УДК 004.413 DOI https://doi.org/10.35546/kntu2078-4481.2024.1.36

Л. М. ОЛЕЩЕНКО
кандидат технічних наук,
доцент кафедри програмного забезпечення комп'ютерних систем
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0000-0001-9908-7422

О. Г. МЕЛЬНИЧУК
студент кафедри програмного забезпечення комп'ютерних систем
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0009-0008-3731-6021

ЗАСТОСУВАННЯ АСАМБЛЕВИХ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ НЕПРАВДИВОГО ТЕКСТУ

У статті представлено ансамблеві методи машинного навчання для підвищення точності виявлення неправдивого тексту та оцінено різні класифікатори з використанням окремих наборів даних. Для дослідження обрано наївний байєсівський класифікатор, пасивно-агресивний класифікатор, метод Support Vector Machine (SVM), логістичну регресію, метод k-найближчих сусідів та класифікатори випадкового лісу. Також протестовано ансамблі, що складаються з комбінацій обраних класифікаторів. Результати дослідження представляють результати класифікації, демонструючи ефективність використання асамблевих методів. Для дослідження використовувалися технології програмування Python (sklearn, pandas, numpy), процесор AMD Ryzen 5 4500U 6 ядер та 16 гігабайт оперативної пам'яті. Дослідження підкреслює важливість використання асамблевих методів для виявлення неправдивих текстів новин. Для TF-IDF – векторизації класифікатор SVM виділяється найвищою середньою точністю 95,74%. Згідно проведених досліджень, SVM досягає найвищої частки правильних прогнозів порівняно з іншими класифікаторами під час навчання на даних, перетворених TF-IDF. При використанні векторизації хешування класифікатор SVM зберігає свою високу продуктивність, досягнувши найвищої середньої точності 97,26%. Ансамблевий метод Voting Ensemble 3 (Ens3 – SVM + PA + LR) досягає середньої точності 96,93%. Основна ідея запропонованого методу полягає в аналізі тексту новин без сторонньої інформації (дати публікації, назви сайтів та додаткових медіа). Текст новин аналізується окремо за трьома показниками: правдивість тексту, сатира чи мова ворожнечі. Для навчання моделей за обраними метриками використано набори даних сервісу Kaggle, а для тестування в «реальних умовах» – довільно вибрані тексти новин і коментарів. Структурою наборів даних є текст та двійкова мітка в іншому стовпці, що відповідає заданому критерію. Набір досліджуваних даних містить 6335 рядків текстів новин і міток «true» або «false». Набір даних сатири – це комбінація двох окремих наборів даних, одного з новинної служби BBC, а іншого – з Onion.

Ключові слова: *асамблеві методи машинного навчання, алгоритми класифікації, аналіз текстів на неправдивість, TF-IDF, Python.*

L. M. OLESHCHENKO
Candidate of Technical Sciences,
Associate Professor at the Department of Computer Systems Software
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0001-9908-7422

O. H. MELNYCHUK
Student at the Department of Computer Systems Software
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0009-0008-3731-6021

MACHINE LEARNING ENSEMBLE METHODS IMPLEMENTATION FOR DECEPTIVE TEXT DETECTION

The article discusses the application of ensemble techniques to improve the accuracy of predictions and conducts an evaluation of different classifiers across distinct datasets. It explores the effectiveness of the Naive Bayesian, Passive-

aggressive, Linear Support Vector, Logistic Regression, k-nearest neighbors, and Random Forest classifiers. Moreover, it investigates the performance of ensembles that integrate these classifiers in various combinations. Python programming technologies (sklearn, pandas, numpy), AMD Ryzen 5 4500U 6-core processor, 16 gigabytes of RAM were used for the research. The findings demonstrate that while individual classifiers achieve commendable accuracy levels, their performance is further enhanced through ensemble approaches. The study details the outcomes of these classifications, highlighting the efficacy of the applied methods. Particularly, the study underscores the value of ensemble strategies in identifying false news text, offering valuable directions for subsequent inquiries. In terms of TF-IDF Vectorization, the Support Vector Machine (SVM) classifier emerges as the most accurate, with an average accuracy rate of 95.74%. This suggests that the SVM is superior in generating correct predictions more frequently than its counterparts when applied to data transformed via TF-IDF Vectorization. With Hashing Vectorization, the SVM consistently outperforms other classifiers, reaching an average accuracy of 97.26%. Among ensemble methods, the Voting Ensemble 3 (Ens3 – SVM + PA + LR) stands out, especially with Hashing Vectorization, achieving an average accuracy of 96.93%.

