ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«МЕТІНВЕСТ ПОЛІТЕХНІКА»

МАШИННЕ НАВЧАННЯ У КІБЕРБЕЗПЕЦІ:

методичні рекомендації до виконання індивідуального завдання №1 «Використання методів Supervised learning для задачі фільтрації спаму»

Запоріжжя 2025

УДК 004.85: 004.056 (072)
М38

Рекомендовано Науково-методичною радою ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА» (протокол № 3 від 26.12.2025 р.)

Укладач:
Москаленко В.В., д.т.н.

М38 **Машинне навчання у кібербезпеці :** методичні рекомендації до виконання індивідуального завдання №1 «Використання методів Supervised learning для задачі фільтрації спаму»/ уклад. В. В. Москаленко. Запоріжжя : ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2025. 38 с.

У методичних рекомендаціях наведено умови, поради і методичні підходи до виконання індивідуального завдання №1 з дисципліни «Машинне навчання у кібербезпеці», вимоги до оформлення, подання та оцінювання результатів виконання індивідуальних завдань.

Рекомендовано для студентів спеціальності 122 - Комп’ютерні науки першого (бакалаврського) рівня освіти.

УДК 004.85: 004.056(072)

ЗМІСТ

ВСТУП

Виконання індивідуального завдання №1 «Використання методів Supervised learning для задачі фільтрації спаму» для дисципліни «Машинне навчання у кібербезпеці» має на меті отримання, закріплення та поглиблення теоретичних знань студентів щодо використання методів машинного навчання (Machine Learning, ML), а саме методів контрольованого навчання (Supervised Learning) для розв'язання задач кібербезпеки. Ці знання отримуються студентами на лекційних та практичних заняттях та закріплюються у ході самостійного виконання індивідуального завдання.

Методичні рекомендації до виконання індивідуальних завдань укладено на підставі Стандарту вищої освіти за спеціальністю 122 «Комп'ютерні науки» у галузі знань 12 «Інформаційні технології» для першого (бакалаврського) рівня вищої освіти (затверджено Наказом Міністерства освіти і науки України від 28.04.2022 р. № 393).

Індивідуальне домашнє завдання (ІДЗ) виконується з використанням отриманих теоретичних та практичних знань за темою змістовного модуля №1 «Методи Supervised Learning для розв'язання задач кібербезпеки» робочої програми дисципліни.

Machine Learning як розділ штучного інтелекту використовується у кібербезпеки для того, щоб аналізувати та співвідносити дані про події та кіберзагрози з різних джерел. Ці результати перетворюються на зрозумілі та дієві аналітичні висновки, які фахівці з безпеки зможуть використати для подальшого розслідування, реагування та звітування.

У курсі розглядаються основні задачі кібербезпеки, які розв'язуються з використанням методів ML, а саме: фільтрація спаму за допомогою методів класифікації та методів обробки природної мови (NLP); виявлення аномалій за даними мережевого трафіку на основі методів неконтрольованого машинного навчання; виявлення зловмисного ПЗ за допомогою штучних нейронних мереж Convolutional Neural Networks.

Вимоги до попереднього рівня знань.

– базові знання з кібербезпеки, вищої математики, чисельних методів, дослідження операцій, теорії ймовірностей та математичної статистики, з основ штучного інтелекту;

– навички програмування, наприклад мовами Python або Java.

Основні завдання першого змістовного модуля включають формування знань щодо методів контрольованого навчання (Supervised Learning) для розв'язання задач кібербезпеки. Вивчення дисципліни дозволить студентам мати розуміння сучасного стану проблем у кібербезпеці, які можуть бути вирішені за допомогою розробки та впровадження систем штучного інтелекту до систем інформаційної безпеки підприємств різних галузей економіки.

Освітня компонента (ОК) «Машинне навчання у кібербезпеці» спрямована на отримання здобувачами наступних загальних та спеціальних (фахових) компетентностей (згідно ОПП «Комп'ютерні науки» першого (бакалаврського) рівня вищої освіти) [1]:

– ЗК1 – здатність до абстрактного мислення, аналізу та синтезу;

– ЗК2 – здатність застосовувати знання у практичних ситуаціях;

– ЗК3 – знання та розуміння предметної області та розуміння професійної діяльності;

– ЗК6 – здатність вчитися й оволодівати сучасними знаннями;

– ЗК7 – здатність до пошуку, оброблення та аналізу інформації з різних джерел;

– ЗК8 – здатність генерувати нові ідеї (креативність);

– ЗК11 – здатність приймати обґрунтовані рішення;

– ЗК12 – здатність оцінювати та забезпечувати якість виконуваних робіт;

– СК2 –здатність до виявлення статистичних закономірностей недетермінованих явищ, застосування методів обчислювального інтелекту, зокрема статистичної, нейромережевої та нечіткої обробки даних, методів машинного навчання та генетичного програмування тощо.

