip(merged_axes, crimes_data.columns[3:], colors):
    ax.hist(crimes_data[column_name], density=True, histtype='bar', color=random_color)
    ax.set_title(column_name)
fig.tight_layout()
plt.show()
```

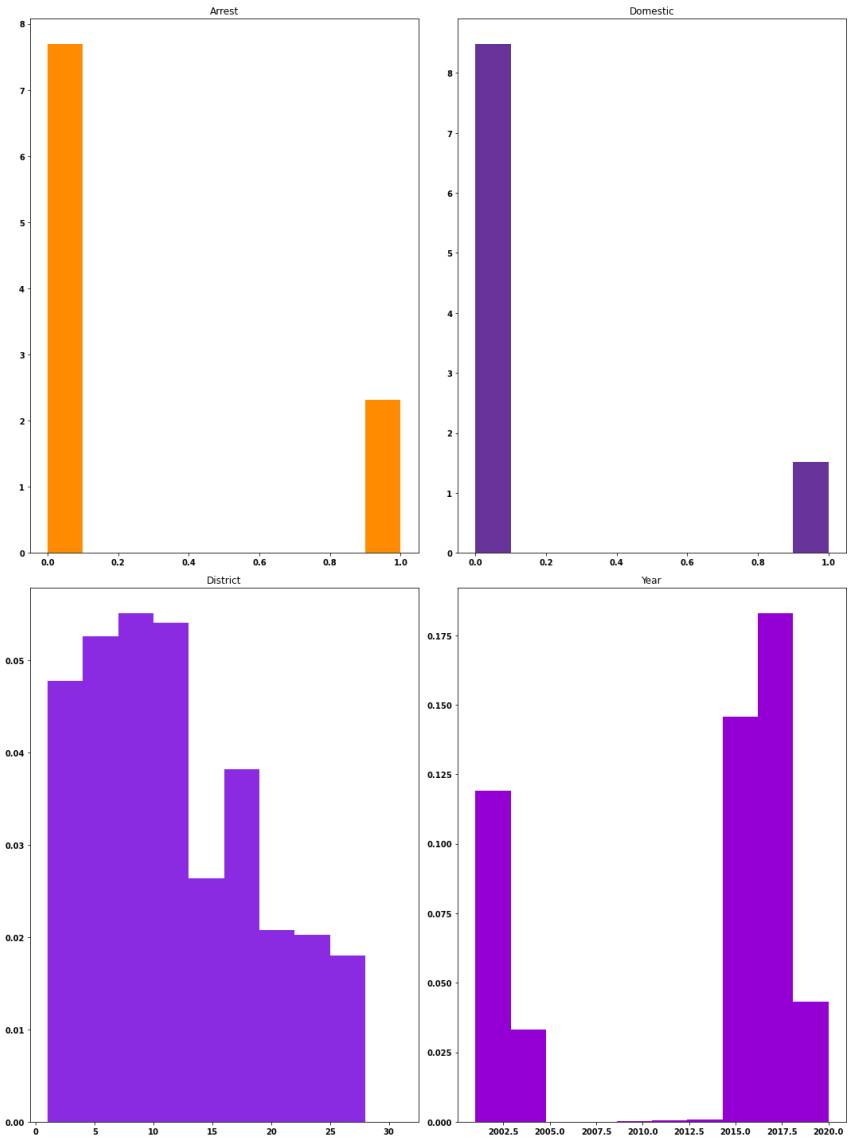


Рисунок 2.13 – Графік розподілу ознак

Найкримінальніший район

