Шифр: «*biotextify***»**

Використання алгоритмів машинного навчання для класифікації генетичних мутацій

АНОТАЦІЯ

Актуальність: Алгоритми машинного навчання знаходять широке застосування в багатьох галузях наукової та практичної діяльності, у тому числі медичній. Одна з важливих практичних задач, що може бути вирішена за допомогою цих алгоритмів, полягає в класифікації генетичних мутацій за їх клінічними текстовими описами. На даний час експерти вирішують цю задачу вручну. Тому вкрай актуальним є впровадження алгоритмів машинного навчання і програмного забезпечення на їх основі для автоматизації процесу класифікації генетичних мутацій.

Об'єкт дослідження: класифікація генетичних мутацій за їх текстовим описанням.

Мета роботи: розв'язати задачу класифікації генетичних мутацій за їх текстовим описанням на основі алгоритмів машинного навчання; проаналізувати навчальну вибірку клінічних описів генетичних мутацій, нормалізувати її, привести до виду, який можна подати на вхід алгоритму класифікації, провести класифікацію та оцінити її якість, порівняти між собою підмножини алгоритмів, обрані на кожному етапі розв'язку задачі та зробити висновки щодо проведеного дослідження.

Інформація про результат: розроблене програмне забезпечення «biotextify» у програмному середовищі Python3.6, яке містить наступну функціональність: методи нормалізації тексту, алгоритми перенесення слів у векторний простір, зменшення простору ознак, класифікації даних; за допомогою програмного забезпечення проведено розв'язання практичної задачі класифікації генетичних мутацій за їх текстовим описанням; проведена оцінка якості виконаної класифікації та інтерпретація одержаних результатів.

Перелік ключових слів: КЛАСИФІКАЦІЯ, ТЕКСТОВИЙ КОРПУС, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ВЧИТЕЛЕМ, ВЕКТОРИ, НОРМАЛІЗАЦІЯ, БАЛАНСУВАННЯ, ГЕНЕТИЧНІ МУТАЦІЇ, NLP.

ЗМІСТ

ВСТУП

Алгоритми машинного навчання знаходять все більш широке застосування в медичній сфері, зокрема через потребу обробки великих даних. Сучасною сферою медицини, що генерує великі об’єми даних, є персоналізована медицина, в рамках якої діагностику та лікування пацієнтів здійснюють на основі їх індивідуальних, зокрема генетичних, особливостей. Такі генетичні особливості використовують, наприклад, у процесі діагностики ракових захворювань. В процесі аналізу генетичної інформації (секвенованого генома пацієнта) лікарі стикаються з тим, що в ній містяться мутації, які ведуть до розвитку пухлини, та мутації, які є нейтральними. Наразі лікарі класифікують їх самостійно, аналізуючи текстову клінічну літературу.

Мета даної роботи полягала у розв’язанні задачі класифікації генетичних мутацій на основі аналізу текстової клінічної літератури за допомогою алгоритмів машинного навчання.

Робота складається з трьох розділів. У першому розділі міститься схема вирішення задачі та огляд алгоритмів, що використовуються для її вирішення, зокрема алгоритмів нормалізації тексту, його представлення у векторному просторі, зменшення розмірності такого простору та безпосередньо класифікації. У другому розділі подано опис програмного забезпечення, створеного для вирішення поставленої задачі. У третьому розділі описано результати обчислювальних експериментів та подано інтерпретацію отриманих результатів.

ПОСТАНОВКА ЗАДАЧІ

Задано навчальну вибірку з інформацією про класи генетичних мутацій та документи з описаннями цих мутацій. Вибірку задано у вигляді матриці:

$$\begin{pmatrix} 1 & G_1 & M_1 & C_1 & d_1 \\ \dots & \dots & \dots & \dots & \dots \\ i & G_i & M_i & C_i & d_i \\ \dots & \dots & \dots & \dots & \dots \\ N & G_N & M_N & C_N & d_N \end{pmatrix},$$

де i – порядковий номер, G_i – ген, в якому відбулась мутація; M_i – тип мутації; C_i – клас, до якого належить дана мутація; d_i – документ, що містить в собі набір слів, які описують один клінічний доказ i-ї мутації; N – кількість документів.

Також задано тестову вибірку, що має такий самий вид, що і навчальна, окрім того, що в ній відсутні мітки класів (C).

Ставиться задача класифікації генетичних мутацій на основі їх текстових клінічних описів, яка вимагає побудови класифікатора на основі навчальної вибірки та перевірки його якості на основі тестової вибірки.

Для розв’язання поставленої задачі необхідно розробити програмне забезпечення з наступним функціоналом:

– видалення шумів з документів текстового корпусу (під текстовим корпусом розуміється набір усіх документів вибірки), за їх наявності;

– зменшення об’єму словника за допомогою морфологічних методів;

– перевірка того, чи є розподіл класів у навчальній вибірці рівномірним і, якщо необхідно, балансування вибірки;

– відображення текстового корпусу у векторний простір різними методами;

– вилучення найменш інформативних ознак з навчальної вибірки;

– класифікація генетичних мутацій на основі векторних даних;

– оцінювання якості класифікації на основі таких метрик як Accuracy, Precision, Recall, F1-Score, Log Loss, AUC.

1. ОГЛЯД АЛГОРИТМІВ ДЛЯ КЛАСИФІКАЦІЇ БІОМЕДИЧНОГО ТЕКСТОВОГО КОРПУСУ

1.1 Схема вирішення задачі класифікації біомедичного текстового корпусу

Під час вирішення будь-якої задачі за допомогою алгоритмів машинного навчання недостатньо лише обрати та застосувати окремий алгоритм. Процес вирішення конкретної задачі включає в себе декілька важливих етапів, особливо, якщо початкові дані мають специфічний формат, наприклад, текстовий, звуковий або формат зображення.

Для вирішення задачі класифікації текстових даних, які мають місце в даній роботі, доцільним буде дотримуватися наступної схеми:

1. Нормалізувати текстовий корпус. Для цього виконати нормалізацію слів, що містяться у корпусі, та корегування розмірів класів у разі незбалансованих класів. Нормалізація слів може проводитися шляхом фільтрації слів, трансформації слів різними морфологічними техніками, об’єднання слів з одного контексту. Корегування розмірів класів може проводитися методами балансування даних.
2. Маючи нормалізовані текстові дані необхідно їх перенести у векторний простір, щоб у подальших кроках з ними міг працювати класифікатор. Існують різні групи методів для перенесення слів у векторний простір, такі як Bag of Words, Word Embedding та група методів Topic Modelling.
3. Провести аналіз ознак числових векторів, одержаних після виконання другого кроку. Даний етап має назву Feature Engineering. Якщо кількість ознак виявиться завеликою, то потрібно використаним алгоритми зменшення простору ознак, наприклад, Random Projection або TruncatedSVD. Також на цьому етапі бажано провести аналіз на наявність додаткових ознак, які можуть значно підвищити якість класифікації. Якщо такі ознаки будуть знайдені, то додати їх до об’єктів вибірки.

4. Безпосередньо провести класифікацію даних, отриманих з попередніх етапів. Алгоритмами, що використовують для класифікації та категоризації текстових даних, є Naiive Bayes, Support Vector Machine, Logistic Regression, Random Forest, XGBoost.

5. Порівняти результати роботи алгоритмів за такими метриками як Accuracy, Log Loss, Precision, Recall, F1-Score, AUC, побудувати Confusion Matrix та ROC Curve. На основі даних метрик зробити висновок щодо найефективніших комбінацій алгоритмів.

Фінальний результат вирішення задачі прямопропорційно залежить від кожного з вищенаведених етапів, тож зупинимося на кожному з них більш детально та розглянемо алгоритми, які можуть бути для них реалізовані.

1.2 Методи нормалізації текстового корпусу

Відбір значимих слів

Загальний обсяг унікальних слів у текстовому корпусі може бути дуже великим. Та чи усі ці слова мають логічний сенс і впливають на якість відокремлення класів один від одного? Скоріше за все - ні. Для того, щоб зрозуміти чи варто використовувати те чи інше слово у подальшому дослідженні і роботі з корпусом, можна проаналізувати слова, які містяться у словнику унікальних слів, побудованому на початковому корпусі. Має зміст проводити аналіз кожного слова на основі таких характеристик як семантична змістовність, відсутність слів, які являють собою комбінацію даного слова з іншими, та інших.

Аналізуючи наукову літературу, зокрема біомедичну, можна відзначити, що в ній наявно багато чисел, які не мають жодного змісту без контексту. Також, присутні скорочення посилань на авторів досліджень типу «et al.», спеціальні символи, що не виражають сенсу для читача та аналітика.

Для боротьби з вищезазначеними шумами в текстовому корпусі можна провести нормалізацію в два етапи:

− видалити усі стоп-слова,

− виділити спеціальні символи з комбінацій слово + спеціальний символ та залишити лише слово.

Стоп-словами або шумовими словами для певної мови називають такі слова, які не несуть жодного смислового навантаження, тому їх користь та роль для пошуку і побудови моделі з тексту не суттєва. До таких загальних слів можна віднести прийменники, суфікси, дієприкметники, вигуки, цифри тощо. Для отримання списку стоп-слів англійської мови можна використовувати бібліотеку NLTK та її пакет corpus, в якому і міститься список шумових слів під назвою stopwords.

Щодо другого етапу нормалізації, ефективним інструментом є регулярні вирази. В даному випадку, вони використовуються саме для замін у словах-комбінаціях або спеціальних шумових словах, специфічних для біомедичного корпусу. Для цієї задачі може використовуватися наступний регулярний вираз:

[,);!\"(\'~:\-+<>#?®™/[\]/+=±%→*&$’”@•†{}|¢§©«_¶½× 〈〉

⩾○♦☆≥≤≈∞∑∗~]|(\d{4})|(et al.)|(\.{2,})

В даному виразі містяться спеціальні символи, що зустрічаються в комбінаціях зі словами (позначені зеленим кольором), 4 цифри, що у багатьох випадках позначають рік певного дослідження (позначені червоним), скорочення авторів досліджень (позначено синім) та групи точок від двох та більше (позначені фіолетовим).