– СК7 – здатність застосовувати теоретичні та практичні основи методології та технології моделювання для дослідження характеристик і поведінки складних об'єктів і систем, проводити обчислювальні експерименти з обробкою й аналізом результатів;

– СК11 – здатність до інтелектуального аналізу даних на основі методів обчислювального інтелекту включно з великими та погано структурованими даними, їхньої оперативної обробки та візуалізації результатів аналізу в процесі розв'язування прикладних задач;

– СК14 – здатність застосовувати методи та засоби забезпечення інформаційної безпеки, розробляти й експлуатувати спеціальне програмне забезпечення захисту інформаційних ресурсів об'єктів критичної інформаційної інфраструктури.

Програмні результати ОК за ОПП «Комп'ютерні науки» [1]:

– ПР4 – використовувати методи обчислювального інтелекту, машинного навчання, нейромережевої та нечіткої обробки даних, генетичного та еволюційного програмування для розв'язання задач розпізнавання, прогнозування, класифікації, ідентифікації об'єктів керування тощо;

– ПР8 – використовувати методологію системного аналізу об'єктів, процесів і систем для задач аналізу, прогнозування, управління та проєктування динамічних процесів в макроекономічних, технічних, технологічних і фінансових об'єктах;

– ПР9 – розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості

застосування для реалізації методів та алгоритмів розв’язання задач в галузі комп’ютерних наук;

– ПР12 –- застосовувати методи та алгоритми обчислювального інтелекту та інтелектуального аналізу даних в задачах класифікації, прогнозування, кластерного аналізу, пошуку асоціативних правил з використанням програмних інструментів підтримки багатовимірного аналізу даних на основі технологій DataMining, TextMining, WebMining;

– ПР15 – розуміти концепцію інформаційної безпеки, принципи безпечного проєктування програмного забезпечення, забезпечувати безпеку комп’ютерних мереж в умовах неповноти та невизначеності вихідних даних.

Інші програмні результати:

– вміти розв’язувати актуальні задачі кібербезпеки, а саме: фільтрація спаму, виявлення аномалій за даними мережевого трафіку, виявлення зловмисного ПЗ та ін. – за допомогою методів ML, наприклад, методів класифікації та інших методів Supervised Learning, методів Unsupervised Learning та штучних нейронних мереж, вміти використовувати методи обробки природної мови (NLP);

– мати спеціалізовані уміння/навички щодо розробки програмного забезпечення (ПЗ) для розв’язання задач кібербезпеки, навички підготовки даних, аналізу та представлення результатів розв’язання задач кібербезпеки, враховуючи їх особливості.

Виконання ІДЗ передбачає інтеграцію навчальної, практичної, комунікативної та інших видів діяльності здобувачів освіти з використанням матеріалів реальних проєктів у сфері ІТ підприємств, зокрема активів Групи МЕТІНВЕСТ [2, 3, 4].

1. ПОРЯДОК ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

1.1 Мета та завдання ІДЗ № 1 «Використання методів Supervised learning для задачі фільтрації спаму»

Мета індивідуального завдання – це розробка ПЗ для розв'язання задачі фільтрації спаму за допомогою методів ML (методів Supervised learning) та аналіз результатів.

За робочою програмою дисципліни виконання ІДЗ №1 передбачає виконання таких основних завдань для студентів:

1) вивчення методів контрольованого навчання для розв'язання задачі кібербезпеки – розпізнавання спаму, а саме:
   - наївний Байєсівський класифікатор (Naïve Bayes);
   - рандомні дерева (Random Forest Classifier);
   - k -найближчих сусідів (K-Neighbors Classifier);
   - машини опорних векторів (Support Vector Machines);
2) підготовка даних для розв'язання задачі фільтрації спаму з використанням методів NLP;
3) розробка ПЗ для розв'язання задачі фільтрації спаму на основі методів контрольованого навчання та NLP (рекомендовано обрати мову Phyton та використати відповідні бібліотеки);
4) аналіз результатів та обґрунтування вибору ефективного методу ML для фільтрації спаму.