```
sns.histplot(crimes_data["Community Area"])
```

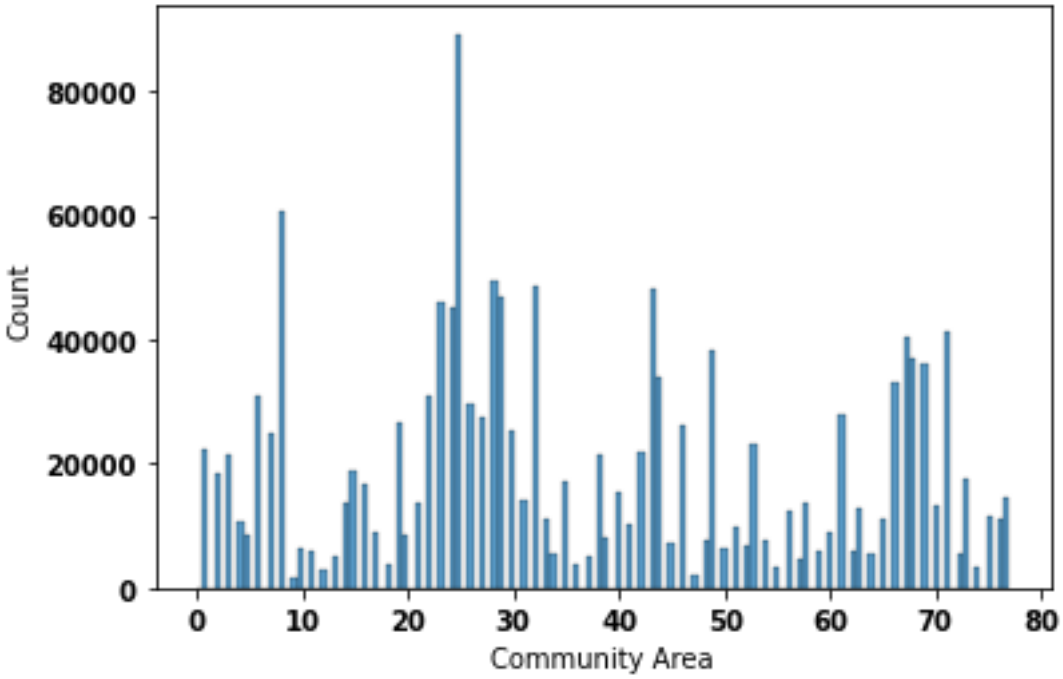


Рисунок 2.14 – Графік кримінальності за районами

Кодування категоріальних ознак.

```
enc = ce.OneHotEncoder(cols=["Primary Type", "Location Description", "Year", "District", "Arrest", "Domestic"], return_df=True)

crimes_data_encoder = enc.fit_transform(crimes_data)
```

Видалення опису правопорушення для подальшої векторізації.

```
crimes_data_encoder.drop(«Description», axis=1, inplace = True)
```

Початок векторізації ознаки Description.

```
vectorizer = CountVectorizer(min_df=0, lowercase=False)
X = vectorizer.fit_transform(crimes_data.Description)
```

```
vectorizer.get_feature_names_out()[:5]
```

```
array([‘10’, ‘10GM’, ‘18’, ‘30’, ‘300’], dtype=object)
```

```
x_array_vector = X.toarray()
def create_feature_name():
    for word_index in range(0, len(x_array_vector[0])):
        yield f"word{word_index}"
```

```
vectorize_sentenses=pd.DataFrame(x_array_vector,
columns=create_feature_name())
```

Злиття датасетів у кінцевий.

Тобто, векторізований опис плюс категоріальні та числові дані відділені до процеса векторізації.

```
end_encoder_data = pd.concat([vectorize_sentenses, crimes_data_encoder],
axis=1)
```

Виділення таргетної змінної

```
X = end_encoder_data.loc[:,:"Year_20"]
y = crimes_data["Community Area"]
```

```
y.info()
```

Стратіфікація, розподілення на тестову та навчальну вибірку з урахуванням класів

```
sss = StratifiedShuffleSplit(n_splits=2, test_size=0.25, random_state=0)
sss.get_n_splits(X, y)
for train_index, test_index in sss.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.loc[train_index], X.loc[test_index]
    y_train, y_test = y.loc[train_index], y.loc[test_index]
```