Морфологічні методи нормалізації текстового корпусу

Аналізуючи велику кількість літератури на одну тематику, наприклад, з однієї наукової області, можна стикнутися з тим, що багато слів зустрічається в тексті у різній формі. Це може бути і форма частини мови (слово зустрічається прикметник і іменник), так і форма відмінку, часу (для дієслів, дієприкметників, дієприслівників) й інше. При побудові словника унікальних слів необхідно враховувати даний факт. Тож, якщо поставлена ціль максимально ефективно скоротити об’єм словника і тим самим не втратити

дисперсію унікальних слів у ньому, можна використати такі морфологічні техніки як стемінг (stemming) та лематизація (lemmatization).

Стемінг – це процес знаходження основи слова для заданого вхідного слова. Основа слова не обов'язково збігається з морфологічним коренем слова. Тобто дана техніка у багатьох випадках зведе однокореневі слова до свого кореня, незважаючи на те, якими частинами мови ці слова є.

Іншою морфологічною технікою, яку буде доцільно розглянути, є лематизація. Лематизація являє собою процес приведення словоформи до леми - її нормальної (словникової) формі. Для кожної частини мови така форма може різнитися. Наприклад, для іменників – це називний відмінок, для дієслів – інфінітивна форма дієслова. Практична реалізація техніки лематизації міститься у пакеті WordNet Lemmatizer бібліотеки NLTK. Даний пакет необхідно попередньо завантажити командою nltk.download('wordnet').

З точки зору нормалізації тексту вищенаведені морфологічні техніки різняться своєю силою. Стемінг доцільно використовувати, коли немає залежності від частини мови слів. Лематизація в свою чергу є більш «м'якою» версією нормалізації і використовується коли слово міститься в різних контекстах і як різна частина мови.

1.3 Алгоритми перенесення тексту у векторний простір

Відображення тексту у векторний простір – це перетворення текстових даних (слів, речень, документів) у векторний вигляд, де кожна така текстова одиниця замінюється на певний числовий вектор. Таке перетворення є необхідним, оскільки усі алгоритми класифікації працюють з числовими (дискретними та неперервними) ознаками, а використовувати метод One Hot Encoding для перетворення великого текстового корпусу є недоречним. Тож далі наведені найбільш ефективні методи відображення тексту у векторний простір.

Методи на основі BagOfWords

Найбільш простим методом подання документів у векторному вигляді є так званий *Bag Of Words* («мішок слів»). В даному випадку на основі набору документів будується словник з усіх, що зустрічаються в ньому N-грам, де N менше або дорівнює заздалегідь заданому значенню. Документ перетворюється в набір ознак, кожному з яких відповідає N-грама зі словника.

BinaryBOW. У найпростішому, бінарному, випадку дана ознака приймає значення 1 в разі, якщо в документі зустрічається відповідна N-грама, і 0 – інакше.

CountBOW. Припускаючи, що значимість появи N-грами в документі тим більше, чим частіше вона в ньому з'являється, цей метод враховує кількість входжень N-грами в документі, крім самого факту входження.

TF-IDF. Дана схема перенесення у векторний простір використовує припущення про те, що значимість N-грами прямо пропорційна частоті її появи в документі і обернено пропорційна частці документів в наборі, в яких ця N-грама зустрічається. Таким чином, найбільшу вагу отримує N-грама, яка часто зустрічається в одному документі, але не зустрічається в інших, а значить – відрізняє цей документ від інших. Ознаки документів в цьому підході є множення двох величин: частоти N-грами і зворотної частоти документа

$$TF \cdot IDF(t_i, d_j, D) = tf(t_i, d_j) \cdot \log \frac{|D|}{|(d_i \supset t_i)|}$$

де $tf(t_i, d_j)$ - частота N-грами t_i в документі

$D -$ набір документів,

$|(d_i \supset t_i)|$- усі документи d_i в наборі, в яких зустрічається N-грама t_i .

Word Embeddings

Word2Vec. Ця модель навчена на корпусі текстів, і відображає слова в векторний простір невеликої розмірності таким чином, що відстань між ними тим менше, чим ближче значення цих слів. Такий ефект досягається за допомогою штучної нейронної мережі, натренованої передбачати по вектору

слова його контекст; таким чином слова, що з'являються в подібних контекстах, відображаються в близькі вектора.

У 2015, після публікації статті про векторне подання слів, були запропоновані два інші методи векторного представлення документів під загальною назвою *Paragraph Vectors*. Вони використовують схожу з word2vec нейромережеву модель, намагаючись по вектору, який можна застосовувати до документу (а не до окремого слова), передбачити чи зустрічаються в ньому певні слова.

1.4 Алгоритми зменшення простору ознак в розріджених даних

Методи перенесення текстового корпусу у векторний простір дають можливість перейти від вигляду $document_i = \{word_{i1}, word_{i2} \dots word_{iN}\}$ до вигляду $document_i = \{vector_1, 0, .. vector_d, \dots, vector_{s,} 0\}$, де $vector_d$ – векторне значення d-го слова у словнику унікальних слів корпусу. У даного вигляду є одна особливість – масив таких векторів має розріджену структуру, тобто велика частина елементів масиву дорівнює 0. Одночасно з цим явищем, розмірність масиву є здебільшого занадто велика для використання його при створенні навчальної вибірки для класифікатора. Тож для зменшення розмірності таких масивів, беручи до уваги розріджену структуру, використовують такі методи як Random Projection і TruncatedSVD. Коротко розглянемо їх математичне пояснення та особливості.

Random Projection

В алгоритмі Random Projection оригінальний простір розмірності d проектується у підпростір розмірності k ($k \ll d$) шляхом побудови випадкової матриці R розміру $k \times d$, стовпчики якої мають одиничну довжину. Використовуючи матричну форму запису, де $X_{d \times N}$ – це оригінальний набір з N спостережень розмірності d,

$$X^{RP}_{k \times N} = R_{k \times d} X_{d \times N}$$

є проекцією даних у k-вимірний підпростір. Ключова ідея випадкового відображення постає з леми Джонсона-Лінденштрауса: якщо точки у векторному підпросторі спроектовані на випадково відібраний підпростір відповідної високої розмірності, то відстані між точками зберігаються з певною апроксимацією.

Singular Value Decomposition

Сингулярне розкладання або Singular Value Decomposition (SVD) являє собою декомпозицію матриці чисел з плаваючою точкою для зведення останньої до канонічного виду. Воно використовується при вирішенні найрізноманітніших завдань – від наближення методом найменших квадратів і рішення систем рівнянь до стиснення зображень. При цьому використовуються різні властивості сингулярного розкладання, наприклад, здатність показувати ранг матриці, наближати матриці даного рангу. SVD дозволяє обчислювати зворотні і псевдообернені матриці великого розміру, що робить його корисним інструментом при вирішенні задач машинного навчання.

Для будь-якої матриці чисел з плаваючою точкою A розміру $n \times n$ існує дві ортогональні $(n \times n)$-матриці U та V такі, що $U^T AV$ – діагональна матриця Λ. Матриці U та V обираються так, щоб діагональні елементи матриці Λ мали вигляд:

$$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_r \geq \lambda_{r+1} = \cdots = \lambda_n = 0,$$

де r – ранг матриці A.

Індекс r елемента λ_r є фактичною розмірністю власного простору матриці A. Стовпчики матриць U та V відповідно називають лівими та правими сингулярними векторами, а значення на діагоналі матриці Λ називають сингулярними числами.

1.5 Алгоритми класифікації векторних даних

Вибір алгоритму класифікації є вирішальним етапом під час проведення задачі класифікації текстових даних. Для порівняння доцільно взяти просту лінійну модель, яку будує логістична регресія, алгоритм Random Forest, здатний побудувати більш складну та якісну модель за рахунок композиції декількох дерев рішень, та Gradient Boosting Tree, який вважають одним з найбільш ефективних алгоритмів побудови ансамблів, а бібліотеку XGBoost, що його реалізує, фактично сучасним стандартом для розв'язання задач машинного навчання. Розглянемо ключові моменти кожного з цих алгоритмів нижче.

Логістична регресія

Побудова моделі логістичної регресії виникає з бажання моделювати ймовірності належності об'єктів до K класів за допомогою лінійних функцій, одночасно гарантуючи, що вони в сумі складатимуть 1 і залишатимуться в діапазоні [0, 1]. Модель задається формулами:

$$\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x$$
$$\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x$$
$$\vdots$$
$$\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x.$$

Ймовірності належності об'єктів до класів визначаються за допомогою логістичної функції, коефіцієнти якої в свою чергу знаходять на основі методу максимальної правдоподібності.

Для поліпшення узагальнюючої здатності вихідної моделі, тобто зменшення ефекту перенавчання, на практиці часто розглядається логістична регресія з регуляризацією.