1.2 Короткі теоретичні відомості щодо методів класифікації

1.2.1 Наївний байєсівський класифікатор

Наївний класифікатор Байєса (Naive Bayes, NB) є імовірнісним класифікатором, заснованим на формулі Байєса зі строгим (наївним) припущенням про незалежність ознак між собою для даного класу, що значно спрощує задачу класифікації за рахунок оцінки одновимірної ймовірностної щильності замість однієї багатовимірної [5, 6].

Одновимірна ймовірнісна щільність - це оцінка ймовірності кожної ознаки окремо за умови, що вони незалежні, а багатовимірна щільність - це оцінка ймовірності поєднання всіх ознак у випадку їх залежності. З цієї причини даний класифікатор називають наївним, оскільки він дозволяє значно спростити розрахунки і підвищити ефективність алгоритму. Однак це припущення не завжди вірно на практиці, у деяких випадках воно може призвести до значного погіршення якості прогнозів.

Формула Байєса записується так [6, 7]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

де *P(A|B)* є апостеріорною ймовірністю настання події А за умови, що подія В вже здійснена;

P(B|A) – умовна ймовірність настання події В при настанні події А;

P(A) і P(B) — апріорні ймовірності подій А і В, відповідно.

У контексті машинного навчання формула Байєса набуває такого вигляду:

$$P(y_k|X) = \frac{P(y_k)P(X|y_k)}{P(X)},$$

де $P(y_k|X)$ – апостеріорна ймовірність приналежності зразка до класу y_k з урахуванням його ознаки X;

$P(X|y_k)$ – правдоподібність, тобто ймовірність ознак X при заданому класі y_k;

$P(y_k)$ – апріорна ймовірність випадково вибраного спостереження (зразка), що належить до класу y_k;

$P(X)$ – апріорна ймовірність ознак X.

Якщо об'єкт описується не однією, а декількома ознаками *X1, X2,...,Xn*, то формула набуває вигляду:

$$P(y_k|X_1, X_2, \ldots, X_n) = \frac{P(y_k)\prod_{i=1}^{n} P(X_i|y_k)}{P(X_1, X_2, \ldots, X_n)}.$$

На практиці найбільший інтерес представляє чисельник цієї формули, так як знаменник залежить тільки від ознак, а не від класу, і тому його часто опускають при порівнянні ймовірностей різних класів. У підсумку правило класифікації буде пропорційним вибору класу з найбільшою апостеріорною ймовірністю:

$$y_k \propto \arg\max_{y_k} P(y_k) \prod_{i=1}^{n} P(X_i|y_k).$$

Для оцінки параметрів моделі, тобто ймовірностей $P(y_k)$ і $P(X_i|y_k)$, зазвичай використовується метод максимальної правдоподібності, який у даному випадку ґрунтується на частотах виникнення класів і ознак у навчальній вибірці.

У бібліотеці scikit-learn є кілька реалізацій наївного байєсівського класифікатора, які відрізняються своїми припущеннями щодо розподілу ознак для даного класу. До них можна віднести наступні:

Гауссівський наївний байєсівський класифікатор (GaussianNB) є варіантом для роботи з неперервними ознаками, які мають нормальний (гауссівський) розподіл. Імовірність наявності ознаки для даного класу обчислюється за формулою:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right),$$

де μ_y і σ_y - середнє і стандартне відхилення ознаки в класі . Ці параметри оцінюються за допомогою методу максимальної правдоподібності за даними навчання.

Мультиноміальний наївний баєсовий класифікатор (MultinomialNB) – це варіант для роботи з дискретними ознаками, які мають мультиноміальний розподіл. Такі ознаки часто зустрічаються у задачах класифікації тексту, де вони представляють кількість входжень у тексті.

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n},$$

де N_{yi} – кількість разів ознака *i* зустрічається у класі *y*;

N_y – загальна кількість усіх ознак у класі *y;*

n – кількість ознак;

α – згладжуючий параметр, що запобігає виникненню нульових ймовірностей.

