Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська дипломна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ РАКУ ЛЕГЕНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____ Сергій НЕВОЛЯ
Керівник: к.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«31» 05 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____ Руслан БЕЛЗЕЦЬКИЙ
«10» 06 2024 р.

Допущено до захисту

Завідувач кафедри САІТ

_____ д.т.н., проф. Віталій МОКІН

«03» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"08" 05 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Неволі Сергію Дмитровичу

1. Тема роботи: «Інформаційна технологія передбачення раку легенів методами машинного навчання»
керівник роботи: Жуков С. О., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 05 2024 року №80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Kaggle Dataset «Lung Cancer Prediction».
4. Зміст текстової частини:
1) Аналіз предметної області передбачення раку легень;
2) Розвідувальний аналіз даних;
3) Розроблення інформаційної технології.
5. Перелік ілюстративного матеріалу:
1) Розвідувальний аналіз даних;
2) Блок-схема алгоритму інформаційної технології;
3) Результати усіх моделей;
4) Оцінка впливу ознак на найкращу модель.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11.03

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз проблематики предметної області	11.03	31.03	вик
2	Аналіз та вибір оптимальних інформаційних технологій	01.04	15.04	вик
3	Розвідувальний аналіз даних	16.04	30.04	вик
4	Розроблення інформаційної технології	01.05	10.05	вик
5	Аналіз результатів дослідження	10.05	20.05	вик
6	Оформлення матеріалів до захисту БДР	20.05	31.05	вик

Студент _____ Сергій НЕВОЛЯ

Керівник роботи _____ Сергій ЖУКОВ

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 88 сторінок формату А4, на яких є 66 рисунка, список використаних джерел містить 20 найменувань.

Метою роботи є підвищення точності передбачення ризику захворіти на рак легень шляхом створення інформаційної технології.

В роботі наведено розроблення інформаційної технології на платформі Kaggle, на мові програмування Python. Проведено розвідувальний аналіз даних, було очищено та оптимізовано вхідні дані. Проведено інтелектуальне моделювання та його налаштування для передбачення ризиків захворіти раком легень. Проаналізовано результати дослідження та прогнозування. Розроблена інформаційна технологія дає можливість передбачати ризики захворіти раком легень у пацієнтів при наявності відповідних даних про цих пацієнтів.

Ключові слова: інформаційна технологія, рак легень, машинне навчання, аналіз даних, моделювання, передбачення, Kaggle, Python.

ABSTRACT

The bachelor's thesis consists of 88 pages of A4 format, on which there are 66 figures, the list of references contains 20 titles.

The aim of the work is to improve the accuracy of predicting the risk of lung cancer by creating information technology.

The paper presents the development of information technology on the Kaggle platform, in the Python programming language. An exploratory analysis of the data was carried out, and the input data was cleaned and optimized. Intelligent modeling and its adjustment to predict the risks of lung cancer were carried out. The results of the study and forecasting are analyzed. The developed information technology makes it possible to predict the risks of lung cancer in patients in the presence of relevant data on these patients.

Keywords: information technology, lung cancer, machine learning, data analysis, modeling, prediction, Kaggle, Python.

ЗМІСТ

ВСТУП

Актуальність теми. Найпоширеніший вид раку це рак легенів, він же й носить статус одного з найбільш смертоносних видів раку по усьому світу.

Через симптоми, якими характеризуються інші захворювання, наприклад кашель із кров'ю та мокротинням, розпізнати у цьому рак легень на початковій стадії мають змогу лише сучасні методи діагностики. До таких належать дослідження за допомогою рентгенівських променів. Даний метод є первинним інструментальним методом діагностики, який дає змогу визначити присутність новоутворення в легенях, але лише на важких кінцевих стадіях. Більш точно діагностувати розмір, місцезнаходження та поширення пухлини дають змогу МРТ (магнітно-резонансна томографія) та КТ (комп'ютерна томографія).

Ліків від даної хвороби, які б могли повністю позбавитись від неї, не існує. Сучасна медицина має змогу допомогти лише на ранніх стадіях з високими шансами на успіх, якщо порівнювати із шансами при пізніх стадіях. Саме тому на даний час найбільш ефективною протидією захворюваності на рак є і залишається своєчасна діагностика та профілактика, тому розробка та дослідження ефективних методів прогнозування даної хвороби є надзвичайно актуальною.

Мета і задачі дослідження. Метою дослідження є підвищення точності передбачення ризику захворіти на рак легень шляхом створення інформаційної технології.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

– провести аналіз проблем із діагностуванням ризиків раку легень;

– здійснити вибір оптимальних моделей машинного навчання та інформаційних технологій;

– здійснити розвідувальний аналіз даних та ознак;

– розробити інформаційну технологію для передбачення ризиків захворіти раком легень.

Об'єктом дослідження є процес розроблення інформаційної технології передбачення ризику раку легень у пацієнтів. методами машинного навчання.

Предметом дослідження є методи машинного навчання і програмні засоби, які використовуються у процесі створення інформаційної технології для передбачення ризиків раку легень у пацієнтів.

Публікації. За результатами роботи було опубліковано тези на тему «Розвідувальний аналіз даних для інформаційної технології передбачення раку легень методами машинного навчання» на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (м. Вінниця, 2023-2024 рр.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПЕРЕДБАЧЕННЯ РАКУ ЛЕГЕНЬ

1.1 Аналіз проблематики раку легень

Одним із найстрашніших захворювань у світі є рак. Такого статусу ця хвороба заслуговує завдяки своїй поширеності та, що найголовніше, високій летальності й невиліковності. Найпоширеніший вид даної хвороби є рак легенів, він же й носить статус одного з найбільш смертоносних видів раку по усьому світу. За даними Всесвітньої організації охорони здоров'я (ВООЗ), у 2022 році було зареєстровано 2,5 мільйона нових випадків раку легенів (12,4% від загальної кількості нових випадків) та 1,8 мільйона смертей від цього захворювання (18,7% від загального числа зареєстрованих смертей від раку) [2].

Повноцінних ліків, які могли б гарантувати повне позбавлення від хвороби на будь яких її стадіях не існує, але воно у активному процесі розробки, та це навряд чи встигне допомогти тим людям, які уже захворіли, адже зазвичай при виявленні хвороби часу на якусь активну протидію залишається дуже мало. Саме тому на даний час найбільш ефективною протидією захворюваності є і залишається рання діагностика та профілактика. Якщо хворобу виявити рано, ефективність лікування та шанси на його успішність сильно зростають, а тому розробка та дослідження ефективних методів прогнозування даної хвороби є надзвичайно актуальною.

1.2 Фактори ризику раку легень

Крім прямих симптомів, які вказують на вірогідність захворювання саме раком легенів існують також так звані фактори ризику. По своїй природі рак це мутація, а у кожної мутації є свій шанс на виникнення і от саме фактори ризику збільшують цей шанс виникнення даної мутації, що в свою чергу означає можливий розвиток цієї мутації та переростання її у захворювання. Тож розглянемо одні із основних факторів ризику, які впливають на шанси захворіти

раком легень, які офіційно дослідженні та науково доведені. До таких факторів належать:

1. Куріння: Куріння є однією з головних причин захворюваності на рак легень. Це частково пояснює причину чому рак легенів тримає першість у рейтингу кількості захворювань саме легеневої форми раку, адже на планеті за останніми даними палить майже 1,1 мільярда людей. Це означає що під прямим впливом даного фактору перебуває майже 14% населення усієї планети.

2. Пасивне куріння: Простими словами це вторинне вдихання тютюнового диму, тобто такого, який видихається курцем або виходить з кінчика сигарети. Проведені раніше дослідження показали, що цей фактор є не менш небезпечним ніж звичайне куріння. Цифри говорять, що люди, які живуть з курцями, мають на 20-30% вищі шанси розвитку хвороби, ніж звичайні люди.

3. Вплив канцерогенів: Канцерогени це речовини, які можуть спричинити появу раку, це відбувається шляхом прямого впливу цих речовин на ДНК структуру клітин легенів, що викликає мутації та розвиток пухлин. До прикладу такими речовинами являються смола, чадний газ, формальдегід, бензол, радон, азбест та інші, усі ці речовини у певній кількості містяться у сигаретах чи утворюються при їх палінні завдяки хімічним реакціям.

4. Забруднене повітря: Це повітря, яке містить шкідливі речовини, зокрема хімічні речовини, гази, дим та пил. Рівень забруднення визначають за різними показниками надходження шкідливих речовин у повітря. Джерелом таких надходжень може слугувати транспорт, промисловість, електропідстанції, сільське господарство, природні джерела (пожежі, виверження вулканів)

5. Генетичні ризики: Генетична схильність до захворювання на рак легень може бути як вродженою так і набутою. Вроджені можуть передаватись, якщо хтось у сім’ї хворів цим захворюванням або ж зароджуватись в процесі якогось збою під час ембріонального періоду. Набута генетична схильність може виникнути при зміні ДНК через навколишні чинники, тютюновий дим, радон, азбест та інше.

6. Вік: Ризик розвитку раку легень зростає із віком, це викликано різними причинами, але в першу чергу через старіння організму та його

слабшання порівняно із молодим віком. Статистика говорить що люди віком до 40 років найменш схильні до такого захворювання, але за умови що на них не впливають ніякі інші чинники, з кожним наступним роком ризики збільшуються. Так наприклад у людей віком 60-69 років ризики найбільші, особливо якщо ця людина курець зі стажем.

7. Інші фактори (хронічні захворювання легень, вплив іонізуючого випромінювання, слабкий імунітет і т. д.) [3].

1.3 Аналіз середовища та мови програмування для розробки технології передбачення раку легень

Передбачення саме раку легень через симптоми ускладняється тим що симптоми, які характерні для даної хвороби досить поширені й для інших захворювань. Навряд чи хтось запідозрить рак через звичайний кашель звичайними методами огляду та діагностики, тому найкращим варіантом залишається завчасно попередити виникнення такої хвороби. Одним із способів такого попередження може бути технологія, яка за набором певних діагностичних даних, які не потребують використання високотехнологічних інструментів діагностики таких як МРТ чи КТ, зможе попереджати про ризики захворіти в майбутньому.

Для вирішення задачі розробки подібної інформаційної технології для прогнозування ризиків найкраще підходить мова програмування Python.

Python – це динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Була розроблена в 1990 році голландським програмістом Гвідо ван Россумом. Автором було вигадано таку назву завдяки популярного на той час британського серіалу який транслювався у 1970-х роках «Летючий цирк Монті Пайтона» [4].

Python використовується для різноманітних цілей: для розроблення ігор і веб-застосунків, розробки інструментів для різноманітних проектів. Ця мова також набула широкого застосування в науковій області, а саме для досліджень і розв’язування прикладних завдань.

Існує сформульована філософія Python Тіма Петерса, він сформулював її у декілька пунктів:

- Практичність важливіша за бездоганність
- Зараз краще, ніж ніколи
- Якщо реалізацію складно пояснити – ідея погана.
- Помилки ніколи не повинні замовчуватися
- Просте краще, ніж складне
- Складне краще, ніж заплутане

Розкриваючи популярні галузі застосування Python необхідно відмітити такі як: робототехніка, машинне навчання, штучний інтелект, інженерія даних, автоматизація, й навіть програмування та кодування мікроконтролерів.

Але якщо дивитись на це під іншим кутом, то такої мови програмування, яка б була універсальною для усього, не існує. Простим прикладом є розробка драйверів або операційних систем на Python, з цим він справляється погано, причиною цьому слугує обмеженість у продуктивності та відсутності безпечної системи типів.

Свою популярністю у сучасному світі Python здобув завдяки його використанню в машинному навчанні та аналізі даних. У Python присутні багаті та різноманітні інструменти лінійної алгебри, обробки сигналів, розвинутих візуалізацій, різних математичних та статистичних підходів. Високе положення Python в рейтингу серед мов програмування достатньо однозначний сигнал, який вказує на те що мова розвивається та її популярність серед розробників збільшується кожного року [5].

Одним з недоліків Python, з якими ви можете стикнутись це його швидкість виконання коду. Оскільки Python це не компільована мовою програмування, код перш за все компілюється у внутрішній так званий байт код, який наступним кроком виконується безпосередньо інтерпретатором Python. Частіше за все коли використовується Python, результатом є програми, які повільніші в порівнянні з такими мовами, як C. Але в сьогоднішніх реаліях комп’ютери мають настільки велику обчислювальну потужність, порівняно із попередніми десятиліттями, що для переважної більшості програм швидкість розробки має вищі пріоритети аніж

швидкість виконання, а як правило, програмний код на Python пишеться набагато швидше. Також не варто забувати, що Python має можливість легко доповнюватись модулями з використанням C або C++. Подібні модулі використовуються для обробки та виконання окремих частин програми, що створюють інтенсивне навантаження на процесор.

Важливо також окремо розглянути важливість та переваги бібліотек Python для машинного навчання. Дана мова програмування пропонує досить широкий спектр бібліотек необхідних для машинного навчання та аналізу даних, від збирання та обробки даних до навчання та навчання й візуалізацію моделей та їх результатів. Ці бібліотеки мають у своєму арсеналі різноманітні інструменти для виконання складних задач машинного навчання, що допомагає зменшувати затрати часу та зусиль розробників. Також мова програмування підтримує багатозадачність, що дозволяє їй ефективно обробляти великі обсяги даних. Це робить Python ідеальним вибором для задач машинного навчання, які потребують високої продуктивності.

Виходячи із того що наша задача пов’язана з медичною сферою для її вирішення чудово підходять такі бібліотеки Python як scikit-learn, TensorFlow, PyTorch. Окремо можна зупинитись на scikit-learn, ця бібліотека пропонує досить обширний спектр різних алгоритмів машинного навчання, вони добре підходять для задач медичного прогнозування, таких як діагностика захворювань, прогнозування ризиків та аналіз зображень. Алгоритми класифікації, такі як Logistic Regression, Decision Trees та Random Forest, можуть використовуватися для прогнозування ймовірностей того, чи має пацієнт певне захворювання на основі даних про його симптоми, історію хвороби й інше. З іншої сторони алгоритми прогнозування до прикладу як «регресія Кокса» та «регресія градієнтного підсилення», використовуються для прогнозування ризику розвитку певних захворювань, до таких можна віднести інсульт, серцевий напад та як у нашому випадку рак легень.

Підводячи підсумок можна сказати що Python з його простотою, гнучкістю та широким спектром бібліотек чудово підходить для вирішення поставленої задачі прогнозування раку легенів у пацієнтів.

Проводити дослідження було вирішено на онлайн-платформі для машинного навчання та науки про дані Kaggle. Хоча платформа Kaggle не пропонує інтегроване середовище розробки, вона має вбудований редактор Jupyter Notebook, який дозволяє користувачам писати та виконувати код Python безпосередньо на платформі. Це може бути зручно для швидкого моделювання моделей та візуалізації даних, а це саме те що нам необхідно для нашого дослідження. Великою перевагою Kaggle є бібліотеки наборів даних, які користувачі поширюють між собою, їх можна використовувати для тренування та оцінювання моделей машинного навчання [6]. Ці набори даних охоплюють широкий спектр галузей, включно з медичною сферою, а це саме те що нам потрібно. Слід зазначити що за допомогою саме таких бібліотек було знайдено набір даних з яким ми будемо працювати.

1.4 Аналіз оптимальних методів машинного навчання

Мова програмування Python містить у собі велику кількість методів, які використовуються для прогнозування та для різних поставлених задач та мети прогнозування. Оскільки головним завданням роботи є прогнозування ризиків захворіти на рак легенів, які поділяються на три категорії, а саме: низький середній, високий, поставлене завдання є задачею класифікації. Розглянемо найбільш поширені приклади методів прогнозування на Python:

Логістична регресія (Logistic Regression) – це статистичний регресійний метод, основне призначення даного методу це аналіз зв’язку між незалежними один від одного змінними (їх ще називають регресорами або предиктори) та залежною змінною. Основується він на логістичній функції, з огляду математики ця модель дозволяє обчислити ймовірність кожного класу для вхідних даних і потім вибирає варіант з найбільшою ймовірністю [7].

Даний метод знайшов своє застосування у випадках коли потрібно передбачити класифікацію, коли вхідні змінні є категоріальними, що означає що це такі змінні, які можуть набувати фіксовану кількість значень.

Метод опорних векторів (Support Vector Machines – SVM) – один із алгоритмів, які використовуються при машинному навчанні для вирішення задач класифікації та регресії. Принцип роботи даного методу такий, що точки з даними розміщуються на певній гіперплощині та розділяються лініями на класи, при появі нових даних зважаючи на попередній розподіл на гіперплощині, модель визначає класифікацію даних.

Подібний метод добре підходить для задач у яких початковий набір даних є невеликим та має високу різноманітність. Вирізняється високою точністю, ефективністю та гнучкістю але також має труднощі із обчислювальною складністю. Також слід зазначити що модель має досить високу чутливість до аномалій [8].

Випадковий ліс (Random Forest) – являє собою ансамблевий метод машинного навчання, використовують його для задач по типу класифікації та регресії. Робота представленого методу ґрунтується на побудові великої кількості дерев рішень, які виконують своє навчання на різних підмножинах даних.

Основним принципи методу являється Баггінг (Bootstrap Aggregating), де кожне дерево в ансамблі для навчання використовує випадкову множину даних. Характеризується дана модель високою точністю, стійкістю до перенавчання, простотою використання, ефективністю як при великій так і при малій кількості даних, але у той же час модель складна для обчислень [9].

Додаткові дерева (Extra Trees) – це метод машинного навчання сімейства ансамблевих. Сутність даного методу полягає у створенні великої множини дерев рішень та комбінування їхніх прогнозів для отримання найбільш оптимального результату.

Принципи методу базуються на випадковому виборі ознак, тобто випадково підбираються ознаки для розгалуження в кожному з вузлів, таким чином це дає перевагу у стійкості до перенавчання порівняно із наприклад Random Forest, де на кожному кроці розгалуження використовуються усі ознаки. Алгоритм зазвичай використовує глибокі дерева, що означає що дерева мають

велику кількість рівнів, таким чином він підвищує ефективність моделей залежності між різними ознаками та класами.

Перевагами методу можна виділити можливість досягнення високої точності класифікації, стійкість до наявного шуму в даних, ефективність при роботі із великою кількістю даних, проста реалізація та налаштування. Також варто згадати можливість перенавчання, така можливість менше аніж у Random Forest, але усе ще може бути суттєвою при використанні завеликої кількості дерев або занадто глибоких [10].

Дерево рішень (Decision Trees) – це одна із популярних моделей прогнозування та класифікації, чудово підходить для задач із чіткою послідовністю прийняття рішень та чіткими правилами, за якими ці рішення приймаються.

Структура дерева рішень має такі компоненти як вузли та ребра. Безпосередньо вузли представляють рішення, у той час як ребра можливі переходи між цими рішеннями. Початковий стан задачі складається із так званого кореневого вузла – це вершина дерева. Вузли, які не мають вихідного ребра називаються листовими або листям дерева, у них фігурують остаточне вирішення задачі. Розчеплення вузла на двоє, або більше відбувається за рахунок рекурсії на основі значення найкращого атрибута, який розділяє дані на підмножини з однаковими цільовими значеннями, до тих пір поки не будуть сформовані листові вузли. На основі листових вузлів відбувається прогнозування нових даних [11].

Перевагами моделі є її проста та зрозуміла структура, ефективність при великій кількості даних, нечутливість до відсутніх значень у даних. Що особливо слід виділити для теми нашого дослідження це те, що дерева рішень одні з найкращих моделей підходять для медичного діагностування захворювань на основі симптомів пацієнтів. Але все ж таки мабуть найбільшою перевагою дерев рішень є можливості їх візуалізації, причому інструментарій для візуалізації досить обширний, що дозволяє не лише легко зрозуміти дані та результати, а ще й отримати красивий графік, але при цьому є й свої недоліки, до яких зокрема

відноситься високі ризики до перенавчання при занадто детальному дереві, також модель має високу чутливість до шуму в даних.

AdaBoost (Adaptive Boosting) – це метод машинного навчання, який належить до сімейства бустингу. Він використовується для покращення продуктивності класифікаторів шляхом послідовного навчання декількох слабких класифікаторів та поєднують їх задля отримання сильнішого класифікатора.

Алгоритм використовує ітеративний процес, під час якого навчаються слабкі класифікатори, кожному окремому класифікатору присвоюється значення ваги, яка в свою чергу демонструє його продуктивність. Під час навчання фокусування відбувається на складних прикладах, на яких слабкі класифікатори майже завжди помиляються. Також алгоритм навчається на своїх попередніх помилках, корегуючи ваги кожної нової ітерації і так до тих пір поки не буде досягнуто мінімальної похибки чи обмежень в кількості ітерацій. Слід відмітити даний метод для покращення продуктивності слабких класифікаторів, він є доволі простим та зрозумілим у використанні, а також має гнучкість у вигляді доступності комбінації з різними класифікаторами. В той же час даний алгоритм має чутливість до шуму та схильність до перенавчання [12].

XGB Classifier – це метод який базується на так званому градієнтному бустингу, який в свою чергу використовується для побудови ансамблю дерев рішень.

Послідовне та поступове навчання слабких моделей дерев рішень та виправлення помилок попередників, зведення таких помилок до мінімуму, завдяки навчанню нових моделей на залишкових помилках – ось це і є принцип роботи даного методу. У процесі обробки даних використовується розподілене обчислення, завдяки чому досягається швидкість даних розрахунків. Одними із плюсів даного методу відзначається його точність та можливість різносортного використання регуляції задля уникнення небажаного перенавчання, до якого так схильний цей метод. Тому важливість регулювання параметрів моделі є запорукою успішного її навчання, це важливо брати до уваги при моделюванні [13].

Багатошаровий перцептрон (Multi-Layer Perceptron) – один із популярних методів машинного навчання, який ще іноді називають нейронна мережа з повним зв’язком. Назва говорить сама за себе, адже використовує для класифікації та прогнозування штучну нейронну мережу з кількома шарами.

Структура такого методу це багатошарова нейронна мережа, де присутні вхідний шар, приховані шари та вихідний шар. Слід зазначити що вхідний шар приймає вхідні данні, у прихованих шарах проводяться обчислення, а от уже на вихідному шарі ми отримуємо результат, в нашому випадку згенерований прогноз класифікації який базується на обчисленнях у попередніх шарах. Мета алгоритму звести до мінімум помилку класифікації, що означає зробити так, щоб відмінність між прогнозованими та фактичними класами зводилась до нуля. Однією з вагомих переваг методу є те що він уміє встановлювати складні залежності між даними, які на перший погляд не є очевидними, також зарекомендував себе як високоефективний у вирішенні задач типу регресії та класифікації. До недоліків методу можна віднести високі вимоги до обчислювальних ресурсів та схильність до перенавчання при умові що кількість даних замала [14, 15].

1.5 Висновки

У цьому розділі було проведено аналіз проблематики предметної області, а саме проблеми лікування та діагностування раку легень, який показав що сучасні методи діагностування виявляють рак лише на пізніх стадіях, що впливає на високу смертність, тому необхідно створити новий метод, який буде прогнозувати та попереджати ризики раку легень для вчасного реагування. Розглянуто основні ризики, які впливають на розвиток даної хвороби. Також досліджено мову програмування Python та середовище розробки Kaggle та їх переваги у контексті машинного навчання. Також проведено аналіз різних методів машинного навчання, для створення інформаційної технології передбачення ризиків раку легень та обрано наступні:Random Forest, AdaBoost, Extra Trees, Decision Tree, XGBoost та Multi-Layer Perceptron.

2 РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ

2.1 Попереднє оброблення даних

Для проведення дослідження було обрано набір даних у Kaggle, що має назву «Lung Cancer Prediction» та опублікований користувачем «The Devastator». Даний датасет має відкритий доступ для загального використання на платформі Kaggle [16]. Представлений датасет містить у собі широкий спектр даних про пацієнтів лікарень з ризиками захворіти раком легень, які надає журнал Nature Medicine. Серед стовпців із параметрами можна виділити такі як вік, вплив забрудненого повітря, куріння, вживання алкоголю, професійні ризики, генетичні ризики та ще 19 інших категорій, кількість записаних пацієнтів дорівнює 1000 (рис. 2.1).

index	Patient Id	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	...	Fatigue	Weight Loss	Shortness of Breath	Wheezing
0	P1	33	1	2	4	5	4	3	2	...	3	4	2	2
1	P10	17	1	3	1	5	3	4	2	...	1	3	7	8
2	P100	35	1	4	5	6	5	5	4	...	8	7	9	2
3	P1000	37	1	7	7	7	7	6	7	...	4	2	3	1
4	P101	46	1	6	8	7	7	7	6	...	3	2	4	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	P995	44	1	6	7	7	7	7	6	...	5	3	2	7
996	P996	37	2	6	8	7	7	7	6	...	9	6	5	7
997	P997	25	2	4	5	6	5	5	4	...	8	7	9	2
998	P998	18	2	6	8	7	7	7	6	...	3	2	4	1
999	P999	47	1	6	5	6	5	5	4	...	8	7	9	2

Рисунок 2.1 – Приклад ознак у наборі даних

Розглянемо повний список ознак, які є у нашому наборі даних, зробимо їхній опис та визначимо тип даних.