Регуляризація полягає в тому, що вектор параметрів θ розглядається як випадковий вектор з деякою заданою апріорної щільністю розподілу $p(\theta)$. Для навчання моделі замість методу максимальної правдоподібності при цьому

використовується метод максимізації апостеріорної оцінки, тобто шукаються параметри θ, що максимізують величину:

$$\prod_{i=1}^{m} P\{y^{(i)}|x^{(i)}, \theta\} \cdot p(\theta)$$

У якості апріорного розподілу часто виступає багатовимірний нормальний розподіл $\mathbb{N}(0, \sigma^2 I)$ з нульовим середнім і матрицею коваріації $\sigma^2 I$, відповідне апріорному переконанню про те, що всі коефіцієнти регресії повинні бути невеликими числами, ідеально – дуже малозначущі коефіцієнти повинні бути нулями. Підставивши щільність цього апріорного розподілу в формулу вище, і застосувавши логарифм, отримаємо наступну оптимізаційну задачу:

$$\sum_{i=1}^{m} \log P\{y^{(i)}|x^{(i)}, \theta\} - \lambda||\theta||^2 \rightarrow max$$

де $\lambda = const/\sigma^2$ – параметр регулярізації. Цей метод відомий як L_2-регуляризована логістична регресія, так як в цільову функцію входить L_2-норма вектора параметрів для регуляризації.

Random Forest

Однією з простих та одночасно ефективних технік побудови ансамблів моделей є техніка під назвою Bagging, окремим випадком реалізації якої є алгоритм Random Forest.

Розглянемо спочатку загальну ідею техніки Bagging. Нехай мається начальна вибірка X . Згенеруємо з неї підвибірки $X_1, \dots, X_M$, такі, що не перетинаються між собою. На кожній вибірці застосуємо базовий алгоритм класифікації $a_i(x)$. Підсумковий класифікатор буде усереднювати відповіді усіх цих алгоритмів (у випадку класифікації це буде відповідати голосуванню): $a(x) = \frac{1}{M}\sum_{i=1}^{M} a_i(x)$.

Алгоритм Random Forest у якості базових алгоритмів використовує дерева, що будуються з використанням алгоритму CART та методу

випадкових підпросторів, застосування якого забезпечує відсутність кореляції між деревами ансамблю.

Алгоритм побудови випадкового лісу, що складається з N дерев, виглядає наступним чином:

Для кожного $n = 1, \dots, N$:

1. Згенерувати підвибірку X_n;
2. Побудувати дерево рішень b_n за вибіркою X_n:
   a. по заданому критерію обирається найкраща ознака, так робимо до моменту, коли всі ознаки будуть використані
   b. дерево будується, поки в кожному листі не менше ніж n_{min} об'єктів або поки не буде досягнута певна максимальна висота дерева
   c. при кожному розбитті спочатку обирається m випадкових ознак із k початкових і оптимальне розбиття вибірки шукається лише серед них.

Підсумковий класифікатор має вигляд: $a(x) = \frac{1}{N}\sum_{i=1}^{N} b_i(x)$. Рекомендується у задачах класифікації обирати $m = \sqrt{k}$, де k – кількість ознак [11].

Gradient Boosting Machine та XGBoost

Розглянемо тепер більш вдосконалений алгоритм побудови ансамблів – алгоритм Gradient Boosting Tree. Він вважається одним з найбільш ефективних алгоритмів у задачах класифікації та регресії, тож його застосування у задачі класифікації текстового корпусу має логічний сенс.

Ідея техніки Boosting, що є альтернативою техніці Bagging, почалася з питання про те, чи можна з великої кількості відносно слабких і простих моделей отримати одну сильну. Під слабкими моделями мається на увазі не просто невеликі і прості моделі на кшталт дерев рішень на противагу більш "сильним" моделям, наприклад, нейронним мережам. У даному випадку слабкі моделі – це довільні алгоритми машинного навчання, точність яких може бути лише трохи вище випадкового вгадування.

Позитивна математична відповідь на це питання знайшлася досить швидко і сама по собі була важливим теоретичним результатом. Її загальний підхід полягав в жадібній побудові лінійної комбінації простих моделей (базових алгоритмів) шляхом повторного зважування вхідних даних. Кожна наступна модель (як правило, дерево рішень) будувалася таким чином, щоб надавати більшу вагу і перевагу раніше некоректно передбаченим спостереженнями.

Саме цю загальну ідею реалізує алгоритм Gradient Boosting Machine (GBM), запропонований Фрідманом. Розглянемо його більш детально.

На вхід алгоритму необхідно подати:

- набір даних $\{(x_i, y_i)\}_{i=1,\dots,n}$;
- кількість ітерацій M
- вибір функції втрат $L(y, f)$ і її градієнт
- вибір групи функцій базових алгоритмів $h(x, \theta)$ з процедурою їх навчання
- додаткові гіперпараметри $h(x, \theta)$, наприклад, глибина дерева у дереві рішень

У якості початкового наближення $f_0(x)$, як правило, використовують просто константне значення γ. Його, а також оптимальний коефіцієнт ρ знаходять бінарним пошуком, або іншим line search алгоритмом щодо вихідної функції втрат (а не градієнта).

Сам GBM складається з таких кроків:

1. Ініціалізувати GBM константним значенням $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma,\ \gamma \in R$

$$\hat{f}_0 = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. Для кожної ітерації $t = 1, \dots, M$ повторювати:
   a. Обчислити псевдо-залишки r_t

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, for\ i = 1, \dots, n$$

   b. Побудувати базовий алгоритм $h_t(x)$ як регресію на псевдо-залишках $\{(x_i, r_{it})\}_{i=1,\dots,n}$;

c. Знайти оптимальний коефіцієнт ρ_t при $h_t(x)$ відносно функції втрат $\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x) + \rho \cdot h(x_i, \theta))$

d. Зберегти $\hat{f}_t(x) = \rho_t \cdot h_t(x)$

e. Оновити поточне наближення $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=1}^{t} \hat{f}_i(x)$$

3. Скомпонувати підсумкову GBM модель $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=1}^{M} \hat{f}_i(x)$$

4. Використовувати модель для класифікації або регресії. [9]

Розглядаючи GBM, не можна не сказати про XGBoost. XGBoost – це популярна бібліотека машинного навчання, яка містить високоефективну реалізацію Gradient Boosting Machine на декількох мовах програмування, та має такі переваги як оптимізація пам’яті, робота у декілька потоків, швидкий алгоритм відсічення гілок дерева та ін.

1.6 Метрики оцінки якості класифікації

Аналіз ефективності того чи іншого алгоритму машинного навчання, обов’язково повинен проводитись на підставі певних об’єктивних метрик. В задачі класифікації такими метриками можуть бути наступні: Accuracy, Log Loss, Precision, Recall, F1-Score та AUC. Були обрані саме такі метрики оцінки якості алгоритму, оскільки кожна з них по різному характеризує помилки алгоритму і разом вони доповнюють одна одну, створюючи повну картину про отримані результати. Розглянемо формули, за якими розраховуються дані метрики (табл. 1.1).

Таблиця 1.1

Метрики оцінки якості класифікації

Метрика	***Формула***	***Інтервал***	***Пояснення***
Accuracy	$\frac{N_{true\ positive} + N_{true\ negative}}{N_{all}}$	[0%; 100%] або [0; 1]	$\boldsymbol{N_{tp}}$ – кількість об'єктів 1-го класу, що були передбачені як об'єкти 1-го класу $\boldsymbol{N_{tn}}$ - кількість об'єктів не 1-го класу, що були передбачені як об'єкти не 1-го класу $\boldsymbol{N_{all}}$ – кількість об'єктів у вибірці $\boldsymbol{N_{fp}}$ - кількість об'єктів не 1-го класу, що були передбачені як об'єкти 1-го класу $\boldsymbol{N_{fn}}$ - кількість об'єктів 1-го класу, що були передбачені як об'єкти не 1-го класу
Multiclass Log Loss	$-\sum_{c=1}^{M} y_{0,c}\log\left(p_{o,c}\right)$	[0; ∞]	M – кількість класів $y_{o,c}$ – бінарний індикатор (0 чи 1), що дорівнює 1 – якщо клас c був правильно передбачений для об'єкту o і 0 – навпаки $p_{o,c}$ – ймовірність належності об'єкту o до класу c
Precision	$\frac{N_{true\ positive}}{N_{true\ positive} + N_{false\ positive}}$	[0%; 100%] або [0; 1]	Наведені у рядку Accuracy
Recall	$\frac{N_{true\ positive}}{N_{true\ positive} + N_{false\ negative}}$	[0%; 100%] або [0; 1]	Наведені у рядку Accuracy
F1-Score	$\frac{2 * Precision * Recall}{Precision + Recall}$	[0%; 100%] або [0; 1]	Наведені у рядку Accuracy

2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ «BIOTEXTIFY»

2.1 Функціональні можливості програми

В ході виконання роботи був розроблений програмний продукт під назвою «biotextify». Програма застосовується для класифікації генетичних мутацій на основі їх текстового опису.

Програмне забезпечення «biotextify» містить наступний функціонал:

– перетворює вхідні текстові дані у навчальну вибірку документів (текстовий корпус), таким чином, що текст перетворюється у матрицю документів та їх ідентифікаторів;

− видаляє зайві символи (шуми) з тексту, які можуть негативно впливати на якість класифікації;

− нормалізує текстовий корпус за допомогою регулярних виразів та морфологічних технік, таких як стемінг та лематизація;

− об’єднує слова з одного контексту у словосполучення;

− перевіряє, чи рівномірно представлені класи в навчальній вибірці і, якщо необхідно, балансує вибірку відносно класів;

− відображає корпус у векторний простір чотирма методами: Count Bag of Words, TF-IDF, Word2Vec, Doc2Vec;

− зменшує простір ознак методами TruncatedSVD та Random Projection, додає нові ознаки до об’єктів;

− проводить класифікацію, використовуючи алгоритми Random Forest, XGBoost та Logistic Regression;

− порівнює результати класифікації за такими метриками як Accuracy, Log Loss, Precision, Recall, F1-Score, AUC, Confusion Matrix;

Програма обробляє вхідні файли форматів CSV та TXT.

Програма витримує навантаження в 450 тис. слів у словнику та запис у файл розміром до 1.5 Гб.

2.2 Організація обчислювального процесу

Програмне забезпечення «biotextify» написано на мові програмування Python3.6 у середовищі розробки PyCharm.