The core concept behind the methodology is the analysis of purely journalistic text, devoid of any irrelevant details such as publication dates, website names, or additional media content. The analysis categorizes the text based on three separate criteria: the veracity of the news, whether it is satirical, or if it constitutes hate speech. To train the models, datasets from the Kaggle platform were utilized according to these criteria, and a selection of arbitrarily chosen news texts and comments were tested under "real-world conditions". The datasets are structured with text and a binary label in another column indicating the respective criterion. The datasets include 6,335 news text entries labeled as true or false. The dataset for satire combines two distinct datasets, one from the BBC news service and another from the satirical news site The Onion.

Key words: *assembly methods of machine learning, classification algorithms, analysis of texts for falsity, TF-IDF, Python.*

Постановка проблеми

Неправдиві новини стали поширеною проблемою цифрової епохи, поширюючись через платформи соціальних медіа та онлайн-видання новин. Швидке поширення неправдивої або оманливої інформації становить серйозну загрозу суспільному дискурсу, політичним процесам і довірі суспільства. Традиційні методи перевірки текстів новин часто не відповідають масштабам і складності сучасних кампаній дезінформації. Отже, існує нагальна потреба в передових обчислювальних техніках для боротьби з поширенням неправдивих новин та збереження цілісності інформаційних каналів.

Широка доступність Інтернету та платформ соціальних мереж прискорила поширення інформації безпрецедентними темпами, заставши суспільство зненацька. Таке швидке поширення створило проблеми для підтримки неупередженості та точності новинного контенту, особливо в середовищі соціальних мереж. Різні онлайн-платформи використовують різноманітні стратегії для класифікації та оцінки схожого вмісту. У той час як традиційні інформаційні агентства покладаються на процеси перевірки вручну, платформи соціальних мереж, такі як Facebook, використовують автоматизовані системи для виявлення ненависті та ворожнечі в коментарях. Такі платформи, як X, використовують «нотатки спільноти», спільні анотації, які додаються до публікацій, пропонуючи додатковий контекст та іноді розвінчують дезінформацію. Аналізуючи величезну кількість текстових даних, моделі машинного навчання можуть робити прогнози та намагатися розпізнати правдивість нової інформації.

Формулювання мети дослідження

Метою статті є розробка асамблевого методу класифікації для аналізу новинних текстів на неправдивість, сатиру та мову ворожнечі. Основна ідея – проаналізувати новинний текст без сторонньої інформації (такої як дата публікації, назви сайтів, додаткових медіа). Текст аналізується окремо за трьома показниками: правдивість новини, сатира чи мова ворожнечі.

Аналіз останніх досліджень і публікацій

У роботах [1–2] досліджується використання програмних методів обробки природної мови та методів машинного навчання для боротьби з неправдивими новинами. Бібліотека Python scikit-learn надає широкий спектр алгоритмів машинного навчання для цієї мети. У статті [3] розглядаються ансамблеві методи та обговорюються їх переваги порівняно з окремими класифікаторами.

У дослідженні [4] порівнюються методи векторизації тексту для обробки природної мови в Text Mining, використовуючи класифікацію, таку як NBC і k-NN. Результати дослідження служать основою для подальших досліджень у сфері автоматизованого аналізу тексту. Стаття [5] розглядає зростання неправдивих новин у соціальних мережах, наголошуючи на їхніх негативних наслідках та складнощах ідентифікації через навмисно оманливий зміст. Автори статті [6] пропонують інноваційну модель FakeDetector для автоматичного визначення достовірності неправдивих новин на основі аналізу текстової інформації та мережевих взаємодій. Стаття [7] досліджує сатиру як форму обману та використовує алгоритм на основі SVM для виявлення сатиричних новин з високою точністю. У роботі [8] розглядаються методи виявлення неправдивих новин через аналіз їхніх характеристик, що публікуються у соціальних мережах. Останні дослідження висвітлюють складнощі та потребу у подальшому розвитку методів машинного навчання для виявлення та стримування поширення неправдивих текстових новин.