Отож даний датасет включає у себе такі колонки:

- Index: Індекс рядка (Numeric);
- Patient Id: ідентифікаційний номер пацієнта (Numeric);

- Age: Вік пацієнта. (Numeric);
- Gender: стать пацієнта. (Categorical);
- Air Pollution: рівень впливу забруднення повітря на пацієнта. (Categorical);
- Alcohol use: рівень вживання алкоголю пацієнтом. (Categorical);
- Dust Allergy: рівень алергії на пил у пацієнта. (Categorical);
- OccuPational Hazards: рівень професійних ризиків пацієнта. (Categorical);
- Genetic Risk: рівень генетичного ризику пацієнта. (Categorical);
- Chronic Lung Disease: рівень хронічного захворювання легенів у пацієнта. (Categorical);
- Balanced Diet: рівень збалансованості дієти пацієнта. (Categorical);
- Obesity: рівень ожиріння пацієнта. (Categorical);
- Smoking: рівень куріння пацієнта. (Categorical);
- Passive Smoker: рівень пасивного куріння пацієнта. (Categorical);
- Chest Pain: рівень болю в грудях пацієнта. (Categorical);
- Coughing of Blood: рівень відкашлювання крові пацієнта. (Categorical);
- Fatigue: рівень втоми пацієнта. (Categorical);
- Weight Loss: рівень втрати ваги пацієнта. (Categorical);
- Shortness of Breath: рівень задишки пацієнта. (Categorical);
- Wheezing: рівень хрипів у пацієнта. (Categorical);
- Swallowing Difficulty: рівень труднощів ковтання пацієнта. (Categorical);
- Clubbing of Finger Nails: рівень збивання нігтів пацієнта. (Categorical).

Зробивши візуальний аналіз наявних ознак, можемо впевнено сказати що індекс (Index) та ідентифікаційні номера пацієнтів (Patient Id) не потрібні для подальшої роботи, тому необхідно провести очищення (рис. 2.2).

```
df.drop(columns=['index', 'Patient Id'], axis=1, inplace=True)
df
```

	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	Balanced Diet	Obesity
0	33	1	2	4	5	4	3	2	2	4
1	17	1	3	1	5	3	4	2	2	2
2	35	1	4	5	6	5	5	4	6	7
3	37	1	7	7	7	7	6	7	7	7
4	46	1	6	8	7	7	7	6	7	7
...	...	...	...	...	...	...	...	...	...	...
995	44	1	6	7	7	7	7	6	7	7
996	37	2	6	8	7	7	7	6	7	7
997	25	2	4	5	6	5	5	4	6	7
998	18	2	6	8	7	7	7	6	7	7
999	47	1	6	5	6	5	5	4	6	7

Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swallowing Difficulty	Clubbing of Finger Nails	Frequent Cold	Dry Cough	Snoring	Level
3	4	2	2	3	1	2	3	4	Low
1	3	7	8	6	2	1	7	2	Medium
8	7	9	2	1	4	6	7	2	High
4	2	3	1	4	5	6	7	5	High
3	2	4	1	4	2	4	2	3	High
...	...	...	...	...	...	...	...	...	...
5	3	2	7	8	2	4	5	3	High
9	6	5	7	2	4	3	1	4	High
8	7	9	2	1	4	6	7	2	High
3	2	4	1	4	2	4	2	3	High
8	7	9	2	1	4	6	7	2	High

Рисунок 2.2 – Результат очищення даних

Далі необхідно дослідити набір на наявність аномалій, або малоймовірних значень, які в майбутньому можуть негативно вплинути при навчанні моделей. Оскільки наш датасет містить малу кількість значень достатньо буде використати функцію describe(), щоб візуально у таблиці визначити наявність аномалій та їхнє місце розташування у датасеті (рис. 2.3).

```
df.describe()
```

	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	Balanced Diet	Obesity
count	1000.000000	1000.000000	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	37.174000	1.402000	3.8400	4.563000	5.165000	4.840000	4.580000	4.380000	4.491000	4.465000
std	12.005493	0.490547	2.0304	2.620477	1.980833	2.107805	2.126999	1.848518	2.135528	2.124921
min	14.000000	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	27.750000	1.000000	2.0000	2.000000	4.000000	3.000000	2.000000	3.000000	2.000000	3.000000
50%	36.000000	1.000000	3.0000	5.000000	6.000000	5.000000	5.000000	4.000000	4.000000	4.000000
75%	45.000000	2.000000	6.0000	7.000000	7.000000	7.000000	7.000000	6.000000	7.000000	7.000000
max	73.000000	2.000000	8.0000	8.000000	8.000000	8.000000	7.000000	7.000000	7.000000	7.000000

Coughing of Blood	Fatigue	Weight Loss	Shortness of Breath	Wheezing	Swallowing Difficulty	Clubbing of Finger Nails	Frequent Cold	Dry Cough	Snoring
1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
4.859000	3.856000	3.855000	4.240000	3.777000	3.746000	3.923000	3.536000	3.853000	2.926000
2.427965	2.244616	2.206546	2.285087	2.041921	2.270383	2.388048	1.832502	2.039007	1.474686
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
3.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
4.000000	3.000000	3.000000	4.000000	4.000000	4.000000	4.000000	3.000000	4.000000	3.000000
7.000000	5.000000	6.000000	6.000000	5.000000	5.000000	5.000000	5.000000	6.000000	4.000000
9.000000	9.000000	8.000000	9.000000	8.000000	8.000000	9.000000	7.000000	7.000000	7.000000

Рисунок 2.3 – Результат виконання функції describe()

За результатами можемо бачити що аномалій не виявлено. Оскільки майже усі значення у наборі є категоріальними, де рівні записано у числовому значенні, мінімальними значеннями для категорій є 1, а максимальні значення не перевищують 9.

Окремо потрібно звернути увагу на вік та гендерну ознаку. Оскільки люди розділяються на чоловіків та жінок, для зручності вони позначаються цифрами такими як 1 та 2 відповідно. Щодо віку, з результатів даної функції можемо зробити висновок, що мінімальний вік пацієнта у даному наборі становить 14 років, а максимальний 73 роки.

Далі необхідно перевірити дані на наявність пропущених значень, тобто такі, які позначаються як NaN, щоб враховувати їх відсутність надалі та при необхідності почистити параметри, які мають занадто багато таких значень й

через що не будуть представляти цінності для прогнозування. Виконувати таку перевірку будемо за допомогою функції info() (рис. 2.4).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 24 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1000 non-null   int64
 1   Gender                    1000 non-null   int64
 2   Air Pollution             1000 non-null   int64
 3   Alcohol use               1000 non-null   int64
 4   Dust Allergy              1000 non-null   int64
 5   OccuPational Hazards      1000 non-null   int64
 6   Genetic Risk              1000 non-null   int64
 7   chronic Lung Disease      1000 non-null   int64
 8   Balanced Diet             1000 non-null   int64
 9   Obesity                   1000 non-null   int64
 10  Smoking                   1000 non-null   int64
 11  Passive Smoker            1000 non-null   int64
 12  Chest Pain                1000 non-null   int64
 13  Coughing of Blood         1000 non-null   int64
 14  Fatigue                   1000 non-null   int64
 15  Weight Loss               1000 non-null   int64
 16  Shortness of Breath       1000 non-null   int64
 17  Wheezing                  1000 non-null   int64
 18  Swallowing Difficulty     1000 non-null   int64
 19  Clubbing of Finger Nails  1000 non-null   int64
 20  Frequent Cold             1000 non-null   int64
 21  Dry Cough                 1000 non-null   int64
 22  Snoring                   1000 non-null   int64
 23  Level                     1000 non-null   object
dtypes: int64(23), object(1)
memory usage: 187.6+ KB
```

Рисунок 2.4 – Результат виконання функції info()

За результатами функції info() можна зробити такі висновки, що у даному наборі даних відсутні пусті значення, тобто усі наявні параметри мають 100% заповнення для кожного пацієнта. Також функція показала кількість параметрів, тип даних для кожного параметру та кількість параметрів з унікальними типами даних, а саме 23 параметри зі значеннями цілих чисел та 1 параметр із значенням об’єкту. Кількість пам’яті, яку займає набір даних становить 187.6 KB.

У параметрі під назвою «Level», із типом даних об’єкт, записані рівні ризику захворіти раком легень (рис. 2.5).

```
print('Cancer Levels: ', df['Level'].unique())
```

```
Cancer Levels:  ['Low' 'Medium' 'High']
```

Рисунок 2.5 – Унікальні значення для параметру «Level»

З рисунку 2.5 бачимо, що у нас є три категорії ризику захворювання, а саме низький, середній та високий. Для розв’язання задач кваліфікації деякі моделі машинного навчання сприймають лише числові значення, тому необхідно перетворити унікальні значення параметру «Level» на числові (рис. 2.6).

```
# Replace "level" with Integer
print('\n')
print('Cancer Levels: ', df['Level'].unique())

# Replacing levels with int
df["Level"].replace({'High': 2, 'Medium': 1, 'Low': 0}, inplace=True)
print('Cancer Levels: ', df['Level'].unique())

print('\nColumns in dataframe: \n', df.columns)
```

```
Cancer Levels:  ['Low' 'Medium' 'High']
Cancer Levels:  [0 1 2]

Columns in dataframe:
 Index(['Age', 'Gender', 'Air Pollution', 'Alcohol use', 'Dust Allergy',
       'OccuPational Hazards', 'Genetic Risk', 'chronic Lung Disease',
       'Balanced Diet', 'Obesity', 'Smoking', 'Passive Smoker', 'Chest Pain',
       'Coughing of Blood', 'Fatigue', 'Weight Loss', 'Shortness of Breath',
       'Wheezing', 'Swallowing Difficulty', 'Clubbing of Finger Nails',
       'Frequent Cold', 'Dry Cough', 'Snoring', 'Level'],
      dtype='object')
```

Рисунок 2.6 – Перетворення унікальних значень параметру «Level» у числові

2.2 Розвідувальний аналіз даних

Далі проведемо аналіз розподілу даних відносно певних параметрів у датасеті.

На рисунку 2.7 зображено діаграми Boxplot та QQ-plot, які показують розподіл даних відповідно до віку пацієнтів.

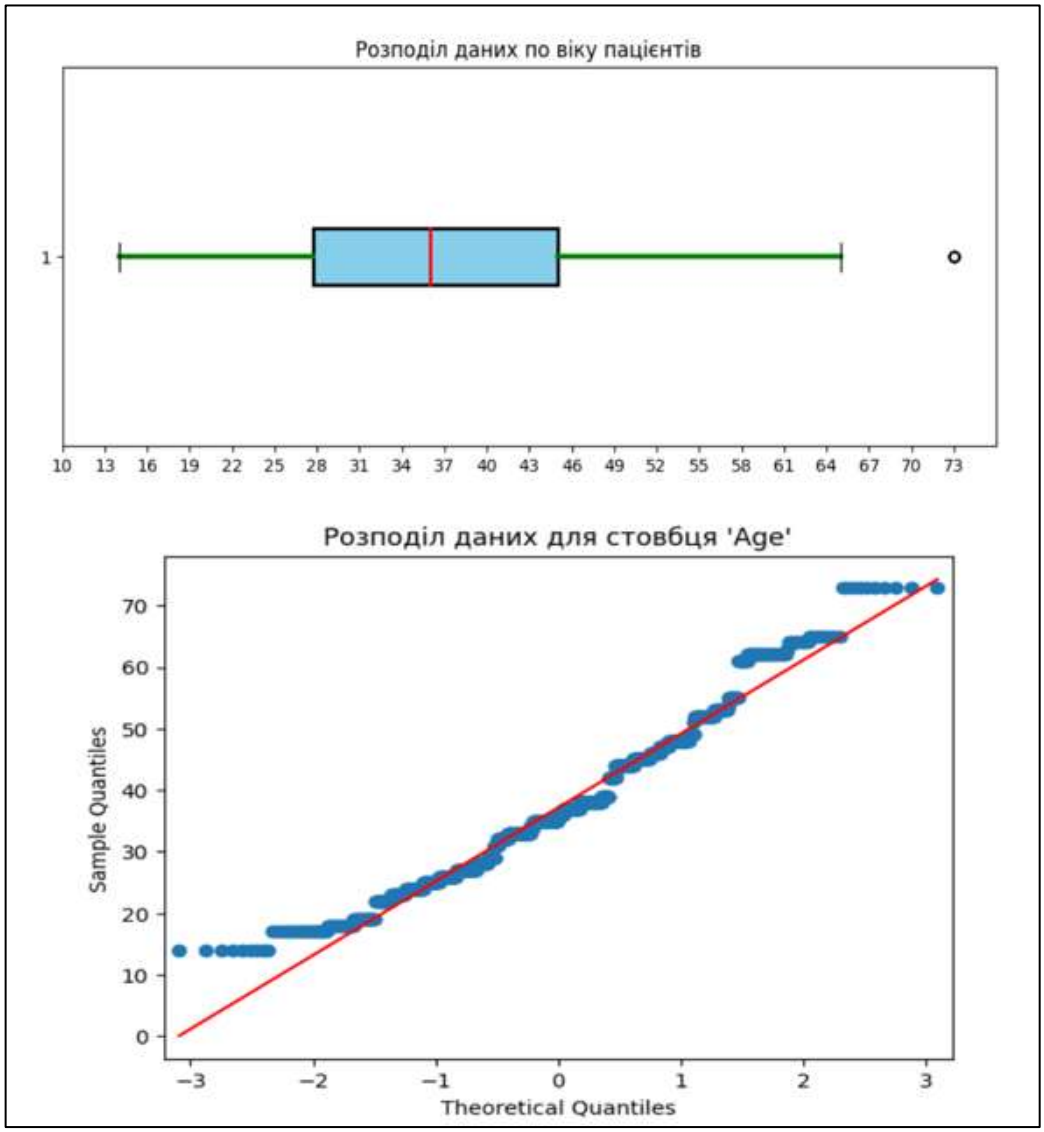


Рисунок 2.7 – Діаграми розподілу даних відповідно до віку пацієнтів

На діаграмах можемо побачити, що більшість даних при розподілі за віком знаходяться у межах від 27 та 44 років, що свідчить про те що вибірка даних здебільшого направлена на людей середнього віку. Також на діаграмі Boxplot

помітно що пацієнт із максимальним віком 73 роки це єдине значення віку людей у проміжку після 65 років, це пояснюється розмірністю набору даних, адже кількість даних у тисячу занадто мала для повної вибірки за віком до 80 років, але навіть при таких значеннях бачимо що на діаграмі QQ-plot кількість даних для пацієнтів середнього віку розподілена рівномірно, що свідчить про відповідність теоретичному розподілу.

Наступним кроком буде визначення кореляції між параметрами, та побудови кореляційних матриць. Для визначення кореляцію будемо використовувати функцію corr() (рис. 2.8).

