НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Прогнозування наявності захворювання печінки на основі медичних показників»

Студента 2 курсу групи ІП-14
Спеціальності: 121
«Інженерія програмного забезпечення»
Хільчука Артема Валерійовича

«ПРИЙНЯВ» з оцінкою

_____

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____
Підпис Дата

Київ - 2023 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-14 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Хільчука Артема Валерійовича

1.Тема роботи Прогнозування наявності захворювання печінки на основі медичних показників

2.Строк здачі студентом закінченої роботи 08.06.2023

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com//: https://www.kaggle.com/datasets/abhi8923shriv/liver-disease-patient-dataset

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Вступ, постановка задачі, аналіз предметної області, робота з даними, інтелектуальний аналіз, висновки, перелік посилань, додаток А.

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6.Дата видачі завдання 16.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.03.23	
2.	Визначення зовнішніх джерел даних	24.03.23	
3.	Пошук та вивчення літератури з питань курсової роботи	01.04.23	
4.	Підготовка даних до аналізу	09.04.23	
5.	Обґрунтування методів інтелектуального аналізу даних	21.04.23	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	23.04.23	
7.	Підготовка пояснювальної записки	05.05.23	
8.	Здача курсової роботи на перевірку	08.06.23	
9.	Захист курсової роботи	09.09.23	

Студент _____ Хільчук Артем Валерійович

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Ліхоузова Т.А

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім'я, по батькові)

"08" червня 2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 47 сторінок, 24 рисунки, 14 посилань.

Об’єкт дослідження: дані про вік, стать, результати аналізу крові пацієнтів, що підозрюють захворювання печінки, а також результат обстеження- хвора печінка пацієнта, чи ні.

Предмет дослідження: створення програмного забезпечення, що проводить класифікацію даних пацієнта з метою подальшого прогнозування наявності в нього захворювання печінки та графічним відображенням результатів.

Мета роботи: розробити програмного забезпечення, що ефективно та точно прогнозує наявність в пацієнта хворобу печінки на основі базових даних.

Дана курсова робота включає в себе: опис створення програмного забезпечення для підготовки вхідних відомостей до аналізу, інтелектуального аналізу результуючого набору даних, їх графічного відображення та прогнозування за допомогою різних моделей.

ДАТАСЕТ, ДАТАФРЕЙМ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, БІНАРНА КЛАСИФІКАЦІЯ, SVM, ЛОГІСТИЧНА РЕГРЕСІЯ, ДЕРЕВА РІШЕНЬ

ЗМІСТ

ВСТУП

У сучасному світі надзвичайно вагомою проблемою є захворювання печніки: за оцінками експертів, від 80 до 100 мільйонів американців мають ту чи іншу форму захворювання печінки, однак діагноз мають лише 4.5 мільйонів, що становить додаткову загрозу здоров'ю громадян.[1]

Печінка не має нервових закінчень, отож, коли вона зазнає пошкоджень, індивід може зовсім не відчувати дискомфорту або болю. Вона є надзвичайно стійким органом: пацієнт може вжити летальну дозу гепатотоксичних препаратів з тих чи інших причин, але печінка відмовить лише через тиждень або й більший термін від часу потрапляння препаратів у ШКТ.[2] Це у свою чергу також вносить свій вклад до схильності індивіда нехтувати здоров'ям цього органу.

До того ж, завдавати шкоди здоров'ю печінки можуть не лише очевидні речі, як от інфекції, надмірне вживання алкоголю, нездорове харчування, застосування медичних препаратів, котрі мають доведений гепатотоксичний ефект, а й, наприклад, прийом деяких трав'яних екстрактів, певного ряду вітамінів у надмірних кількостях, тощо.[3]

Дана кількість неочевидних факторів ризику, а також приголомшливо неприємна частка діагностованих носіїв хвороб печінки створює потребу в здатності швидко та точно встановлювати наявність захворювань даного органу.

У рамках даної курсової роботи були проаналізовані дані лікарень про пацієнтів з різними важливими характеристиками здоров'я печінки та підготовано їх до аналізу для встановлення наявності захворювання з використанням декількох методів для прогнозування нездужання даного органу.

Дана курсова робота буде розроблена з використанням технологій та бібліотек Python 3[4], Pandas[5], Seaborn[6], Matplotlib[7], Sklearn[8], numpy[9], missingo[10],imblearn[11].

1 ПОСТАНОВКА ЗАДАЧІ

Виконання даної курсової роботи потребує вирішення декількох завдань:

Аналізу предметної області, роботи з датасетом: завантаження, дослідження його структури та виправлення наявних помилок, кореляційний аналіз даних для вибору предикторів; вибір методів для прогнозування та обґрунтування даного вибору; аналіз отриманих результатів кожного з методів та порівняння отриманих результатів ефективності.

Створення застосунку, що зчитує дані зі вхідного файлу, відображає отримані дані, розподіляє їх на тестову та тренувальну вибірки й проводить інтелектуальний аналіз для отримання передбачення за допомогою моделей прогнозування: логістичної регресії, дерева рішень та SVM.

Маніпуляція гіперпараметрами даних моделей та аналіз отриманих результатів, порівняння різних моделей прогнозування на даній вибірці, встановлення найоптимальнішого методу та його конфігурації.

Вхідними даними будуть вік пацієнта, стать, загальний рівень білірубіну, прямий білірубін, рівні ензимів АЛТ, АСТ, АЛП, загальний білок крові, рівень альбуміну, відношення альбуміну до глобуліну та висновок, чи має в результаті пацієнт захворювання печінки, чи ні.

2 Аналіз предметної області

Печінка в людському організмі відповідає за безліч ключових метаболічних функцій, включаючи виведення токсинів, травлення та засвоєння низки поживних речовин, зокрема білка та ряду амінокислот. Інфекції, такі як гепатити B і C, або генетичні аномалії можуть спричинити захворювання печінки. Автоімунні реакції або отруєння гепатотоксичними медикаментами також можуть викликати нездужання цього органу.

Найпоширеніші захворювання печінки включають: гепатити, неалкогольну жирову хворобу печінки (НЖХП), алкогольну хворобу печінки (АХП), цироз та рак печінки, аутоімунний гепатит, хворобу Вільсона.

Як було зазначено в постановці завдання, печніка надзвичайно стійка й спричиняє рівень дискомфорту, нерозмірно малий отриманим пошкодженням, що призводить до того, що доволі часто пацієнти потрапляють в лікарню з просунутими ступенями захворювань, деколи навіть потребуючи трансплантації. Як результат, за оцінками WHO від захворювань печінки в середньому помирає 2.4% людей від загальної кількості смертей.[12]

Виявлення пошкоджень печінки - тривалий та складний процес, що включає в себе аналіз крові, комп’ютерну томографію, еластографію, серологічний тест, функціональний тест та навіть деколи генетичний тест і біопсію печінки.

Однак чи можна оптимізувати даний процес діагностики, а відповідно полегшити навантаження на лікарів, за допомогою технологій інтелектуального аналізу даних на первинному етапі- аналізі вмісту крові?

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

- завантаження та дослідження структури датасету з інформацією про стать та вік пацієнтів та результатів аналізів крові
- підготовка зчитаних даних до аналізу
- використання декількох моделей прогнозування даних з налаштуванням

гіперпараметрів

- прогнозування за характеристиками наявність у пацієнта захворювання печінки
- відображення отриманих результатів та їх аналіз
- порівняння використаних моделей.

3 Робота з даними

3.1 Опис обраних даних

Для виконання курсової роботи було обрано набір даних пацієнтів з підозрами на захворювання - “Liver Disease Patients”: https://www.kaggle.com/datasets/abhi8923shriv/liver-disease-patient-dataset

Даний датасет містить 30000 записів пацієнтів та складається з 11 стовпців:

- Age of the patient – вік пацієнта
- Gender of the patient – стать
- Total Bilirubin – рівень загального білірубіну в крові (mg/dL)
- Direct Bilirubin – рівень прямого білірубіну (mg/dL)
- Alkphos Alkaline Phosphotase – рівень АЛП (IU/L)
- Sgpt Alamine Aminotransferase – рівень АЛТ (IU/L)
- Sgot Aspartate Aminotransferase – рівень АСТ (IU/L)
- Total Protiens – рівень білку в крові (g/dL)
- ALB Albumin – рівень альбуміну (g/dL)
- A/G Ratio Albumin and Globulin Ratio – співвідношення альбуміну до глобуліну
- Result – вказує на те, чи має пацієнт захворювання печінки, чи ні

3.2 Перевірка даних

Зчитуємо та візуалізуємо датафрейм зі вхідними даними:

	Age of the patient	Gender of the patient	Total Bilirubin	\
0	65.0	Female	0.7	
1	62.0	Male	10.9	
2	62.0	Male	7.3	
3	58.0	Male	1.0	
4	72.0	Male	3.9	
...	...	...	...	
30686	50.0	Male	2.2	
30687	55.0	Male	2.9	
30688	54.0	Male	6.8	
30689	48.0	Female	1.9	
30690	30.0	Male	3.1	

	Direct Bilirubin	Alkphos Alkaline Phosphotase	\
0	0.1	187.0	
1	5.5	699.0	
2	4.1	490.0	
3	0.4	182.0	
4	2.0	195.0	
...	...	...	
30686	1.0	610.0	
30687	1.3	482.0	
30688	3.0	542.0	
30689	1.0	231.0	
30690	1.6	253.0	

Рисунок 3.1 – перші 5 стовпців початкового датафрейму

	Sgpt Alamine Aminotransferase	Sgot Aspartate Aminotransferase	\
0	16.0	18.0	
1	64.0	100.0	
2	60.0	68.0	
3	14.0	20.0	
4	27.0	59.0	
...	...	...	
30686	17.0	28.0	
30687	22.0	34.0	
30688	116.0	66.0	
30689	16.0	55.0	
30690	80.0	406.0	

	Total Protiens	ALB Albumin	A/G Ratio Albumin and Globulin Ratio	\
0	6.8	3.3	0.90	
1	7.5	3.2	0.74	
2	7.0	3.3	0.89	
3	6.8	3.4	1.00	
4	7.3	2.4	0.40	
...	...	...	...	
30686	7.3	2.6	0.55	
30687	7.0	2.4	0.50	
30688	6.4	3.1	0.90	
30689	4.3	1.6	0.60	
30690	6.8	3.9	1.30	