Дані для дослідження та тренування моделей

Для навчання моделей у даній роботі використовувалися набори даних сервісу Kaggle, а для тестування в «реальних умовах» – довільно вибрані тексти новин і коментарів. Наборів даних містять текст і двійковий знак в стовпцях, які відповідають критерію. Набір даних для дослідження містить 6335 рядків новинних текстів та мітки true або false. Набір даних для перевірки на сатиру – це комбінація двох окремих наборів даних: один із служби новин BBC, інший – із гумористичного вебсайту сатиричних новин Onion. Об'єднаний набір даних налічує 8341 рядок. Останній набір даних містить коментарі мережі X з більш ніж 24 000 рядків даних.

Найбільшу складність у такому комбінованому аналізі метрик становить різноманіття даних для окремих метрик, оскільки дані з неправдивою інформацією є довгими, офіційними текстами, а коментарі користувачів використовуються для перевірки на мову ворожнечі.

Перед навчанням моделей необхідно певним чином змінити тексти наборів даних, для цього використовується перетворення «необроблених» даних у формат, придатний для моделей машинного навчання. Це перший і обов'язковий етап створення моделей. Особливо це стосується аналізу текстових даних людського мовлення. Такі текстові дані мають «шум», який проявляється в пунктуації, емоціях і регістрі тексту, а також у відмінюванні слів. Стоп-слова – це найпоширеніші слова в тексті, які не несуть жодної інформації. Оскільки мова досліджуваних наборів даних є англійською, стоп-слова також включають артиклі. Для видалення таких слів було використано бібліотеку NLTK, що містить близько 180 стоп-слів.

Скорочення слів, або коріння, полягає у відокремленні та видаленні допоміжних частин слова від кореня, або основної форми. Бібліотека NLTK використовується для скорочення слів. Зокрема, у цьому дослідженні використано лематизацію. На відміну від звичайного стемінгу, де під час скорочення можлива втрата початкового контексту та значення слова (universal – всесвіт), під час лематизації перед скороченням відбувається морфологічний аналіз слова, хоча це займає більше часу (рис. 1).

Залежно від наборів даних, цифри також видаляються або замінюються словами в даних, або скорочені слова «розширюються».

Рис. 1. Попередня підготовка текстових даних

Алгоритми машинного навчання найчастіше приймають числові значення для навчання, тому спочатку потрібно перетворити всі текстові документи в їх числове представлення. Прикладом такого числового представлення є вектор, індекси якого є індексами унікальних слів з документів, а значення – кількість входжень цих слів у документи. Стоп-слова також можна знову видалити під час векторизації.

У даному дослідженні було обрано та використано два векторизатори: TD – IDF та векторизатор із хешуванням. TF – IDF (Term Frequency – Inverse Document Frequency) є показником важливості слів у документах. Важливість слова – це кількість випадків/вживань слова в документах. Зворотна частота стосується унікальності

слова в колекції документів. Велике значення TF – IDF досягається, коли значення TF є великим і коли вага в знаменнику у формулі IDF не має великого значення.

Один із векторизаторів, який вбудовано в бібліотеку scikit – learn і є найпростішим у реалізації. Він створює матрицю з усіх документів, де кожне унікальне слово є стовпцем, а документи є рядками. Значення в кожній клітинці є числовим і означає, скільки разів це слово з'являється в документі. У scikit – learn слова реалізації зберігаються не як текст, а як індекси.

Хешований векторизатор не зберігає словник індексів слів. Кожне слово/токен хешується за допомогою функції Murmurhash 3 і відображається безпосередньо в стовпці матриці. Це має перевагу, оскільки доступні набори даних досить великі.