```
df_corr = df.corr(method='spearman')
df_corr
```

	Age	Gender	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	Genetic Risk	chronic Lung Disease	Balanced Diet	Obesity
Age	1.000000	-0.165672	0.062103	0.139518	0.040532	0.054822	0.040005	0.108806	-0.033680	0.060273
Gender	-0.165672	1.000000	-0.243384	-0.236563	-0.213486	-0.181904	-0.223501	-0.212435	-0.101899	-0.118361
Air Pollution	0.062103	-0.243384	1.000000	0.694256	0.640920	0.553041	0.654058	0.597733	0.496763	0.579275
Alcohol use	0.139518	-0.236563	0.694256	1.000000	0.837587	0.851185	0.848291	0.750601	0.577756	0.624148
Dust Allergy	0.040532	-0.213486	0.640920	0.837587	1.000000	0.875420	0.819746	0.691008	0.647398	0.680932
OccuPational Hazards	0.054822	-0.181904	0.553041	0.851185	0.875420	1.000000	0.877435	0.859535	0.681119	0.712281
Genetic Risk	0.040005	-0.223501	0.654058	0.848291	0.819746	0.877435	1.000000	0.791538	0.637459	0.702909
chronic Lung Disease	0.108806	-0.212435	0.597733	0.750601	0.691008	0.859535	0.791538	1.000000	0.624321	0.588089
Balanced Diet	-0.033680	-0.101899	0.496763	0.577756	0.647398	0.681119	0.637459	0.624321	1.000000	0.638906
Obesity	0.060273	-0.118361	0.579275	0.624148	0.680932	0.712281	0.702909	0.588089	0.638906	1.000000

Рисунок 2.8 – Значення кореляції між параметрами

На рисунках 2.9 та 2.10 продемонстровані матриці кореляції та їх теплові мапи. Особливістю таких матриць є те що на них можна візуально побачити залежності між параметрами завдяки палітрі кольорів, які відповідають рівням залежності, чим більш насичений колір відповідно тим більша залежність.

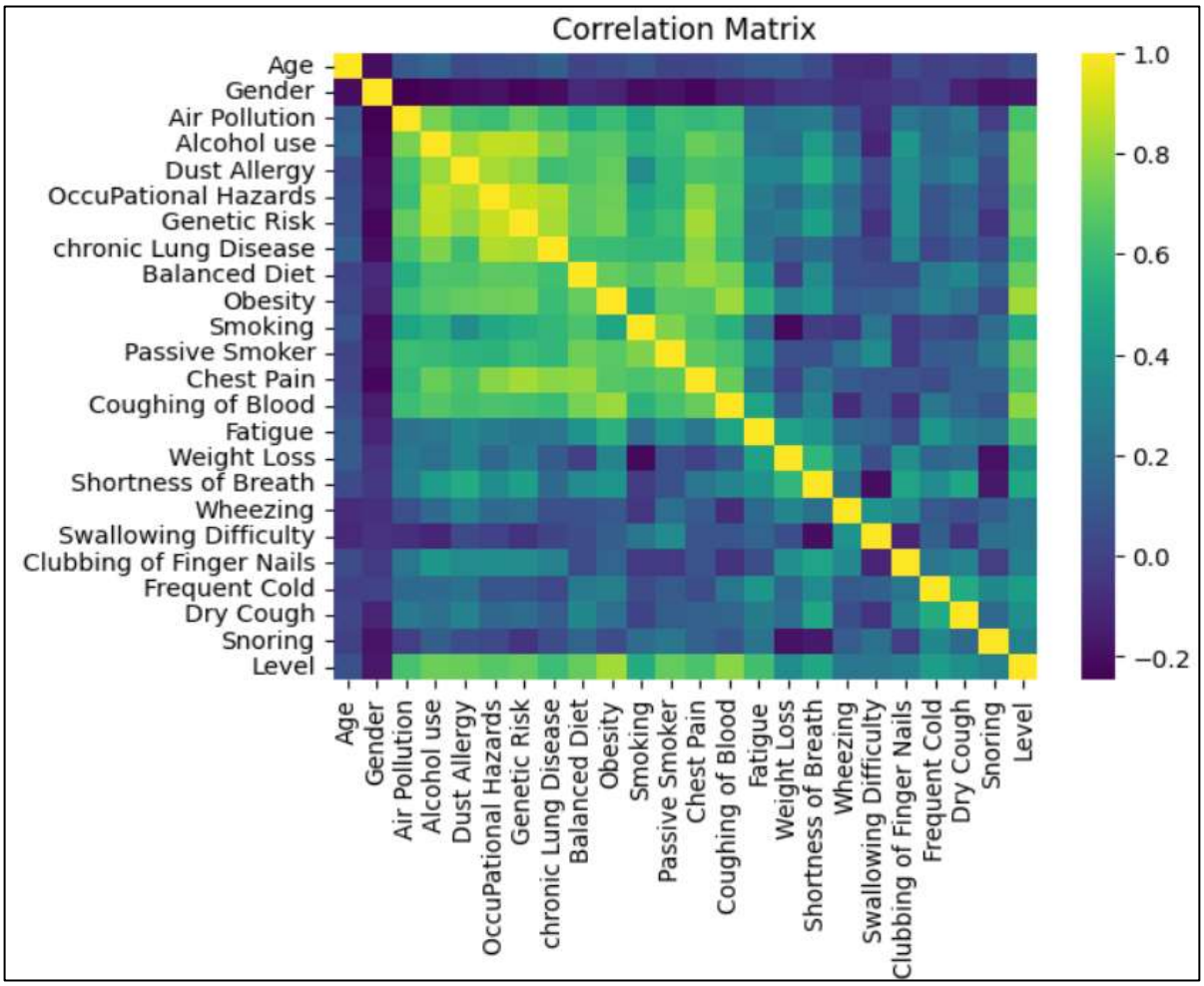


Рисунок 2.9 – Теплова мапа матриці кореляції

Такий тип матриць є досить простим у побудові за допомогою відповідних бібліотек та зручним у використанні, адже не потребує детального дослідження оцінки кореляції використовуючи сухі числа, а дозволяє при першому ж перегляді оцінити залежності між певними параметрами, та прочитати загальну картину наявних даних. Це є дуже корисним як для професіоналів так і для початківців. Також перевагою таких матриць є те що вони інтуїтивно зрозумілі навіть тим людям, які не розбираються в статистиці, що допомагає таким людям пояснювати результати дослідження набагато швидше [17].

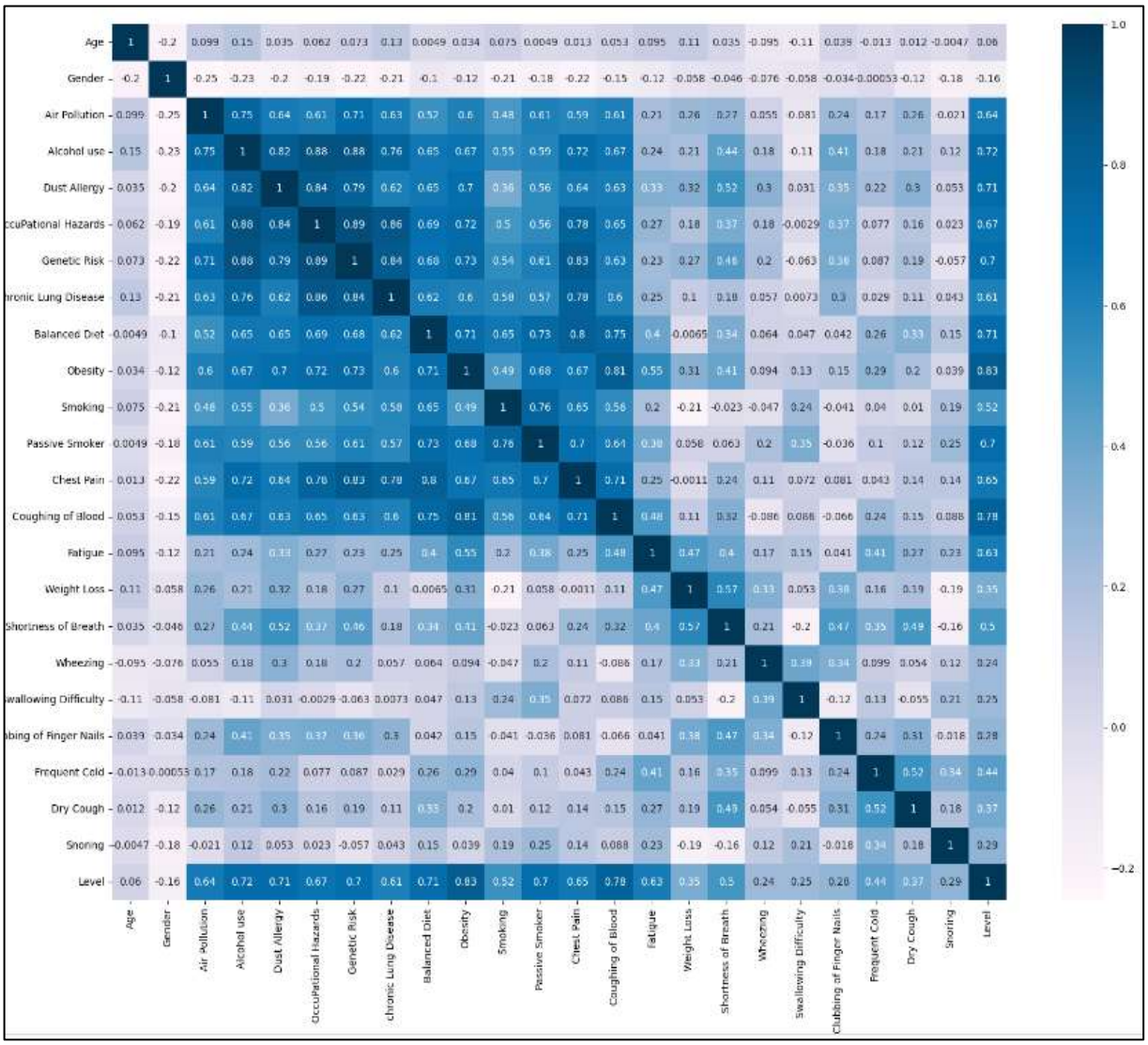


Рисунок 2.10 – Теплова мапа матриці кореляції із значеннями

З рисунку 2.10 можна зробити висновок що залежність прослідковується у верхній лівій частині матриці, що означає що між собою корелюються такі параметри як рівень забрудненості повітря (Air Pollution), рівень вживання алкоголю(Alcohol use), рівень алергії на пил (Dust Allergy), рівень куріння (Smoke), рівень ожиріння (Obesity), рівень пасивного куріння (Passive Smoker). Залежність подібних параметрів один від одного в цілому прогнозована, адже погані звички такі як куріння чи вживання алкоголю доповнюють один одного, пилова алергія можу бути викликана забрудненим повітрям. Якщо до прикладу людина довгий період часу знаходиться під впливом іншої людини, яка часто курить, тобто являється пасивним курцем, то шанси що така людина сама стане палити будуть значно вищі ніж у людей, які не піддаються такому впливу.

Не менш цікавою залежністю від інших параметрів володіє параметр рівня ризику захворіти на рак легень (Level). Найбільша кореляція спостерігається на таких параметрах як ожиріння (Obesity), рівень вживання алкоголю (Alcohol use), генетичні ризики (Genetic Risk), рівень відкашлювання крові (Coughing of Blood). Для цих показників кореляція коливається від 0.7 до 0.83, що є достатнім приводом вважати що така залежність відповідає дійсності.

Далі спробуємо побудувати графік із рівень розподілу даних відповідно до віку пацієнтів та їх ризиків захворіти (рис. 2.11).

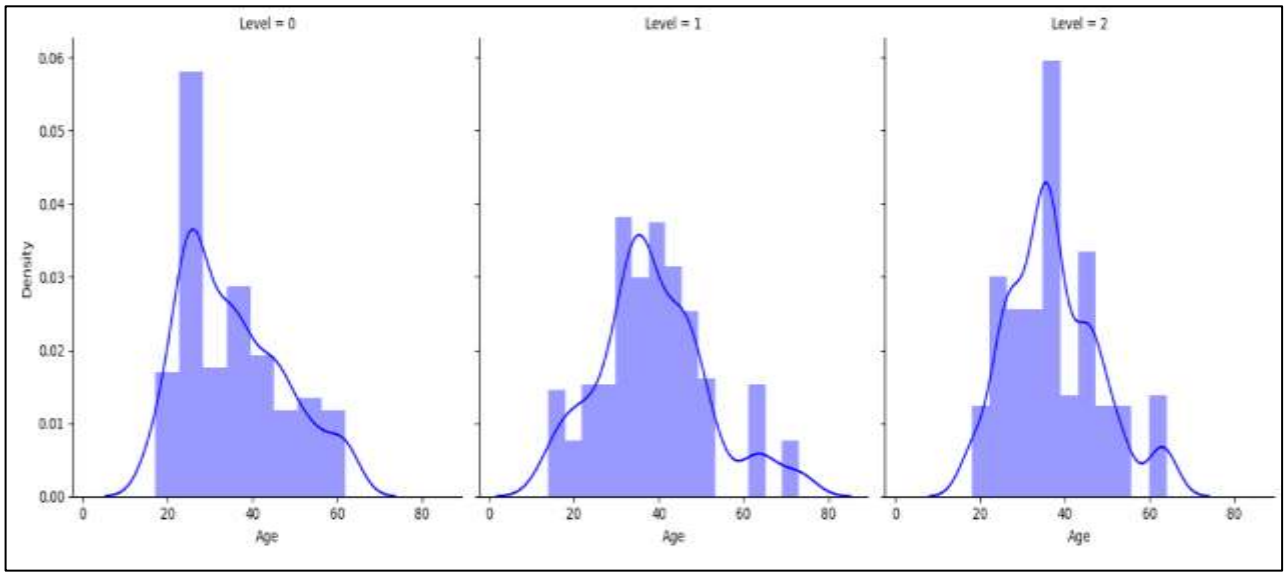


Рисунок 2.11 – Графік розподілу кількості пацієнтів за віком відповідно до ризику захворіти раком легень.

З візуалізації видно, що у нашій вибірці кількість людей віком 20 років має найменші ризики захворіти, середні ризики рівномірно розподілені між людьми віком між від 30 до 45 років, тоді як кількість людей із високими ризиками захворювання припадає більшістю на пацієнтів віком 35-40 років. Це може означати що наші дані здебільшого розподілені відповідно світовій тенденції по захворюванню на рак. Як ми вже знаємо молодь на такий вид раку, а саме раку легенів, хворіє набагато рідше, оскільки молодий організм краще бориться та більш захищений від можливих збудників такої хвороби. Також слід відмітити, що багато головних чинників, які називають лікарі при діагностуванні хвороби, таких як куріння, пасивне куріння, вживання алкоголю, забруднене повітря,

ожиріння і так далі, мають підвищені ризики саме при регулярному впливі на людину та людський організм. Це означає що навіть якщо ми візьмемо до прикладу людину, яка почала курити у 20 років та якій на даний момент 25, то порівняно із людиною, яка також почала курити у 20 років, але на даний момент їй 45, ризики викликати рак легень у них будуть різні, але не через вік сам по собі, а через час впливу сигарет на організм та на легені зокрема. Тобто стаж куріння у людини з віком збільшується, якщо вона не покидає цю погану звичку. Така ж ситуації і з іншими параметрами, через які ризики захворіти збільшуються, але при умові довготривалого впливу цих параметрів на організм.

Побудуємо подібний графік до попереднього, але уже на основі гендерної ознаки замість вікової (рис. 2.12).

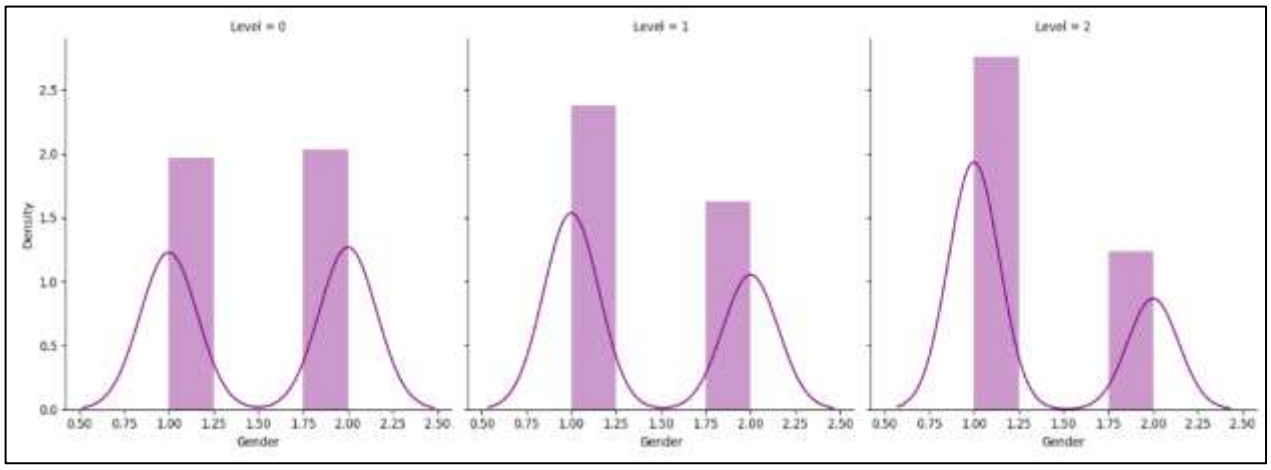


Рисунок 2.12 – Графік розподілу кількості пацієнтів за гендерною ознакою відповідно до ризику захворіти раком легень.

Аналізуючи результати візуалізації гендерної ознаки можемо бачити, що низький ризик майже рівномірно розподілений між жінками та чоловіками із невеликим відхиленням у сторону жінок. У той же самий час при аналізі середнього та високого ризиків чоловіки мають явну перевагу над жінками (чоловіки позначені 1, жінки позначені 2). Це може свідчити про те, що чоловіки більше схильні до впливу негативних параметрів аніж жінки. Це можна пояснити тим що за статистикою кількість курців серед чоловіків набагато вища ніж серед жінок, більше чоловіків працює на промисловому виробництві де зазнають

впливу різних шкідливих речовин починаючи від металургії та заводів закінчуючи різними фермерськими підприємствами, де вони зазнають шкоди для здоров'я за рахунок хімічних компонентів, газу, металургійних відходів, пестицидів та іншого, що використовується на таких підприємствах, дане твердження демонструє графік розподілу на рисунку 2.13.

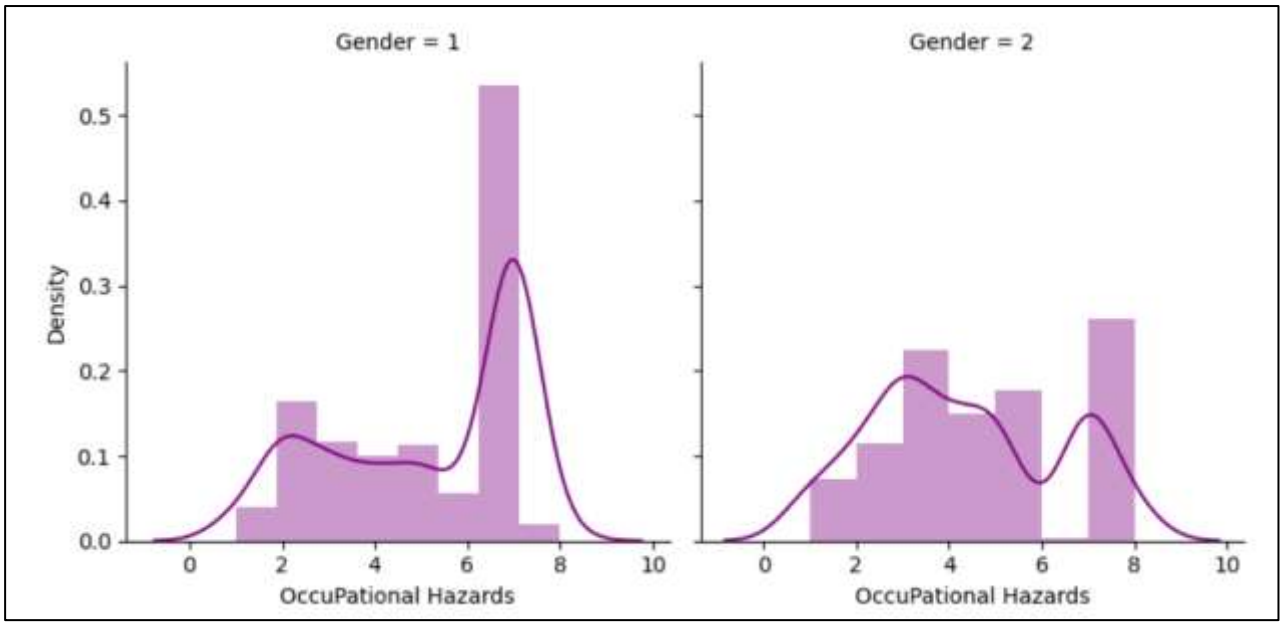


Рисунок 2.13 – Графіки розподілу кількості чоловіків та жінок відповідно до професійного ризику захворіти раком легень.

Переглянемо розподіл по кількості даних у наборі за переліком інших ознак, які наявні у нашому датасеті (рис. 2.14-2.26).

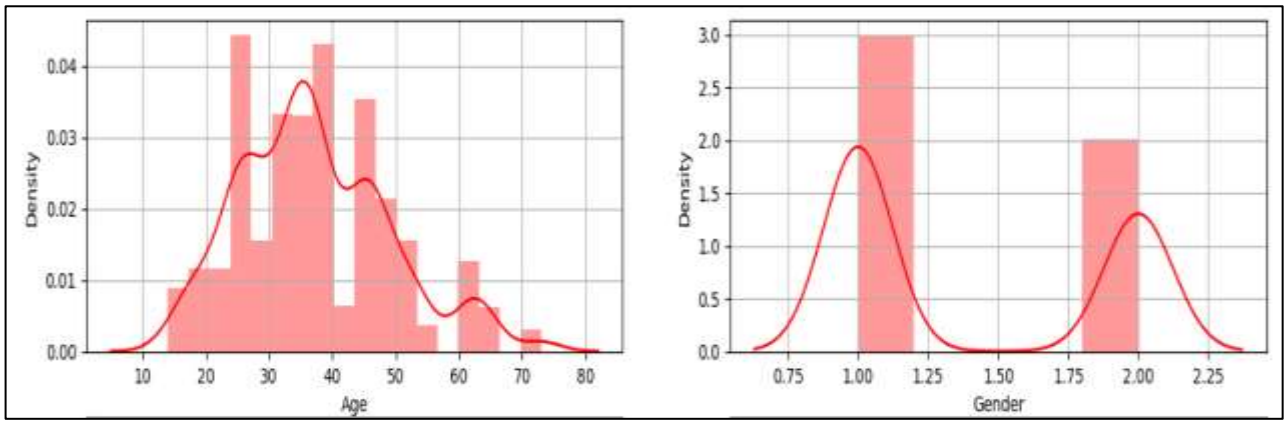


Рисунок 2.14 – Кількісний розподіл вибірки датасету за ознаками віку та статі

На графіку рисунку 2.14 бачимо не дуже рівномірний розподіл за віковою категорією у пацієнтів. Найбільша кількість людей у віці 25 та 35 років. Люди віком 55 років узагалі відсутні, але такий розподіл зумовлений малою кількістю даних у датасеті у противагу великій кількості значень віку від 0 до 100, тому кількість у тисячу значень є недостатньою для рівномірного розподілу у рамках вікової категорії.

На графіку із гендерними ознаками бачимо, що кількість чоловіків переважає жінок, але у той же час жінок не настільки мало, щоб не брати цей параметр до уваги.

Відобразимо кількісний розподіл за ознаками рівня забрудненого повітря та рівня вживання алкоголю (рис. 2.15).

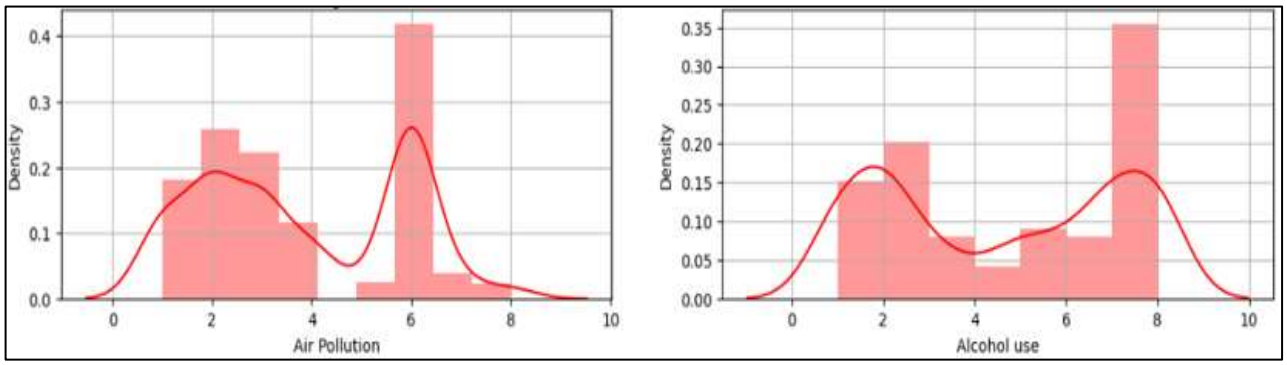


Рисунок 2.15 – Кількісний розподіл вибірки датасету за ознаками рівня забрудненого повітря та рівня вживання алкоголю

Візуалізація рисунку 2.15 показує що вибірка даних припадає на рівень забруднення повітря, у якому мешкають та піддаються впливу пацієнти, більшою своєю частиною на середньо вищий, у той час низький та середні рівні розподілені рівномірно між собою, а кількість пацієнтів із високим рівнем впливу забрудненого повітря є досить низькою та рідкісною ознакою.

Розподіл пацієнтів за рівнем вживання алкоголю стабільний при низьких показниках, зменшується при середніх показниках та дуже сильно зростає при максимальному показнику у 8 балів. Це показує що велика частка людей у вибірці мають серйозні проблеми із залежністю від спиртних напоїв.

Відобразимо кількісний розподіл за ознаками рівня алергії на пил та рівня професійних ризиків (рис. 2.16).

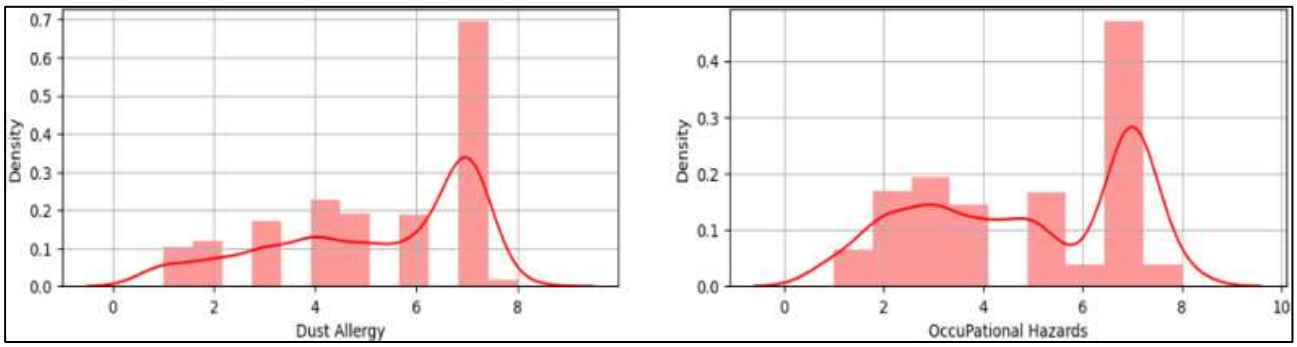


Рисунок 2.16 – Кількісний розподіл вибірки датасету за ознаками рівня алергії на пил та рівня професійних ризиків

Графіки на рисунку 2.16 демонструють що розподіл пацієнтів за параметрами рівня алергії на пил та рівня професійних ризиків схожі між собою. Обидва графіки показують відносно рівномірний розподіл при низьких рівнях показників, у той час як на високому рівні кількість людей різко зростає.

Відобразимо кількісний розподіл за ознаками рівня генетичного ризику та рівня хронічного захворювання легень (рис. 2.17).

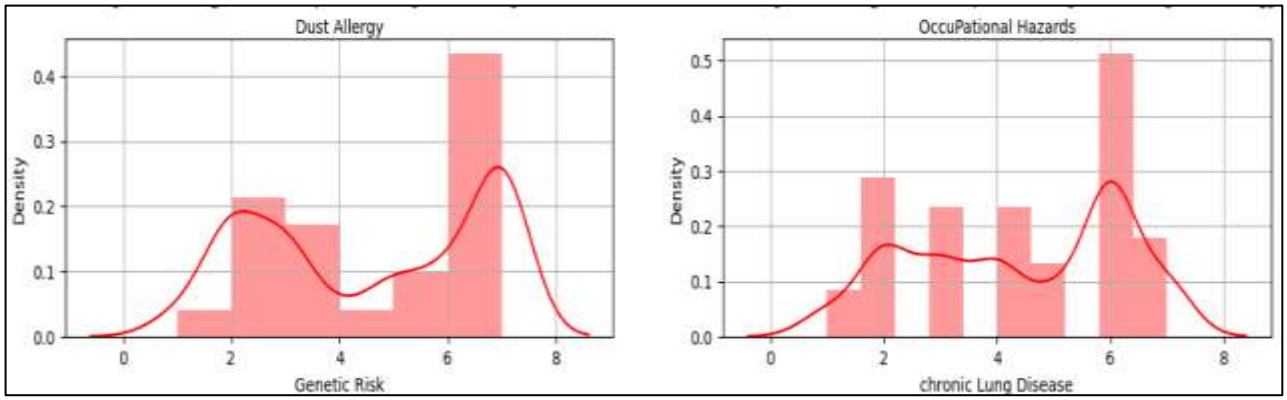


Рисунок 2.17 – Кількісний розподіл вибірки датасету за ознаками рівня генетичного ризику та рівня хронічного захворювання легень

Перший продемонстрований графік на рисунку 2.17 показує велику кількість людей із найвищим рівнем генетичних ризиків, при чому просто високі рівні ризику залишаються у меншій кількісті між усіма рівнями.

Кількість людей із рівнем хронічних захворювань легень розподілена рівномірно між усіма оцінками ризиків окрім 6, де спостерігається кількісна перевага майже у два рази при порівнянні із кожним окремою оцінкою.

Далі відобразимо кількісний розподіл за ознаками рівня збалансованості дієти (правильного харчування) та рівня ожиріння у пацієнтів (рис. 2.18).

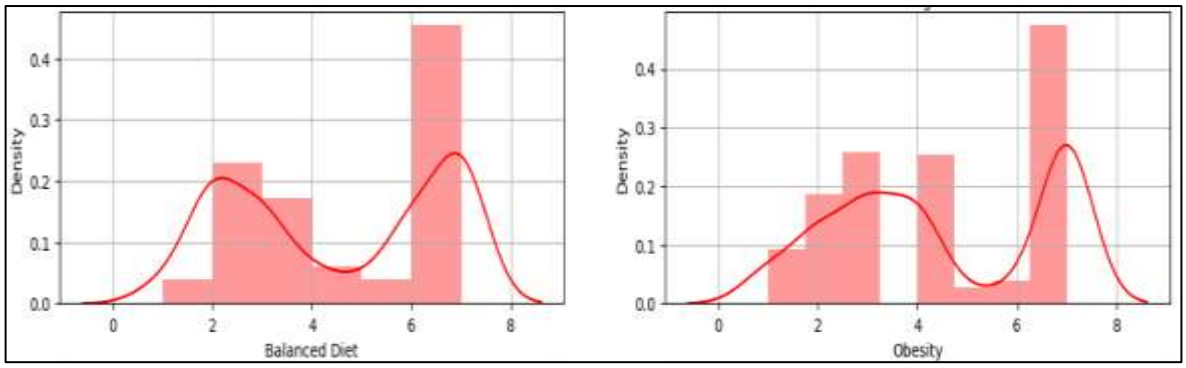


Рисунок 2.18 – Кількісний розподіл вибірки датасету за ознаками рівня збалансованості дієти (правильного харчування) та рівня ожиріння у пацієнтів

Кількісний розподіл зображений на рисунку 2.18 показує що більшість людей харчуються шкідливою для організму їжею, що відповідає помітці на графіку у 7 балів. Такі результати відображення частково пояснюють результати другого графіка, на якому показано рівень ожиріння пацієнтів де люди з високим рівнем зайвої ваги переважають над низькими рівнями.

Далі відобразимо кількісний розподіл за ознаками рівня куріння та рівня пасивного куріння (рис. 2.19).

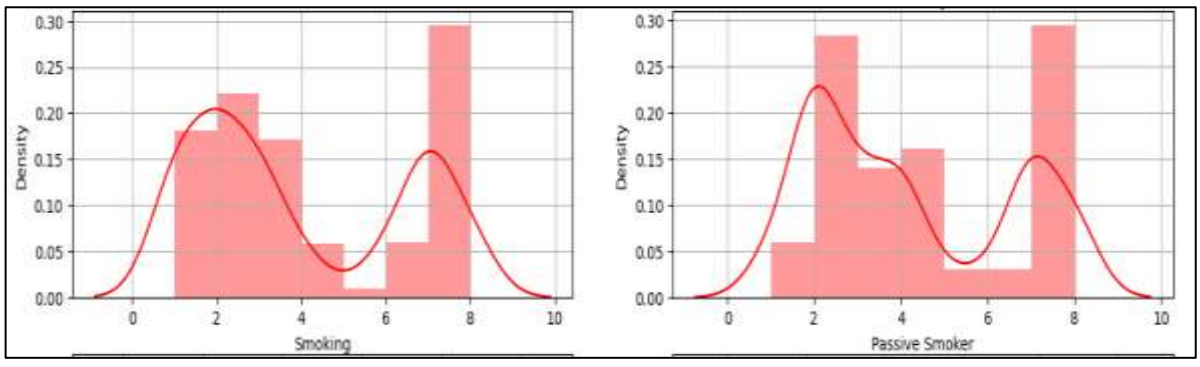


Рисунок 2.19 – Кількісний розподіл вибірки датасету за ознаками рівня куріння та рівня пасивного куріння

Графіки рисунку 2.19 показують рівномірний розподіл між низьким та середнім рівнем впливу сигарет на курців, малу кількість даних про курців із рівнем вищим за середній, та невелику перевагу над низькими на найвищому рівні.

У свою чергу розподіл між рівнями впливу пасивного куріння рівномірно розподілений серед низького рівня та найвищого. Щодо середнього та високого рівнів оцінки то тут кількість менша та дуже мала відповідно.

Наступним кроком відобразимо кількісний розподіл за ознаками рівня болю у грудях та рівня відкашлювання крові (рис. 2.20).

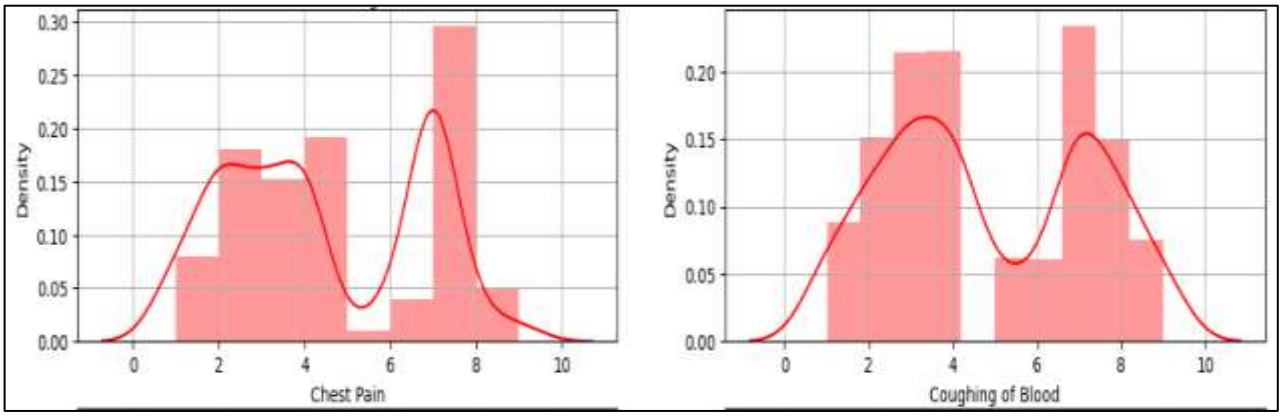


Рисунок 2.20 – Кількісний розподіл вибірки датасету за ознаками рівня болю у грудях та рівня відкашлювання крові

На рисунку 2.20 зображено графіки кількісного розподілу, які демонструють що кількість пацієнтів із середнім рівнем болю у грудях рівномірно розподілена між собою, значення оцінки рівня болю які відповідають 5, 6, та 8 є досить низькими, але при цьому кількість пацієнтів із рівнем болі у грудях 7 є досить високою.

У свою чергу розподіл між рівнями відкашлювання крові є рівномірним для низьких та високих рівнів, але рівномірно низькою для середніх рівнів.

Наступним кроком відобразимо кількісний розподіл за ознаками рівня загальної втоми та рівня втрати ваги у пацієнтів (рис. 2.21).

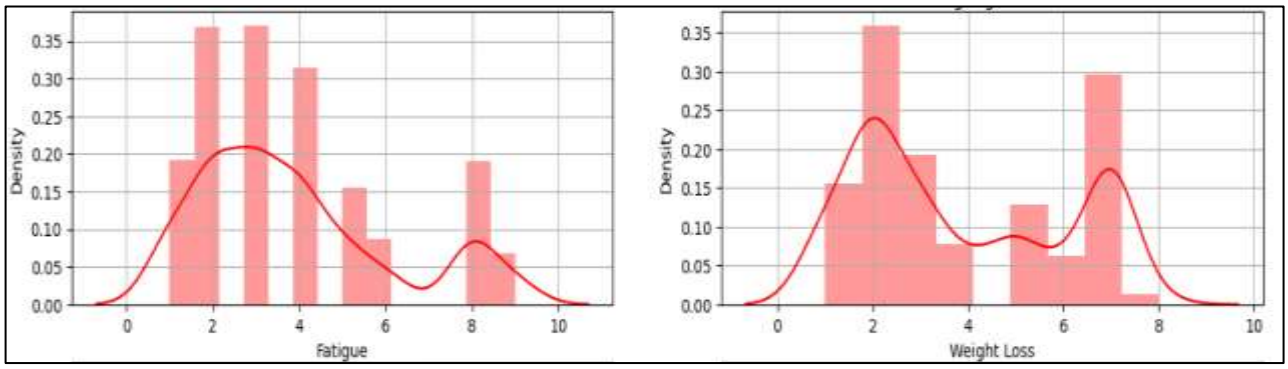


Рисунок 2.21 – Кількісний розподіл вибірки датасету за ознаками рівня загальної втоми та рівня втрати ваги у пацієнтів

Відповідно до рисунку 2.21 можна зробити висновок що втома для пацієнтів у наборі даних є незначною, адже вона рівномірно розподілена між низькими рівнями, а кількість низьких рівнів набагато вища ніж високих. Відповідно до цього розподіл між високими рівнями втоми також є рівномірним, але низьким.

На графіку із візуалізацією рівня втрати ваги бачимо паритет між низьким рівнем та високим рівнем. Середні значення мають меншу кількість порівняно із крайніми значеннями на графіку, але між собою розподілені однаково.

Наступним кроком відобразимо кількісний розподіл за ознаками рівня задишки та рівня хрипіння пацієнтів (рис. 2.22).

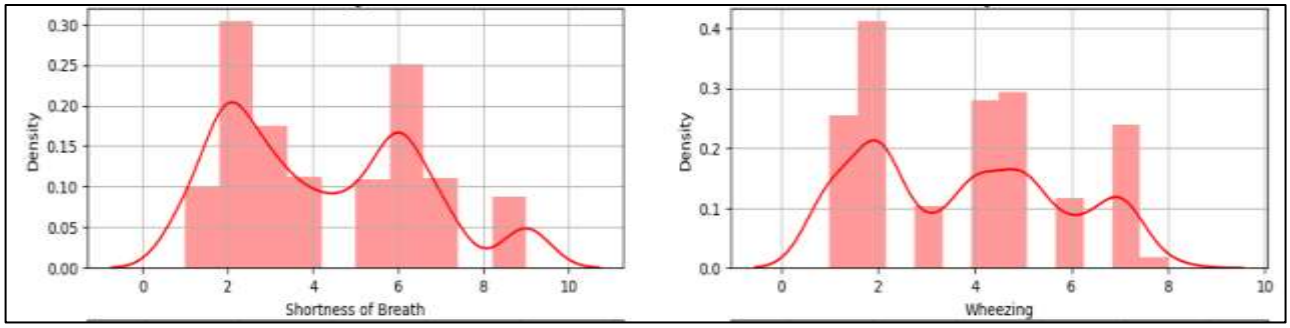


Рисунок 2.22 – Кількісний розподіл вибірки датасету за ознаками рівня задишки та рівня хрипіння пацієнтів

Рисунок 2.22 відображає графіки, які нам демонструють що розподіл кількості пацієнтів за рівнем задишки є низьким та рівномірним для середнього та високого рівня оцінки, але високим та рівномірним для оцінок 2 та 6.

Як показує графік розподілу за параметром рівня хрипіння, то кількість пацієнтів із низьким рівнем переважає своєю більшістю над іншими, але розподіл між середніми та високими рівнями приблизно однаковий. Це означає що вибірка схиляється до низького рівня кількості пацієнтів з недомаганням через задишку чи хрипіння.

Наступним кроком відобразимо кількісний розподіл за ознаками рівня труднощів при ковтанні та рівня булин нігтів у пацієнтів (рис. 2.23).

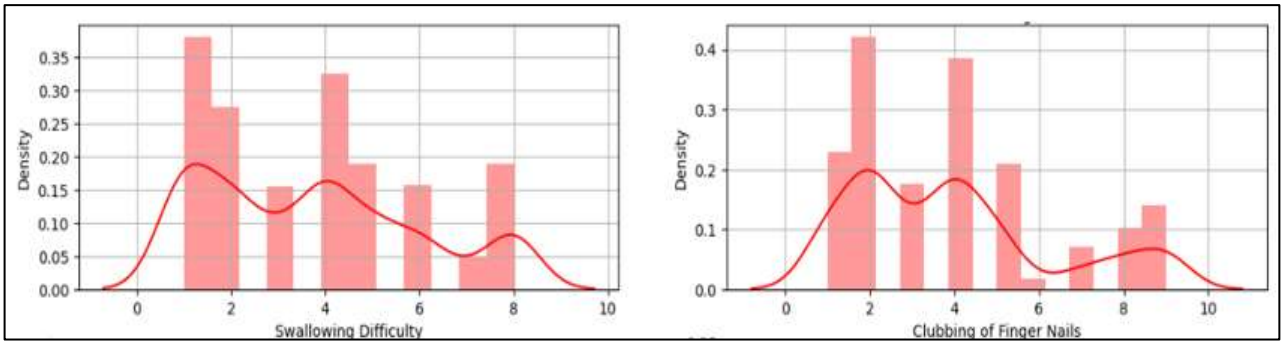


Рисунок 2.23 – Кількісний розподіл вибірки датасету за ознаками рівня труднощів при ковтанні та рівня булин нігтів у пацієнтів

На графіках зображених на рисунку 2.23 бачимо що розподіл у кількості між параметром, який визначає рівень труднощів при ковтанні, досить рівномірний, але має невеличку перевагу на низьких рівнях.

Схожа картина спостерігається і для графіку справа, який відображає рівень булин нігтів у пацієнта, але кількість із низькими і високими рівнями відрізняється від лівого графіка оскільки високих рівнів даного показника набагато менше порівняно із низькими. Варто зазначити що булини нігтів це симптом при якому нігті на руках стають округлими та сферичними, ніби набухають і чим більше вони заокруглюються тим вищій позначений рівень даного симптому. Такі симптоми дуже поширені серед хворих у яких діагностували саме рак легень.

Відобразимо кількісний розподіл за ознаками рівня частоти захворюваності грипом та рівня сухого кашлю (рис. 2.24).

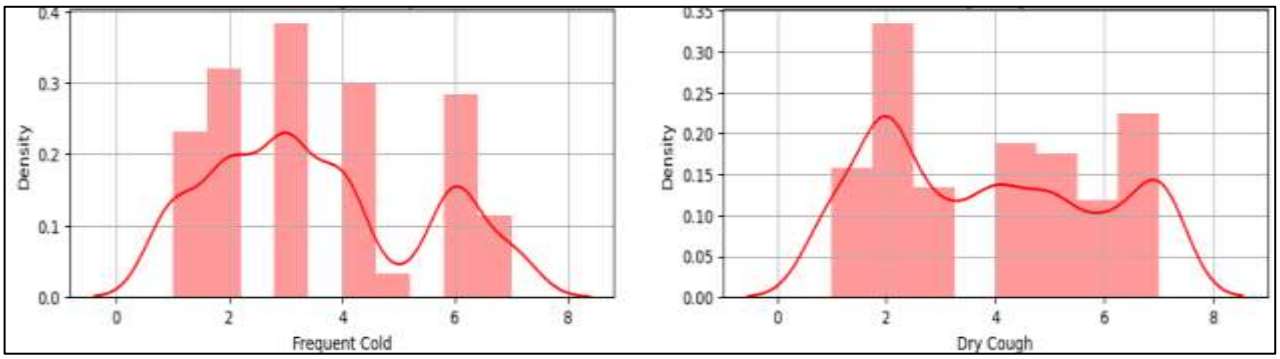


Рисунок 2.24 – Кількісний розподіл вибірки датасету за ознаками рівня частоти захворюваності грипом та рівня сухого кашлю

На рисунку 2.24 на лівому графіку, який відображає рівні частоти при яких пацієнт хворіє на грип, тобто як часто він хворіє на грип, видно що розподіл по кількості здебільшого рівномірний між усіма значеннями окрім 5 де кількість дуже низька.

Правий графік, з ознакою рівня сухого кашлю, показує схожі результати що й лівий, але рівномірність між значеннями тут, на відмінну від першого графіку, порушує значення рівня 2, яке перевищує кількісно усі інші.

Відобразимо кількісний розподіл за ознакою рівня хропіння у пацієнтів (рис. 2.25).

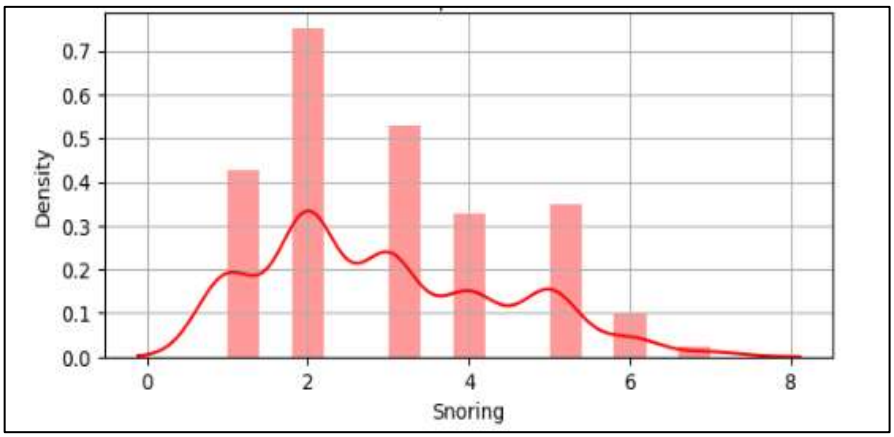


Рисунок 2.25 – Кількісний розподіл вибірки датасету за ознакою рівня хропіння у пацієнтів

Кількісний розподіл відображений на рисунку 2.25 показує що кількість пацієнтів у вибірці із хропінням розподілена приблизно однаково, але піковим значенням по кількості є 2.

Цільовим значенням яке ми будемо прогнозувати являється рівень ризику захворіти раком легень, який у датасеті позначається як Level. Тому для нього потрібно провести ширший аналіз розподілу.

На рисунку 2.26 зображено розподіл кількості унікальних значень для цього параметра, а саме Low, Medium, High, які позначаються як 0, 1 та 2 відповідно.

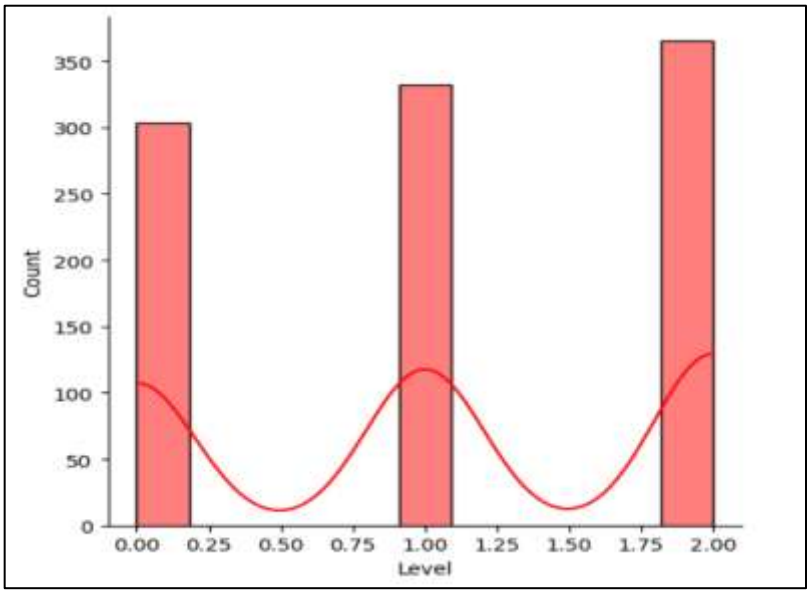


Рисунок 2.26 – Кількісний розподіл вибірки датасету по параметру Level

Даліч на рисунку 2.27 зображено відсоткову кругову діаграму, яка відображає кількісний розподіл унікальних значеннях параметру Level у відсотках.

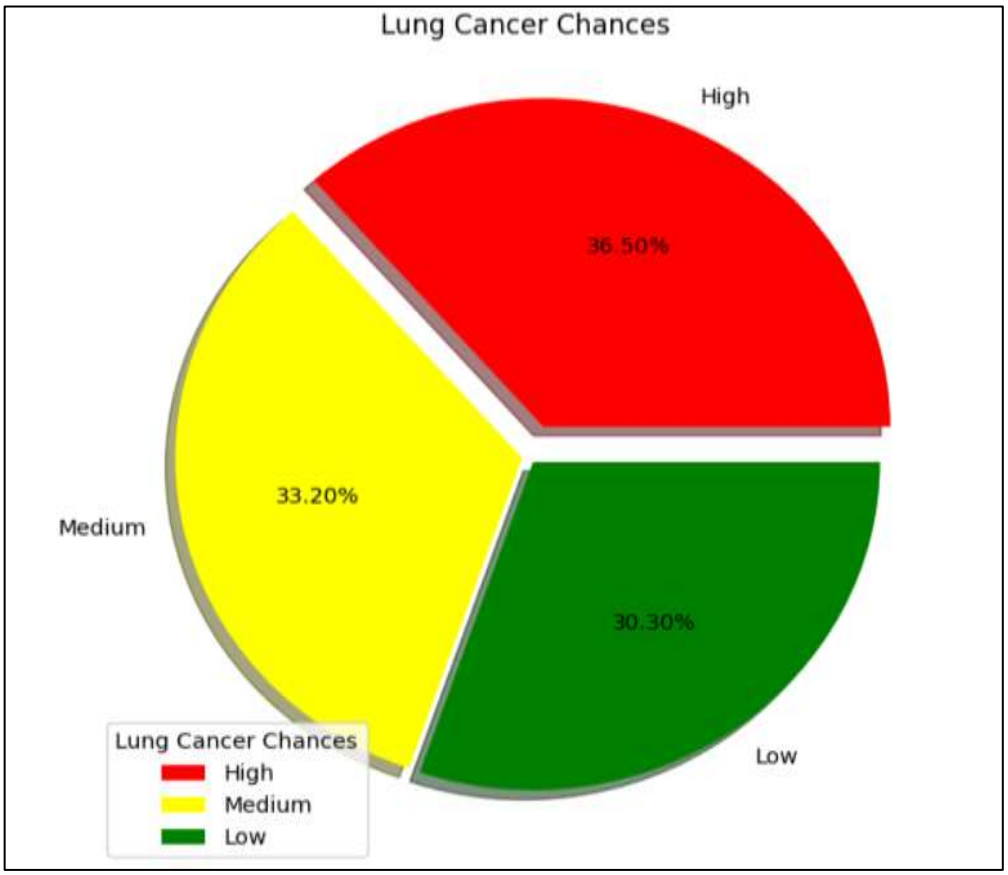


Рисунок 2.27 – Кількісний розподіл вибірки датасету у відсотках по параметру Level

Із результатів аналізу кількості значень параметру Level можна впевнено сказати що розподіл між унікальними значеннями ризику захворіти раком легень серед пацієнтів у даному датасеті є рівномірним, це демонструється мінімальною різницею у кількості між усіма унікальними параметрами ознаки, отже для перевірки результатів дослідження при побудові моделей машинного навчання можна використовувати метрики F1_Score та R2_Score. Але для задачі класифікації доцільніше все ж опиратись на метрику F1_Score, тому остаточна оцінка моделей буде залежати саме від цієї метрики.

Для розвідувального аналізу як приклад було використано ноутбук користувача «Subhajeet Das» [18].

2.3 Висновки

В цьому розділі було здійснено попередню обробку даних, а саме очищення зайвих ознак таких як Index та «Номер пацієнта» та заміна типу даних цільової ознаки «Рівень ризику» на числові, для того щоб їх можна було використовувати для прогнозування методами машинного навчання описаних у попередньому розділі. Також був проведений розвідувальний аналіз даних з використанням візуалізації для демонстрації розподілу даних за різними ознаками, який показав що більша частина пацієнтів відповідають середньому віку та мають певні проблеми різного характеру, які можуть впливати на ризики захворіти раком легень. Кількісний розподіл по цільовій ознаці «Рівень ризику» рівномірний.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Розроблення інформаційної технології

Для створення інформаційної технологія для передбачення ризиків захворіти на рак легень необхідно розробити комплекс методів машинного навчання, які будуть навчатися на попередньо оброблених даних, та тестуватись на цих даних. Для цієї задачі було вирішено обрати наступні методи: Random Forest, AdaBoost, Extra Trees, Decision Tree, XGBoost та Multi-Layer Perceptron.

Для налаштування даних моделей необхідно задати їм список параметрів, за якими вони будуть корегуватися. Для знаходження оптимальної комбінації параметрів було вирішено використовувати метод GridSearchCV, як ефективний та простий у виконанні. Якщо оцінка точності моделі не буде задовольняти вимоги поставленої задачі необхідно буде змінити список параметрів до моменту отримання хороших результатів.

Після отримання результатів необхідно порівняти моделі між собою та обрати найкращу з них для подальшого дослідження. Для найкращої моделі необхідно проаналізувати результати шляхом оцінки впливовості ознак на навчання моделі, тобто який рівень впливу кожної окремої наявної у наборі даних ознаки на рішення моделі під час вибору категорії класифікації для даних.

На рисунку 3.1 наведено блок-схему алгоритму роботи інформаційної технології передбачення ризиків захворіти на рак легень.

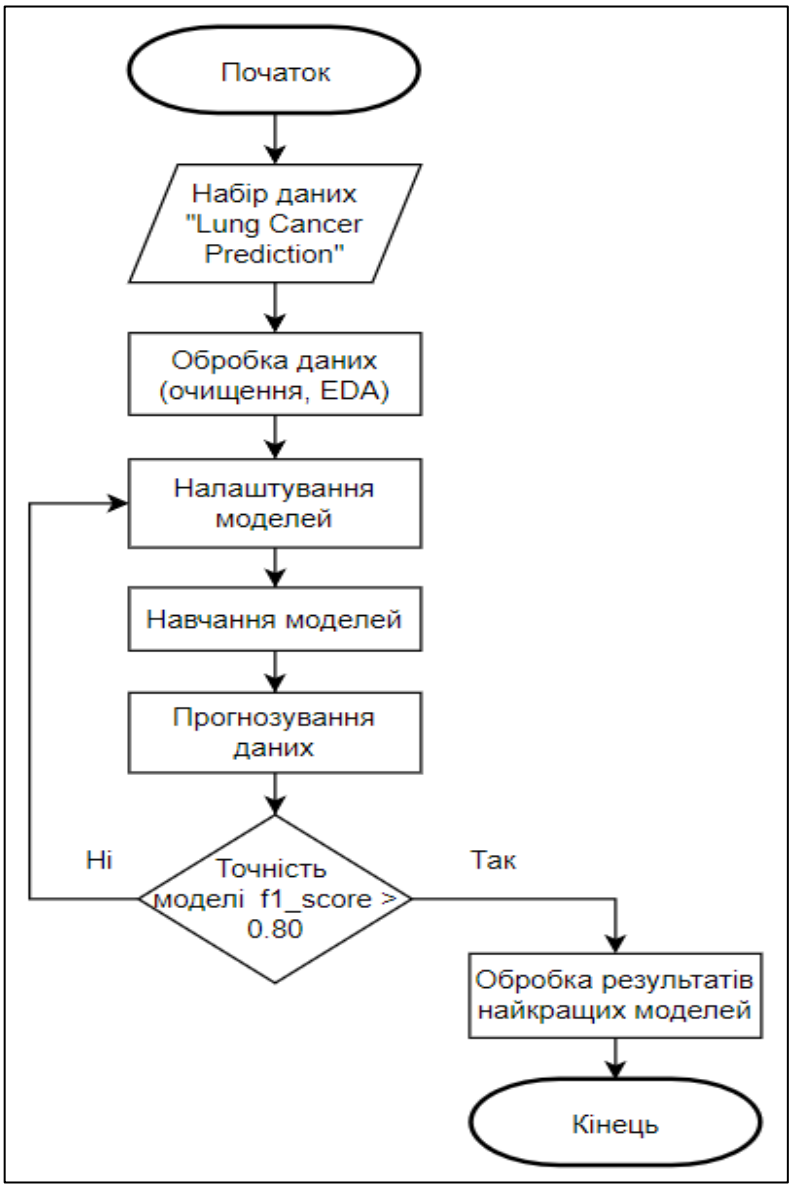


Рисунок 3.1 – Блок-схема алгоритму роботи інформаційної технології передбачення ризиків захворіти на рак легень

Дана блок-схема демонструє алгоритм роботи, який показує як працює інформаційна технологія. Вона демонструє процес оброблення даних, які потім передаються для навчання моделей із попереднім їх налаштуванням. За результатами навчання ми використовуємо моделі для прогнозування, під час якого порівнюємо їхню точність. Якщо точність моделі менша ніж 80%, ми відправляємо її на перенавчання до моменту поки не отримаємо задовільний результат. Потім визначаємо найкращу модель та обробляємо її результати шляхом дослідження впливовості ознак на дану модель.

3.2 Розбиття даних

При роботі із моделями як приклад використовувався ноутбук «Used Cars : Analysis and Prediction» користувача «Vitalii Mokin» [19].

Для того щоб розпочати роботу із моделюванням необхідно розбити наш набір даних на трейнові та тестові (валідаційні), для цього буде використовуватись функція train_test_split імпортована із бібліотеки Sklearn (рис. 3.2).