Рисунок 3.2 – перші 5 стовпців початкового датафрейму

	Sgpt Alamine Aminotransferase	Sgot Aspartate Aminotransferase	\
0	16.0	18.0	
1	64.0	100.0	
2	60.0	68.0	
3	14.0	20.0	
4	27.0	59.0	
...	...	...	
30686	17.0	28.0	
30687	22.0	34.0	
30688	116.0	66.0	
30689	16.0	55.0	
30690	80.0	406.0	

	Total Protiens	ALB Albumin	A/G Ratio Albumin and Globulin Ratio	\
0	6.8	3.3	0.90	
1	7.5	3.2	0.74	
2	7.0	3.3	0.89	
3	6.8	3.4	1.00	
4	7.3	2.4	0.40	
...	...	...	...	
30686	7.3	2.6	0.55	
30687	7.0	2.4	0.50	
30688	6.4	3.1	0.90	
30689	4.3	1.6	0.60	
30690	6.8	3.9	1.30	

Рисунок 3.3 – наступні 5 стовпців початкового датафрейму

	Result
0	1
1	1
2	1
3	1
4	1
...	...
30686	1
30687	1
30688	1
30689	1
30690	1

Рисунок 3.4 – останній стовпець датафрейму початкового датафрейму

Проаналізуємо датафрейм на наявність відсутніх значень. Перш за все, побудуймо теплову карту відсутніх значень:

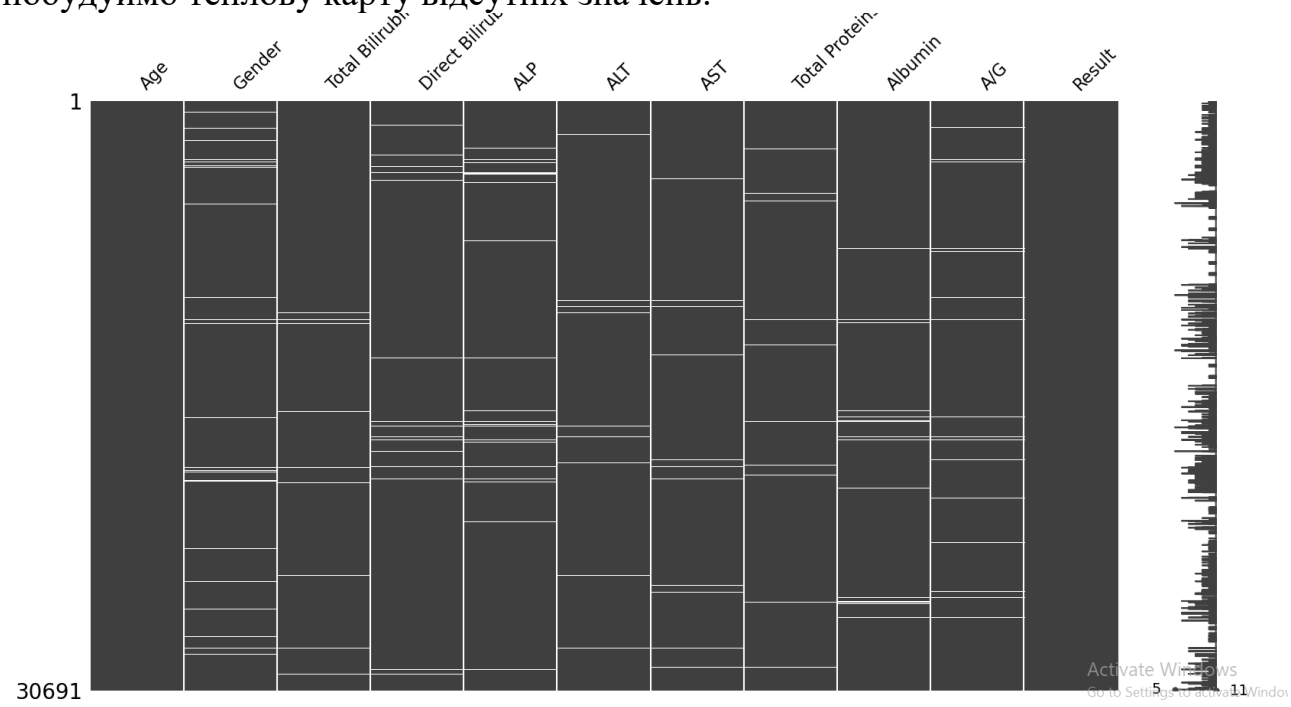

Рисунок 3.5 – теплова карта відсутніх значень у датафреймі

Як бачимо, відсутні значення наявні в 9 з 11 стовпців, однак наявні вони, зогляду, у невеликих кількостях і доволі часто відсутність полів рядка може збігатися для кількох стовпців. Проаналізуємо кількості появ відсутніх значень по стовпцях:

```
Total rows: 30691
Absent values per column:
    Age - 2
    Gender - 902
    Total Bilirubin - 648
    Direct Bilirubin - 561
    ALP - 796
    ALT - 538
    AST - 462
    Total Proteins - 463
    Albumin - 494
    A/G - 559
    Result - 0
```

Рисунок 3.6 – кількості появ відсутніх значень по стовпцях

Усе таки, в Age також наявні два відсутні значення. У цілому, порівняно із загальним обсягом набору даних відсутніх значень незначний відсоток, отож

було прийнято рішення рядки з відсутніми показниками просто видаляти. Після видалення, об’єм даних скоротився з 30691 запису до 27158 записів.
Візуалізуємо дані частоти значень стовпця Result:

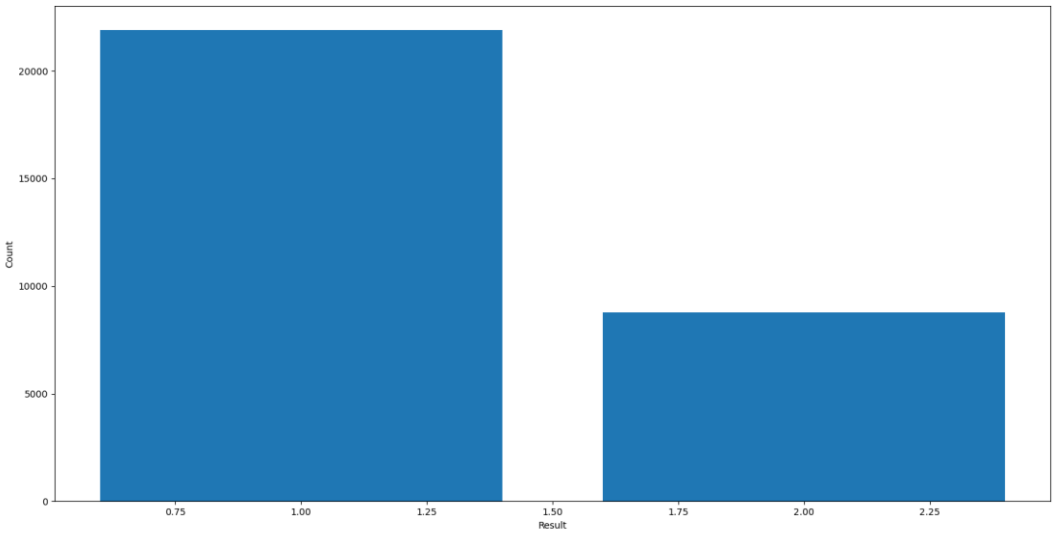

Рисунок 3.7 – стовпчикова діаграма частот

Як бачимо, є значна незбалансованість класів. Двійка відображає те, що пацієнт виявився зі здоровою печінкою, а одиниця вказує на те, що захворювання є. Пояснюється така упередженість тим, що дані збиралися в лікарнях, а туди звертаються частіше пацієнти, що справді мають підстави підозрювати у себе захворювання печінки. Дану незбалансованість класів необхідно буде враховувати при подальших кроках виконання курсової роботи.
Тепер переглянемо статистики кожного зі стовпців:

```
                Age  Total Bilirubin  Direct Bilirubin          ALP  \
count  27158.000000     27158.000000      27158.000000  27158.000000
mean      44.125046         3.407909          1.541630    290.142021
std       15.971563         6.332486          2.895084    239.595473
min        4.000000         0.400000          0.100000     63.000000
25%       33.000000         0.800000          0.200000    175.000000
50%       45.000000         1.000000          0.300000    209.000000
75%       55.000000         2.700000          1.300000    298.000000
max       90.000000        75.000000         19.700000   2110.000000

                ALT           AST  Total Proteins       Albumin           A/G  \
count  27158.000000  27158.000000    27158.000000  27158.000000  27158.000000
mean      81.279292    112.102879        6.472605      3.124044      0.943567
std      181.571537    283.616005        1.081477      0.792329      0.324205
min       10.000000     10.000000        2.700000      0.900000      0.300000
25%       23.000000     26.000000        5.800000      2.600000      0.700000
50%       36.000000     42.000000        6.600000      3.100000      0.900000
75%       62.000000     88.000000        7.200000      3.700000      1.100000
max     2000.000000   4929.000000        9.600000      5.500000      2.800000

             Result
count  27158.000000
mean       1.282790
std        0.450363
min        1.000000
25%        1.000000
50%        1.000000
75%        2.000000
max        2.000000
```

Рисунок 3.8 – статистики стовпців датафрейму

Бачимо, що для багатьох стовпців наявні значення, що лежать поза межами математичного сподівання з трьома стандартними девіаціями. Для оцінки кількості викидів та прийняття подальших рішень щодо них, візуалізуємо числові дані за допомогою діаграми розмаху:

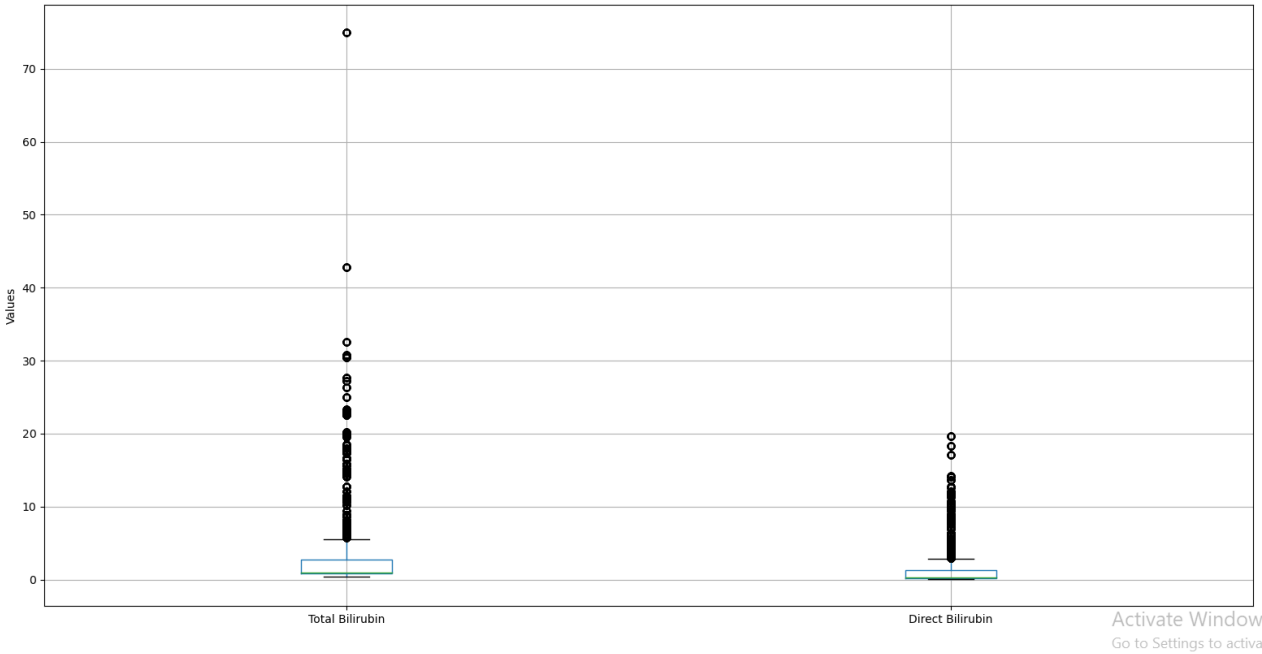

Рисунок 3.9 – діаграми розмаху для загального та прямого білірубіну

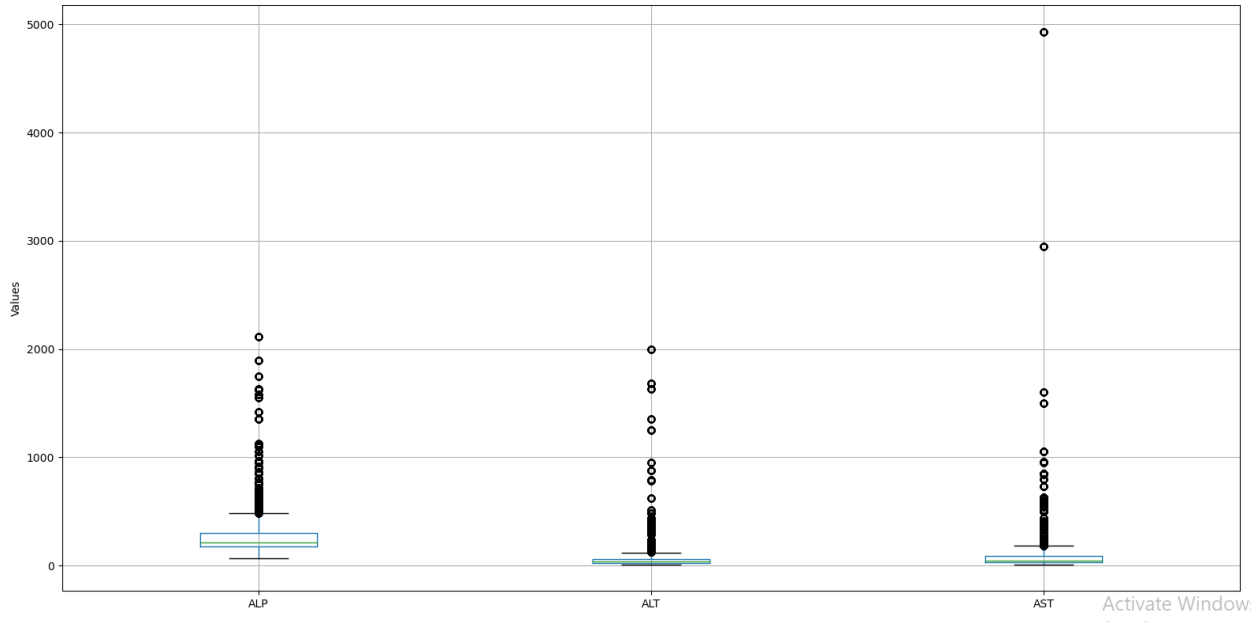

Рисунок 3.10 – діаграми розмаху для АЛП, АЛТ, АСТ

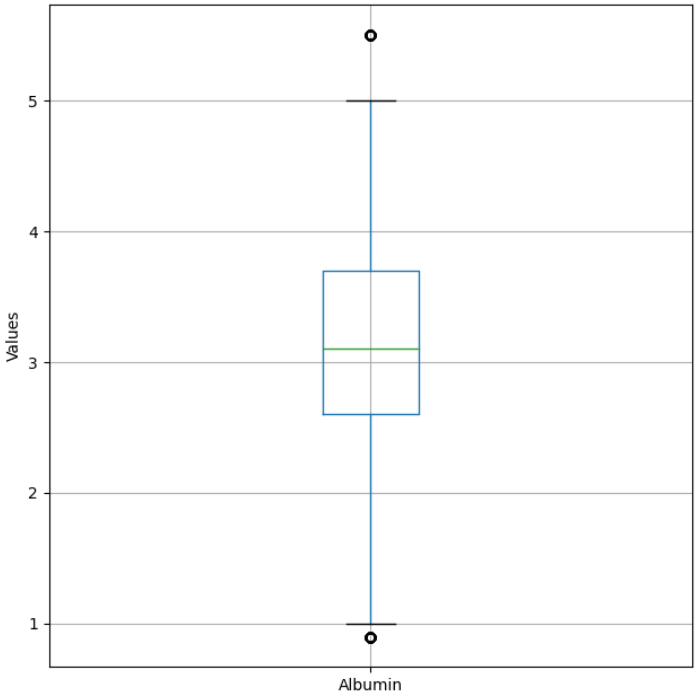

Рисунок 3.11 – діаграми розмаху для альбуміну

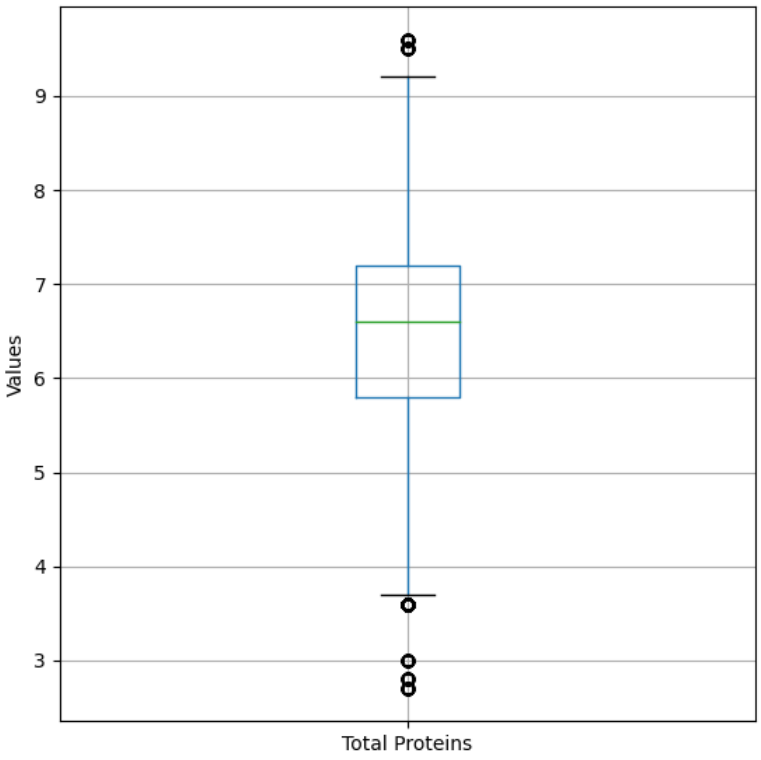

Рисунок 3.12 – діаграма розмаху загального протеїну

Як бачимо, найбільша кількість викидів наявна в рівнях білірубіну та ензимів. Проаналізуємо конкретну кількість викидів для всіх стовпців:

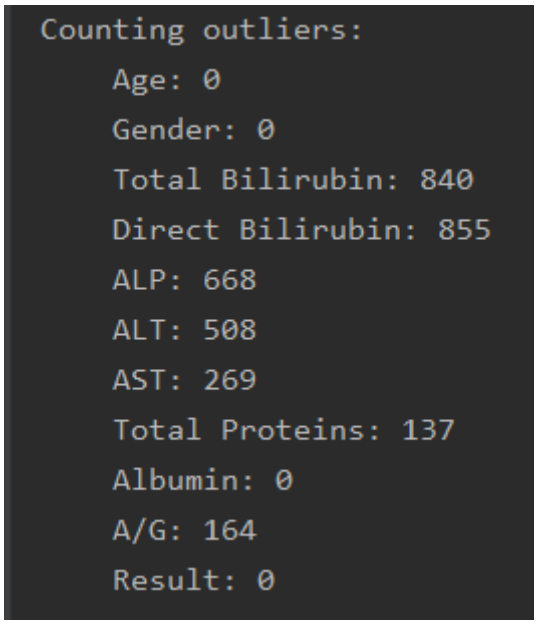

Рисунок 3.13 – кількості викидів по стовпцях

Викидів теж незначна кількість. Питання полягає в тому, чи видаляти рядки, що містять викиди, чи ні? На мою думку, в контексті даної предметної області викиди несуть доволі цінну інформацію, адже, наприклад, рівні ензимів у понад тисячу можуть вказувати на некроз печінки – тобто, тяжкий ступінь захворювання. Відповідно, було прийнято рішення рядки з викидами не видаляти. Наявність викидів необхідно буде враховувати під час вибору моделі прогнозування.