Обґрунтування використання класифікаторів

У порівнянні з глибоким навчанням, моделями мовлення або згортковими чи рекурентними нейронними мережами, звичайні класифікатори зазвичай поступаються точністю прогнозування. По-перше, було висунуто гіпотезу, що якщо окремі класифікатори поступаються в точності більш складним моделям, то можна спробувати використовувати кілька класифікаторів і обрати найкращі прогнози для досягнення максимальної точності. По-друге, класифікатори набагато простіші в реалізації та швидше навчаються, і тому вони ідеально підходять для роботи в умовах обмеженого часу або коли потрібно перевірити інші налаштування чи параметри. По-третє, класифікатори набагато краще працюють з невеликими наборами даних, ніж складніші моделі. Усі використані набори даних відносно невеликі за обсягом і були взяті для дослідження без використання додаткових носіїв чи вебскребків і не мають масштабу для навчання складних моделей рівня ChatGPT. Мовні моделі, з іншого боку, вимагають досить великої кількості даних для отримання оптимальних результатів.

У дослідженні тексти розділені на три категорії: неправдиві новини, сатира та мова ворожнечі. Щоб спростити навчання та використання класифікаторів, було вирішено навчати кожну метрику окремо, тобто спростити задачу класифікації до бінарної. Навчені класифікатори зберігаються для подальшого використання в ансамблевих методах.

Однією з цілей дослідження є визначення ймовірності належності тексту до категорії. Бібліотека scikit – learn має вбудовану функцію predict proba (), яка працює не з усіма використовуваними класифікаторами, тому деякі з них були змінені.

Наївний байєсівський класифікатор підходить для класифікації коротких текстів і припускає, що ознаки тексту не пов'язані між собою, але таке припущення не завжди вірне. Один із наборів даних наповнений коментарями з мережі X, і цей фактор також вплинув на вибір класифікатора.

Мультиноміальний варіант Байєса, обраний для дослідження, може ефективно моделювати спільний розподіл ознак і пов'язаних з ними класів за допомогою мультиноміального розподілу. Розподіл частоти для кожного терміна можна покращити, включивши міру дисперсії, таку як частота терміну, зворотна частоті документа (TF-IDF), яка враховує кількість документів, у яких кожен зустрічається. Це може суттєво підвищити продуктивність, надавши більшої ваги термінам, які з'являються в меншій кількості документів, і, таким чином, покращити їх здатність розрізняти.

Незважаючи на те, що мультиноміальний розподіл добре працює при безпосередньому використанні з частотами термінів, він також добре працює для дробових значень, таких, як значення TF-IDF.

Пасивно-агресивний класифікатор вважається одним із найкращих серед розглянутих у даній роботі. Він часто використовується для поділу тексту на дві групи, і тому дуже добре підходить для використання в цьому дослідженні. Крім того, він часто використовується для фільтрації спаму та виявлення шахрайства. Пасивно-агресивний класифікатор не має ймовірнісного методу, тому було вирішено створити для нього своєрідну обгортку:

```
def predict_proba (self, x_test):
arr1= 1-(1. / (1. + np.exp (– self.decision_function (x_test))))
arr2= -(arr1-1)
return np.stack ((arr2, arr1), axis =1)
```

Вбудована функція decision_function() показує, як алгоритм приймає рішення щодо класифікації та повертає масив numpy, де кожен елемент надається для значень x вибірки x _ test (дані для тестування) показує відстань від гіперплощини з різних сторін, зі значеннями від -1 до 1, з 0 точно посередині.

Метод опорних векторів Support Vector Machine (SVM), хоча й працює повільніше, ніж байєсівський класифікатор, проте вважається одним із найкращих і використовується як для класифікації, так і для регресії.

У scikit-learn модуль SVM є оболонкою для бібліотеки libsvm і підтримує різні ядра. У дослідженні використовується функція LinearSVC (), яка підтримує лише лінійне ядро, але є швидшою, ніж модуль SVM. Ця функція потім калібрується за допомогою CalibratedClassifierCV. Це було зроблено для того, щоб отримати ймовірності відповідно текстовому типу даних.