```
# Data splitting
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Рисунок 3.2 – Розбиття датасету на трейнові та валідаційні дані

На рисунку вище бачимо що границя розбиття задана у 20% параметром test_size=0,2 що означає що у відсотковому співвідношенні частка тренувальних даних буде дорівнювати 80%, оскільки наш датасет містить рівно 1000 значень то тренувальних даних у нас буде 800, а тестових 200 (рис. 3.3).

```
Train Shape

X train shape:  (800, 23)
Y train shape:  (800,)


Test Shape

X test shape:  (200, 23)
Y test shape:  (200,)
```

Рисунок 3.3 – Результат розбиття даних

Після цього необхідно вибрати метрику вимірювання точності прогнозування моделей. Різні метрики використовуються для різного роду задач, наприклад для задач регресії доцільніше використовувати метрику R2_Score, а для задач класифікації, доцільніше використовувати метрику F1_Score. Але

оскільки розподіл даних у нашому датасеті рівномірний ми будемо використовувати обидві метрики для порівняння результатів та записувати їх у окрему таблицю (рис. 3.4).

```
# Results of prediction
results = pd.DataFrame(columns = ['model', 'f1_train', 'f1_test', 'r2_train', 'r2_test'])
```

Рисунок 3.4 – Створення таблиці для запису результатів із метриками точності

Під час процесу моделювання та навчання моделей необхідно підібрати параметри, які найбільше підходять для наших моделей та дають максимальні результати точності, для цього ми будемо використовувати метод GridSearchCV, який буде підбирати зі списку представлених параметрів комбінацію найкращих з них шляхом перебору усіх можливих комбінацій, які будуть давати найвищу точність.