Бібліотеки, використані під час розробки «biotextify»:

- *pandas* - перетворення навчальної вибірки у DataFrame
- *nltk* - проведення стеммінгу, лематизації та видалення стоп-слів
- *gensim* - побудова векторної моделі, зменшення кількості ознак словника, об'єднання слів у словосполучення
- *sklearn* - побудова векторної моделі, алгоритми класифікації та зменшення простору ознак, обчислення метрик якості класифікації
- *xgboost* – алгоритм класифікації

Структурна схема програмного продукту «biotextify» зображена у на рисунку нижче (рис. 2.1).

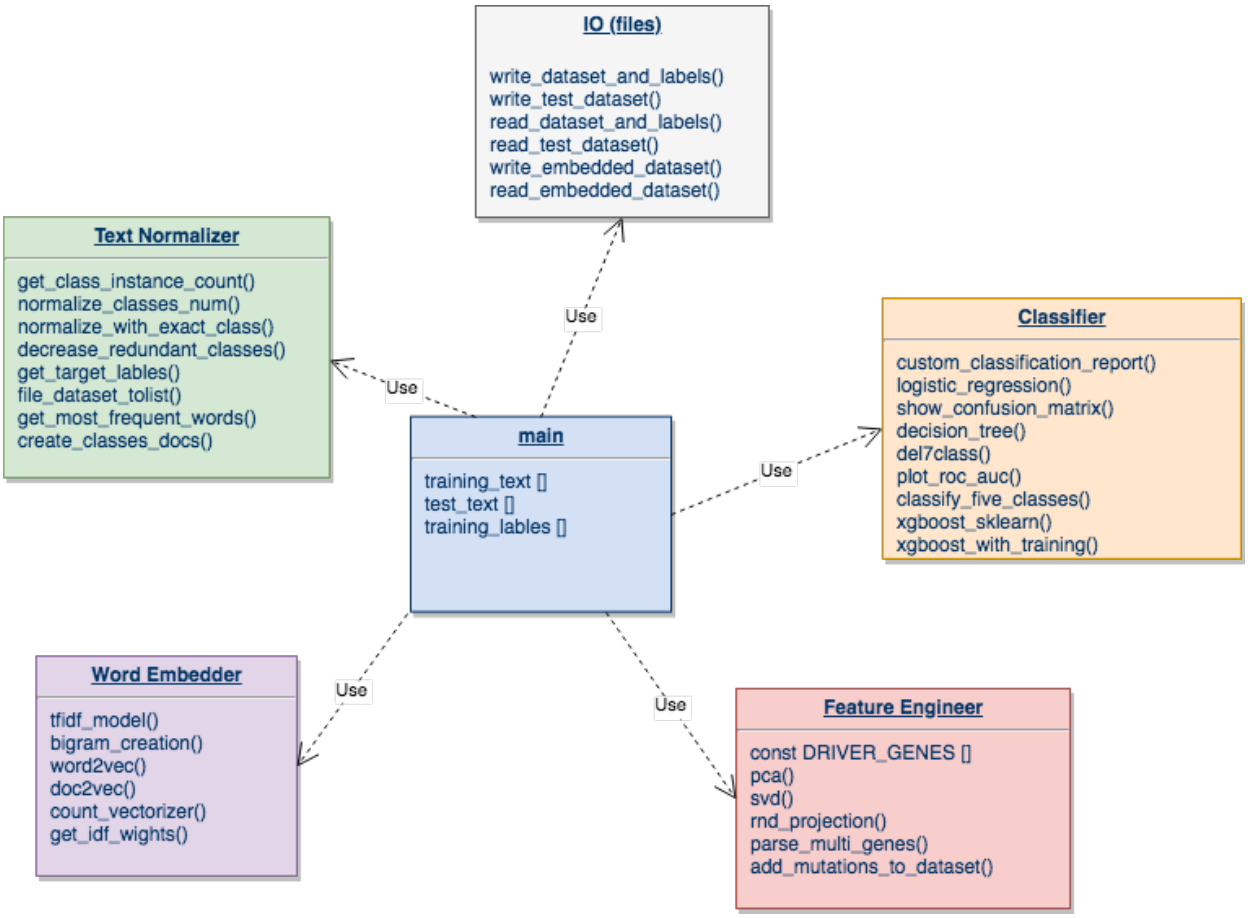

Рис. 2.1 Структурна схема програми «biotextify»

Як можна побачити зі схеми, програма складається з п’яти модулів і кожен з них взаємодіє лише з вхідною точкою програми – main. Таке рішення було прийняте для того, щоб виділити різні етапи вирішення задачі у різні файли та модулі.

Має зміст звернути увагу на модуль *_io,* завдяки якому була проведена декомпозиція задачі і результати кожного етапу декомпозиції записані у текстові файли за допомогою методів саме модуля *_io*. Такий підхід до вирішення задачі скоротив час тестування програми більше ніж у 10 разів.

Кожний модуль відповідає за певний етап вирішення задачі, тож опишемо задачі кожного модуля:

– Text Normalizer – містить методи нормалізації текстового корпусу, балансування початкових даних

– Word Embedder – містить методи перенесення слів у векторний простір

– IO – містить методи запису і читання навчальної та текстової вибірок в і з .txt файлів у спеціальному зручному форматі

– Feature Engineer – містить методи зменшення простору ознак та додання нових ознак до об’єктів

– Classifier – містить ряд методів класифікації, обчислення метрик якості класифікаторів та побудову графічних метрик.

3. РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНИХ ЕКСПЕРИМЕНТІВ

З метою вирішення задачі класифікації генетичних мутацій на основі їх текстових описів було проведено ряд експериментів за допомогою програмного забезпечення «biotextify». Для проведення експериментів використовувалась навчальна і тестова вибірки розмірами 1500 та 300 документів відповідно. Кожний з документів у початковому вигляді містив приблизно 2,5 тис. слів. На вхід класифікатора були подані вектори розмірністю 700, в подальшому скороченні за одним з алгоритмів зменшення розмірності простору ознак.

Для вирішення задачі було проведено наступні експерименти. Спочатку, було проведено тестування базової комбінації алгоритмів векторизації, класифікації та отримання початкових результатів на незбалансованому наборі даних. Після цього було проведено балансування даних та виконане тестування алгоритмів перенесення текстового корпусу у векторний простір та їх вплив на якість класифікації алгоритмом Logistic Regression. Наступним етапом було порівняння алгоритмів зменшення розмірності простору ознак та чи впливає додавання нових ознак на якість класифікації алгоритмом Logistic Regression. Нарешті, проводилося безпосереднє тестування та порівняння алгоритмів класифікації, беручи до уваги результати попередніх експериментів.

Додатково, була проведена інтерпретація результатів класифікації шляхом аналізу коефіцієнтів моделі Logistic Regression.

Розглянемо проведені експерименти більш детально.

3.1 Проведення базової класифікації на незбалансованих даних

Під час проведення аналізу розподілу документів початкової навчальної вибірки було помічено, що даний розподіл достатньо сильно відрізняється від рівномірного. Таким чином, було зроблено висновок про незбалансованість вхідних даних (рис 3.1).

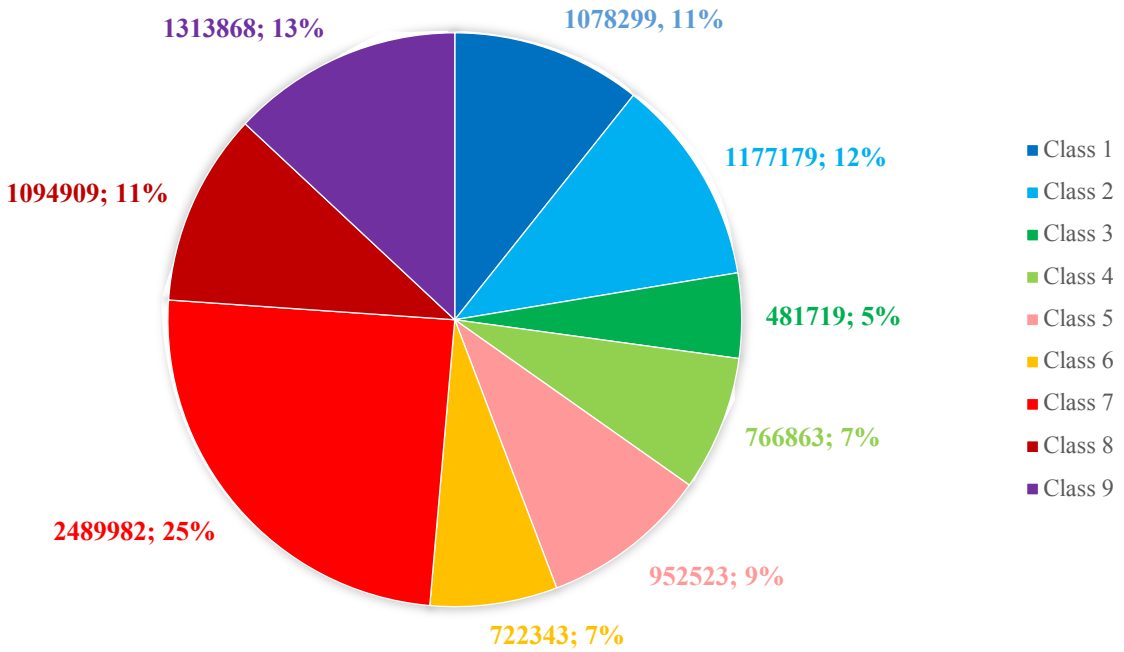

Рис 3.1 Абсолютна та відсоткова кількість слів у класах навчальної вибірки

Відомо, що у деяких алгоритмів класифікації немає залежності від розподілу вхідних даних. Таким алгоритмом є, наприклад, алгоритм Naïve Bayes. Було вирішено провести тестування з наступною комбінацією алгоритмів: TF-IDF для перенесення слів у векторний простір та алгоритми Naïve Bayes і Logistic Regression для безпосередньої класифікації векторів. Оскільки ця комбінація алгоритмів вважалася базовою, оцінка якості була проведена лише за метрикою Accuracy, яка склала 41% для алгоритму Naïve Bayes та 48% для Logistic Regression. Після цього було прийняте рішення про необхідність балансування вхідних даних методами Random Undersampling та Random Oversampling для приведення розподілу даних до рівномірного. Після даної процедури якість класифікації за алгоритмом Logistic Regression склала 65%, тож у подальших експериментах використовувався саме збалансований варіант даних. Алгоритм Naïve Bayes було вирішено не використовувати у подальших експериментах через складну структуру вхідної вибірки.