Також для зручності закодуємо деякі значення:

- У стовпці Gender замість Male встановимо 0, а замість Female – 1
- У стовпці Result замість 2 встановимо 0. Двійка вказує на те, що пацієнт не був діагностований захворюванням печінки, а 1, що був. За допомогою такого кодування інтерпретація буде інтуїтивно більш зрозумілою

3.3 Вибір предикторів

Перш за все, будуємо матрицю кореляцій та відображаємо її у вигляді теплової карти:

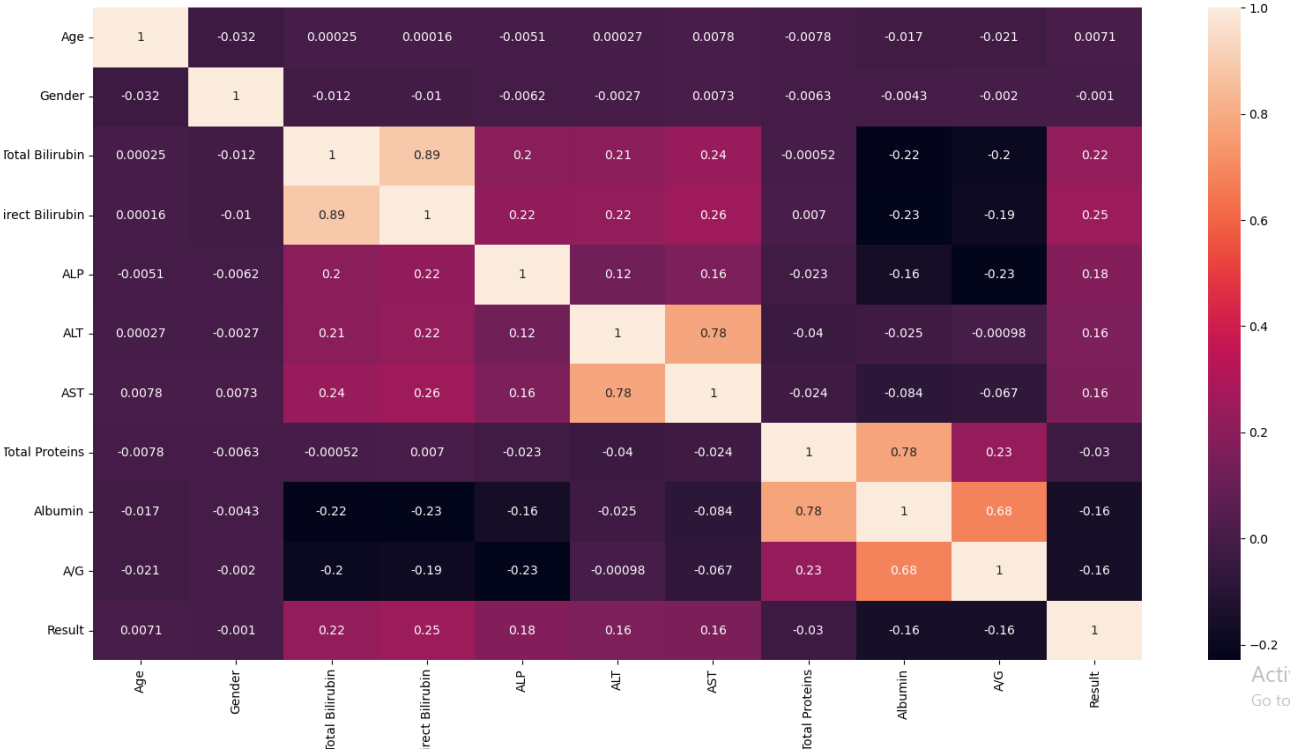

Рисунок 3.14 – теплова мапа кореляцій

Очікувано, наявність діагнозу позитивно корелює з рівнями білірубіну: у здорової печінки білірубін руйнується під дією ензимів, однак дизфункційна печінка справляється з даною задачею все гірше. Білірубін має характерний жовтий колір, отож, коли в людини виникають проблеми з печінкою, шкіра та білки очей починають забарвлюватися відповідним чином.

Також позитивно корелюють з діагнозом рівні ензимів АЛТ, АСТ та АЛП у крові: зазвичай вони перебувають всередині печінки, однак коли клітина руйнується, дані ензими вивільняються в кровотік організму. Іншими словами, рівні даних ензимів є прямими індикаторами інтенсивності пошкоджень печінки.

Неочікувано, вік та стать фактично не впливають на діагноз. Загальний рівень протеїну в крові також фактично не є індикатором.

Негативно корелюють з діагнозом рівні альбуміну та відношення альбуміну до глобуліну.

Отож, відкидаємо стовпці віку, статі, загального рівню протеїну.

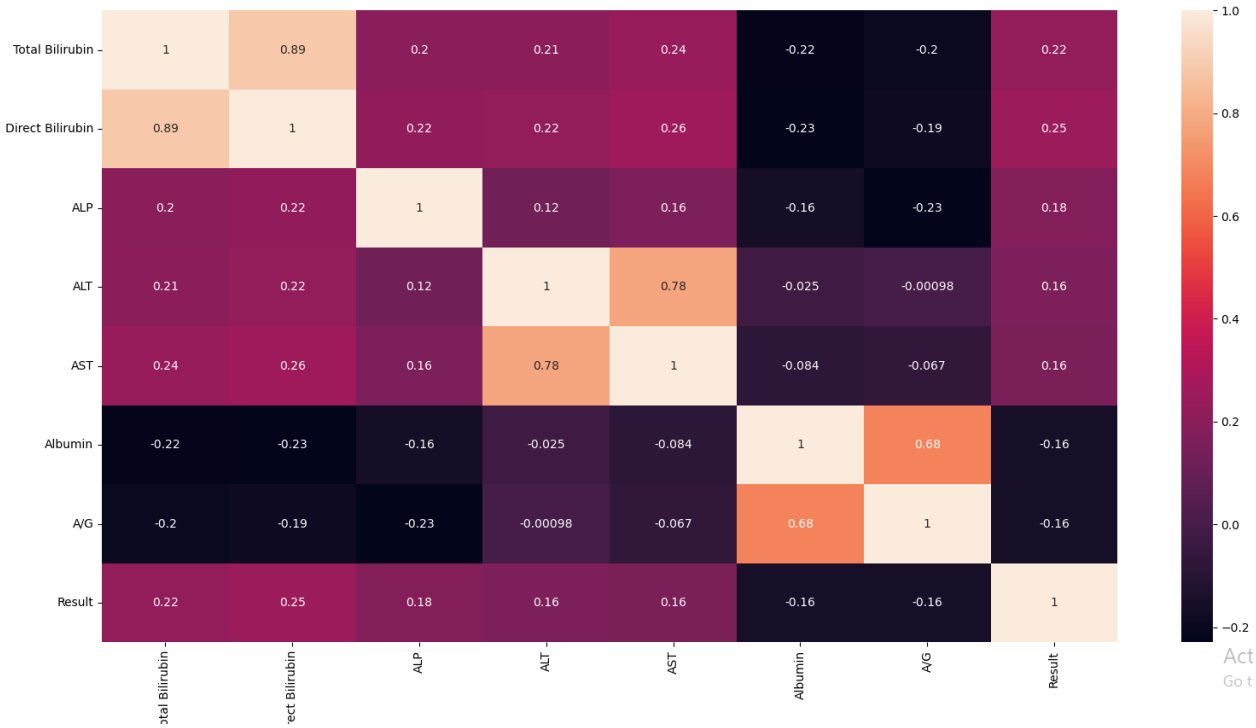

Рисунок 3.15 – теплова мапа кореляцій після видалення некорельованих стовпців

Тепер необхідно ліквідувати мультиколінеарність.

Загальний білірубін має сильну кореляцію з прямим білірубіному- видаляємо загальний білірубін. Аналогічно АЛТ сильно корелює з АСТ – видаляємо АСТ. Альбумін та співвідношення альбуміну до глобуліну теж мають неабияку кореляцію між собою. Очевидно, це через наявність альбуміну в даному відношенні. Спробуємо встановити рівень глобуліна звідси й встановити, чи впливатиме він на result.

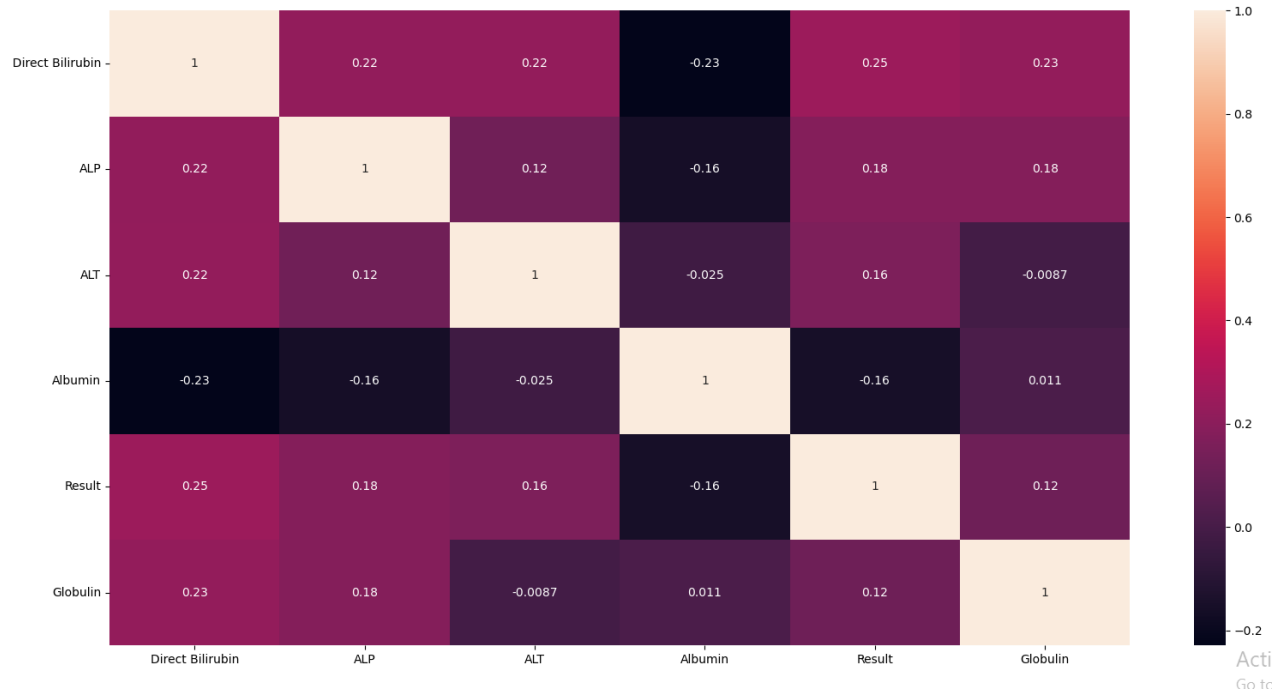

Рисунок 3.16 – теплова мапа кореляцій після ліквідації мультиколінеарних стовпців та додання стовпця глобуліну