3.3 Результати застосування Random Forest Classifier

Отож спробуємо налаштувати модель Random Forest, спочатку визначим найкращі параметри (рис. 3.5).

```
param_RF = {
    'n_estimators': [25, 50, 75, 100],
    'max_depth': [2, 3],
    'min_samples_split': [2,3,4],
    'min_samples_leaf': [1, 2, 3],
    'max_features': ['sqrt'],
    'max_samples': [0.5, 0.6, 0.7]
}
```

```
%%time
model_tuning = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_RF, cv=
5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_rf = RandomForestClassifier(**best_params)
model_rf.fit(x_train, y_train)
```

```
Best Parameters: {'max_depth': 3, 'max_features': 'sqrt', 'max_samples': 0.6, 'min_sam
ples_leaf': 3, 'min_samples_split': 3, 'n_estimators': 100}
CPU times: user 2min 51s, sys: 1.04 s, total: 2min 52s
```

Рисунок 3.5 – Налаштування параметрів моделі Random Forest

Наступним кроком буде процес навчання моделі на найкращих параметрах та виведення й збереження результатів цього тренування (рис. 3.6).

```
# Train
train_predictions = model_rf.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_rf.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[0,'model'] = 'RandomForest Classifier'
results.loc[0,'f1_train'] = f1_train
results.loc[0,'f1_test'] = f1_test
results.loc[0,'r2_train'] = r2_train
results.loc[0,'r2_test'] = r2_test
results.loc[0,'short'] = 'RF'
```

Рисунок 3.6 – Навчання моделі Random Forest

Результати навчання виводимо для тренувальних та тестових даних для порівняння точності (рис. 3.7).

```
Train Data Metrics:
Precision :  0.9675
Recall :  0.9675
Accuracy :  0.9675
F1 Score :  0.9675
R2 Score :  0.9055657203564894


Test Data Metrics:
Precision :  0.98
Recall :  0.98
Accuracy :  0.98
F1 Score :  0.98
R2 Score :  0.925012185519853
```

Рисунок 3.7 – Результати навчання моделі Random Forest

Як бачимо результати за оцінками майже ідеальні, це означає що модель навчилась дуже добре, тож виведемо матрицю плутанини для тестових даних (рис. 3.8).

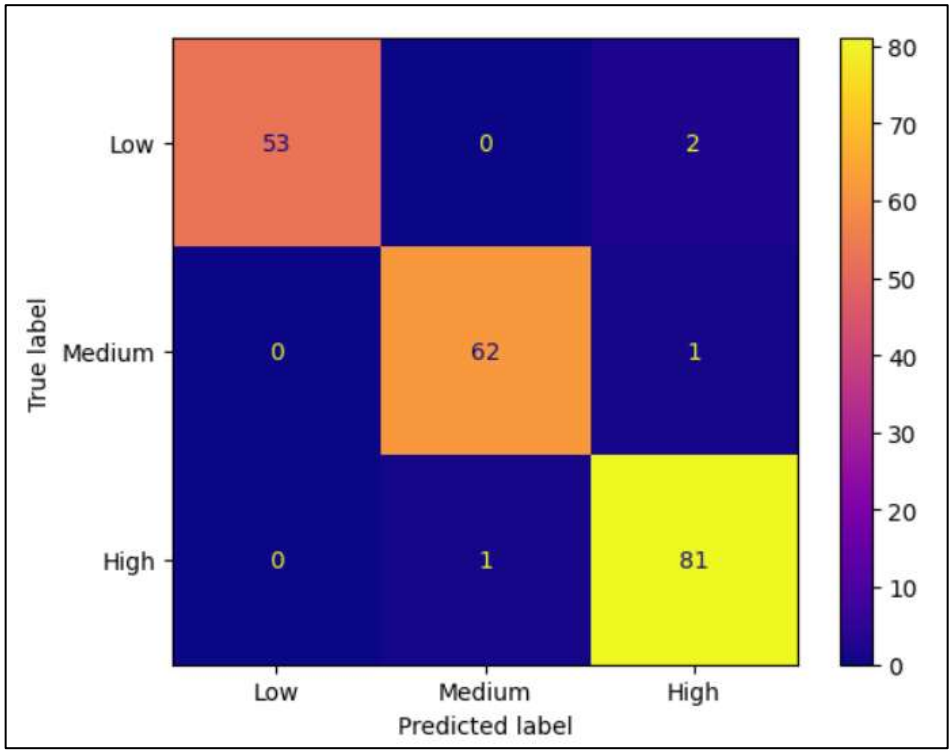


Рисунок 3.8 – Матриця плутанини моделі Random Forest

3.4 Результати застосування AdaBoost Classifier

Далі будемо налаштовувати модель Ada Boost, алгоритм налаштування такий самий як і для попередньої моделі, спочатку визначаємо найкращі параметри, потім навчаємо та виводимо результати (рис. 3.9).

```
param_ADA = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1],
    'algorithm': ['SAMME.R'],
}
```

```
%%time
model_tuning = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_ADA, cv=5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_ada = AdaBoostClassifier(**best_params)
model_ada.fit(x_train, y_train)
```

```
Best Parameters: {'algorithm': 'SAMME.R', 'learning_rate': 0.01, 'n_estimators': 50}
CPU times: user 12.8 s, sys: 75.1 ms, total: 12.9 s
Wall time: 12.9 s
```

Рисунок 3.9 – Налаштування параметрів моделі Ada Boost

Виконуємо навчання моделі із заданими найкращими параметрами, які були тюнінговані за допомогою GridSearchCV (рис. 3.10).

```
# Train
train_predictions = model_ada.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_ada.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[1,'model'] = 'ADABoost Classifier'
results.loc[1,'f1_train'] = f1_train
results.loc[1,'f1_test'] = f1_test
results.loc[1,'r2_train'] = r2_train
results.loc[1,'r2_test'] = r2_test
results.loc[1,'short'] = 'ADA'
```

Рисунок 3.10 – Навчання моделі Ada Boost

Виводимо результати точності для порівняння між тренувальними та тестовими даними (рис. 3.11).

```
Train Data Metrics:
Precision :  0.90125
Recall :  0.90125
Accuracy :  0.90125
F1 Score :  0.90125
R2 Score :  0.8507938381632533


Test Data Metrics:
Precision :  0.895
Recall :  0.895
Accuracy :  0.895
F1 Score :  0.895
R2 Score :  0.8425255895916913
```

Рисунок 3.11 – Результати навчання моделі Ada Boost

Як бачимо із результатів оцінки навчання моделі, вона навчилась добре, можливо для покращення оцінки необхідно буде змінити деякі параметри та можливо збільшити їх кількість.

Подивимось що ж покаже нам матриця плутанини для тестових даних (рис. 3.12).

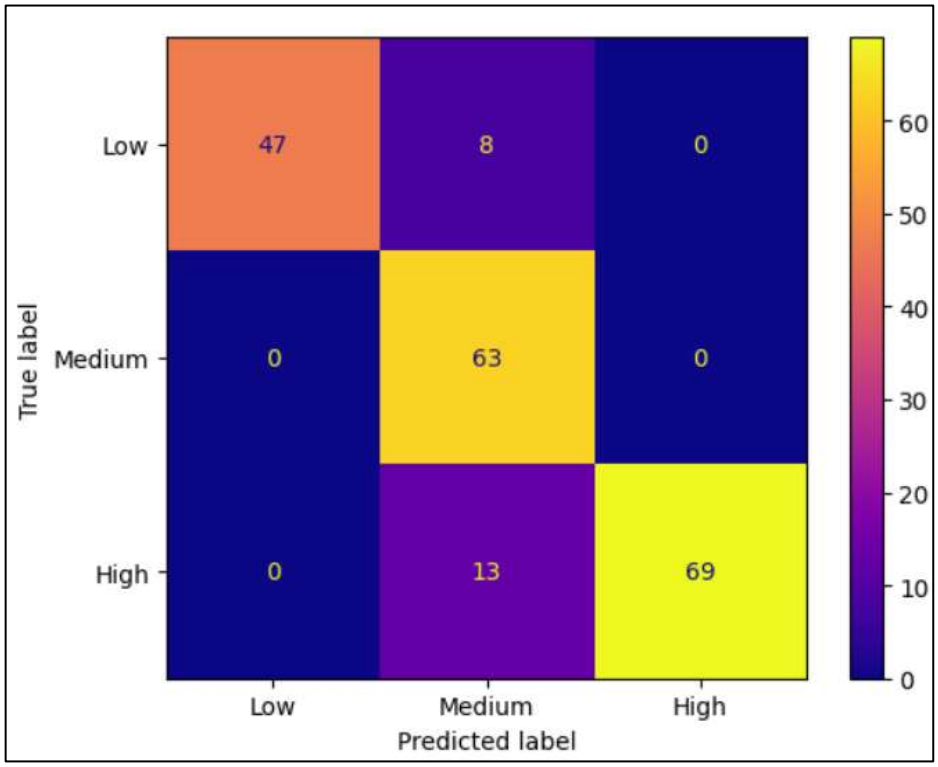


Рисунок 3.12 – Матриця плутанини моделі Ada Boost

Як бачимо із результатів матриці плутанини модель навчилась непогано, вона має невелику похибку для високий рівня ризику та для низького рівня ризику.

3.5 Результати застосування Extra Trees Classifier

Наступним кроком задаємо параметри для моделі Extra Trees (рис.3.13).

```
params_etc ={'n_estimators':[100, 200, 500],
             'max_depth': [3, 4, 5],
             'min_samples_split':  [2, 3],
             'min_samples_leaf':  [1, 2],
             'max_features': ['sqrt'],
             'ccp_alpha': [0.1]}
```

+ Code + Markdown

```
%%time
model_tuning = GridSearchCV(estimator=ExtraTreesClassifier(), param_grid=params_etc, cv=5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_etc = ExtraTreesClassifier(**best_params)
model_etc.fit(x_train, y_train)
```

```
Best Parameters: {'ccp_alpha': 0.1, 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split':
2, 'n_estimators': 200}
CPU times: user 1min 16s, sys: 490 ms, total: 1min 17s
Wall time: 1min 17s
```

Рисунок 3.13 – Налаштування параметрів моделі Extra Trees

На рисунку 3.13 зображено налаштування параметрів для моделі Extra Trees. Наступним кроком буде навчання моделі з параметрами які вибрала програма, як найкращі (рис. 3.14) та виведення результатів навчання (рис. 3.15).

```
# Train
train_predictions = model_etc.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_etc.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[2,'model'] = 'ExtraTrees Classifier'
results.loc[2,'f1_train'] = f1_train
results.loc[2,'f1_test'] = f1_test
results.loc[2,'r2_train'] = r2_train
results.loc[2,'r2_test'] = r2_test
results.loc[2,'short'] = 'ExT'
```

Рисунок 3.14 – Навчання моделі Extra Trees

```
Train Data Metrics:
Precision :  0.8875
Recall :  0.8875
Accuracy :  0.8875
F1 Score :  0.8875
R2 Score :  0.7846898424127958


Test Data Metrics:
Precision :  0.895
Recall :  0.895
Accuracy :  0.895
F1 Score :  0.895
R2 Score :  0.7975329009036032
```

Рисунок 3.15 – Результати навчання моделі Extra Trees

Результати навчання показують досить хороший результат, можливо більш детальне налаштування параметрів покращить модель, щоб вона демонструвала кращий результат. Особливо гарний результат демонструє модель за метрикою F1_Score, а це гарний показник. Виведемо матрицю плутанини (рис. 3.16).

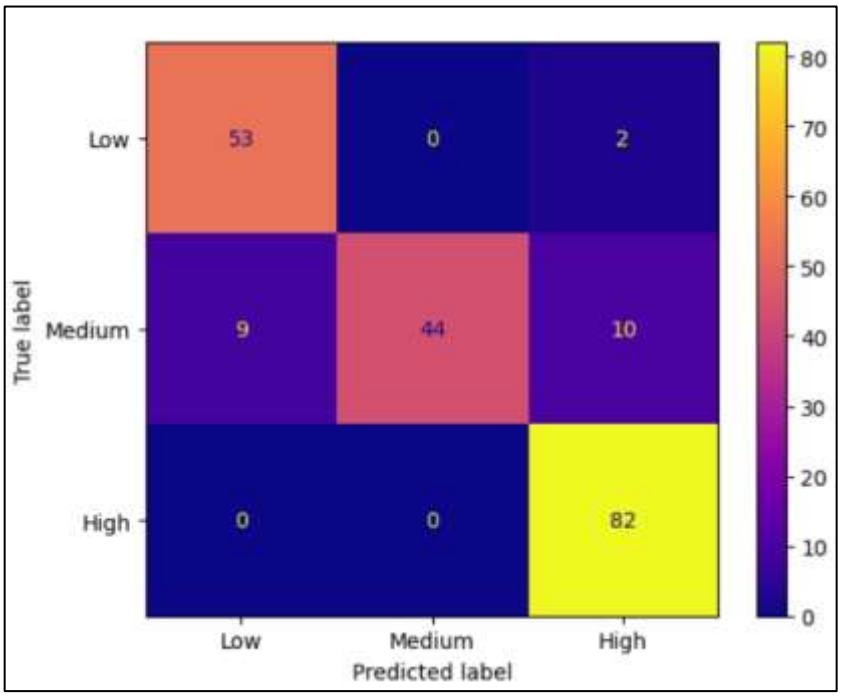
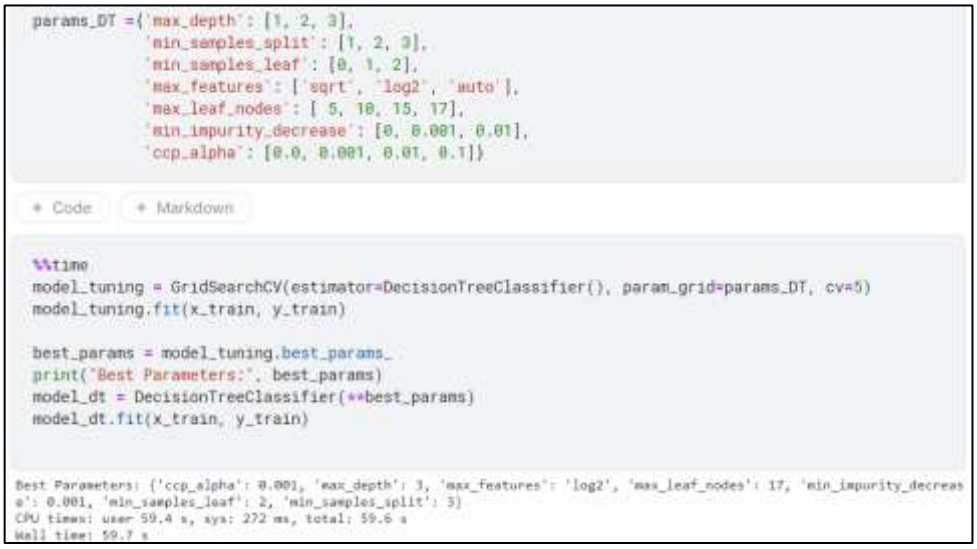


Рисунок 3.16 – Матриця плутанини моделі Extra Trees

Матриця плутанини на рисунку 3.16 демонструє хороші результати для параметру низького та високого ризиків, але для середнього рівня ризику є частка похибок.

3.6 Результати застосування Decision Tree Classifier

Далі проведемо налаштування моделі Decision Tree (рис. 3.17).



```
params_DT ={'max_depth': [1, 2, 3],
            'min_samples_split': [1, 2, 3],
            'min_samples_leaf': [0, 1, 2],
            'max_features': ['sqrt', 'log2', 'auto'],
            'max_leaf_nodes': [ 5, 10, 15, 17],
            'min_impurity_decrease': [0, 0.001, 0.01],
            'ccp_alpha': [0.0, 0.001, 0.01, 0.1]}
```

+ Code + Markdown

```
%%time
model_tuning = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=params_DT, cv=5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_dt = DecisionTreeClassifier(**best_params)
model_dt.fit(x_train, y_train)
```

```
Best Parameters: {'ccp_alpha': 0.001, 'max_depth': 3, 'max_features': 'log2', 'max_leaf_nodes': 17, 'min_impurity_decrease': 0.001, 'min_samples_leaf': 2, 'min_samples_split': 3}
CPU times: user 59.4 s, sys: 272 ms, total: 59.6 s
Wall time: 59.7 s
```

Рисунок 3.17 – Налаштування параметрів моделі Decision Tree

На рисунку 3.17 зображено вибір найкращих параметрів для моделі Decision Tree, за допомогою цих параметрів будемо навчати модель (рис. 3.18).

```
# Train
train_predictions = model_dt.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_dt.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[3,'model'] = 'DecisionTree Classifier'
results.loc[3,'f1_train'] = f1_train
results.loc[3,'f1_test'] = f1_test
results.loc[3,'r2_train'] = r2_train
results.loc[3,'r2_test'] = r2_test
results.loc[3,'short'] = 'DT'
```

Рисунок 3.18 – Навчання моделі Decision Tree

Результати навчання за оцінками метрик приведено на рисунку 3.19. Вони показують що модель має хороші показники за оцінкою метрик, що означає що модель навчилась добре.

```
Train Data Metrics:
Precision :  0.9625
Recall :  0.9625
Accuracy :  0.9625
F1 Score :  0.9625000000000001
R2 Score :  0.8980109779850085


Test Data Metrics:
Precision :  0.95
Recall :  0.95
Accuracy :  0.95
F1 Score :  0.9500000000000001
R2 Score :  0.8800194968317648
```

Рисунок 3.19 – Результати навчання моделі Decision Tree

Також виведемо результати за допомогою матриці плутанини, зображеної на рисунку 3.20.

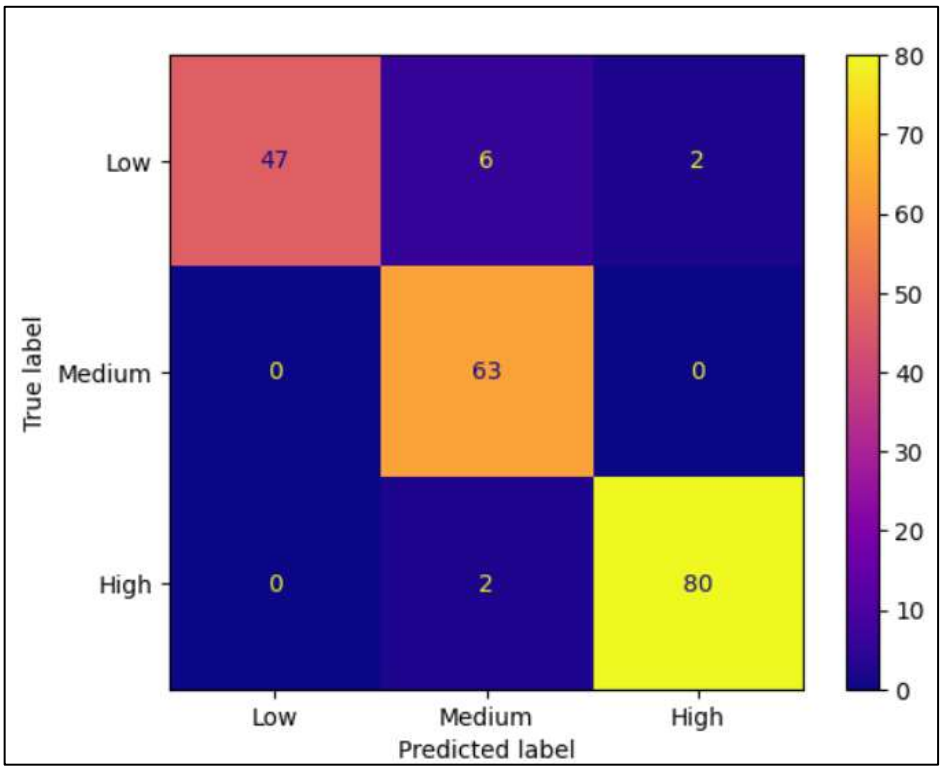


Рисунок 3.20 – Матриця плутанини моделі Decision Tree

Модель Decision Tree має широкі можливості для візуалізації результатів, скористаємось даними інструментами для побудови дерева рішень (рис. 3.21).

Подібні інструменти широко використовуються для гарного відображення результатів даної моделі. Що не менш важливо вони є легкими у розумінні. Також за допомогою візуалізації можна детально роздивитись кожний вузол дерева та його листя, що дасть змогу прослідкувати та детально розібрати кожен окремий шлях від початку дерева, тобто його коріння, до його результатів, тобто листя.

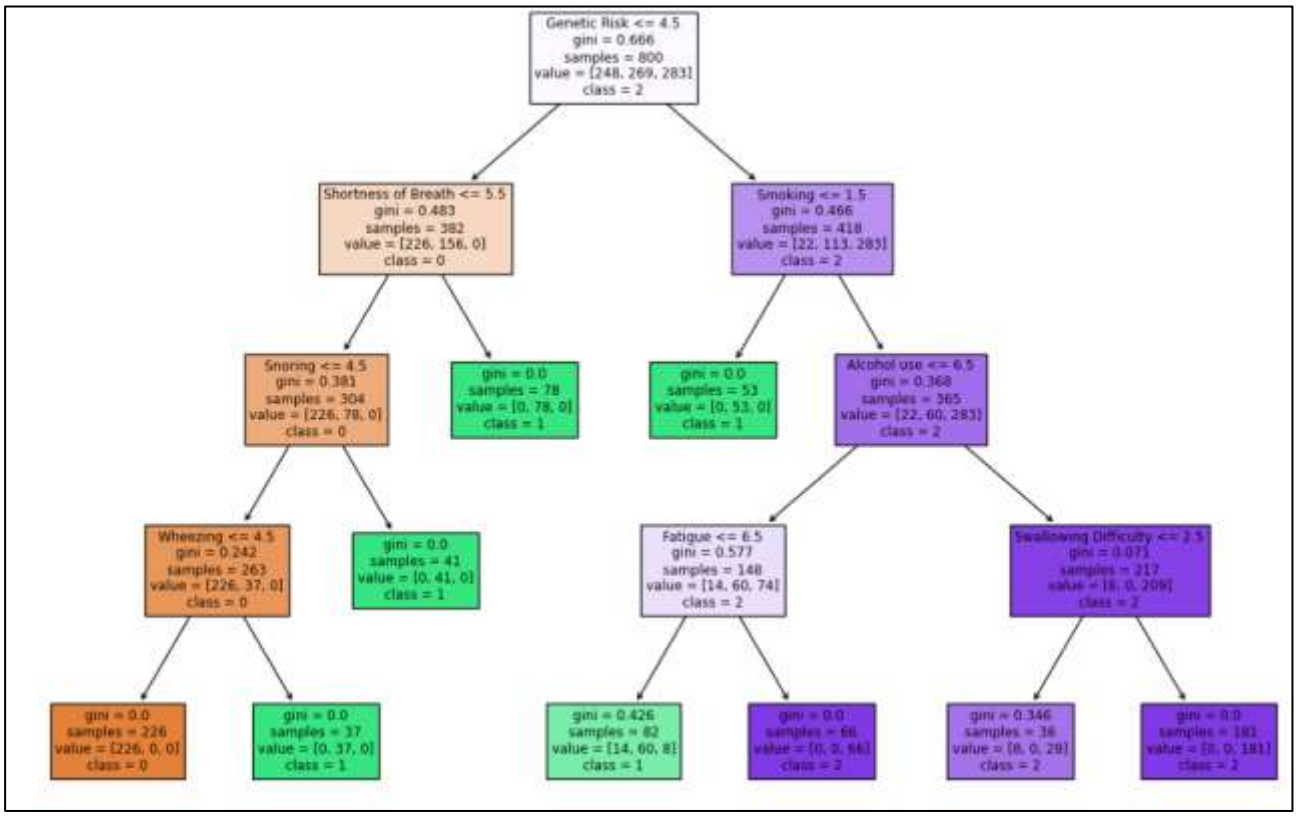


Рисунок 3.21 – Побудоване дерево рішень для моделі Decision Tree

Передчасно завантаживши, імпортуємо бібліотеку dtreeviz, яка надає ширші можливості використання графіків для відображення вузлів та листя на дереві рішень. На рисунку 3.22 задано параметри для побудови графіка за результатами навчання моделі та збереження результатів (рис. 3.23).

```
import dtreeviz

viz_model = dtreeviz.model(model_dt,
                           X_train=x_train, y_train=y_train,
                           feature_names=feature_names,
                           target_name='Lung Cancer',
                           class_names=['Low', 'Medium', 'High'])

v = viz_model.view()     # render as SVG into internal object
v.save("Lung Cancer.svg")  # save as svg
```

Рисунок 3.22 – Параметри візуалізації дерева рішень для Decision Tree за допомогою бібліотеки dtreeviz

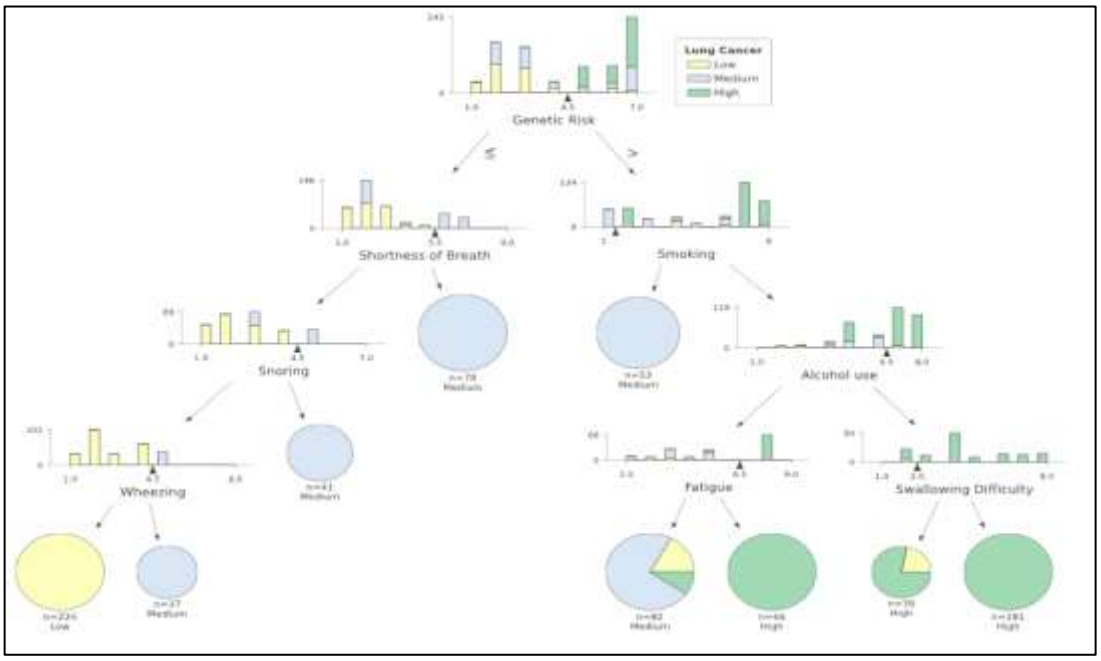


Рисунок 3.23 – Візуалізація дерева рішень для Decision Tree за допомогою бібліотеки dtreeviz

На рисунку 3.23 зображено використання інструментів, які додає бібліотека dtreeviz та за їх допомогою побудовано дерево рішень з використанням діаграм для детальної візуалізації вузлів та листя дерева.