Комплементарний наївний байєсівський класифікатор (ComplementNB) – це покращена версія *MultinomialNB*, яка підходить для незбалансованих наборів даних. Замість того, щоб оцінювати ймовірність ознаки для даного класу, алгоритм оцінює нормовану вагу ω_{ci} ознаки для класу ***c*** як ймовірність ознаки, коли клас доповнюється, тобто для всіх інших класів, що робить його менш чутливим до упередженості вибірки. Формула для розрахунку ймовірності ознаки при доповненні класу виглядає наступним чином:

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = \log \hat{\theta}_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|},$$

де $\widehat{\theta_{ci}}$ – оцінка ймовірності виникнення ознаки *i* при доповненні класу ***c***;

α_i – параметр згладжування;

d_{ij} – кількість випадків виникнення ознаки *i* у класі *j* (частота прояви ознаки *i* у всіх класах, крім ***c***);

ω_{ci} – нормалізована вага ознак *i* для класу ***c***.

Прогнозований клас $\hat{c}$ для заданого вектора ознак ***t*** буде виглядати так:

$$\hat{c} = \arg\min_c \sum_i t_i w_{ci}$$

.

Бернуллієвський наївний байєсівський класифікатор (BernoulliNB) є ще одним варіантом роботи з дискретними ознаками, але які мають бернуліанський розподіл [8]. У цьому випадку ознаки є бінарними показниками наявності або відсутності певних властивостей (ознак) у об'єкта. Наприклад, у задачі класифікації тексту це може бути наявність або відсутність певних слів в тексті:

$$P(x_i = t \mid y = c\,;\,\alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

де $P(x_i = 1|y)$ – ймовірність того, що ознака *i* набуває значення (істина) за умови, що об'єкт належить до класу ***y***;

x_i – значення атрибуту ознаки *i* (0 або 1).

Категоричний NB – це варіант категоріально розподілених даних, заснований на припущенні, що кожна ознака, описана індексом, має свій категоріальний розподіл. Імовірність настання ознаки для даного класу обчислюється за формулою:

$$P(x_i = t \mid y = c\,;\,\alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

де $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$ – кількість разів, коли ознака x_i приймає значення *t* у класі ***c***;

$N_c = |\{j \in J \mid y_j = c\}|$ – сумарна кількість усіх ознак у класі ***c*** у тренувальних даних;

α – параметр згладжування;

n_i – це число доступних значень ознаки ***i***.

Принцип роботи NB класифікатора з розподілом Гаусса опишимо за допомогою алгоритму, який складається з таких кроків:

1) обчислюються апріорні ймовірності класів;

2) обчислюються середнє і стандартне відхилення ознак за класами;

3) обчислюється імовірнісна щільність тестових ознак за гауссовим розподілом на основі отриманих відхилень ознак за класами;

4) обчислюються апостеріорні ймовірності як добуток апріорних ймовірностей класів і імовірнісних щільностей тестових ознак;

5) класи, які мають найбільшу апостеріорну ймовірність будуть остаточним прогнозом щодо спаму.

Наївний баєсівський класифікатор програмно реалізовано у Python [9, 10] у бібліотеках для різних задач класифікації.

1.2.2 Використання класифікатора наївного Байєса для фільтрації спаму

У контексті фільтрації спаму, наївний баєсівський класифікатор базується на частоті появи слів у спамі та неспам-повідомленнях, а також максимізації добутку їхніх ймовірностей [11 - 15].

«Наївність» у цьому випадку буде полягати у припущенні, що слова у повідомленні не залежать від порядку та контексту.

Тоді формула Байєса набуває наступного вигляду:

$$P(C|M) \propto P(C) \prod_{i=1}^{n} P(w_i|C), \;\; w_i \in M$$

,

де *C* – клас: спам або не спам;

M – повідомлення;

w_i - *i*-е слово у повідомленні *M*;

$\propto$ – знак пропорційності.

Приклад. Припустимо, треба класифікувати повідомлення **«***Hi, you won a discount and you can get the prize this evening.».*

Є зразок тренінгу, що складається з повідомлень, які наведені у таблиці 1.