Як бачимо, кореляція доволі слабка позитивна існує, однак вона все ж занадто далека від порогу 0.2, тому стовпець глобуліну, усе ж, буде відкинуто. Отож, результуючий набір даних має наступний вигляд:

	Direct Bilirubin	ALP	ALT	Albumin	Result
0	0.1	187.0	16.0	3.3	1
1	5.5	699.0	64.0	3.2	1
2	4.1	490.0	60.0	3.3	1
3	0.4	182.0	14.0	3.4	1
4	2.0	195.0	27.0	2.4	1
...	...	...	...	...	...
30686	1.0	610.0	17.0	2.6	1
30687	1.3	482.0	22.0	2.4	1
30688	3.0	542.0	116.0	3.1	1
30689	1.0	231.0	16.0	1.6	1
30690	1.6	253.0	80.0	3.9	1

Рисунок 3.17 – вигляд результуючого датафрейму

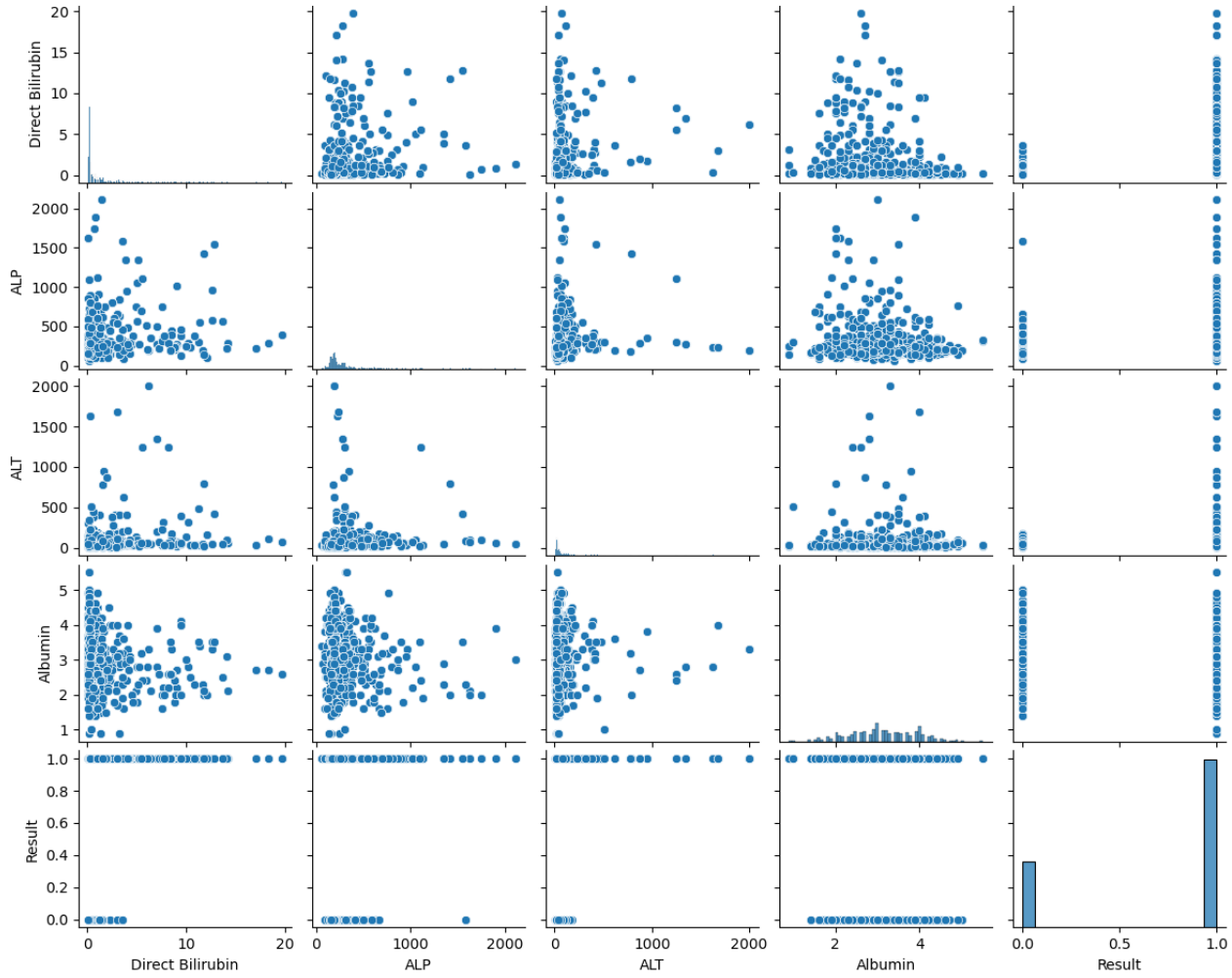

Рисунок 3.18– візуалізація результуючих даних за допомогою матриці розсіювань

4 Інтелектуальний аналіз даних

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Отож, задача полягає в бінарній класифікації для набору даних, що великий за об’ємом, незбалансований за класами та містить у собі частку викидових значень.

Перше, що асоціюється з бінарною класифікацією – SVM. SVM- один з найпопулярніших та найбільш відомих алгоритмів класифікації даних. Концепція машини допоміжних векторів надзвичайно проста- необхідно провести у просторі з ознаками у якості вимірів площину таким чином, аби відстані між площиною та найближчими до неї точками класів були якомога більшими. SVM виявляє високу стійкість у роботі з викидами та незбалансованими класами.[13] До всього ж, згідно з деякими оцінками, SVM є найпоширенішим алгоритмом у працях, що пов’язані з галуззю медицини, перевершуючи в популярності навіть нейронні мережі.[14] Недоліком, однак, є велика потреби в обчислювальних потужностях, що неодмінно усугубиться великим об’ємом даних. Це у свою чергу може значно ускладнити перебір гіперпараметрів моделі.

Іншою моделлю для порівняння було обрано логістичну регресію. Логістична регресія є одним з основних алгоритмів, 3 за популярністю в медичній літературі, з багатьма змінними для моделювання дихотомічних результатів, тобто таких, що передбачують дві категорії.[14] Логістична регресія використовується для отримання співвідношення шансів, коли присутні більше однієї пояснювальної змінної. Процедура схожа на множинну лінійну регресію, за винятком того, що відповідна змінна є біноміальною. Вона показує вплив кожної змінної на співвідношення шансів спостережуваної події. Моделі логістичної регресії виявляють низьку чутливість до впливу викидів, оскільки сигмоїдна функція згладжує їх. До всього ж, логістична регресія має низькі вимоги до обчислювальних можливостей, що буде дуже корисно при роботі з великим датасетом. Недоліком же логістичної регресії є її припущення

про лінійність між предикторами та логарифмом вірогідностей, однак у житті все не завжди так просто.

Урешті-решт, третім алгоритмом було обрано дерево вибору. Дерево рішень використовує структуру даних, що нагадує дерево, для передбачення результатів конкретної проблеми. Дерева рішень працюють за вертикальним принципом обходу зверху-вниз, що означає, що кореневий вузол завжди знаходиться у верхній частині структури, а результати представлені листям дерева. Кожен вузол після кореневого розбивається на кілька вузлів. Основна ідея полягає в тому, щоб використовувати дерево рішень для розбиття простору даних на області. Розбиття бінарного дерева може бути бінарним або багатошляховим. Алгоритм продовжує розбивати дерево, доки дані не стануть достатньо однорідними. Після тренування повертається дерево рішень, яке може використовуватися для здійснення оптимальних категоризованих прогнозів. Дерево рішень також має низьку чутливість до викидів: вони будуються таким чином, що акцентується на загальних закономірностях та тенденціях в даних, а не на окремих точках даних. Загальна структура та прогнози дерева залежать від колективної поведінки даних. Однак неділокому даної моделі є чутливість до незбаолансованості класів, тому при його застосуванні необзідно буде застосовувати техніки балансування.

4.2 Аналіз отриманих результатів для методу SVM

Для підбору найоптиммальніших гіперпараметрів для даного алгоритму застосуємо sklearn.model_selection.GridSearchCV. У рамках даного аналізу будуть перебрані наступні параметри:

- У якості ядра будуть перебиратися лінійне, поліноміальне та радіальне
- У якості коефіцієнту ядра будуть перебиратися значення ‘scale’ та ‘auto’

У зв’язку з тим, що складність алгоритму SVM становить у найліпшому випадку O(n^2), де n – число рядків даних, у рамках даної курсової роботи скоротимо тренувальну вибірку для перебору значень до 1000 випадкових екземплярів з основного набору.

За результатами пошуку найоптимальнішими гіперпараметрами виявилися linear для ядра та scale для коефіцієнту ядра.