3.2 Порівняння алгоритмів перенесення текстового корпусу у векторний простір

Наступним питанням, що постає у вирішенні задачі класифікації є вибір алгоритму перенесення слів у векторний простір. Як вже відомо, існує декілька груп алгоритмів, які використовують принципово різні підходи до векторизації слів.

Для порівняння було обрано наступні алгоритми векторизації тексту: CBOW, TF-IDF, Word2Vec, Doc2Vec. Вибір цих алгоритмів пов'язаний з тим, що кожен з них є достатньо поширеним при роботі з текстом, одночасно маючи унікальну схему відображення слів або документів у векторний простір. Вищезазначені алгоритми належать до двох груп методів векторизації: BagOfWords та Word Embeddings.

Нижче наведено результати класифікації текстового корпусу в залежності від вибору алгоритму перенесення слів у векторний простір, кількості елементів у словнику унікальних слів та гіперпараметрів алгоритмів. В якості класифікатору використано Logistic Regression.

Таблиця 3.1

Результати класифікації у разі використання різних алгоритмів векторизації слів

№ тесту	Алгоритм векторизації	Параметри алгоритму	MultiClass Loss
1	CBOW	розмір словника = 2000	11.58
2	TF-IDF	розмір словника = 2000	2.61
3	Word2Vec	мінімальна частота слова = 100, вікно = 5, розмір вектора слова = 1	4.97
4	Word2Vec	максимальна довжина словника = 30000, вікно = 10, розмір вектора слова = 300	2.20

Продовження Таблиці 3.1

5	Doc2Vec	мінімальна частота слова = 200, вікно = 5, розмір вектора речення = 500	3.74
6	CBOW	розмір словника = 190000	9.75
7	TF-IDF	розмір словника = 190000	2.30

Результати показують, що якість класифікації має сильну залежність від гіперпараметрів алгоритмів. Метод CBOW має найнижчий показник, що пояснюється його простотою та наявністю великого розкиду поміж координатами векторів кожного документа. Метод Doc2Vec показав нижчу якість, ніж TF-IDF, у більшій мірі через те, що біомедичний корпус наукових текстів, на якому було проведено тестування, має достатньо складну структуру та низьку ймовірність повтору контекстів, що є вирішальним для даного алгоритму.

Було визначено, що найбільш ефективними алгоритмами векторизації розглянутого корпусу є TF-IDF та Word2Vec (при правильному заданні гіперпараметрів алгоритму). Оскільки метод Word2Vec вважається більш ефективним та новішим, було вирішено використовувати саме його у подальших експериментах. Результати поданих вище експериментів можуть бути корисні також під час обробки інших корпусів наукової біомедичної літератури, оскільки розподіл слів та структура речень є схожими з розглянутою навчальною вибіркою.

3.3 Тестування впливу множини ознак на якість класифікації

Наступний етап проведення експериментів був спрямований на виявлення залежності результатів класифікації від набору ознак, які подаються на вхід класифікатора. Алгоритм Logistic Regression було протестовано на наборі даних з розмірністю 190000, 10000, 2000 та 1000. Зменшення простору ознак відбувалося за допомогою алгоритмів Random

Projection та TruncatedSVD. Відхилення результатів було дуже незначним, приблизно 0.05 у метриці Multi Class Log Loss. Це означає, що з 360-ти тисяч унікальних слів (розмір словника після нормалізації тексту) лише 1 тисяча слів насправді є інформативною та впливає на результати класифікації.

На даному етапі проведення експериментів з'явилася ідея покращення результатів шляхом додання нових ознак. Перед цим модель включала в себе лише показники, що являли собою одновимірні вектори з простору Word2Vec.

Після проведення дослідження про типи генів та генетичних мутацій і їх розподіл у випадках появи певних ракових захворювань, було вирішено використати додаткову інформацію про властивості генів. Графік з Додатку А відображає таку інформацію про групи генів. Деякі гени мають драйверні мутації у всіх групах ракових хвороб, тому на графіку вони винесені у групу Multiclass Driver Mutations (верхній правий кут). Слід нагадати, що кожний об'єкт навчальної вибірки має атрибути ген (Gene) та Мутація (Mutation). Тобто для кожного об'єкту можна визначити, чи належить його ген до групи Multiclass Driver Mutations чи ні. Ось звідки і з'являються значення 0 та 1 для One-Hot-Encoding ознаки, яку було додано до моделі.

Вищенаведена техніка допомогла зменшити Multi Class Log Loss на 0.1, що лише підкреслює важливість вибору ознак під час вирішення задачі класифікації.

3.4 Порівняння результатів роботи алгоритмів класифікації

Було проведено оцінювання якості класифікації на двох вибірках: начальній та тестовій. Були обрані обидві вибірки, щоб подивитися на різницю результатів класифікації на навчальній вибірці (на якій будувався класифікатор) та тестовій вибірці (та, що не використовувалась під час побудови моделі) і переконатися у відсутності перенавчання.

Результати тестування алгоритмів Logistic Regression, Random Forest та GBM (реалізація XGBoost) на навчальній та тестовій вибірці можна побачити

у Додатку В. Оскільки майже усі метрики розраховуються окремо для кожного класу, то можна порівняти якість класифікації для кожного із них.

З таблиць Б.1, Б.2 видно загальну тенденцію на кращу якість класифікації на навчальній вибірці, ніж на тестовій. Тобто міститься певний ефект перенавчання, присутній і після налаштування гіперпараметрів алгоритмів. Також видно, що алгоритми Random Forest та XGBoost показали кращі результати, ніж Logistic Regression, оскільки вони здатні будувати нелінійні роздільні поверхні. Порівнюючи між собою результати XGBoost та Random Forest можна сказати, що вони дуже схожі, але модель XGBoost має більший потенціал покращення результатів при подальших налаштування гіперпараметрів та трохи кращі результати у певних метриках.

Іноді буває корисним переглянути результати не лише у табличному форматі, а й за допомогою «графічних» метрик, таких як Confusion Matrix (матриця відповідності справжніх та передбачуваних моделлю класів) та ROC Curve (Receiver Operating Characteristic). Розглянемо результати даних метрик окремо для кожного алгоритму.

З наведених у Додатку В графічних результатів можна побачити, що трьома алгоритмами класи 8 та 9 розпізнаються зі 100% точністю, найгірший показник розпізнавання має клас 7, оскільки після балансування вибірки за кількістю слів в ньому залишилося менше текстових документів ніж в інших класах.

Тож на даному етапі було зроблено висновок про найвищу ефективність алгоритму GBM (XGBoost). Тепер можна перейти до інтерпретації результатів класифікації.

3.5 Інтерпретація результатів моделі Logistic Regression

Відомо, що алгоритми машинного навчання іноді називають «чорною скринею», оскільки невідомо, чому саме були отримані певні результати у певній задачі. Деякі алгоритми мають властивість важкої інтерпретації,

наприклад нейронні мережі та ансамблі і композиції моделей. Але деякі лінійні і нелінійні моделі мають навпаки властивість легкої інтерпретації. Такими моделями є, наприклад, дерева рішень та логістична регресія. Завдяки властивості легкої інтерпретації будь-яка людина, навіть незнайома з машинним навчанням, зможе отримати користь від результатів роботи моделі.

Проблема класифікації генетичних мутацій, що вирішувалася в рамках даної роботи, особливо потребує чіткої інтерпретації, оскільки якість класифікації моделей можна оцінювати не лише за метричними даними, а й за логічною оцінкою спеціалістів даної предметної області, тобто лікарів.

Однією з моделей, що використовувалися для задачі класифікації біомедичних даних, є логістична регресія (Logistic Regression). Завдяки своїй природі як моделі регресії, вона проставляє коефіцієнти кожній з ознак. А ознака в свою чергу – це векторне представлення унікального слова зі словника методу Word2Vec. Тобто якщо для кожного слова словника створити відображення типу {слово:ознака} одночасно виписавши коефіцієнти моделі Logistic Regression, то можна отримати співвідношення коефіцієнту (ваги) певного слова для певного класу і представити ймовірність належності поточного об’єкта вибірки X до кожного з класів як:

$$Pr(G = class_i \mid X = x) = \frac{\exp(g_i)}{1 + \sum_{l=1}^{K-1} \exp(g_l)}, for\ i = 1, \dots, K$$

де $G -$ клас, який ми передбачаємо для поточного об’єкта вибірки x; $Pr(G = class_i) -$ ймовірність того, що передбачуваний клас – це клас $class_i$; для якого лінійний вираз $g_i = \sum_{j=1}^{N} coef_{ij} * word_j$ – це сума добутків унікальних слів на їх ваги; K – кількість класів; N – кількість слів у словнику унікальних слів алгоритму перенесення слів у векторний простір, $coef_{ij}$ – коефіцієнт моделі Logistic Regression для j-го слова i-го класу.