```
TRAIN: [1141503  690929  664381 ... 1316113  547421  576560] TEST: [ 176620
1328122 1154987 ... 1115127  581886  170596]
TRAIN: [1226823 1302829  583073 ...  686129  636121 1445654] TEST: [ 65542
9  315432 1321719 ...  991778  883837 1275786]
```

Збереження стратифікованих даних

```
X_train.to_csv("X_train", index=False)
X_test.to_csv("X_test", index=False)
y_train.to_csv("Y_train", index=False)
y_test.to_csv("Y_test", index=False)
```

### 2.2.3 Побудова моделей для класифікації

```
cls = DecisionTreeClassifier(max_depth=19)
```

```
model = cls.fit(X_train, y_train)
cls.score(X_train, y_train)
```

```
0.660296
```

```
cls.score(X_test, y_test)
```

```
0.65606545049213204
```

```
knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)
```

```
accuracy = knn.score(X_test, y_test)
print (accuracy)
model = VWMultiClassifier()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print(model.score(X_train, y_train))
model.score(X_test, y_test)
```

## 2.3 Висновок

У технічному розділі було вирішено дві задачі з принципово різних областей соціальної значимості. В першій задачі була проаналізовані область лісних пожеж  та контролю за вогнем. Побудована придикатна модель, яка з високою точністю вирішує задачу класифікації пожеж за ступенем ризику для навколишнього середовища. В другій задачі була проаналізована область приступності та вирішена задача класифікації районів за ступенем скорених правопорушень.

# ВИСНОВКИ

У кваліфікаційній роботі була розглянута проблема контролю лісних пожеж та проблема контролю злочинності у містах. Виявлено ключові кроки для попередньої обробки даних. Розроблені та налаштовані моделі машинного навчання для вирішення проблем. Інструментом рішення для проблеми контролю пожеж були обрані: задача регресії та кластеризації. Інструментом рішення для проблеми контролю злочинності була обрана задача кластеризації.

Для першої задачі передбачення можливої площі пожежі, був розроблений алгоритм попередньої обробки даних та побудови моделей машинного навчання.

Алгоритм попередньої обробки даних: отримані дані задачі, виведена загальна статистика за даними задачі, побудовані гістограми, побудовані графіки залежностей, досліджені взаємозв'язки між факторами, знайдені та відкориговані аномалії та пропуски в даних, нормалізовані дані.

Алгоритм побудови моделей машинного навчання: побудовані регресори для моделей машинного навчання, налаштовані моделі машинного навчання, отримана оцінка точності моделей машинного навчання.

Окремим підзавданням було виділено: зведення задачі передбачення можливої площі пожежі до задачі класифікації пожеж за ступенем небезпеки.

Алгоритм вирішення підзавдання: встановлені мітки для класифікатора, стратифіковані дані, побудований класифікатор, налаштовані моделі машинного навчання, отримані оцінки точності моделі класифікації.

Підсумком рішення проблеми першої задачі є такі твердження:

- для задачі контролю над лісними пожежами, а саме, передбачення площі пожежі, потрібна велика кількість даних, якою немає у відкритому доступі зібрана за допомогою різноманітних датчиків.
- у рамках роботи з даними першої задачі було прийняте рішення в неспроможності моделі передбачення площі пожеж, та відмічена хороша точність моделі класифікації лісних пожеж за ступенем небезпеки.
- для задачі класифікації пожеж за ступенем небезпеки, велика кількість даних не потрібна, і вдається побудувати достатньо точну модель на малих та середніх об’ємах даних.
- для другої задачі була необхідна робота з текстовими даними, їх обробка, лематизація та векторизація, що було зроблено у роботі.