Отримуємо найкращу модель від GridSearchCV та встановлюємо необхідні метрики на випробувальних даних. У якості метрик оцінки якостей моделей будуть аналізуватися точність, повнота передбачень, F-score для знаходження компромісу між вищевказаними (обчислюється за формулою (2*Accuracy*Recall)/(Accuracy+Recall)), а також сутність помилок (кількість помилок першого та другого родів).

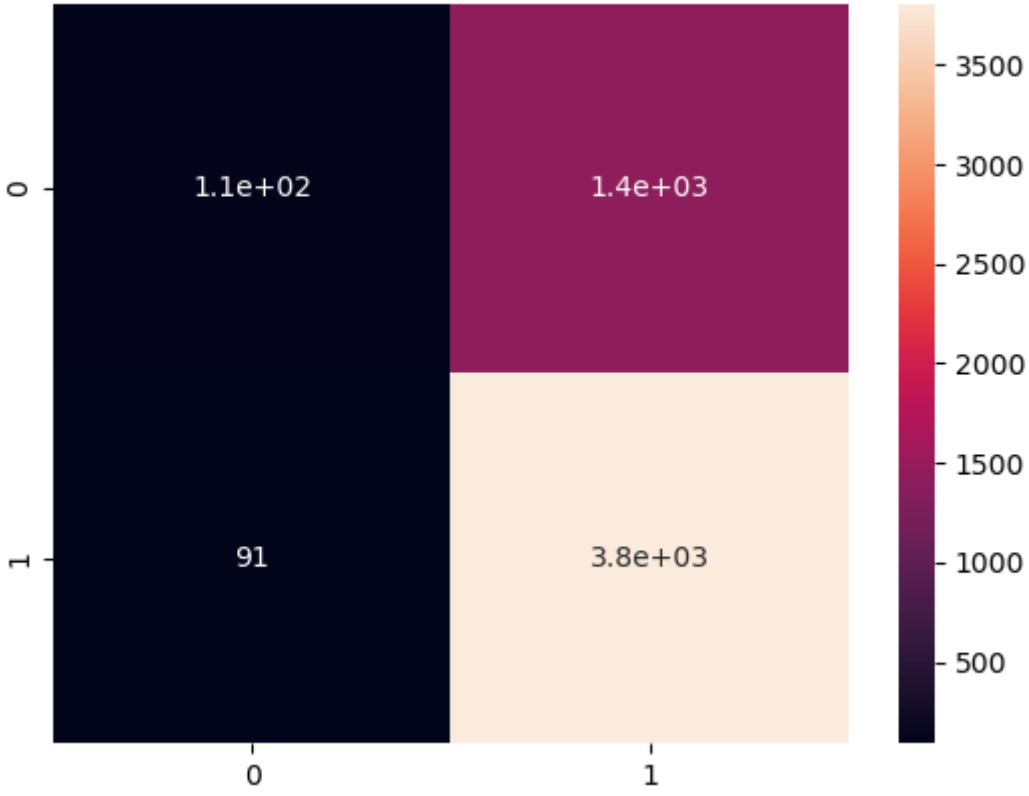

Рисунок 4.1 – матриця невідповідностей для SVC

Результуючими метркиами для даного алгоритму є:

- 72% точності
- 97.6% повноти
- 82.97% F-score

Неприємно, що наявна певна кількість помилок другого роду, хоч їх частка й незначна. Якщо брати за основу дану модель, необхідно буде встановити чітку політику інтерпретації результатів передбачення з урахуванням наявності настільки небезпечної похибки.

Помилки першого роду також присутні в значних кількостях, однак у контексті даної предметної області вони не є критичними, оскільки це лиш ознчатиме, що пацієнтові більш вірогідно необхідно буде пройти решту огляду.

Чудові показники має повнота аналізу. Точність також статистично доволі непогана.

4.3 Аналіз отриманих результатів для методу логістичної регресії

Для масштабування даних застосуємо sklearn.preprocessing.StandardScaler. Задля зручності, будуємо пайплайн за допомогою sklearn.pipeline.Pipeline, у який включаємо скейлер та логістичну регресію. Алгоритм пошуку найкращої конфігурації той самий. Будуть перебиратися наступні гіперпараметри:

- Обернені значення сили регуляризації 0.1, 1 та 10
- Типи регуляризації L1 та L2
- Алгоритми оптимізаційної задачі liblinear та saga

За результатами пошуку найоптимальнішими гіперпараметрами виявилися 0.1, l1 та libilinear.

Отримуємо найкращу модель від GridSearchCV та встановлюємо необхідні метрики на випробувальних даних. Метрики оцінки якості залишаються незмінні.

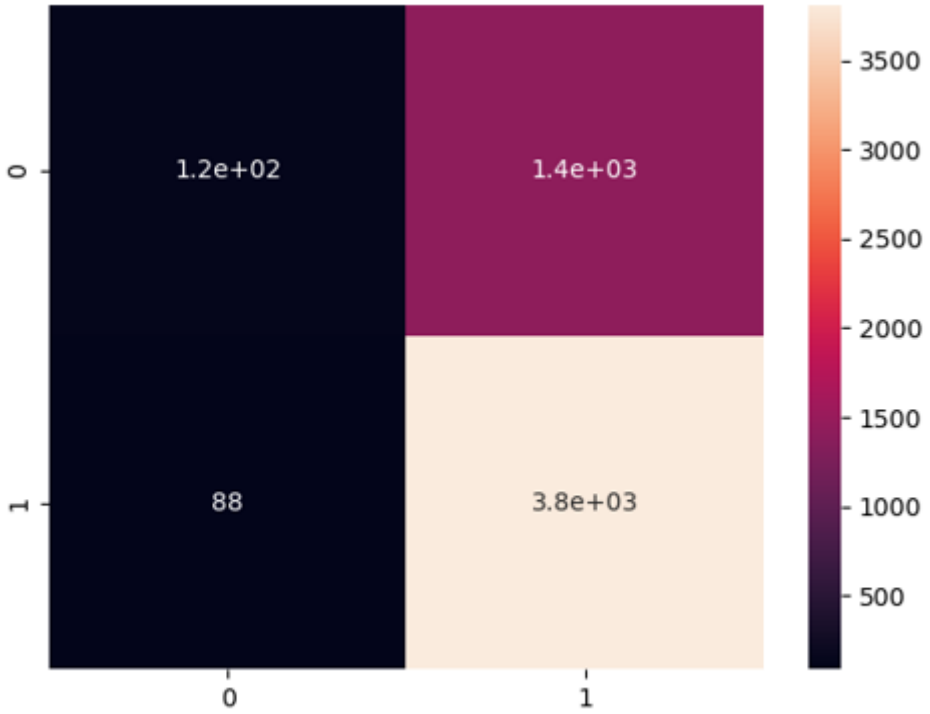

Рисунок 4.2 – матриця невідповідностей для логістчиної регресії

Результуючими метркиами для даного алгоритму є:

- 72.1% точності
- 97.7% повноти
- 83.1% F-score

У даної моделі також є помилки другого роду, однак вже трохи менше, ніж у SVM. Помилки першого роду також присутні в об’ємах, приблизно рівних попередній моделі.

4.4 Аналіз отриманих результатів для методу дерева рішень

Перш за все, необхідно повторно наголосити, що набір даних є незбалансованим, що може впливати на якість результату даної моделі.

Для дерева рішень будуть перебиратися наступні параметри:

- Критерії оцінки розбиття gini та enthropy
- Максимальні глибини дерева: від 1 до 8
- Мінімальні кількість вибірок для розбиття вузла 2, 5 та 10
- Мінімальні кількість вибірок для розбиття дистка 1, 2 та 4

За результатами пошуку найоптимальнішими гіперпараметрами виявилися відповідно gini, 8, 1 та 2.

Отримуємо найкращу модель від GridSearchCV та встановлюємо необхідні метрики на випробувальних даних. Метрики оцінки якості залишаються незмінні.

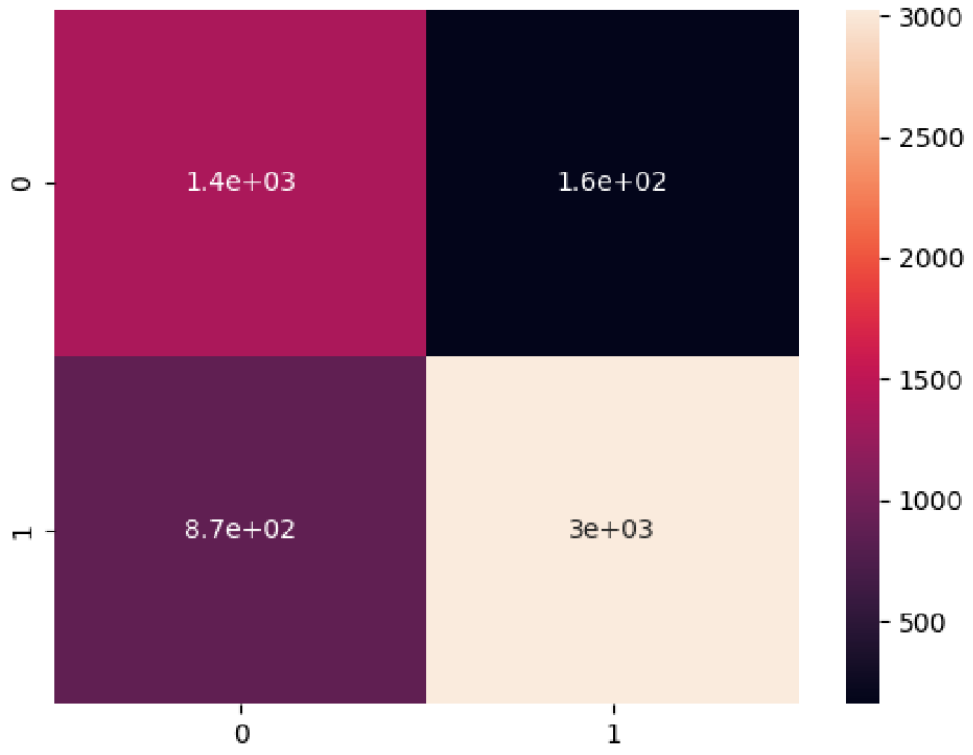

Рисунок 4.3 – матриця невідповідностей для дерева рішень

Результуючими метриками для даного алгоритму є:

– 80.9% точності
– 77.6% повноти
– 79.2% F-score

Як бачимо, модель має вийграш у точності, порівняно з розглянутими альтернативами. Однак, на відміну від конкурентних алгоритмів, повнота зазнає нерозмірних втрат, що, безумовно, є неприємно.