Логістична регресія підходить для двійкової класифікації та одразу повертає ймовірність. У scikit – learn функція LogisticRegression () реалізує регуляризовану логістичну регресію за допомогою бібліотеки liblinear.

Класифікатор k-найближчих сусідів (KNN) є одним із найпростіших і найпоширеніших алгоритмів, які використовуються для класифікації тексту. У деяких конкретних випадках KNN може перевершити громіздкі моделі типу BERT. У 2023 році група дослідників з Університету Ватерлоо опублікувала зразок коду, у якому алгоритм використовувався з бібліотекою gzip для стиснення показників відстані. Це було зроблено для того, щоб довести, що подібність між двома текстовими документами тісно пов'язана з їх стисливістю у файли меншого розміру.

Класифікатор випадкового лісу є популярним вибором для класифікації тексту, оскільки він добре обробляє великі обсяги текстових даних і може обробляти складні зв’язки між словами в тексті та категоріями, до яких вони належать. Одна з головних переваг цього класифікатора полягає в тому, що він здатний уникнути перенавчання, коли модель є надто складною та надто точно відповідає навчальним даним, що призводить до низької продуктивності нових даних.

У даному дослідженні використовували такі комбіновані ансамблеві методи:

1. наївний байєсівський класифікатор + KNN + логістична регресія;
2. класифікатор KNN + пасивно-агресивний класифікатор;
3. лінійний класифікатор опорних векторів + пасивно-агресивний класифікатор + логістична регресія.

Ці комбінації вибиралися за різними, але не жорсткими принципами. Найпопулярнішими серед обраних алгоритмів є Naive Bayes, KNN та логістична регресія, тому було цікаво перевірити їх ефективність відносно один одного. SVM, пасивно-агресивний класифікатор (PA) і логістична регресія (LR) продемонстрували найкращі результати в попередніх тестах на доступних наборах даних.

Результати дослідження

У Python навчання класифікатора включає декілька кроків. По-перше, ми імпортуємо необхідні бібліотеки, такі як scikit-learn, яка надає широкий спектр алгоритмів машинного навчання. Потім ми готуємо дані, розділяючи їх на навчальні та тестові набори за допомогою таких функцій, як train_test_split() . Далі ми ініціалізуємо наш класифікатор і «підганяємо» його до навчальних даних за допомогою методу fit(). Під час цього процесу класифікатор вивчає закономірності та зв’язки в даних. Після навчання ми оцінюємо продуктивність класифікатора на основі даних тестування, використовуючи такі показники, як точність, точність, запам’ятовування або оцінка F1. Нарешті, ми точно налаштовуємо гіперпараметри моделі за допомогою таких методів, як пошук у сітці або перехресна перевірка, щоб оптимізувати її продуктивність.

Векторизація класифікатора передбачає перетворення необроблених текстових даних у числові характеристики, які можуть використовуватися алгоритмами машинного навчання. Зазвичай це робиться за допомогою таких методів, як CountVectorizer або TF-IDFVectorizer з бібліотеки scikit-learn для векторизації текстових даних:

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = ["This is the first document.", "This document is the second document.", "And this is the third one.", "Is this the first document?"]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
```

Навчання класифікатора виконується циклами, окремо для кожного класифікатора/комбінації, окремо для кожного набору даних і окремо за допомогою векторизаторів. Після навчання класифікаторів проводиться їх тестування на прикладах кожної текстової категорії. Класифікатори повинні давати ймовірність того, що текст належить або не належить до обраної категорії. Для цього використовується функція scikit – learn predict() і predict proba(). Метод predict визначає клас даних, а метод predict proba визначає ймовірність того, що дані належать до певного класу для кожного з них, і повертає масив цих значень.

Для векторизації TF-IDF класифікатор опорних векторів SVM виділяється найвищою середньою точністю 95,74%. Це вказує на те, що SVM досяг найвищої частки правильних прогнозів порівняно з іншими класифікаторами під час навчання на даних, перетворених TF-IDF. Крім того, класифікатор SVM також досяг найвищого середнього правильного прогнозу 4,67, що свідчить про його надійність у точній категоризації текстових даних новин. З іншого боку, при використанні векторизації хешування класифікатор SVM зберіг свою продуктивність, досягнувши найвищої середньої точності 97,26% і найвищого середнього правильного передбачення 4,67. Результати демонструють ефективність SVM у обробці простору ознак великої розмірності, створеного методом векторизації хешування.

Крім того, ансамблеві методи, зокрема Voting Ensemble 3 (SVM + PA + LR), показали високу продуктивність, особливо з векторизацією хешування. Цей ансамбль досяг середньої точності 96,93% і середнього правильного прогнозу 4,67, ще більше підкреслюючи переваги поєднання кількох стратегій класифікатора для підвищення точності та правильності класифікації (табл. 1).

Таблиця 1

Порівняння результатів класифікації

Класифікатор	Середня точність	Середня кількість вдалих передбачень
Наївний Баєсів (NB)	93.23%	4
Пасивно-агресивний (PA)	94.8%	4.67
Метод опорних векторів (SVM)	95.58%	4.67
Логістична регресія (LR)	94.27%	4.5
k-найближчих сусідів (KNN)	90.79%	4.83
Випадкового лісу (FOREST)	86.74%	4
NB + KNN + LR	94.54%	4.33
KNN + PA	78.72%	4
SVM + PA + LR	94.98%	4.83

Висновки та перспективи подальших досліджень

У статті розглянуто використання ансамблевих методів класифікації неправдивих текстів для підвищення точності прогнозування та оцінено різні моделі класифікації тексту за допомогою окремих наборів даних. Оцінено наївний байєсівський, пасивно-агресивний класифікатори, метод опорних векторів, логістичну регресію, метод k-найближчих сусідів та класифікатори випадкового лісу. Для подальших досліджень можна розширити аналіз застосування ансамблевих методів класифікації на різноманітніші типи даних та ситуації, що можуть виникнути у контексті виявлення неправдивих текстів. Також можна дослідити вплив використання різних комбінацій класифікаторів та алгоритмів на точність та надійність прогнозування, а також ефективність цих методів у різних областях, таких як соціальні мережі, новинні портали чи форуми. Такі дослідження можуть принести нові важливі відомості про те, як оптимізувати використання ансамблевих методів для боротьби з поширенням неправдивої інформації в мережі Інтернет.

Список використаної літератури

1. Vasu Agarwal, H. Parveen Sultana, Srijan Malhotra, Amitrajit Sarkar (2019). Analysis of classifiers for fake news detection, Procedia Comput. Sci., 165 (2019), pp. 377–383, DOI: 10.1016/j.procs.2020.01.035.

2. Chary Deekshith P., Singh R.P. (2020). Review on Advanced Machine Learning Model: Scikit-Learn (July 4, 2020), International Journal of Scientific Research and Engineering Development (IJSRED) Vol. 3, Issue 4, 526–529.

3. Dietterich T.G. (2000). Ensemble Methods in Machine Learning. In: Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol 1857. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45014-9_1.

4. Urszula Krzeszewska, Aneta Poniszewska-Maranda, Joanna Ochelska-Mierzejewska (2022). Systematic Comparison of Vectorization Methods in Classification Context. Applied Sciences. 12. 5119. DOI: 10.3390/app12105119.

5. Shu K., Sliva A., Wang S., Tang J., & Liu H. (2017). Fake News Detection on Social Media: A Data Mining Perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22–36. DOI: 10.1145/3137597.3137600.

6. Wang W., Cui P., Zhu W., & Yang S. (2018). Fake News Detection with Deep Diffusive Neural Network. Proceedings of the 2018 World Wide Web Conference on World Wide Web (pp. 797–806).

7. Rubin V. L., Conroy N. J., & Chen Y. (2015). Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. Proceedings of the Association for Information Science and Technology, 52(1), 1–4. DOI: 10.18653/v1/W16-0802.

8. Reis J. C., Correia A., Murai F., Veloso A., Benevenuto F., & Cambria E. (2019). Supervised Learning for Fake News Detection. IEEE Intelligent Systems, 34(2), 76–81. DOI: 10.1109/MIS.2019.2899143.