3.7 Результати застосування XGBoost Classifier

Далі на рисунку 3.24 зображено налаштування параметрів моделі XGBoost.

```
params_XGB ={'n_estimators': [100, 200, 300],
             'num_leaves': [2, 5, 7, 10],
             'max_depth': [2, 3, 5],
             'subsample': [0.01],
             'learning_rate': [0.01],
             'objective': ['multi:softmax'],
             'num_class': [3]}
```

+ Code + Markdown

```
%%time
model_tuning = GridSearchCV(estimator=XGBClassifier(), param_grid=params_XGB, cv=5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_xgb = XGBClassifier(**best_params)
model_xgb.fit(x_train, y_train)
```

```
Best Parameters: {'learning_rate': 0.01, 'max_depth': 2, 'n_estimators': 300, 'num_class': 3, 'num_leaves': 2
e': 'multi:softmax', 'subsample': 0.01}
CPU times: user 1min 51s, sys: 2.29 s, total: 1min 53s
Wall time: 30.9 s
```

Рисунок 3.24 – Налаштування параметрів моделі XGBoost

Рисунок 3.24 демонструє налаштування параметрів XGBoost моделі, які тюнінгуються за допомогою методу GridSearchCV та використовуються для навчання моделі (рис. 3.25).

```
# Train
train_predictions = model_xgb.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_xgb.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[4,'model'] = 'XGB Classifier'
results.loc[4,'f1_train'] = f1_train
results.loc[4,'f1_test'] = f1_test
results.loc[4,'r2_train'] = r2_train
results.loc[4,'r2_test'] = r2_test
results.loc[4,'short'] = 'XGB'
```

Рисунок 3.25 – Навчання моделі XGBoost

Результати навчання та матриця плутанини моделі продемонстровані на рисунку 3.26 й 3.27 відповідно.

```
Train Data Metrics:
Precision :  0.89
Recall :  0.89
Accuracy :  0.89
F1 Score :  0.89
R2 Score :  0.7884672135985362


Test Data Metrics:
Precision :  0.935
Recall :  0.935
Accuracy :  0.935
F1 Score :  0.935
R2 Score :  0.8575231524877207
```

Рисунок 3.26 – Результат навчання моделі XGBoost

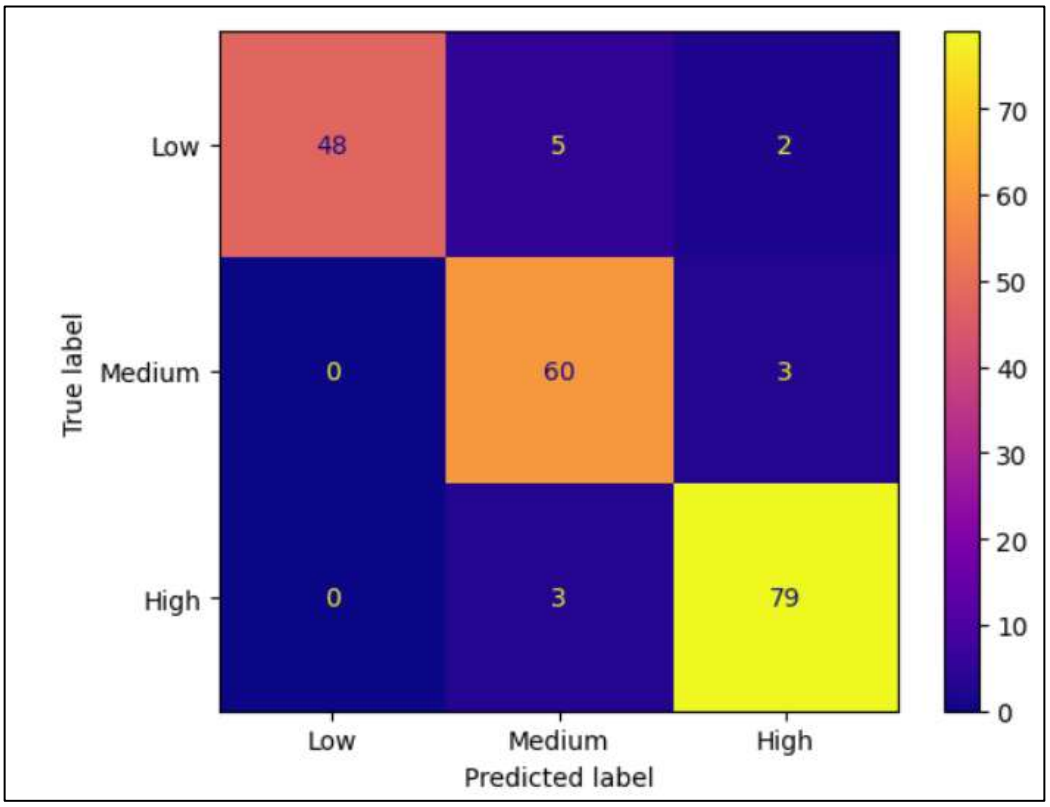


Рисунок 3.27 – Матриця плутанини моделі XGBoost

Результати навчання за метриками показують дуже хороші результати. Подібні оцінки метрики дуже важко отримати при звичайних умовах, але при правильному налаштуванні параметрів та хорошому наборі даних виходить продемонстрований вище результат, який і відображає матриця плутанини моделі XGBoost.

3.8 Результати застосування MLP Classifier

Ще одним кроком буде навчання моделі Multi-Layer Perceptron (MLP). Для цього вибираємо список параметрів та визначаємо найкращі комбінації (рис. 3.28).

```
params_mlp ={'hidden_layer_sizes': [50, 75, 100],
             'random_state': [2],
             'alpha': [0.001],
             'activation': ['relu', 'tanh'],
             'learning_rate_init': [0.001],
             'max_iter': [100, 200, 300],
             'solver': ['adam'],
             'early_stopping': [True],
             'validation_fraction': [0.1],

            }
```

```
%%time
model_tuning = GridSearchCV(estimator=MLPClassifier(), param_grid=params_mlp, cv=5)
model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_mlp = MLPClassifier(**best_params)
model_mlp.fit(x_train, y_train)
```

```
Best Parameters: {'activation': 'tanh', 'alpha': 0.001, 'early_stopping': True, 'hidden_layer_sizes': 100, 'learning_rate
_init': 0.001, 'max_iter': 100, 'random_state': 2, 'solver': 'adam', 'validation_fraction': 0.1}
CPU times: user 59.1 s, sys: 40.8 s, total: 1min 39s
Wall time: 29.7 s
```

Рисунок 3.28 – Налаштування параметрів моделі MLP

Далі на рисунку 3.29 зображено навчання моделі із заданими попередньо найкращими параметрами, які були обрані шляхом комбінування усіх можливих варіантів із запропонованих користувачем.

```
# Train
train_predictions = model_dt.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_dt.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[5,'model'] = 'MLP Classifier'
results.loc[5,'f1_train'] = f1_train
results.loc[5,'f1_test'] = f1_test
results.loc[5,'r2_train'] = r2_train
results.loc[5,'r2_test'] = r2_test
results.loc[5,'short'] = 'MLP'
```

Рисунок 3.29 – Навчання моделі MLP

На рисунку 3.30 нижче зображено результати які надала модель після навчання, оцінки її точності за різними метриками.

```
Train Data Metrics:
Precision :  0.9125
Recall :  0.9125
Accuracy :  0.9125
F1 Score :  0.9125
R2 Score :  0.8621259517204746


Test Data Metrics:
Precision :  0.94
Recall :  0.94
Accuracy :  0.94
F1 Score :  0.94
R2 Score :  0.9100146226238236
```

Рисунок 3.30 – Результат навчання моделі MLP

Бачимо що результати які надала модель можна назвати хорошими за обома метриками F1_Score та R2_Score, що означає що модель навчилась непогано.

Виведемо матрицю плутанини для відображення результатів (рис. 3.31).

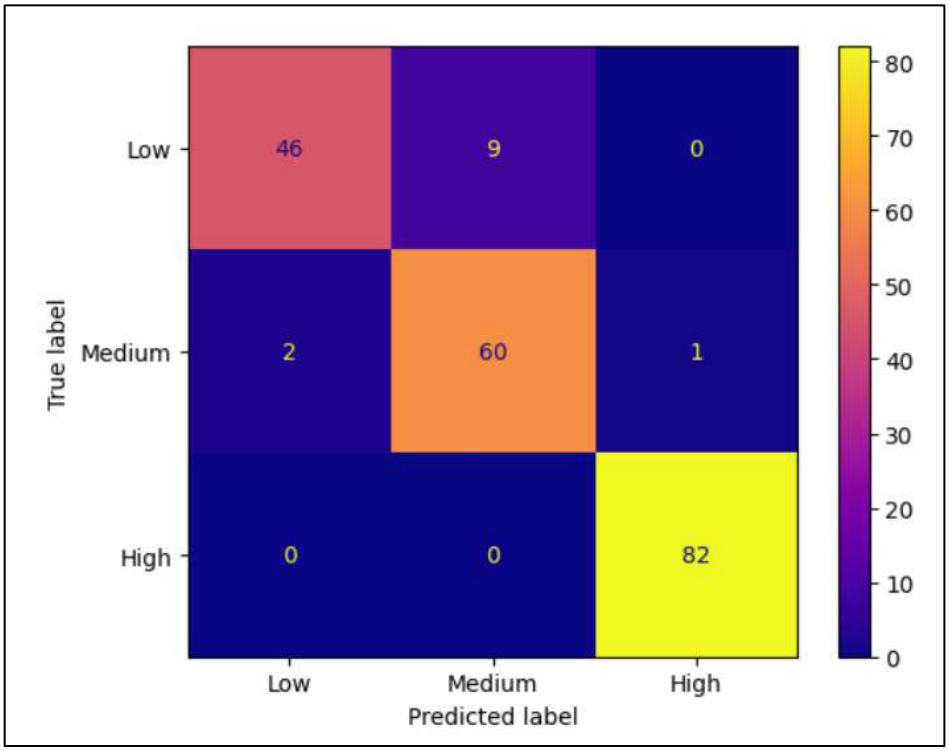


Рисунок 3.31 – Матриця плутанини моделі MLP

Результати навчання моделі MLP показують непогані результати, але з певною похибкою у визначенні класифікації для низького рівня, що візуально ми бачимо на матриці плутанини.

3.9 Дослідження впливу ознак на результати найкращих моделей

Отже ми отримали результати навчання моделей за оцінками метрики F1_Score та R2_Score для тестових та тренувальних даних по усіх моделях (рис. 3.32-3.33).

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	0.9675	0.98	0.905566	0.925012	RF
1	ADABoost Classifier	0.90125	0.895	0.850794	0.842526	ADA
2	ExtraTrees Classifier	0.8875	0.895	0.78469	0.797533	ExT
3	DecisionTree Classifier	0.9625	0.95	0.898011	0.880019	DT
4	XGB Classifier	0.89	0.935	0.788467	0.857523	XGB
5	MLP Classifier	0.9125	0.94	0.862126	0.910015	MLP

Рисунок 3.32 – Результати усіх моделей

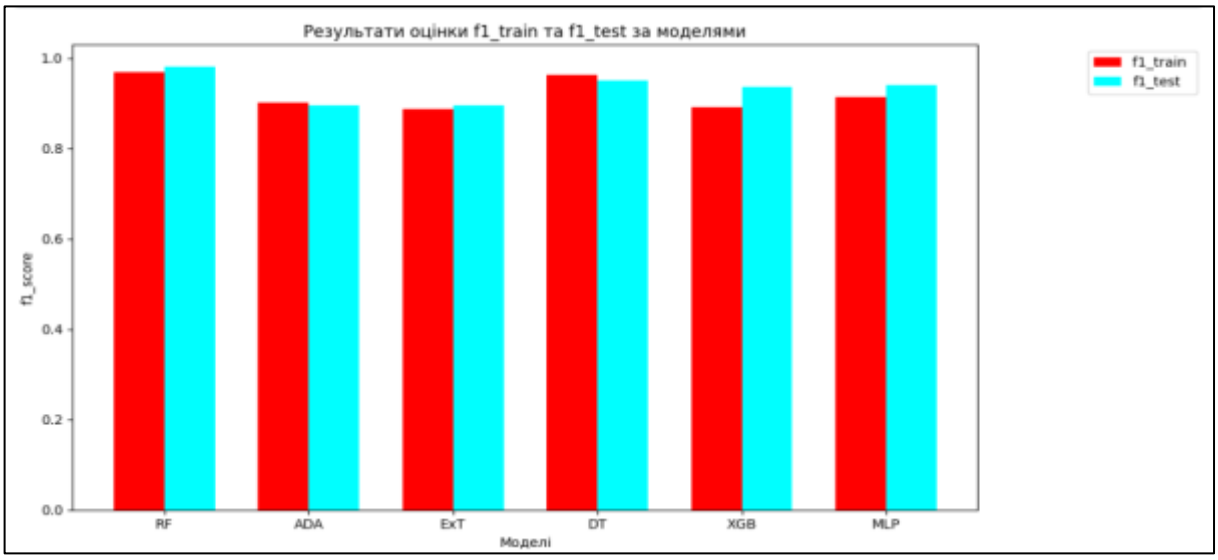


Рисунок 3.33 – Гістограма результатів навчання моделей за метрикою F1_Score для тренувальних та тестових даних

Як бачимо із результатів візуалізації на рисунку 3.33, найкращими моделями являються Random Forest (RF) та Decision Tree (DT) із оцінкою метрики F1_Score для тестових даних у 0.98 та 0.95 відповідно. Попри це інші моделі також мають хороші результати вище 80% точності. Що не менш важливо гістограма демонструє мінімальні відхилення за метрикою для тестових та тренувальних даних по кожній моделі, це свідчить про те що усі моделі мають хорошу узгодженість із навчальними даними.

Далі скористаємось атрибутом feature_importances, який надає змогу отримати оцінку важливості ознак, тобто таких які були найбільш важливими під час прийняття рішень моделлю (рис. 3.34-3.35).

	feature	score_model_rf
13	Coughing of Blood	0.167293
11	Passive Smoker	0.120029
9	Obesity	0.113776
12	Chest Pain	0.075676
8	Balanced Diet	0.063681
10	Smoking	0.055698
17	Wheezing	0.051315
2	Air Pollution	0.047879
14	Fatigue	0.039554
6	Genetic Risk	0.036527
16	Shortness of Breath	0.035017
5	OccuPational Hazards	0.031461
3	Alcohol use	0.030822
4	Dust Allergy	0.030180
19	Clubbing of Finger Nails	0.023836
20	Frequent Cold	0.023188
18	Swallowing Difficulty	0.021595
15	Weight Loss	0.013723
22	Snoring	0.008110
21	Dry Cough	0.006579
7	chronic Lung Disease	0.003073

Рисунок 3.34 – Список важливості ознак за оцінкою feature_importances для моделі Random Forest

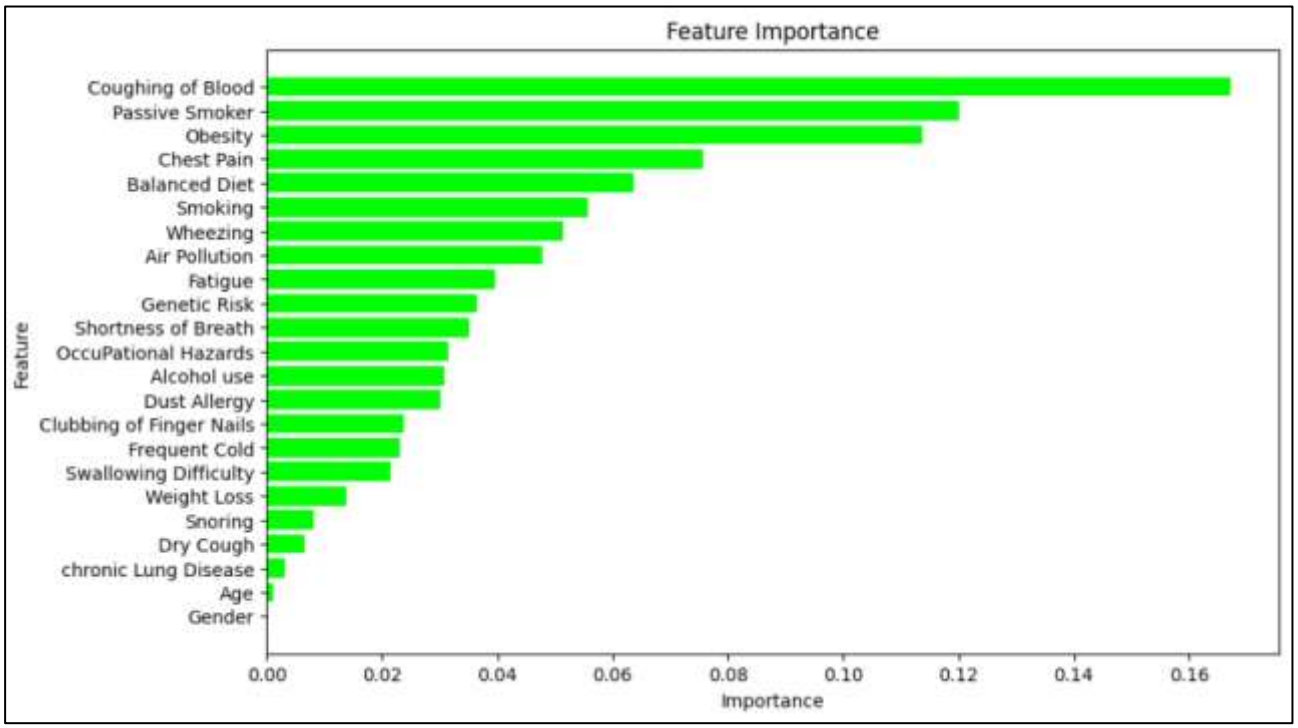


Рисунок 3.35 – Візуалізація важливості ознак за оцінкою feature_importances для моделі Random Forest

Як бачимо з рисунку 3.35, найбільш впливовою ознакою за оцінкою feature_importances є «Кашель із кров'ю» після нього «Пасивне куріння» та «Ожиріння». Найменшого впливу на рішення моделі надають параметри «Віку» та параметри «Гендерної ознаки».

Далі скористаємось бібліотекою SHAP. Бібліотека SHAP (SHapley Additive exPlanations) - це потужний інструмент для пояснення прогностичних моделей штучного інтелекту, зокрема моделей машинного навчання. SHAP використовує концепцію значень Шеплі (Shapley values) з теорії ігор для розкриття внеску кожного ознаки у прогнозоване значення моделі. Це дозволяє розуміти, як саме кожна ознака впливає на прогноз та які взаємодії між ознаками мають місце (рис. 3.36).

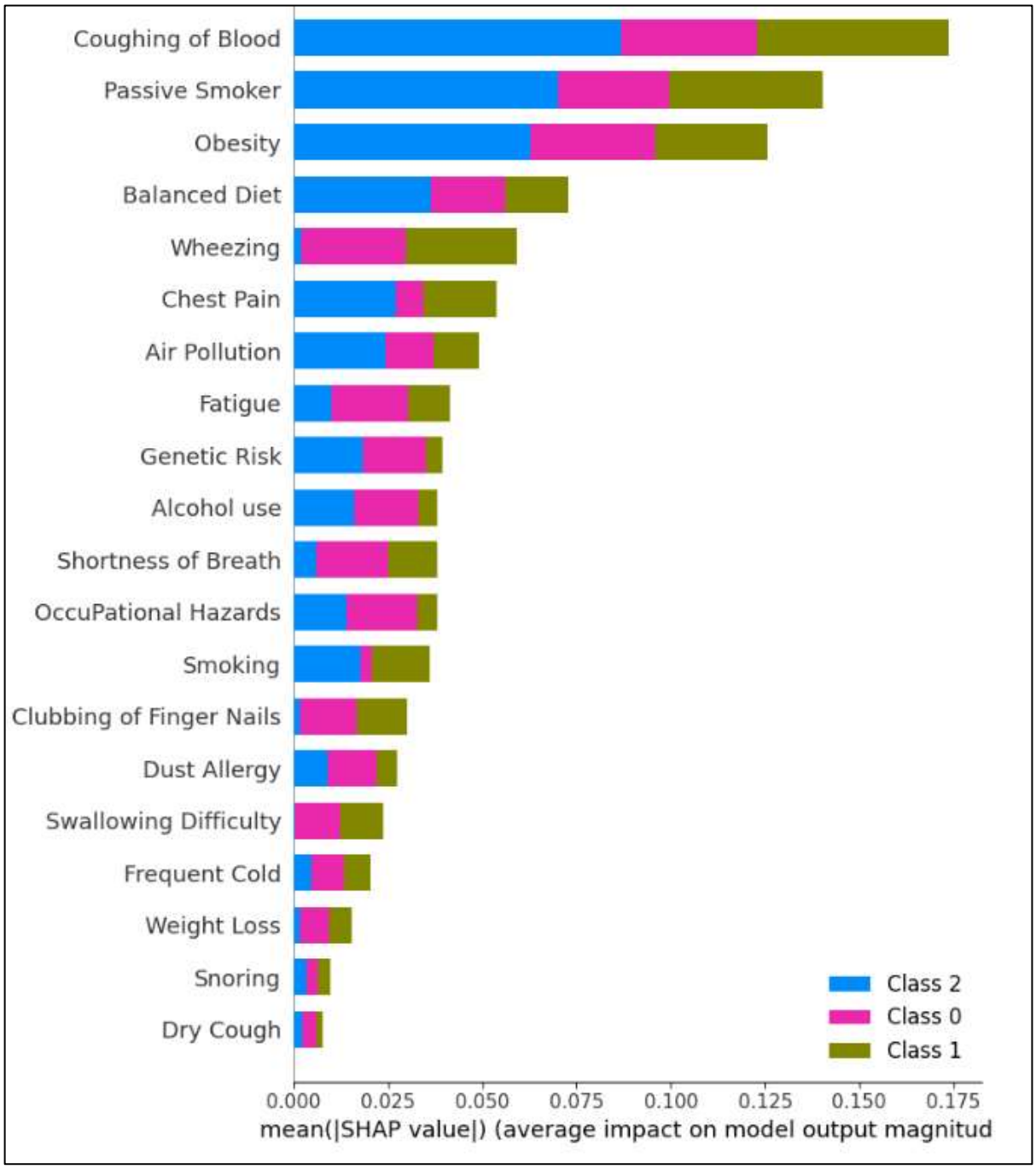


Рисунок 3.36 – Візуалізація важливості ознак за оцінкою яку надає бібліотека SHAP для моделі Random Forest

На рисунку 3.36 бачимо оцінки впливу ознак на кожну окрему категорію класифікації, яку надає бібліотека SHAP. Нагадаємо що категорії «0», «1» та «2» відповідають рівням ризику захворіти на рак легень, а саме «Low», «Medium», «High».

Порівнюючи результати оцінок за feature_importances та за SHAP бачимо, що їхні результати збігаються для перших трьох ознак, а саме «Кашляння

кров’ю», «Пасивне куріння» та «Ожиріння», що дає підстави вважати що ці три фактори дійсно найбільше впливають на рішення моделі при класифікації.