Неприємлимим є те, що модель має надзвичано велику кількість помилок 2 роду, що в десятки разів перевищує аналогічні, що, на мою думку, робить цей алгоритм неконкурентним.

4.5 Порівняння отриманих результатів

Отож, урешті-решт маємо наступні дані стосовно результативності:

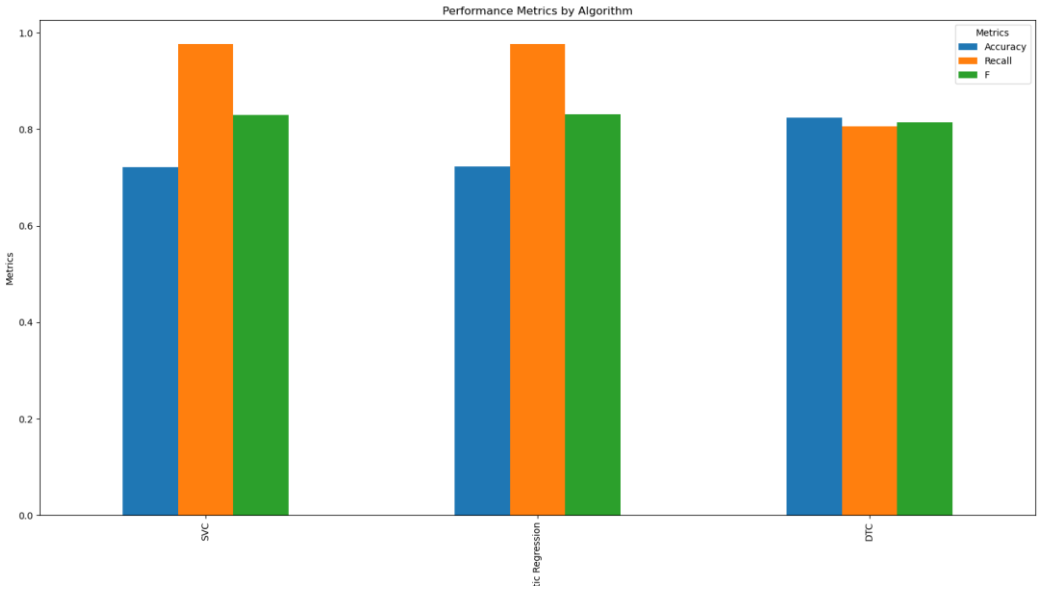

Рисунок 4.4 – значення точності, повноти та F-Score для різних алгоритмів

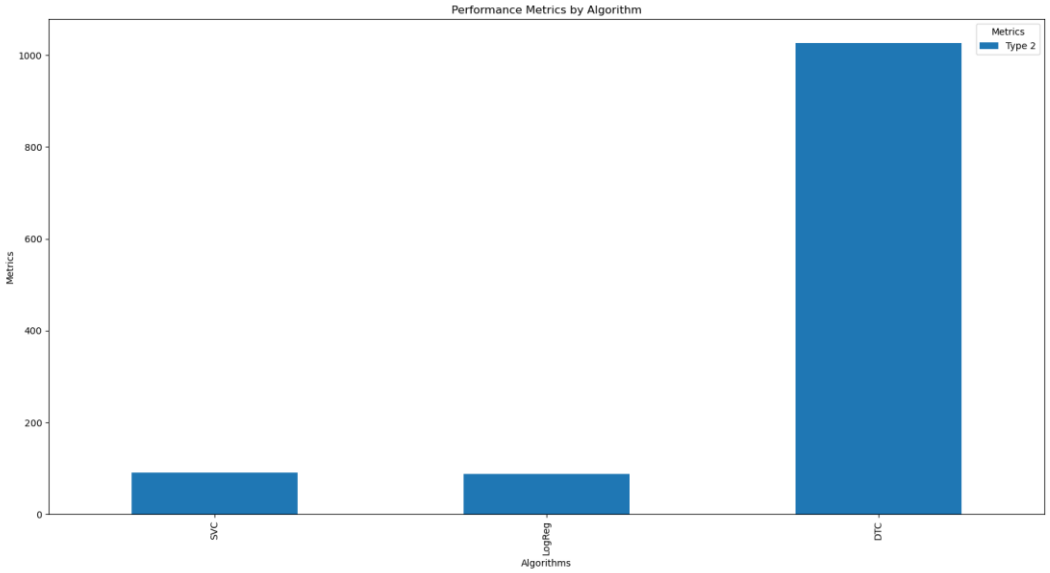

Рисунок 4.5 – кількості помилок 2 роду для різних алгоритмів

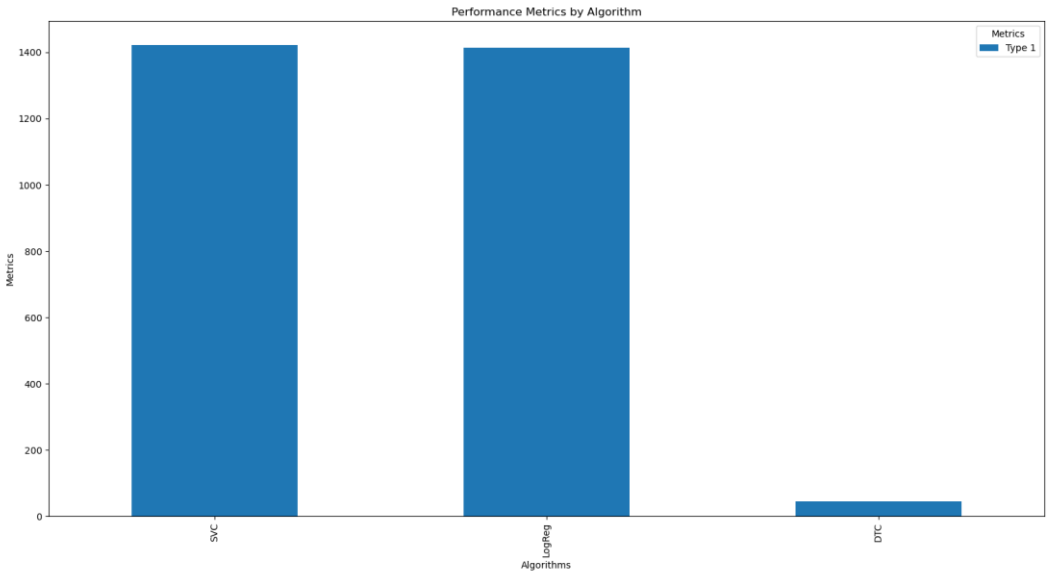

Рисунок 4.6 – кількості помилок 1 роду для різних алгоритмів

Зрозуміло, модель DTC не підходить під дану задачу у зв’язку з високою частотою помилок 2 роду, що, по суті, зробить дане ПЗ для прогнозування непотрібним у контексті даної предметної області.

Відповідно, залишається SVM та логістична регресія. Логістична регресія випереджає SVM за F-score, має трохи меншу кількість помилок другого роду та приблизно однакову кількість 1 роду. До всього ж, тренується вона швидше, отож, вибір очевидний.

ВИСНОВКИ

Проблема захворювань печінки є надзвичайно важливою в сучасному світі, однак нерідко знехтувана. Процес діагностики хвороб печінки є комплексним та тривалим процесом, що нерідко може складатися з кількох етапів. Оптимізація даного процесу за допомогою технологій інтелектуального аналізу дозволить значною мірою зменшити навантаження на лікарів та прискорити діагностику пацієнта.

У рамках виконання курсової роботи було було реалізовано програмне забезпечення, що приймає дані про результати аналізів крові, віку та статі пацієнтів і тренує на їх основі моделі трьох методів: SVM, логістичну регресію та дерево рішень.

Було зчитано відповідно дані з відповідного файлу, перейменовано деякі стовпці на більш лаконічні, досліджено структуру даних. Встановлено, що для деяких стовпців нерідкісним явищем є поява викидів. Видалено відсутні значення. Закодовано значення стовпців статі та результату.

Методом кореляційного аналізу було встановлено набір ознак, за якими буде робитися прогноз: ними виявилися прямий білірубін, АЛП, АЛТ та альбумін.

Обгрунтовано вибір трьох методів для прогнозування результату на основі особливостей вхідних даних та проаналізовано результати кожного з методів окремо.

Аналізуючи отримані результати, було встановлено, що найкращою для прогнозування буде модель логістичної регресії. Точність та повнота даної моделі є співрозмірною з SVM та перевершує результати дерева рішень. До того ж, за кількістю помилок 2 роду, а також метрикою F-score вона показує трохи ліпшу результативність за метод допоміжних векторів, при цьому маючи однакові результати за кількістю помилок 1 роду. До того ж, логістична регресія не вимагає таких високих обчислювальних потужностей, як SVM, що теж схиляє до цього алгоритму

Отож, найпліпшою моделлю для подальшого прогнозування наявності захворювань печінки в пацієнтів найоптимальним варіантом буде заастсоування логістичної регресії.