- побудовано модель класифікації скоєного правопорушення за районом міста, яка має достатню точність та є масштабованою.

# СПИСИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. [Baumer-2017] Baumer, Benjamin, Daniel Kaplan, and Nicholas Horton. Modern Data Science with R. Boca Raton, Fla.: Chapman & Hall/CRC Press, 2017. [bokeh] Bokeh Development Team. “Bokeh: Python library for interactive visualiza- tion” (2014).
2. [Deng-Wickham-2011] Deng, Henry, and Hadley Wickham. “Density Estimation in R.” September 2011.
3. [Donoho-2015] Donoho, David. “50 Years of Data Science.” September 18, 2015. https://oreil.ly/kqFb0. [Duong-2001] Duong, Tarn. “An Introduction to Kernel Density Estimation.” 2001.
4. [Few-2007] Few, Stephen. “Save the Pies for Dessert.” Visual Business Intelligence Newsletter. Perceptual Edge. August 2007.
5. [Freedman-2007] Freedman, David, Robert Pisani, and Roger Purves. Statistics. 4th ed. New York: W. W. Norton, 2007. [Hintze-Nelson-1998] Hintze, Jerry L., and Ray D. Nelson. “Violin Plots: A Box Plot– Density Trace Synergism.” The American Statistician 52, no. 2 (May 1998): 181–84. [Galton-1886] Galton, Francis. “Regression Towards Mediocrity in Hereditary Stat- ure.” The Journal of the Anthropological Institute of Great Britain and Ireland 15 (1886): 246–63.
6. [ggplot2] Wickham, Hadley. ggplot2: Elegant Graphics for Data Analysis. New York: Springer-Verlag New York, 2009.
7. [Hyndman-Fan-1996] Hyndman, Rob J., and Yanan Fan. “Sample Quantiles in Statis- tical Packages.” American Statistician 50, no. 4 (1996): 361–65. [lattice] Sarkar, Deepayan. Lattice: Multivariate Data Visualization with R. New York: Springer, 2008.
8. [Legendre] Legendre, Adrien-Marie. Nouvelle méthodes pour la détermination des orbites des comètes. Paris: F. Didot, 1805. “Measures of Skewness and Kurtosis.” In NIST/SEMATECH e-Handbook of Statistical Methods. 2012.
9. [R-base-2015] R Core Team. “R: A Language and Environment for Statistical Com- puting.” R Foundation for Statistical Computing. 2015.
10. [Salsburg-2001] Salsburg, David. The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century. New York: W. H. Freeman, 2001. [seaborn] Waskom, Michael. “Seaborn: Statistical Data Visualization.” 2015
11. [Trellis-Graphics] Becker, Richard A., William S.Cleveland, Ming-Jen Shyu, and Ste- phen P. Kaluzny. “A Tour of Trellis Graphics.” April 15, 1996
12. [Tukey-1962] Tukey, John W. “The Future of Data Analysis.” The Annals of Mathemat- ical Statistics 33, no. 1 (1962): 1–67.

# Додаток А. Відомість матеріалів кваліфіційної роботи

| № з/п | Позначення | Назва | Кількість | Примітки |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | Документація | | |
| 3 | | | | |
| 4 | САіУ.РД.21.08.ПЗ | Пояснювальна записка | | Формат А4 |
| 5 | | | | |
| 6 | САіУ.РД.21.08.ДМ | Демонстраційні матеріали | | Презентація наCD-R |
| 7 | | | | |
| 8 | САіУ.РД.21.08.КР | Копія роботи | 1 | Диск CD-R |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | САіУ.РД.21.08.ДА.ПЗ. | | | |
| Змін. | Аркуш | № докум. | Підпис | Дата | | | | |
| Розроб. | | Рибалко | | | Матеріали кваліфіка-ційної роботи | Літ. | Аркуш | Аркушів |
| К. розд. | | Купенко | | | | | | |
| Керівн. | | Купенко | | | | НТУ «ДП»; 124м-21-1 | | |
| Н.контр. | | Хом’як | | | | | | |
| Зав. каф. | | Купенко | | | | | | |

## Додаток Б. Відгук на кваліфікаційну роботу бакалавра

Відгук
на кваліфікаційну роботу бакалавра
студента групи 124-17-2 Рибалка І.О.
спеціальності 124 «Системний аналіз»

## Додаток В. Рецензія на кваліфікаційну роботу бакалавра