Далі використавши атрибут feature_importances для моделі Decision Tree бачимо наступний результат (рис. 3.37).

	feature	score_model_dt
6	Genetic Risk	0.315914
16	Shortness of Breath	0.141415
17	Wheezing	0.131062
10	Smoking	0.124292
22	Snoring	0.107967
14	Fatigue	0.103928
3	Alcohol use	0.069309
18	Swallowing Difficulty	0.006112
13	Coughing of Blood	0.000000
21	Dry Cough	0.000000
20	Frequent Cold	0.000000
19	Clubbing of Finger Nails	0.000000
15	Weight Loss	0.000000
0	Age	0.000000
12	Chest Pain	0.000000
1	Gender	0.000000
9	Obesity	0.000000
8	Balanced Diet	0.000000
7	chronic Lung Disease	0.000000

Рисунок 3.37 – Список важливості ознак за оцінкою feature_importances для моделі Decision Tree

Далі візуалізуємо отримані результати на (рис. 3.38).

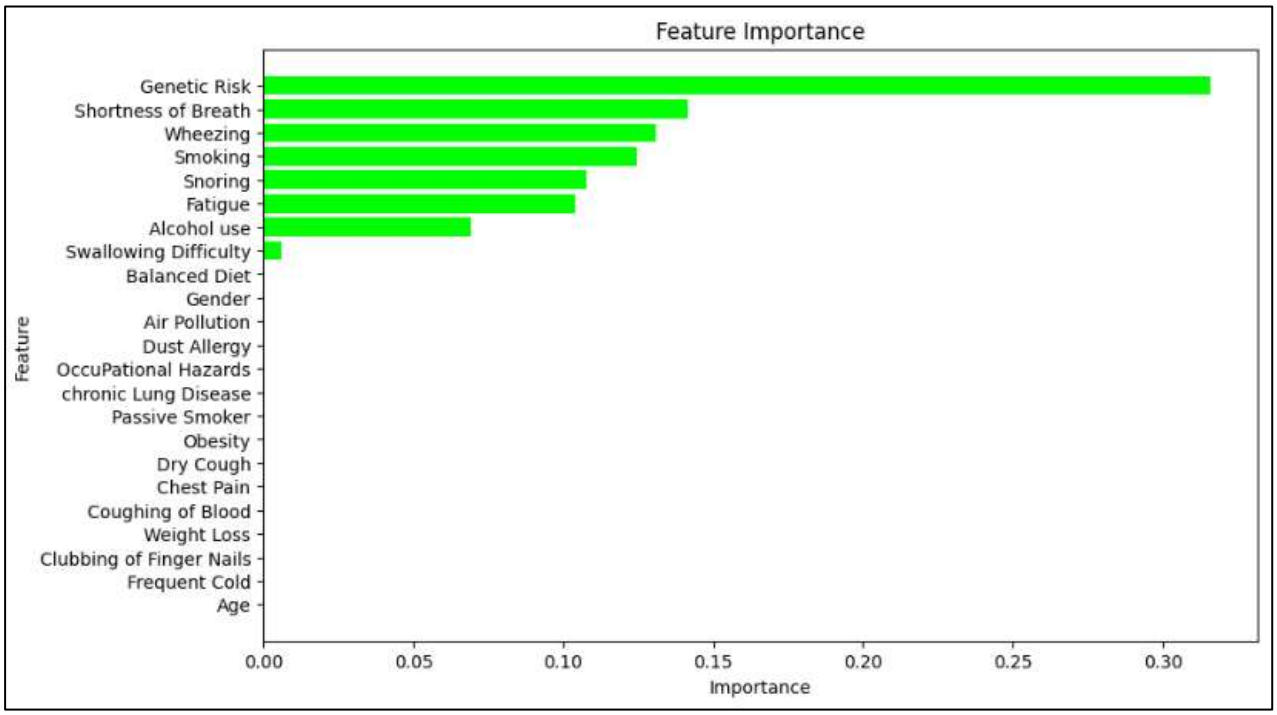


Рисунок 3.38 – Візуалізація важливості ознак за оцінкою feature_importances для моделі Decision Tree

Як можемо бачити із результатів візуалізації, кількість ознак які впливають на рішення моделі Decision Tree набагато менша аніж у Random Forest. Це пов’язано із обмеженнями для максимальної глибини дерева «max_depth» для моделі Decision Tree, оскільки кількість даних замала таке обмеження необхідне для уникнення перенавчання моделі. Глибина дерев напряму впливає на кількість ознак, які модель може використати при навчанні.

Тим не менш найбільш впливовою ознакою виявилася «Генетичні ризики», дана ознака суттєво випереджає інші, які схожі за впливовістю між собою, окрім «Рівня труднощів ковтання». Ця ознака найменше із усіх обраних впливає на рішення моделі при класифікації.

Візуалізуємо важливості ознак за оцінкою бібліотеки SHAP (рис. 3.39).

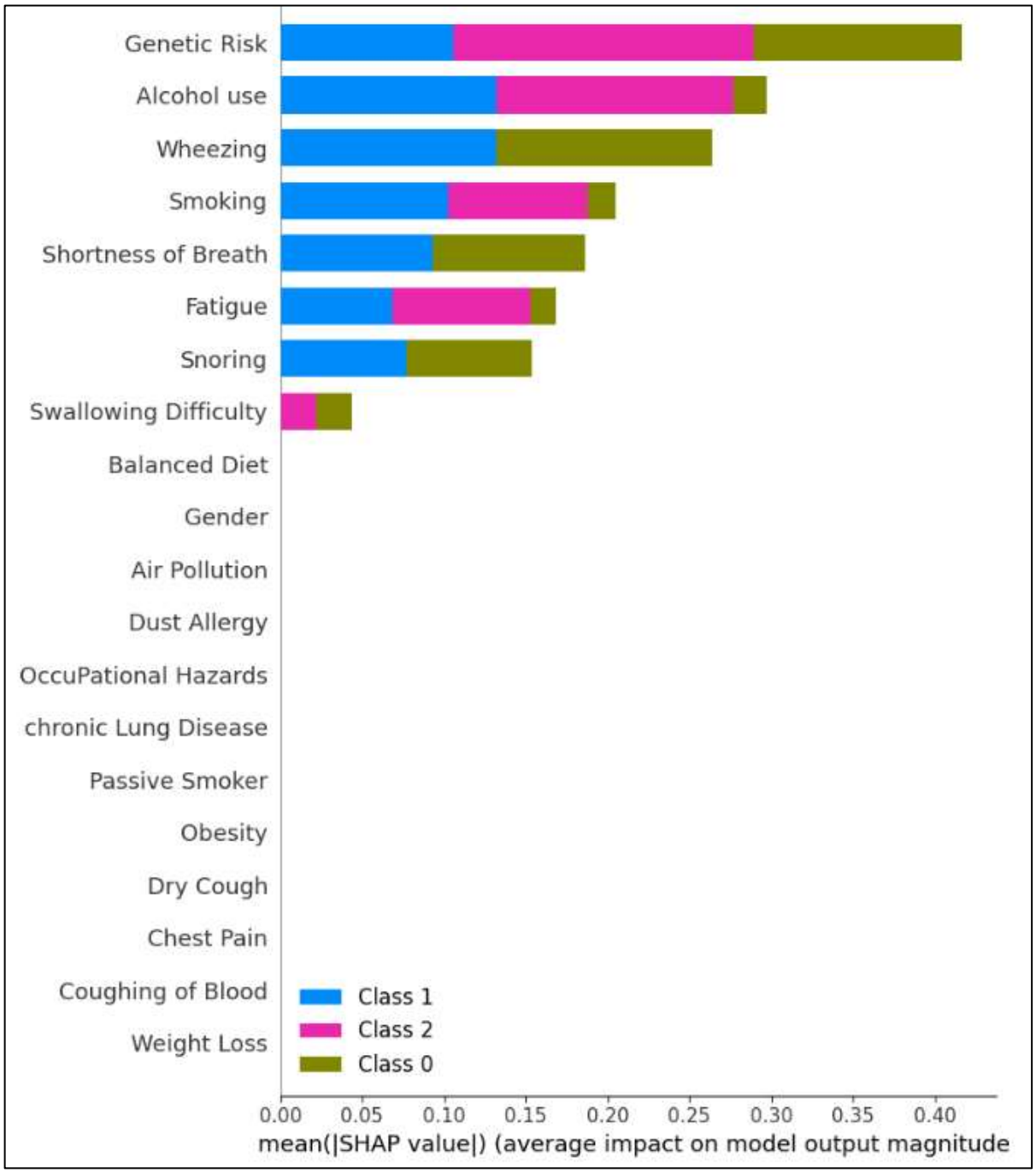


Рисунок 3.39 – Візуалізація важливості ознак за оцінкою яку надає бібліотека SHAP для моделі Decision Tree

На рисунку 3.39 зображена візуалізація важливості ознак за оцінкою бібліотеки SHAP. Порівнюючи результати оцінок за feature_importances та за SHAP бачимо, що SHAP надає більшу вагу для ознаки «Вживання алкоголю» але лише переважно для класифікації категорії «Medium» та «High», у той час як для категоріях «Low» вплив даної ознаки мінімальний. Найбільш впливова визначена ознака «Генетичного ризику» рівномірно розподіляє між собою

категорії «Low» та «Medium» але віддає більше значення для категорії «High». Цікаво що найменш впливова ознака із списку, а саме «Рівень труднощів ковтання», узагалі не має ніякого впливу на категорію «Medium».

3.10 Висновки

В даному розділі було розроблено інформаційну технологію для передбачення ризиків захворіти на рак легень. Були застосовані наступні моделі машинного навчання: Random Forest, AdaBoost, Extra Trees, Decision Tree, XGBoost та Multi-Layer Perceptron, виконано їх налаштування. Зроблено порівняння оцінок моделей за різними метриками, а саме F1_Score та R2_Score. За результатами оцінки даних метрик усі моделі показали результат точності вище 89%, що є дуже хорошим результатом. Також було обрано найкращі моделі за метрикою F1_Score, ними виявились Random Forest із точністю 98% для тестових даних та Decision Tree із точністю 95% для тестових даних. Також проведено дослідження впливу різних ознак на ці моделі та на кожне рішення моделі відповідно до категорій класифікацій з використанням бібліотеки SHAP. Дослідження показали що для моделі Random Forest найвпливовішою ознакою є «Рівень відкашлювання крові» у той час як для моделі Decision Tree такою ознакою є «Рівень генетичних ризиків».

ВИСНОВКИ

В процесі виконання бакалаврської дипломної роботи проведено аналіз предметної області, здійснено аналіз проблематики предметної області, а саме проблеми смертності та проблеми діагностування на ранніх стадіях. Розглянуто основні фактори ризику, які впливають на виникнення даного захворювання та обрано середовище розробки Kaggle та мову програмування Python. Також було описано різні методи машинного навчання, які підходять для вирішення нашої задачі прогнозування та обрано оптимальні.

Здійснена попередня обробка даних та їх очищення для використання в прогнозуванні методами машинного навчання. Також був проведений розвідувальний аналіз даних, який показав що більша частина пацієнтів відповідають середньому віку, а кількісний розподіл по цільовій ознаці «Рівень ризику» рівномірний.

Розроблено інформаційну технологію для передбачення ризиків захворіти на рак легень за допомогою різних методів машинного навчання, а саме Random Forest, AdaBoost, Extra Trees, Decision Tree, XGBoost та MLP. Зроблено порівняння оцінок моделей за метриками F1_Score та R2_Score. За результатами оцінки даних метрик усі моделі показали результат точності вище 89%, що є дуже хорошим результатом. Також було обрано найкращі моделі за метрикою F1_Score, ними виявились Random Forest із точністю прогнозування 98% та Decision Tree із точністю 95%. Проведено дослідження впливу різних ознак на ці моделі. Дослідження показали, що для моделі Random Forest найвпливовішою ознакою є «Рівень відкашлювання крові» у той час як для моделі Decision Tree такою ознакою є «Рівень генетичних ризиків» [20].

За результатами роботи було опубліковано тези на тему «Розвідувальний аналіз даних для інформаційної технології передбачення раку легень методами машинного навчання» на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (м. Вінниця, 2023-2024 рр.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Неволя С. Д., Жуков С. О. Розвідувальний аналіз даних для інформаційної технології передбачення раку легенів методами машинного навчання. *Конференції ВНТУ*, Молодь в науці: дослідження, проблеми, перспективи. 2024. 9Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/author/submission/21234.

2. Прогнозування захворюваності на рак, ВООЗ [Електронний ресурс]. – URL: https://phc.org.ua/news/vooz-prognozue-zrostannya-zakhvoryuvanosti-na-rak-yak-zniziti-rizik-rozvitku-onkonedugi.

3. Що важливо знати про рак легень [Електронний ресурс]. URL: https://www.phc.org.ua/news/scho-vazhlivo-znati-pro-rak-legen.

4. Путівник мовою програмування Python [Електронний ресурс]. – URL: https://pythonguide.rozh2sch.org.ua/#_короткий_опис .

5. Популярність Python [Електронний ресурс]. URL: http://www.itschool.vn.ua/the-popularity-of-python/ .

6. Kaggle Wiki [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Kaggle.

7. Yousheng Zhou. A privacy-preserving logistic regression-based diagnosis scheme for digital healthcare. *Elsevier*, 2023, 63-73p. [Електронний ресурс]. Режим доступу: https://research.google/pubs/logistic-regression-the-importance-of-being-improper/.

8. Early Prediction of Dementia Using Feature Extraction Battery (FEB) and Optimized Support Vector Machine (SVM) for Classification/by Ashir Javeed, Ana Luiza Dallora and others. *Biomedicines*, 2023, 439 [Електронний ресурс]. DOI: https://doi.org/10.3390/biomedicines11020439 .

9. Manzali, Y., Elfar, M. Random Forest Pruning Techniques: A Recent Review. *Oper. Res. Forum* 4, 43 (2023). [Електронний ресурс]. DOI: https://doi.org/10.1007/s43069-023-00223-6 .

10. Amala Sonny, Abhinav Kumar, Linga Reddy Cenkeramaddi. Carry Object Detection Utilizing mmWave Radar Sensors and Ensemble-Based Extra Tree

Classifiers on the Edge Computing Systems. *IEEE Sensors Journal*, 2023, 20137 - 20149p. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/abstract/document/10188595.

11. Intrusion detection using decision tree classifier with feature reduction technique/ Syed Atir Raza, Sania Shamim, Abdul Hannan Khan, Aqsa Anwar. *Mehran University Research Journal Of Engineering & Technology*, 2023, 30-37p. [Електронний ресурс]. DOI: https://search.informit.org/doi/abs/10.3316/informit.002355033595975 .

12. Chunyu Piao , Nan Wang , and Chaofeng Yuan. Rebalance Weights AdaBoost-SVM Model for Imbalanced Data. *Hindawi*, 2023, 26p. [Електронний ресурс]. URL: https://downloads.hindawi.com/archive/2023/4860536.pdf.

13. Intrusion Detection using hybridized Meta-heuristic techniques with Weighted XGBoost Classifier/ Ghulam Mohiuddin, Zhijun Lin та ін. за ред. Ghulam Mohiuddin. *Expert Systems with Applications*, 2023. [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/pii/S0957417423010989 .

14. K, S., Thilagam, P.S. Multi-layer perceptron based fake news classification using knowledge base triples. *Appl Intell* 53, 6276–6287 (2023). [Електронний ресурс]. URL: https://link.springer.com/article/10.1007/s10489-022-03627-9 .

15. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця: ВНТУ, 2024. – 258 с. [Електронний ресурс]. URL: https://docs.vntu.edu.ua/card.php?id=8163 .

16. Lung Cancer Prediction Dataset. Kaggle. 2023 [Електронний ресурс]. URL: https://www.kaggle.com/datasets/thedevastator/cancer-patients-and-air-pollution-a-new-link .

17. Кореляційна матриця Пірсона та Спірмена [Електронний ресурс]. URL: https://www.guru99.com/uk/r-pearson-spearman-correlation.html .

18. Lung Cancer Prediction by SUBHAJEET DAS. Kaggle. 2023 [Електронний ресурс]. URL: https://www.kaggle.com/code/subhajeetdas/lung-cancer-prediction .

19. Used Cars : Analysis and Prediction by Vitalii Mokin. Kaggle. 2024 [Електронний ресурс]. URL: https://www.kaggle.com/code/vbmokin/used-cars-analysis-and-prediction .

20. Lung Cancer Prediction 2.0 by SergiiN. Kaggle. 2024 [Електронний ресурс]. URL: https://www.kaggle.com/sergiin/lung-cancer-prediction-2-0 .

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ РАКУ ЛЕГЕНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

08-34.БДР.010.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Сергій ЖУКОВ
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____ Сергій НЕВОЛЯ
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки:

1) Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця: ВНТУ, 2024. – 258 с. [Електронний ресурс]. URL: https://docs.vntu.edu.ua/card.php?id=8163;

2) Used Cars: Analysis and Prediction by Vitalii Mokin. Kaggle. 2024 [Електронний ресурс]. URL: https://www.kaggle.com/code/vbmokin/used-cars-analysis-and-prediction.

3. Мета і призначення роботи.

Метою дослідження є підвищення точності передбачення ризику захворіти на рак легень шляхом створення інформаційної технології.

4. Вихідні дані для проведення робіт:

Kaggle Dataset «Lung Cancer Prediction» [Електронний ресурс]. URL: https://www.kaggle.com/datasets/thedevastator/cancer-patients-and-air-pollution-a-new-link.

5. Методи дослідження:

Використання мови програмування Python та методів машинного навчання:Random Forest, AdaBoost, Extra Trees, Decision Tree, XGBoost та Multi-Layer Perceptron.

6. Етапи роботи і терміни їх виконання:

a) Аналіз проблематики предметної області _____ – _____
b) Аналіз оптимальних інформаційних технологій _____ – _____
c) Розвідувальний аналіз даних _____ – _____
d) Розроблення інформаційної технології _____ – _____
e) Аналіз результатів дослідження _____ – _____
f) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології для передбачення ризиків захворіти раком легень методами машинного навчання.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.
Початок розробки «___» _____ 2024 р.
Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____ Сергій НЕВОЛЯ

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технологія передбачення раку легенів методами машинного навчання»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 97.68% Схожість 2.32%

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
o Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
o Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Сергій НЕВОЛЯ

Керівник роботи _____ (підпис) Сергій ЖУКОВ

Додаток В

(довідниковий)