Не має необхідності відображати усі N коефіцієнтів і слів, а відобразити певну підмножину максимальних коефіцієнтів, наприклад, п‘ять. Дану підмножину максимальних коефіцієнтів можна отримати за допомогою методів argpartition та argsort з бібліотеки numpy.

Тож було отримано наступні вирази g_i:

$\boldsymbol{g_1} = 7.25 * \text{known\_deleterious} + 5.45 * p53 + 4.55 * tp53 + 4.47 * tumor\ supressior + 3.89 * methylation$

$\boldsymbol{g_2} = 5.29 * \text{brct\_domain} + 5.08 * \text{neutral} + 4.68 * aberrat\ splicing + 4.64 * plasmid + 4.45 * transcriptional\ activation$

$\boldsymbol{g_3} = 5.06 * \text{akt1} + 4.89 * \text{cloned} + 4.19 * survival + 3.90 * brkt\_domain + 3.72 * driver$

$\boldsymbol{g_4} = 6.73 * \text{cycle arrest} + 6.19 * \text{p16ink4a} + 5.52 * tumor\ supressor + 5.09 * p53 + 4.39 * cdkn2a$

$\boldsymbol{g_5} = 5.38 * \text{fgfr2} + 4.88 * \text{fgfr1} + 3.48 * control + 3.25 * compared\ wild + 3.07 * potential$

$\boldsymbol{g_6} = 5.06 * \text{akt1} + 4.89 * \text{cloned} + 4.19 * survival + 3.90 * brkt\_domain + 3.72 * potential$

$\boldsymbol{g_7} = 4.29 * \text{signal transduction} + 4.17 * \text{oncogenic} + 3.75 * glioma + 3.75 * pi3k + 3.36 * pathogenic$

$\boldsymbol{g_8} = 9.08 * \text{idh1\_r132h} + 7.93 * glioma + 7.43 * \text{phosphatase} + 6.72 * sarcoma + 5.67 * histone$

$\boldsymbol{g_9} = 8.39 * \text{myelodysplastic\_syndrome} + 8.07 * \text{splicing\_abnormality} + 7.35 * \text{differential\_splicing} + 5.50 * \text{intron\_retention} + 5.00 * \text{r132h}$

Звичайно, множини з п’яти коефіцієнтів недостатньо для розуміння властивостей класу, але даний приклад показує, що інтерпретація коефіцієнтів моделі Logistic Regression дійсно вказує на те, чому класифікатор відніс певний документ до певного класу, і розширюючи кількість коефіцієнтів можна домогтися більш точної інтерпретації.

ВИСНОВКИ

У роботі було розглянуто вирішення задачі класифікації генетичних мутації на основі їх текстового опису за допомогою алгоритмів машинного навчання. Для вирішення задачі було створено програмне забезпечення “biotextify”, головною задачею якого є якісна класифікація біомедицинського тексту, пов’язаного з генетичними мутаціями та їх клінічними описами. Технологією розробки програми була вибрана мова Python, оскільки вона має багато додаткових бібліотек для зручної роботи з текстовими даними.

Під час роботи були досліджені та використані різноманітні техніки обробки природньої мови, такі як стеммінг, лематизація, які допомогли зменшити кількість унікальних N-gram у словнику. Для відображення текстового корпусу у векторний простір було використано моделі TF-IDF, Count Bag of Words, Word2Vec, Doc2Vec, Bi-grams. Для вирішення задачі класифікації використовувались алгоритми Logistic Regression, Random Forest та XGboost. Найвища якість класифікації була досягнута комбінацією алгоритмів Word2Vec, що дозволяє розставити семантично близькі слова близько один до одного у векторному просторі, та XGBoost, що визнаний одним з найпотужніших ансамблевих моделей для класифікації.

Особливо цікавою частиною роботи було проведення інтерпретації моделі Logistic Regression, після якої було виведено «рівняння» для еталонних документів кожного класу.

Аналіз алгоритмів та методів роботи з текстовим корпусом наукової біомедичної літератури, проведений в даній роботі, може використовуватися не лише для безпосередньої класифікації таких вхідних даних, а й для інших пов’язаних задач, наприклад, задачі генерації відповіді на питання, кластеризації тексту, узагальнення тексту, скорочення простору слів у тексті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hastie T., The Elements Of Statistical Learning, Data Mining, Inference, and Prediction / [Електронний ресурс] Hastie T., Tibshirani R., Friedman J.// Springer Science+Business Media.- 2017. – с. 119 – 127. Режим доступу: https://web.stanford.edu/~hastie/Papers/ESLII.pdf
2. Пархоменко П. А. Обзор и экспериментальное сравнение методов кластеризации текстов / [Електронний ресурс]. Пархоменко П. А,. Григорьев А.А.,. Астраханцев Н.А. // Труды ИСП РАН. - 2017. - том 29. - выпуск 2. – с. 161–200, Режим доступу. https://doi.org/10.15514/ISPRAS-2017-29(2)-6
3. Чудесный мир Word Embeddings: какие они бывают и зачем нужны? [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: https://habrahabr.ru/company/ods/blog/329410/
4. Учим машину разбираться в генах человека [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: https://habrahabr.ru/company/microsoft/blog/343604/
5. Mikolov T. Efficient Estimation of Word Representations in Vector Space/[Електронний ресурс], Mikolov T, Chen K., Corrado G., Dean J.// 2014. – с. 12. Електронні дані. - Режим доступу: https://arxiv.org/pdf/1301.3781.pdf
6. Оценка классификатора (точность, полнота, F-мера) [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html
7. Personalized Medicine: Redefining Cancer Treatment [Інтернет-портал] – Електронні дані. - Режим доступу: https://www.kaggle.com/c/msk-redefining-cancer-treatment/data
8. Bingham E. Random projection in dimensionality reduction: Applications to image and text data/[Електронний ресурс], Bingham E., Manilla H.// 2015. -

Електронні дані. - Режим доступу: http://www.ime.unicamp.br/~wanderson/Artigos/randon_projection_kdd.pdf

9. Открытый курс машинного обучения. Тема 10. Градиентный бустинг. [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: https://habr.com/company/ods/blog/327250/
10. Сингулярное разложение в алгоритмах машинного обучения. [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=SVD
11. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: https://habr.com/company/ods/blog/324402/
12. Дьяконов А. Градиентный бустинг/[Електронний ресурс], Дьяконов А// 2017. – с.1-20. - Електронні дані. - Режим доступу: https://alexanderdyakonov.files.wordpress.com/2017book_boosting_pdf.pdf
13. Practical guide to deal with Imbalanced Classification Problems in R. [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу:https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/
14. Bailey M. Comprehensive Charachterization of Cancer Driver Genes and Mutations/[Електронний ресурс], Bailey M., Tokheim K., Porta-Pardo E., Mills G., Karchin R., Ding L.// 2018. - с.32. - Електронні дані. - Режим доступу: https://www.cell.com/cell/pdf/S0092-8674(18)30237-X.pdf
15. United States National Library of Medicine/Genetics Home Reference [Електронний ресурс] : [Інтернет-портал] – Електронні дані. – Режим доступу: https://ghr.nlm.nih.gov/gene

ДОДАТОК А

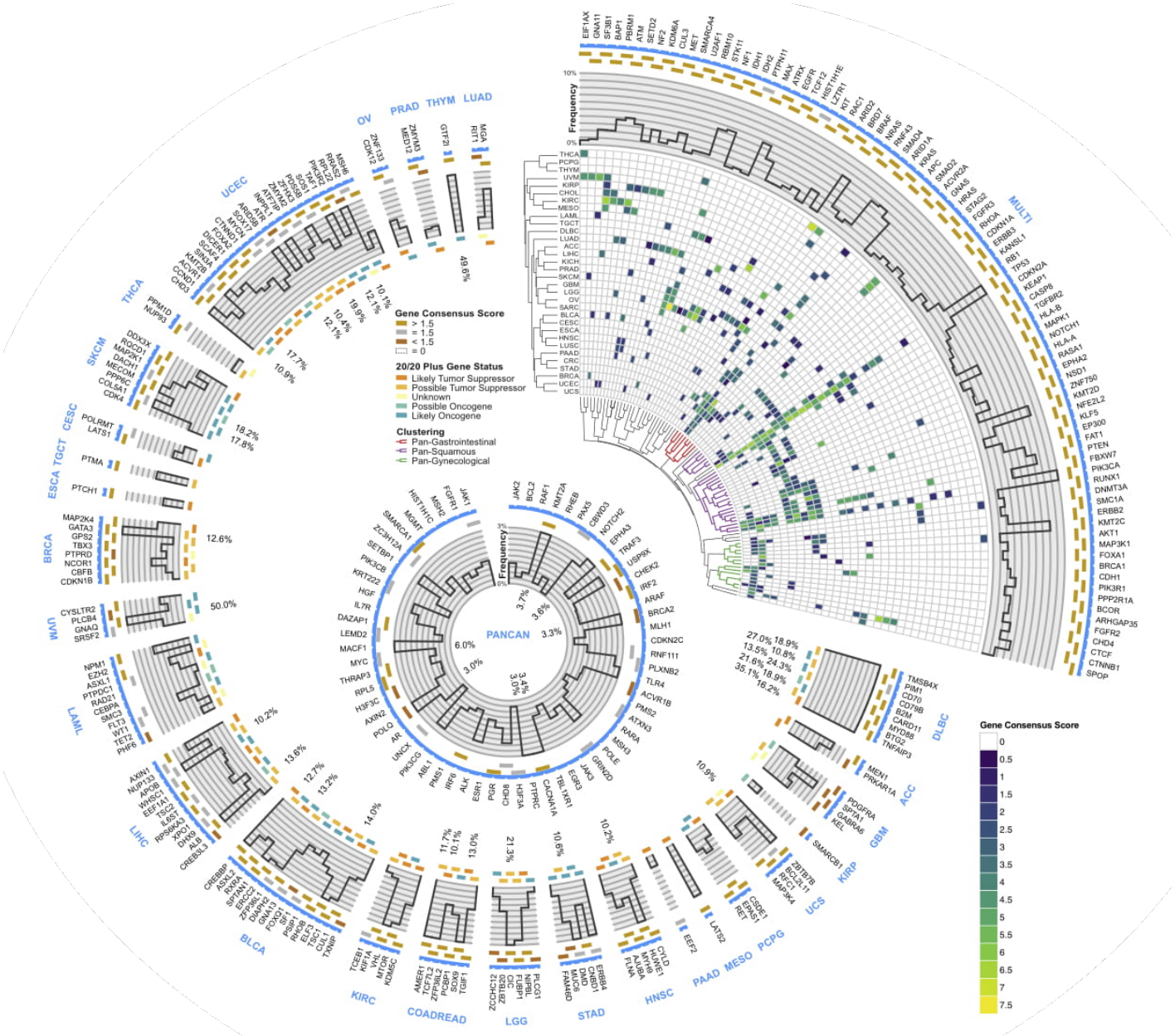

Рис. А.1 Графічне відображення драйверних генів для різних груп ракових захворювань [14]