ПЕРЕЛІК ПОСИЛАНЬ

1. *How Many People Have Liver Disease?* (5 August 2022 р.). Отримано з https://liverfoundation.org/: https://liverfoundation.org/about-your-liver/facts-about-liver-disease/how-many-people-have-liver-disease/
2. *Paracetamol Poisoning*. (17 April 2023 р.). Отримано з en.wikipedia.org: https://en.wikipedia.org/wiki/Paracetamol_poisoning
3. M., G.-C., Robles-Díaz , M., Ortega-Alonso, A., & Medina-Caliz, I. (9 April 2016 р.). Hepatotoxicity by Dietary Supplements: A Tabular Listing and Clinical Characteristics.
4. Python 3.11.4 documentation. (7 June 2023 р.). Отримано з https://docs.python.org/3/
5. pandas documentation. (28 May 2023 р.). Отримано з https://pandas.pydata.org/docs/
6. seaborn: statistical data visualization. (без дати). Отримано з https://seaborn.pydata.org/tutorial/introduction
7. Matplotlib 3.7.1 documentation. (2023). Отримано з https://matplotlib.org/stable/index.html
8. scikit-learn: Machine Learning in Python. (без дати). Отримано з https://scikit-learn.org/stable/
9. NumPy documentation. (без дати). Отримано з https://numpy.org/doc/stable/
10. Missingno. (без дати). Отримано з https://github.com/ResidentMario/missingno
11. imbalanced-learn documentation. (28 December 2022 р.). Отримано з https://imbalanced-learn.org/stable/
12. Huang, D., Terrault, N., & Tacke , F. (28 March 2023 р.). Global epidemiology of cirrhosis — aetiology, trends and predictions. doi:https://doi.org/10.1038/s41575-023-00759-2
13. How does SVM ignore outliers ? (27 May 2017 р.). Отримано з https://www.reddit.com/r/MachineLearning/comments/8mihni/d_how_does_svm_ignore_outliers/

14. Sciforce. (24 April 2019 р.). Top AI algorithms for Healthcare. Отримано з https://medium.com/sciforce/top-ai-algorithms-for-healthcare-aa5007ffa330

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду прогнозування наявності захворювання печінки на основі медичних показників

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.,

студента групи ІП-14 ІІ курсу

Хільчука А.В.

```
import pandas as pd
import missingno as msno
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, recall_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

if __name__=="__main__":
    pd.set_option('display.max_columns', None)

    data=pd.read_csv('data/Liver Patient Dataset
(LPD)_train.csv',encoding='cp1252')
    #Some of the column names have redundant spaces
    for col in data.columns:
        data.rename(columns={col: col.strip()}, inplace=True)

    #renaming columns into more terse alternatives
    data.rename(columns={'Age of the patient': 'Age',
                         'Gender of the patient':'Gender',
                         'Alkphos Alkaline Phosphotase':'ALP',
                         'Sgpt Alamine Aminotransferase':'ALT',
                         'Sgot Aspartate Aminotransferase':'AST',
                         'Total Protiens': 'Total Proteins',
```

```
                    'ALB Albumin':'Albumin',
                    'A/G Ratio Albumin and Globulin Ratio':'A/G'},inplace=True)
        print(data)

        #removing NaN values
        msno.matrix(data)
        plt.show()

        print(f"\nTotal rows: {len(data)}\nAbsent values per column:")
        for col in data.columns:
            print(f"\t{col} - {data[col].isna().sum()}")


        print(f"\nProceeding to delete rows that contain NaN values\nRows before deletion: {len(data)}")
        data.dropna(inplace=True)
        print(f"Rows after deletion:{len(data)}")

        counts = data['Result'].value_counts()
        labels = counts.index
        values = counts.values
        plt.bar(labels, values)
        plt.xlabel('Result')
        plt.ylabel('Count')
        plt.show()

        print(data.describe())

        #check for outliers
        data[['Total Bilirubin','Direct Bilirubin']].boxplot()
```

```
plt.ylabel('Values')
plt.show()

data[['ALP', 'ALT','AST']].boxplot()
plt.ylabel('Values')
plt.show()

data[['Albumin']].boxplot()
plt.ylabel('Values')
plt.show()

data[['Total Proteins']].boxplot()
plt.ylabel('Values')
plt.show()



# Function to count outliers using Z-score
def count_outliers_zscore(column):
    if pd.api.types.is_numeric_dtype(column):
        z_scores = (column - column.mean()) / column.std()
        outliers = np.abs(z_scores) > 3  # Those tha lie after 3 standart
deviations
        return outliers.sum()
    else:
        return 0  # Skip non-numeric columns



# Print outlier counts for each column
print('Counting outliers:')
```

```
for col in data.columns:
    outlier_count = count_outliers_zscore(data[col])
    print(f'\t{col}: {outlier_count}')



#some encoding
#gender
genders={'Male':0,'Female':1}
data['Gender']=data['Gender'].map(genders).astype(int)

#those with 2 tend to have normal enzymes level, so they arelikely suffering from liver diseases
#this mappping seems more logical
results={2:0,1:1}
data['Result']=data['Result'].map(results)

#Determining predictors
sns.heatmap(data.corr(),annot=True)
plt.show()
data.drop(columns=['Total Proteins','Age','Gender'],inplace=True)

sns.heatmap(data.corr(),annot=True)
plt.show()

data['Globulin']=data['Albumin']/data['A/G']
data.drop(columns=['Total Bilirubin','AST','A/G'],inplace=True)

sns.heatmap(data.corr(),annot=True)
plt.show()
```

```
        data.drop(columns=['Globulin'],inplace=True)
        print(data)

        sns.pairplot(data)
        plt.show()

        #splitting data
        X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1],
data['Result'], test_size=0.2, random_state=42)

        print("Testing SVC:")



        param_grid = {
            'kernel': ['linear', 'poly','rbf'],
            'gamma': ['scale', 'auto']
        }
        svc_clf = SVC()
        tmp=data.sample(1000)
        X_svc_train=tmp.iloc[:, :-1]
        y_svc_train=tmp['Result']

        grid_search = GridSearchCV(svc_clf, param_grid, cv=5)
        grid_search.fit(X_svc_train, y_svc_train)

        best_params = grid_search.best_params_#linear, scale
        best_model = grid_search.best_estimator_
        print(f'Best parameters for SVC are: {grid_search.best_params_}')

        #best_model = SVC(kernel='linear',gamma='scale').fit(X_train, y_train)
```

```
        y_pred = best_model.predict(X_test)
        svc_acc = accuracy_score(y_test, y_pred)
        svc_recall = recall_score(y_test, y_pred)
        print(f'For SVC metrics are:\n\tAccuracy:
{svc_acc}\n\tRecall:{svc_recall}\n\tF-score:{2 * svc_acc * svc_recall / (svc_acc +
svc_recall)}')
        cm=confusion_matrix(y_test,y_pred)
        svc_type2_err=cm[1, 0]
        svc_type1_err=cm[0, 1]

        sns.heatmap(cm,annot=True)
        plt.grid(False)
        plt.show()

        #logistic regression
        param_grid = {
           'logisticregression__C': [0.1, 1, 10],
           'logisticregression__penalty': ['l1', 'l2'],
           'logisticregression__solver': ['liblinear', 'saga']
        }

        logreg = pipeline = make_pipeline(StandardScaler(), LogisticRegression())

        grid_search = GridSearchCV(logreg, param_grid, cv=5)
        grid_search.fit(X_train, y_train)
        best_params = grid_search.best_params_
        print(f'Best parameters for logistic regression are:
{grid_search.best_params_}')
        best_model = grid_search.best_estimator_
```

```
        # Evaluate the model
        accuracy = best_model.score(X_test, y_test)
        logreg.fit(X_train, y_train)
        y_pred = best_model.predict(X_test)
        logreg_acc = accuracy_score(y_test, y_pred)
        logreg_recall = recall_score(y_test, y_pred)
        print(f'For logistic regression metrics are:\n\tAccuracy:
{logreg_acc}\n\tRecall:{logreg_recall}\n\tF-score:{2 * logreg_acc * logreg_recall /
(logreg_acc + logreg_recall)}')
        cm = confusion_matrix(y_test, y_pred)
        logreg_type2_err=cm[1, 0]
        logreg_type1_err=cm[0, 1]

        sns.heatmap(cm,annot=True)
        plt.grid(False)
        plt.show()
        # Define the parameter grid
        param_grid = {
            'criterion': ['gini', 'entropy'],
            'max_depth': [ 1,2,3,4,5,6,7,8],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4],
        }

        # Create an instance of SMOTE
        oversampler = SMOTE()

        # Apply oversampling to the training data
        X_train_resampled, y_train_resampled = oversampler.fit_resample(X_train,
y_train)
```

```
tree_clf = DecisionTreeClassifier()
grid_search = GridSearchCV(tree_clf, param_grid, cv=5)
grid_search.fit(X_train_resampled, y_train_resampled)

best_params = grid_search.best_params_

print(f'Best parameters for DTC are: {grid_search.best_params_}')
best_model= grid_search.best_estimator_
best_model.fit(X_train_resampled, y_train_resampled)
y_pred = best_model.predict(X_test)
DTC_acc = accuracy_score(y_test, y_pred)
DTC_recall = recall_score(y_test, y_pred)
print(f'For random forest metrics are:\n\tAccuracy:
{DTC_acc}\n\tRecall:{DTC_recall}\n\tF-score:{2 * DTC_acc * DTC_recall /
(DTC_acc + DTC_recall)}')
cm = confusion_matrix(y_test, y_pred)
dtc_type2_err = cm[1, 0]
dtc_type1_err = cm[0, 1]
sns.heatmap(cm,annot=True)
plt.grid(False)
plt.show()

results=pd.DataFrame({
        'Accuracy':[svc_acc,logreg_acc,DTC_acc],
        'Recall':[svc_recall,logreg_recall,DTC_recall]
}, index=['SVC','Logistic Regression','DTC'])

results['F']=2*results['Accuracy']*results['Recall']/(results['Accuracy']+results['Recall'
])
```

```
results.plot(kind='bar')
plt.xlabel('Algorithms')
plt.ylabel('Metrics')
plt.title('Performance Metrics by Algorithm')
plt.legend(title='Metrics')
plt.show()

results = pd.DataFrame({
    'Type 1': [svc_type1_err, logreg_type1_err, dtc_type1_err]
}, index=['SVC', 'LogReg', 'DTC'])
results.plot(kind='bar')
plt.xlabel('Algorithms')
plt.ylabel('Metrics')
plt.title('Performance Metrics by Algorithm')
plt.legend(title='Metrics')
plt.show()

results = pd.DataFrame({
    'Type 2': [svc_type2_err, logreg_type2_err, dtc_type2_err]
}, index=['SVC', 'LogReg', 'DTC'])
results.plot(kind='bar')
plt.xlabel('Algorithms')
plt.ylabel('Metrics')
plt.title('Performance Metrics by Algorithm')
plt.legend(title='Metrics')
plt.show()
```