Фрагмент лістингу програми

Код ноутбуку «Lung cancer Prediction 2.0»

```
!pip install dtreeviz

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import xgboost as xgb

import matplotlib.gridspec as gridspec

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neural_network import MLPClassifier

from xgboost import XGBClassifier

from sklearn.metrics import precision_score, recall_score, accuracy_score, r2_score, f1_score,
confusion_matrix, ConfusionMatrixDisplay, classification_report

from sklearn import preprocessing

from sklearn.model_selection import GridSearchCV

import shap

import warnings

warnings.filterwarnings("ignore")

df = pd.read_csv("/kaggle/input/cancer-patients-and-air-pollution-a-new-link/cancer patient data sets.csv")

df.drop(columns=['index', 'Patient Id'], axis=1, inplace=True)

df.describe()

# Replace "level" with Integer

print('Cancer Levels: ', df['Level'].unique())

df["Level"].replace({'High': 2, 'Medium': 1, 'Low': 0}, inplace=True)

print('Cancer Levels: ', df['Level'].unique())

df.info()

EDA

plt.figure(figsize=(10,4))

plt.boxplot(df['Age'], vert=False, patch_artist=True,

            boxprops=dict(facecolor='skyblue', linewidth=2),

            whiskerprops=dict(color='green',linewidth=3),

            medianprops=dict(color='red',linewidth=2)

           )

plt.title('Розподіл даних по віку пацієнтів')
```

```
plt.xticks(np.arange(10, max(df['Age'])+1, 3))

plt.show()

import statsmodels.api as sm

data = data = df['Age'].dropna()

# Побудова QQ-графіку

sm.qqplot(data, line='s')

plt.title("Розподіл даних для стовбця 'Age'")

plt.show()

df_corr = df.corr()

df_corr

plt.title("Correlation Matrix")

sns.heatmap(df_corr, cmap='viridis')

print('\n')

plt.figure(figsize=(20,15))

sns.heatmap(df.corr(), annot=True, cmap=plt.cm.PuBu)

plt.show()

print('\n')

for i in range(24):

    plt.subplot(16, 2, i+1)

    sns.distplot(df.iloc[:, i], color = 'red')

    plt.grid()

plt.figure(figsize = (15,7))

colors = ['red', 'yellow', 'green']

plt.title("Lung Cancer Chances ")

plt.pie(df['Level'].value_counts(), explode = (0.1, 0.02, 0.02), labels = ['High', 'Medium', 'Low'], autopct
= "%1.2f%%", shadow = True, colors = colors)

plt.legend(title = "Lung Cancer Chances", loc = "lower left")

sns.displot(df['Level'], kde=True, color = 'red')

dfviz = df.copy()

y = df.pop('Level')

x = df

Train & Test Splitting the Data

# Data splitting

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Results of prediction

results = pd.DataFrame(columns = ['model', 'f1_train', 'f1_test', 'r2_train', 'r2_test'])

Random Forest

from sklearn.model_selection import GridSearchCV

param_RF = {

    'n_estimators': [50, 100, 150],

    'max_depth': [2],

    'criterion':['gini'],
```

```
    'min_samples_split': [2, 3],
    'min_samples_leaf': [2, 3],
    'random_state' : [42],
    'max_samples': [0.4]}
%%time
model_tuning = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_RF, cv=5)
model_tuning.fit(x_train, y_train)
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_rf = RandomForestClassifier(**best_params)
model_rf.fit(x_train, y_train)
# Train
train_predictions = model_rf.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')
# Test
test_predictions = model_rf.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')
#Result
performTrain(train_predictions)
performTest(test_predictions)
score_model_rf = model_rf.score(x_test, y_test)
ADABoost Classifier
param_ADA = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1],
    'random_state' : [42],
    'algorithm': ['SAMME.R'],  }
%%time
model_tuning = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_ADA, cv=5)
model_tuning.fit(x_train, y_train)
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_ada = AdaBoostClassifier(**best_params)
model_ada.fit(x_train, y_train)
# Train
train_predictions = model_ada.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')
# Test
```

```
test_predictions = model_ada.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')
#Result
performTrain(train_predictions)
performTest(test_predictions)
score_model_ada = model_ada.score(x_test, y_test)
Extra Trees Classifier
params_etc ={'n_estimators':[200, 300, 500],
             'max_depth': [3, 4, 5],
             'min_samples_split':  [2, 3],
             'min_samples_leaf':  [2, 3],
             'max_features': ['sqrt'],
             'ccp_alpha': [0.1],
             'random_state' : [42]}
%%time
model_tuning = GridSearchCV(estimator=ExtraTreesClassifier(), param_grid=params_etc, cv=5)
model_tuning.fit(x_train, y_train)
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_etc = ExtraTreesClassifier(**best_params)
model_etc.fit(x_train, y_train)
# Train
train_predictions = model_etc.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')
# Test
test_predictions = model_etc.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')
#Result
performTrain(train_predictions)
performTest(test_predictions)
score_model_etc = model_etc.score(x_test, y_test)
Decision Tree
params_DT ={'max_depth': [3, 4],
             'min_samples_split': [1, 2, 3],
             'min_samples_leaf': [0, 1, 2],
             'max_features': ['sqrt', 'log2', 'auto'],
             'max_leaf_nodes': [ 5, 10, 15, 17],
             'random_state' : [42]}
```

```
%%time

model_tuning = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=params_DT, cv=5)

model_tuning.fit(x_train, y_train)

best_params = model_tuning.best_params_

print("Best Parameters:", best_params)

model_dt = DecisionTreeClassifier(**best_params)

model_dt.fit(x_train, y_train)

# Train

train_predictions = model_dt.predict(x_train)

r2_train = r2_score(y_train, train_predictions)

f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test

test_predictions = model_dt.predict(x_test)

r2_test = r2_score(y_test, test_predictions)

f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result

performTrain(train_predictions)

performTest(test_predictions)

score_model_dt = model_dt.score(x_test, y_test)

Decision Tree Visualization

feature_names = dfviz.columns[0:23]

viz = dfviz.copy()

viz["Level"]=viz["Level"].values.astype(str)

print(viz.dtypes)

target_names = viz['Level'].unique().tolist()

from sklearn.tree import plot_tree # tree diagram

plt.figure(figsize=(15, 10))

plot_tree(model_dt, feature_names = feature_names, class_names = target_names, filled = True, rounded =
False)

plt.savefig('tree_visualization.png')

import dtreeviz

viz_model = dtreeviz.model(model_dt,

                           X_train=x_train, y_train=y_train,

                           feature_names=feature_names,

                           target_name='Lung Cancer',

                           class_names=['Low', 'Medium', 'High'])

v = viz_model.view()     # render as SVG into internal object

v.save("Lung Cancer.svg")  # save as svg

viz_model.view()

XGBoost Classifier

params_XGB ={'n_estimators': [100, 200, 300],

             'num_leaves': [2, 5, 7, 10],
```

```
              'max_depth': [2, 3, 5],
              'subsample': [0.01],
              'learning_rate': [0.01],
              'objective': ['multi:softmax'],
              'random_state' : [42],
              'num_class': [3]}
%%time
model_tuning = GridSearchCV(estimator=XGBClassifier(), param_grid=params_XGB, cv=5)
model_tuning.fit(x_train, y_train)
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_xgb = XGBClassifier(**best_params)
model_xgb.fit(x_train, y_train)
# Train
train_predictions = model_xgb.predict(x_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')
# Test
test_predictions = model_xgb.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')
#Result
performTrain(train_predictions)
performTest(test_predictions)
score_model_xgb = model_xgb.score(x_test, y_test)
Multi-Layer Perceptron Classifier
params_mlp ={'hidden_layer_sizes': [50, 75, 100],
              'learning_rate_init': [0.001],
              'max_iter': [100, 200, 300],
              'solver': ['adam'],
              'early_stopping': [True],
              'random_state' : [42]}
%%time
model_tuning = GridSearchCV(estimator=MLPClassifier(), param_grid=params_mlp, cv=5)
model_tuning.fit(x_train, y_train)
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_mlp = MLPClassifier(**best_params)
model_mlp.fit(x_train, y_train)
# Train
train_predictions = model_mlp.predict(x_train)
```

```
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')
# Test
test_predictions = model_mlp.predict(x_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')
#Result
performTrain(train_predictions)
performTest(test_predictions)
Comparison
display(results)
# Feature importance for Random Forest Clasifire
feature_importances_model_rf = pd.DataFrame(x_train.columns)
feature_importances_model_rf.columns = ['feature']
feature_importances_model_rf["score_model_rf"] = pd.Series(model_rf.feature_importances_)
feature_importances_model_rf.sort_values(by='score_model_rf', ascending=False)
importances = model_rf.feature_importances_
feature_names = df.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=True)
importances = model_rf.feature_importances_
feature_names = df.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=True)
# Feature importance for Decision Tree Clasifire
feature_importances_model_dt = pd.DataFrame(x_train.columns)
feature_importances_model_dt.columns = ['feature']
feature_importances_model_dt["score_model_dt"] = pd.Series(model_dt.feature_importances_)
feature_importances_model_dt.sort_values(by='score_model_dt', ascending=False)
# Feature importance
importances = model_dt.feature_importances_
feature_names = df.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=True)
explainer = shap.Explainer(model_dt, x)
shap_values = explainer.shap_values(x)
shap.summary_plot(shap_values, x)
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ РАКУ ЛЕГЕНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

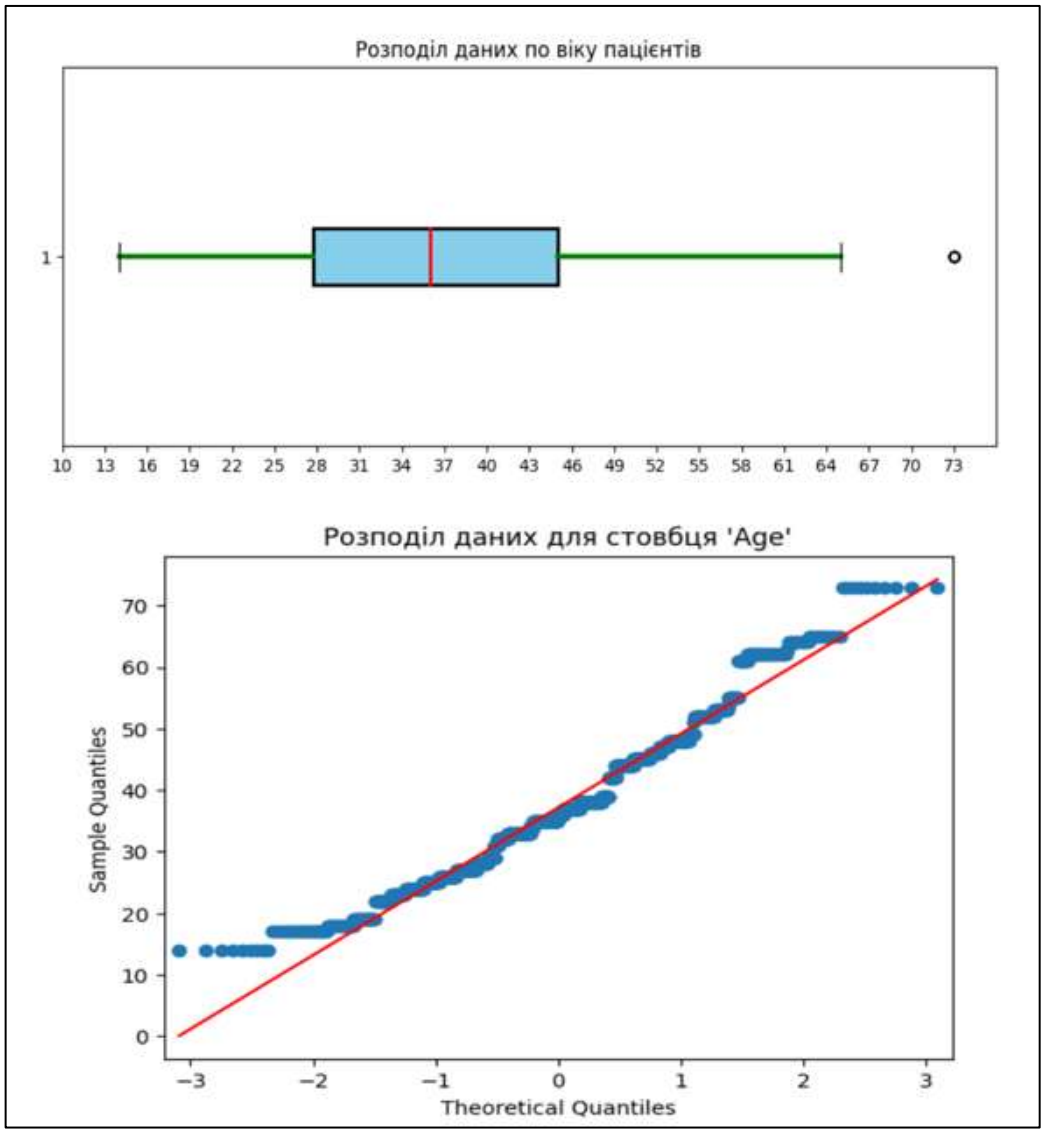


Рисунок Г.1 – Діаграми розподілу даних відповідно до віку пацієнтів

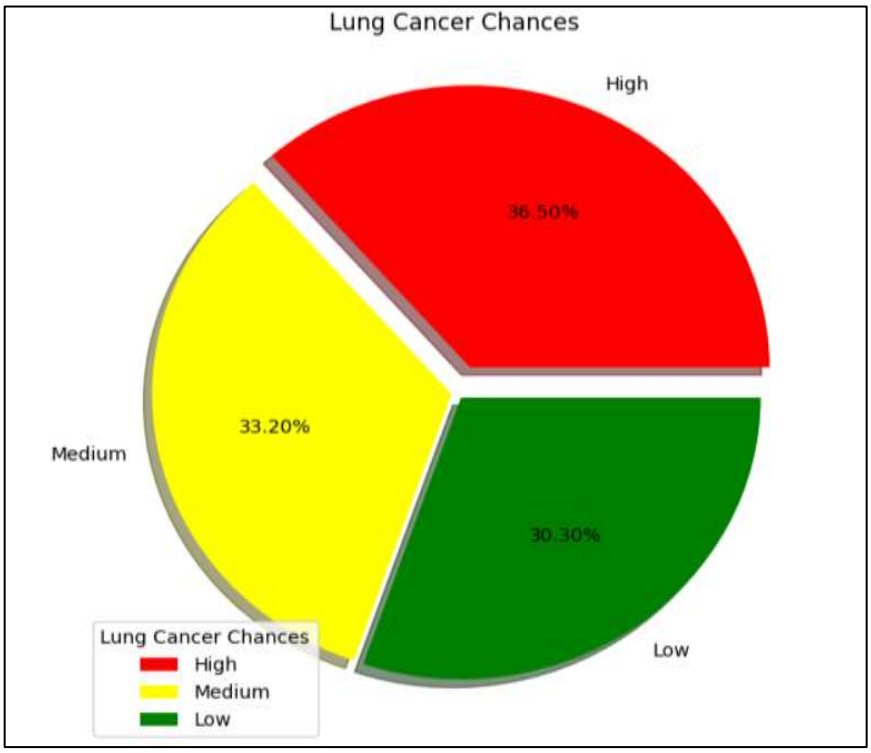


Рисунок Г.2 – Кількісний розподіл вибірки датасету у відсотках по цільовій ознаці Level

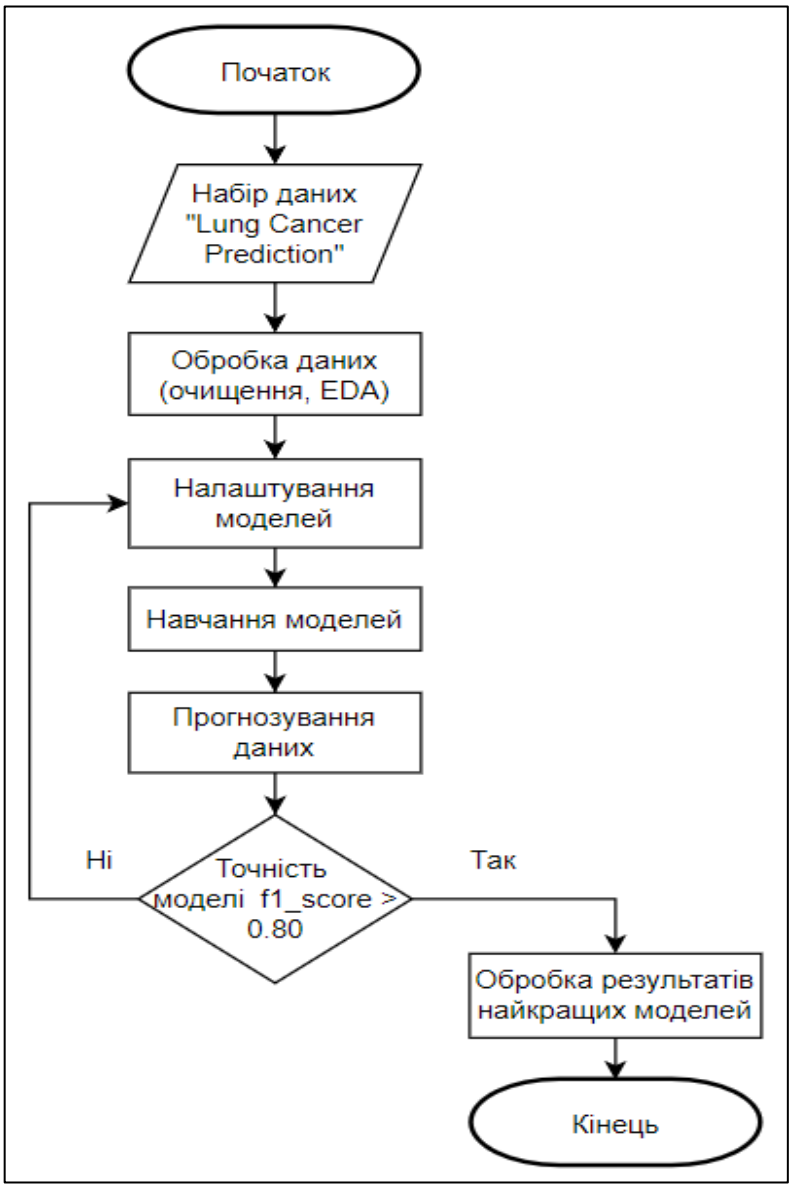


Рисунок Г.3 – Блок-схема алгоритму роботи інформаційної технології

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	0.9675	0.98	0.905566	0.925012	RF
1	ADABoost Classifier	0.90125	0.895	0.850794	0.842526	ADA
2	ExtraTrees Classifier	0.8875	0.895	0.78469	0.797533	ExT
3	DecisionTree Classifier	0.9625	0.95	0.898011	0.880019	DT
4	XGB Classifier	0.89	0.935	0.788467	0.857523	XGB
5	MLP Classifier	0.9125	0.94	0.862126	0.910015	MLP

Рисунок Г.4 – Результати навчання усіх моделей

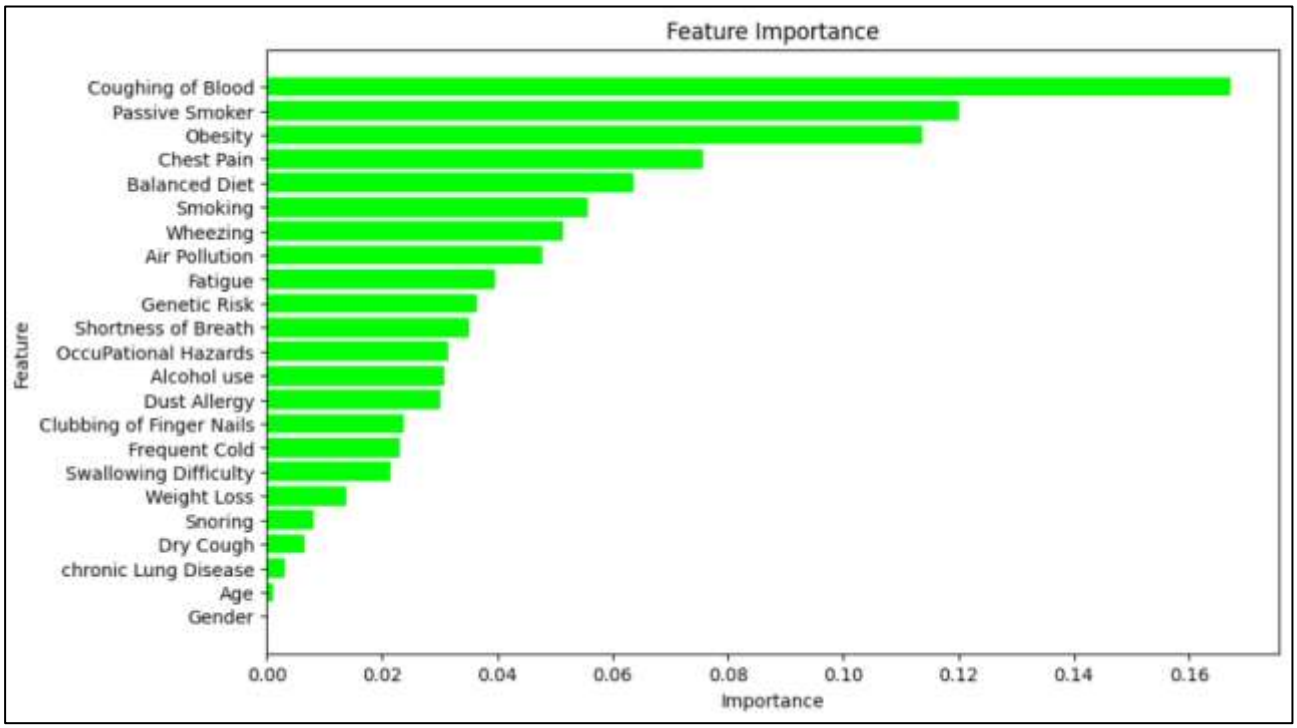


Рисунок Г.5 –Діаграма важливості ознак для моделі Random Forest

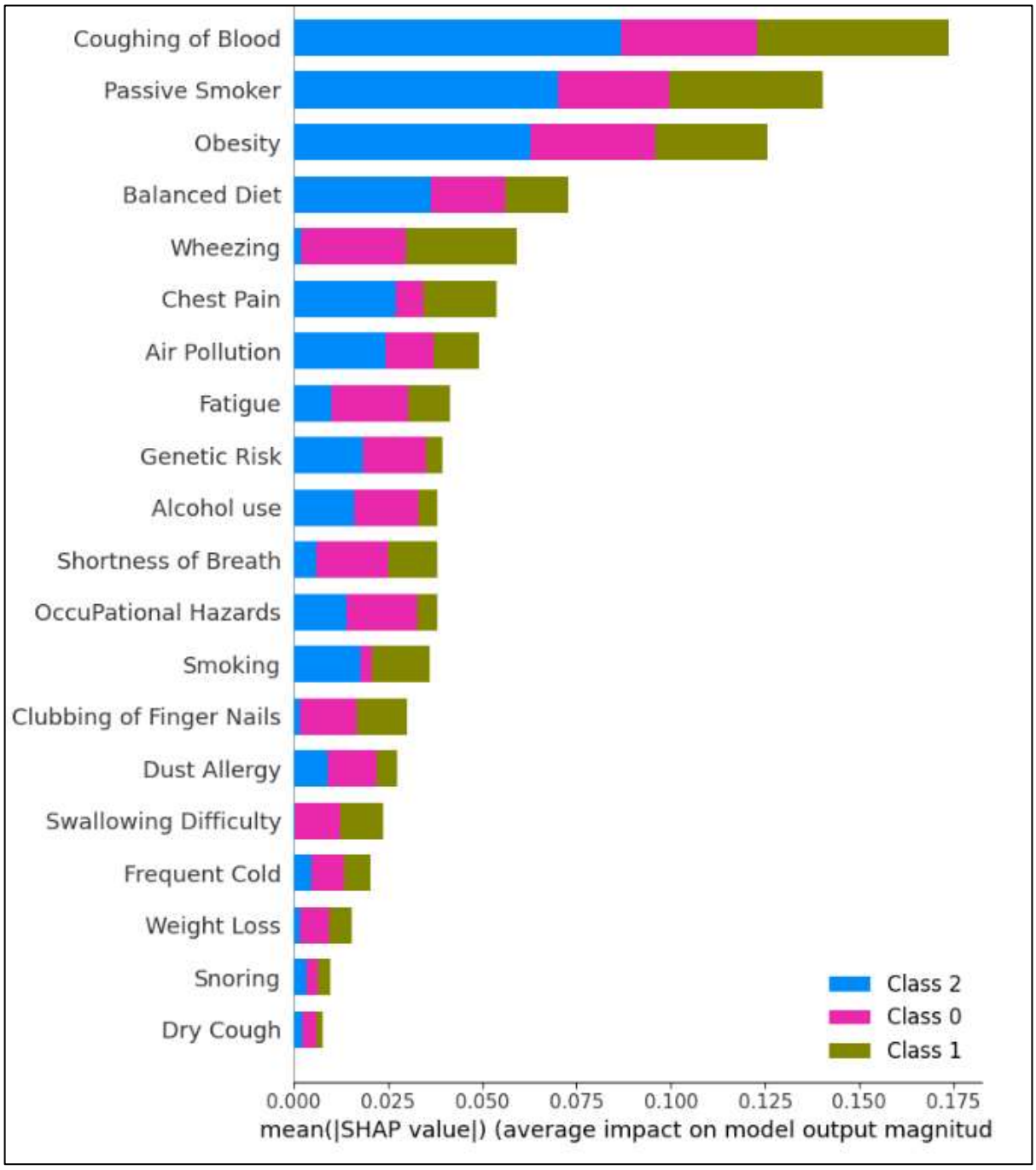


Рисунок Г.6 –Діаграма важливості ознак за оцінкою бібліотеки SHAP для моделі Random Forest

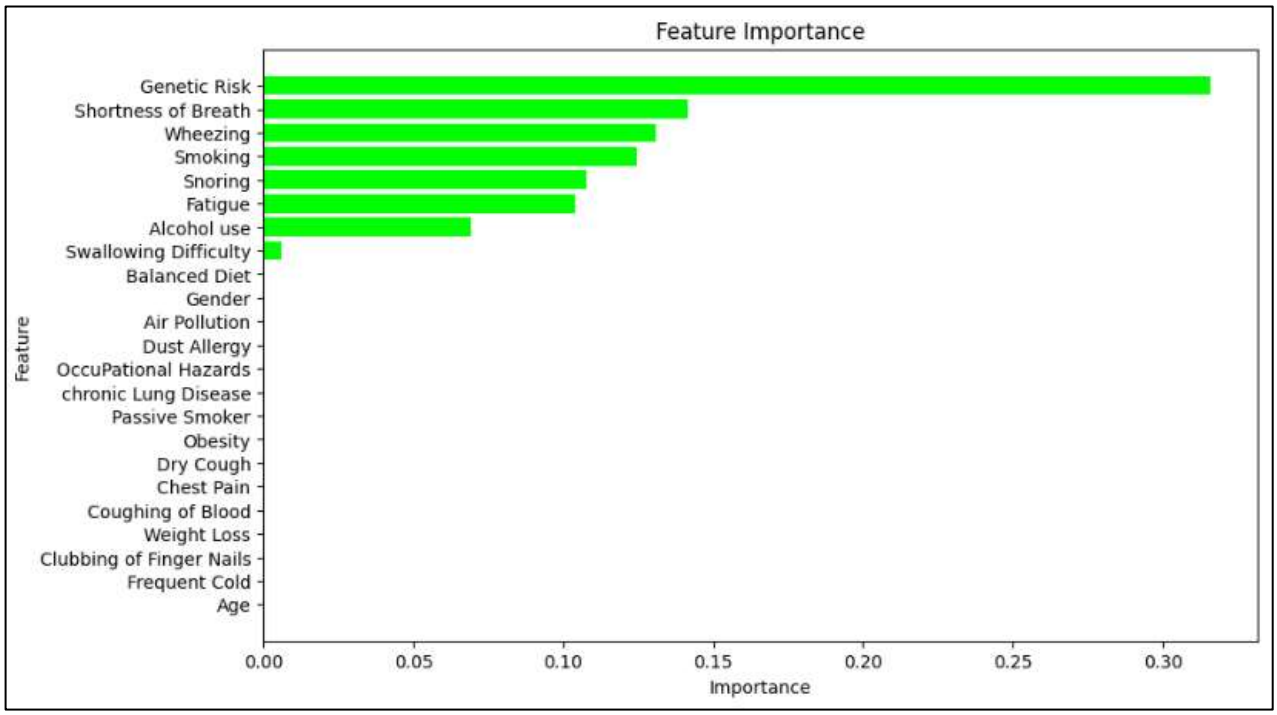


Рисунок Г.7 – Діаграма важливості ознак для моделі Decision Tree

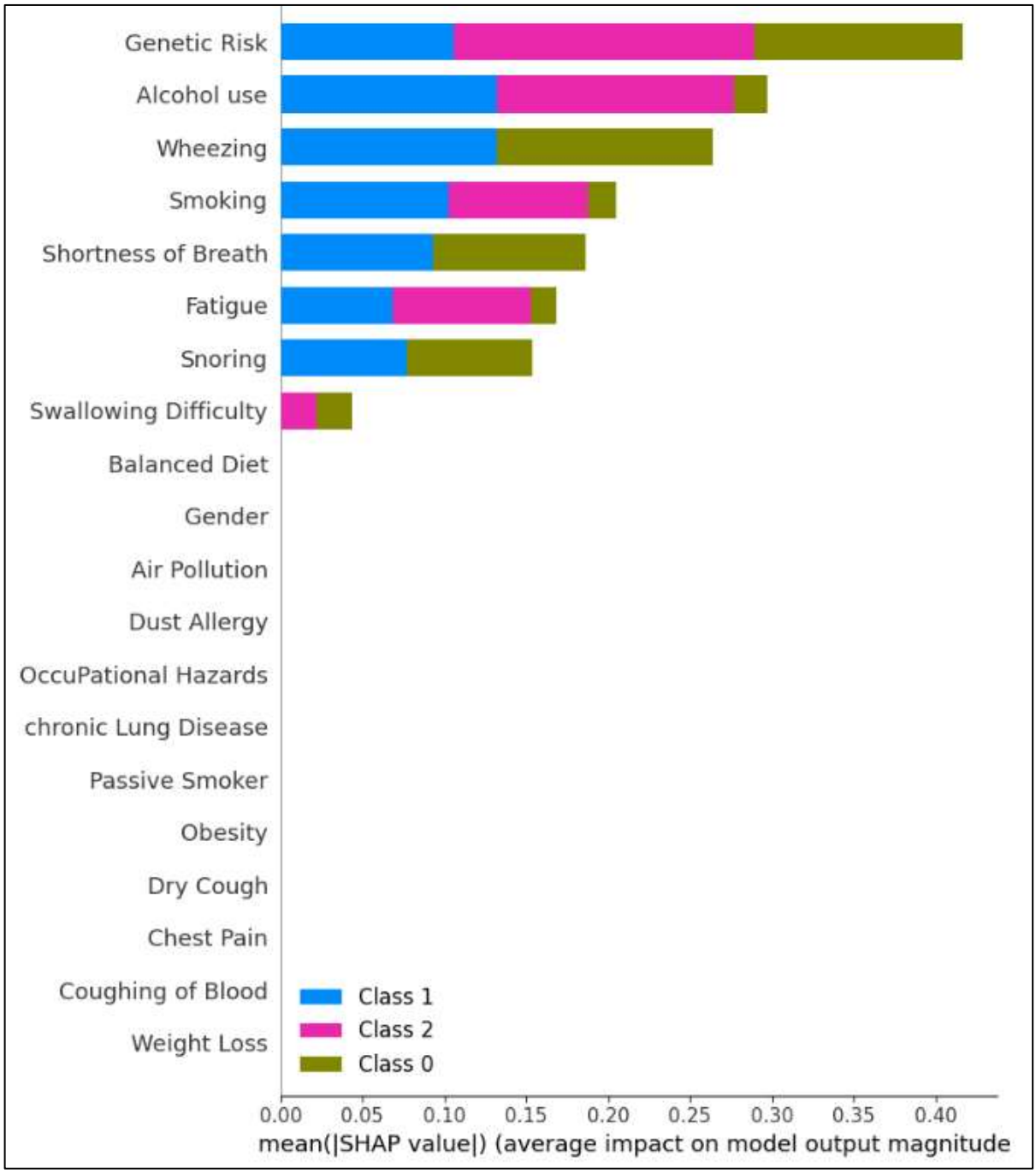


Рисунок Г.8 – Діаграма важливості ознак за оцінкою бібліотеки SHAP для моделі Decision Tree