ДОДАТОК Б

Таблиця Б.1

Результати оцінки якості класифікації на основі навчальної вибірки

Алгоритм	Гіперпараметри	Accuracy	Log Loss	Precision	Recall	F1-Score	AUC
XGBoost	eta = 0.01 max_depth = 10 gamma = 5 colsample_bytree = 0.4 num_rounds = 500	95 %	0.32	class_1: 0.91 class_2: 0.97 class_3: 0.93 class_4: 0.94 class_5: 0.87 class_6: 0.91 class_7: 1.00 class_8: 1.00 class_9: 1.00 *Average: 0.95*	class_1: 0.89 class_2: 1.00 class_3: 0.97 class_4: 0.91 class_5: 0.85 class_6: 1.00 class_7: 0.95 class_8: 1.00 class_9: 0.97 *Average: 0.95*	class_1: 0.90 class_2: 0.98 class_3: 0.95 class_4: 0.92 class_5: 0.86 class_6: 0.95 class_7: 0.97 class_8: 1.00 class_9: 1.00 *Average: 0.95*	class_1: 0.99 class_2: 0.99 class_3: 0.99 class_4: 0.99 class_5: 0.98 class_6: 0.99 class_7: 0.99 class_8: 1.00 class_9: 1.00 *Average: 0.99*
Logistic Regression	penalty='l2' C = 25	83 %	0.80	class_1: 0.71 class_2: 0.73 class_3: 0.90 class_4: 0.78 class_5: 0.66 class_6: 0.93 class_7: 0.84 class_8: 0.98 class_9: 1.00 *Average: 0.84*	class_1: 0.69 class_2: 0.95 class_3: 0.90 class_4: 0.83 class_5: 0.77 class_6: 0.67 class_7: 0.53 class_8: 1.00 class_9: 0.97 *Average: 0.83*	class_1: 0.70 class_2: 0.83 class_3: 0.90 class_4: 0.80 class_5: 0.71 class_6: 0.78 class_7: 0.65 class_8: 0.99 class_9: 0.98 *Average: 0.83*	class_1: 0.96 class_2: 0.98 class_3: 0.99 class_4: 0.96 class_5: 0.93 class_6: 0.97 class_7: 0.96 class_8: 1.00 class_9: 1.00 *Average: 0.98*
Random Forest	n_estimators = 500 max_depth = 9 max_features = 15	93 %	0.67	class_1: 0.94 class_2: 0.89 class_3: 0.89 class_4: 0.95 class_5: 0.79 class_6: 0.97 class_7: 1.00 class_8: 1.00 class_9: 1.00 *Average: 0.93*	class_1: 0.92 class_2: 1.00 class_3: 0.93 class_4: 0.91 class_5: 0.86 class_6: 0.85 class_7: 0.90 class_8: 1.00 class_9: 1.00 *Average: 0.93*	class_1: 0.93 class_2: 0.94 class_3: 0.91 class_4: 0.93 class_5: 0.82 class_6: 0.90 class_7: 0.94 class_8: 1.00 class_9: 1.00 *Average: 0.93*	class_1: 0.99 class_2: 0.99 class_3: 0.98 class_4: 0.99 class_5: 0.96 class_6: 0.99 class_7: 0.99 class_8: 1.00 class_9: 1.00 *Average: 0.99*

Таблиця Б.2

Результати оцінки якості класифікації на основі тестової вибірки

Алгоритм	Гіперпараметри	Accuracy	Log Loss	Precision	Recall	F1-Score	AUC
XGBoost	eta = 0.01 max_depth = 10 gamma = 5 colsample_bytree = 0.4 num_rounds = 500	73 %	0.84	class_1: 0.55 class_2: 0.52 class_3: 0.88 class_4: 0.78 class_5: 0.63 class_6: 1.00 class_7: 0.30 class_8: 1.00 class_9: 1.00 *Average: 0.73*	class_1: 0.54 class_2: 0.81 class_3: 0.94 class_4: 0.71 class_5: 0.59 class_6: 0.69 class_7: 0.29 class_8: 1.00 class_9: 1.00 *Average: 0.72*	class_1: 0.54 class_2: 0.63 class_3: 0.95 class_4: 0.68 class_5: 0.60 class_6: 0.78 class_7: 0.15 class_8: 0.12 class_9: 1.00 *Average: 0.72*	class_1: 0.88 class_2: 0.95 class_3: 0.99 class_4: 0.93 class_5: 0.89 class_6: 0.94 class_7: 0.91 class_8: 1.00 class_9: 1.99 *Average: 0.96*
Logistic Regression	penalty='l2' C = 25	83 %	1.01	class_1: 0.45 class_2: 0.57 class_3: 0.82 class_4: 0.42 class_5: 0.67 class_6: 0.92 class_7: 0.31 class_8: 0.86 class_9: 1.00 *Average: 0.70*	class_1: 0.35 class_2: 0.67 class_3: 0.81 class_4: 0.68 class_5: 0.56 class_6: 0.63 class_7: 0.36 class_8: 1.00 class_9: 0.89 *Average: 0.68*	class_1: 0.40 class_2: 0.62 class_3: 0.82 class_4: 0.52 class_5: 0.61 class_6: 0.75 class_7: 0.34 class_8: 0.92 class_9: 0.94 *Average: 0.68*	class_1: 0.89 class_2: 0.94 class_3: 0.97 class_4: 0.89 class_5: 0.87 class_6: 0.89 class_7: 0.91 class_8: 0.99 class_9: 1.00 *Average: 0.94*
Random Forest	n_estimators = 500 max_depth = 9 max_features = 15	74 %	0.95	class_1: 0.50 class_2: 0.56 class_3: 0.85 class_4: 0.45 class_5: 0.72 class_6: 0.96 class_7: 0.75 class_8: 1.00 class_9: 1.00 *Average: 0.76*	class_1: 0.33 class_2: 0.83 class_3: 0.87 class_4: 0.71 class_5: 0.64 class_6: 0.78 class_7: 0.15 class_8: 1.00 class_9: 1.00 *Average: 0.74*	class_1: 0.40 class_2: 0.67 class_3: 0.86 class_4: 0.55 class_5: 0.68 class_6: 0.86 class_7: 0.26 class_8: 1.00 class_9: 1.00 *Average: 0.73*	class_1: 0.90 class_2: 0.95 class_3: 0.99 class_4: 0.92 class_5: 0.90 class_6: 0.95 class_7: 0.89 class_8: 1.00 class_9: 1.00 *Average: 0.96*

ДОДАТОК В

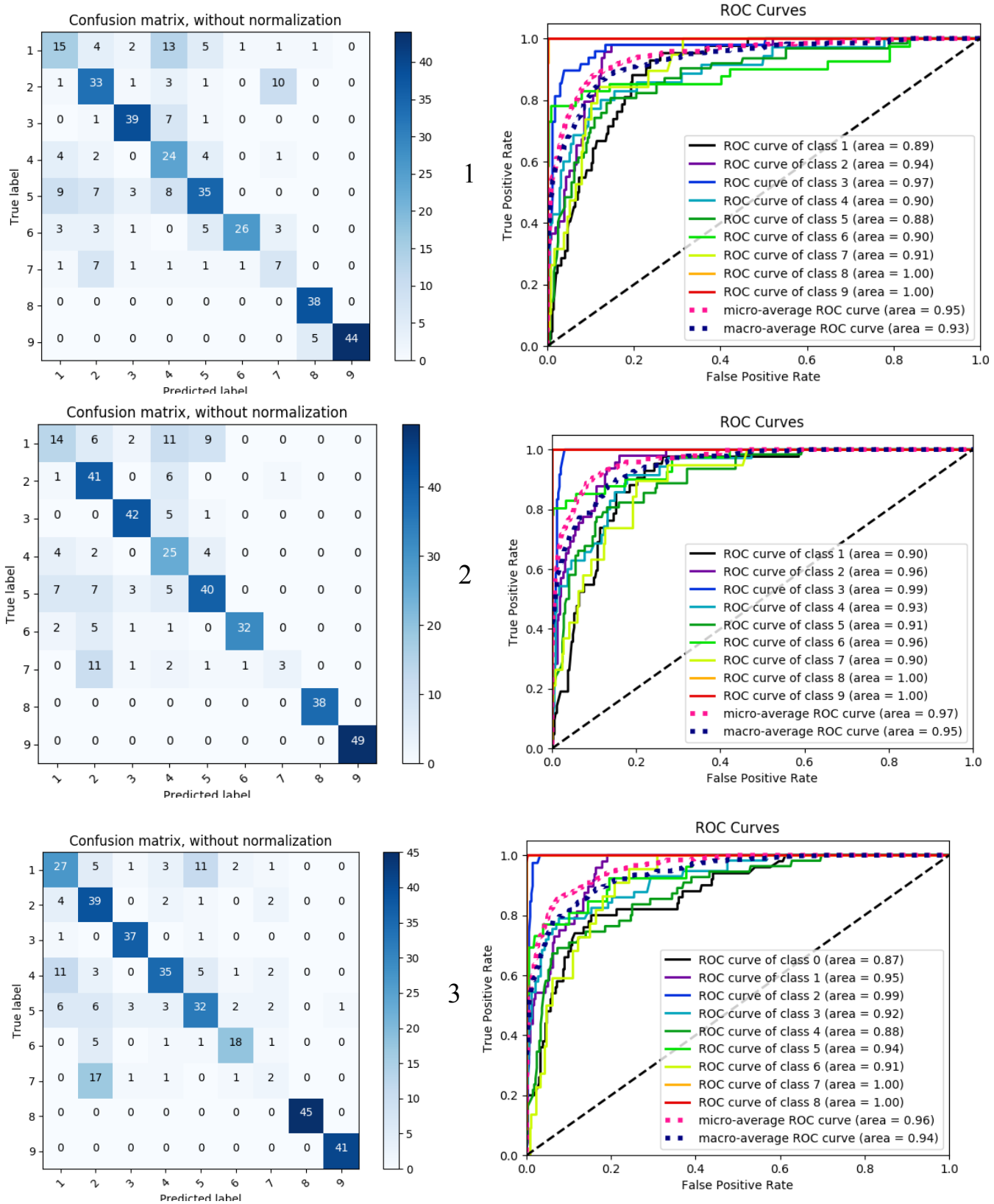

Рис В.1 Порівняння якості класифікації у разі використання різних алгоритмів класифікації (1 – Logistic Regression, 2 – Random Forest, 3 – XGBoost)

ДОДАТОК Д

ПРОБЛЕМИ

ПРИКЛАДНОЇ МАТЕМАТИКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Тези доповідей

конференції за підсумками науково-дослідної роботи

Дніпровського національного університету

імені Олеся Гончара

За 2017 рік

Дніпро
2018

ВИКОРИСТАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ КЛІНІЧНО АКТИВНИХ ГЕНЕТИЧНИХ МУТАЦІЙ

Тищенко В.О., Мацуга О.М.
Факультет прикладної математики

Алгоритми машинного навчання знаходять все більш широке застосування в медичній сфері, зокрема через потребу обробки великих даних. Сучасною сферою медицини, що генерує великі об'єми даних, є персоналізована медицина, в рамках якої діагностику та лікування пацієнтів здійснюють на основі їх індивідуальних, зокрема генетичних, особливостей. Такі генетичні особливості використовують, наприклад, у процесі діагностики ракових захворювань. В процесі аналізу генетичної інформації (секвенованого генома пацієнта) лікарі стикаються з тим, що в ній містяться мутації, які ведуть до розвитку пухлини, та мутації, які є нейтральними. Наразі лікарі класифікують їх самостійно, аналізуючи текстову клінічну літературу. Задля автоматизації даного процесу було оголошено змагання «Classifying Clinically Actionable Genetic Mutations» [1]. У даному змаганні необхідно було розробити алгоритм машинного навчання для класифікації генетичних мутацій на основі текстових клінічних даних. Дана робота присвячена розв'язанню задачі, сформульованої в контексті вищезазначеного змагання.

Під час розв'язання даної задачі було створено програмне забезпечення засобами мови python3 та бібліотек gensim, nltk, sklearn, scipy. Було реалізовано два методи представлення текстових клінічних даних у векторній формі: TF-IDF та Word2Vec [2]. Для класифікації генетичних мутацій були використані алгоритми Support Vectors Machine, Naive Bayes та Logistic Regression. Задля покращення якості текстових документів були проведені лематизація, стемінг, видалення стоп-слів, фільтрація даних за допомогою регулярних виразів. Слова, що часто зустрічалися разом у тексті, були об'єднані у біграми, тобто словосполучення з двох слів.

За результатами експериментів найвища якість класифікації була досягнута комбінацією алгоритмів TF-IDF, що призначає найвищі коефіцієнти словам, які краще відокремлюють документи один від одного, та Logistic Regression з регулялізацією L_2. В процесі експериментів розмірність векторів, які представляли текстові дані, змінювалась від 5 до 200 тисяч. Найкраща якість класифікації спостерігалася у разі розмірності 180 тисяч.

Література

1. Personalized Medicine: Redefining Cancer Treatment URL: https://www.kaggle.com/c/msk-redefining-cancer-treatment/data (Last accessed: 20.03.2018).

2. Пархоменко П.А., Григорьев А.А., Астраханцев Н.А. Обзор и экспериментальное сравнение методов кластеризации текстов. *Труды ИСП РАН*. 2017. Том 29, Вып. 2. С. 161–200. DOI: 10.15514/ISPRAS-2017-29(2)-6.

Київський університет імені Бориса Грінченка
Факультет інформаційних технологій та управління
Кафедра інформаційних технологій і математичних дисциплін

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ – 2018

Збірник тез
V Всеукраїнської науково-практичної конференції
молодих науковців

17 травня 2018 року
м. Київ

Київ – 2018

освіта надає можливості для розвитку дослідницьких вмінь учнів, і тому, беззаперечно, є актуальним напрямком розвитку національної освіти. Перспективи подальших розвідок вбачаємо в аналізі вітчизняних та зарубіжних надбань з проблеми запровадження STEM-освіти.

ДЖЕРЕЛА

1. Фіцула М. М. Педагогіка / М. М. Фіцула. – Тернопіль: Навчальна книга-Богдан, 2005. – 232 с

2. Набільська О. Використання досліду на уроках природознавства Електронний ресурс] – Режим доступу : http://ukped.com/statti/teorijanavchannja/5856-vykorystannya-doslidu-na-urokakh-pryrodoznavstva.html

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ВЕКТОРИЗАЦІЇ ТЕКСТУ В ПРОЦЕСІ КЛАСИФІКАЦІЇ БІОМЕДИЧНОГО ТЕКСТОВОГО КОРПУСУ

Тищенко В.О., Мацуга О.М.

Дніпровський національний університет імені Олеся Гончара, м. Дніпро

Алгоритми роботи з текстовими даними або Natural Language Processing знаходять все більш широке застосування у різних сферах наукової та практичної діяльності. Наприклад, потреба в таких алгоритмах виникає в задачі класифікації генетичних мутацій на основі аналізу текстової клінічної літератури. З метою розробки алгоритму для розв'язання цієї задачі на платформі Kaggle було оголошеного змагання «Classifying Clinically Actionable Genetic Mutations» [1].

Під час участі у вищезазначеному змаганні було помічено, що результати класифікації значно відрізняються у разі використання різних алгоритмів представлення тексту в векторній формі (або алгоритмів векторизації). Тож дана робота присвячена порівнянню практичної ефективності найбільш відомих алгоритмів векторизації в процесі класифікації біомедичного текстового корпусу, наданого у вищезгаданому змаганні.

У ході розв'язання даної задачі було створено програмне забезпечення засобами мови python3 та бібліотек gensim, nltk, sklearn, scipy. Було реалізовано лематизацію, стемінг, видалення стоп-слів, фільтрацію даних за допомогою регулярних виразів задля покращення якості текстових документів. Слова, що часто зустрічалися разом у тексті, було об'єднано у біграми, тобто словосполучення з двох слів. Після усіх перетворень, текстовий корпус містив 389 тис. унікальних слів, а кожний документ – 3 тис. слів.

Для дослідження було обрано наступні алгоритми векторизації тексту: CBOW, TF-IDF, Word2Vec, Doc2Vec. Їх вибір пов'язаний з тим, що

вони є достатньо поширеними при роботі з текстом та належать до двох різних груп методів векторизації: BagOfWords та Word Embeddings.

Задача класифікації вирішувалась на основі алгоритму Logistic Regression з регулялізацією L_2. Було обрано саме цей алгоритм, оскільки він добре адаптується до векторних даних, може працювати з вибірками, які мають велику кількість ознак, та має низький поріг перенавчання. Тестування було проведено й на інших методах класифікації, таких як Naiive Bayes та Support Vector Machines, але якість класифікації на їх основі була нижче.

Результати класифікації генетичних мутацій алгоритмом Logistic Regression з регулялізацією L_2 у разі використання різних алгоритмів векторизації подано нижче (табл. 1).

Таблиця 1. Результати класифікації у разі використання різних алгоритмів векторизації

№ тесту	Алгоритм векторизації	Параметри алгоритму	MultiClass Loss
1	CBOW	розмір словника = 2000	11.58
2	TF-IDF	розмір словника = 2000	2.61
3	Word2Vec	мінімальна частота слова = 100, вікно = 5, розмір вектора слова = 1	4.97
4	Doc2Vec	мінімальна частота слова = 200, вікно = 5, розмір вектора речення = 500	3.74
5	CBOW	розмір словника = 190000	9.75
6	TF-IDF	розмір словника = 190000	2.05

За результатами експериментів найвища якість класифікації була досягнута у разі використання алгоритму TF-IDF, що призначає найвищі коефіцієнти словам, які краще відокремлюють документи один від одного. Більш того, різниця результатів класифікації при різному об'ємі словника в цьому алгоритмі, є незначущою відносно інших методів. Метод CBOW має найнижчий показник, що пояснюється його простотою та наявністю великого розкиду поміж координатами векторів кожного документа. Методи Word2Vec та Doc2Vec показали нижчу якість ніж TF-IDF у більшій мірі через те, що біомедичний корпус наукових текстів, на якому було проведено тестування, має достатньо складну структуру та низьку вірогідність повтору контекстів, що є вирішальним для даних алгоритмів.