Таблиця 1 – Зразок тренінгових повідомлень

Message	Class
Hi, how are you?	Not spam
Congratulations, you won a prize!	Spam
Buy the product now and get a discount!	Spam
Let's walk this evening	Not spam

Насамперед необхідно розрахувати частоту входження всіх унікальних слів та їх загальну кількість у спам і не спам-повідомленнях. Потім розрахувати ймовірність появи кожного слова у спам і не спам-повідомленнях на основі цих частот.

Коли у повідомленні є слова, які раніше не з'являлися у навчальному зразку, використовується згладжування. Існує багато різних типів згладжування, але сутність найпростішого згладжування полягає у тому, щоб до частотності слів у повідомленнях додати 1. Цей прийом дозволяє уникнути проблеми нульової ймовірності.

Розрахунки ймовірностей для усіх слів наведені у табличній формі на рис. 1

Word	Frequency in Not Spam	Frequency in Spam	Probability in Not Spam	Probability in Spam
Hi	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
how	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
are	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
you	1 + 1 = 2	1 + 1 = 2	2 / 28 = 0.0714	2 / 33 = 0.06
Congratulations	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
won	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
a	0 + 1 = 1	2 + 1 = 3	1 / 28 = 0.0357	3 / 33 = 0.09
prize	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
Buy	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
the	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
product	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
now	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
and	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
get	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
discount	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
Let's	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
walk	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
this	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
evening	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
can	0 + 1 = 1	0 + 1 = 1	1 / 28 = 0.0357	1 / 33 = 0.03
Total amount of words	**28**	**33**		

Рисунок 1 – Розрахунки ймовірностей слів повідомлення

Далі обчислюються ймовірності того, що повідомлення є спамом або не є спамом, і остаточним прогнозом буде клас з найбільшою ймовірністю:

$$P(C|M) = P(C) \cdot P('Hi'|C) \cdot P('you'|C) \cdot P('won'|C) \cdot P('a'|C) \cdot$$
$$\cdot P('discount'|C) \cdot P('and'|C) \cdot P('you'|C) \cdot P('can'|C) \cdot P('get'|C) \cdot$$
$$\cdot P('the'|C) \cdot P('prize'|C) \cdot P('this'|C) \cdot P('evening'|C)$$

де $C \in (Spam,\ \ Not\ \ Spam)$;

$$P(Spam) = P(Not\ \ Spam) = \frac{2}{4} = 0.5.$$

Імовірність того, що повідомлення є спамом:

$$P(Spam|M) = 0.5 \cdot 0.03 \cdot 0.06 \cdot 0.06 \cdot 0.09 \cdot 0.06 \cdot 0.06 \cdot 0.06 \cdot 0.03 \cdot$$
$$\cdot 0.06 \cdot 0.06 \cdot 0.06 \cdot 0.03 \cdot 0.03 \approx 6.12 \cdot 10^{-18}$$

Імовірність того, що повідомлення не є спамом:

$$P(Not\ \ Spam|M) = 0.5 \cdot 0.0714 \cdot 0.0714 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot$$
$$\cdot 0.0714 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot 0.0714 \cdot 0.0714 \approx 2.45 \cdot 10^{-18}$$

Тому що $P(Spam|M) > P(Not\ \ Spam|M) \rightarrow$ повідомлення є спамом.

Варто додати, що на практиці для зручності обчислень замість самої ймовірності часто використовується логарифм ймовірності.

Переваги наївного байєсівського класифікатора:

– простота реалізації та тлумачення;

– практично не потрібне коригування параметрів

– висока швидкість роботи і точність прогнозів у багатьох ситуаціях;

– він має відносно непогану стійкість до шумів і викидів, оскільки заснований на розподілах ймовірностей і наївному припущенні про незалежність ознак.

Недоліки наївного байєсівського класифікатора:

– при порушенні припущення про незалежність ознак точність прогнозів може бути значно знижена;

– може віддавати перевагу класам з більшою кількістю вибірок у разі незбалансованих даних.

1.2.3 Метод *k* -найближчих сусідів

Метод *k*-найближчих сусідів (k-nearest neighbors algorithm, k-NN) – це контрольований алгоритм машинного навчання, який використовує інформацію перед міткою для прогнозування подальшої інформації. Він особливо корисний у проблемах класифікації, як-от виявлення зловмисного програмного забезпечення [16, 17]. У K-NN алгоритм зберігає усі доступні випадки та класифікує нові випадки на основі міри подібності (функції відстані) (рис.2).

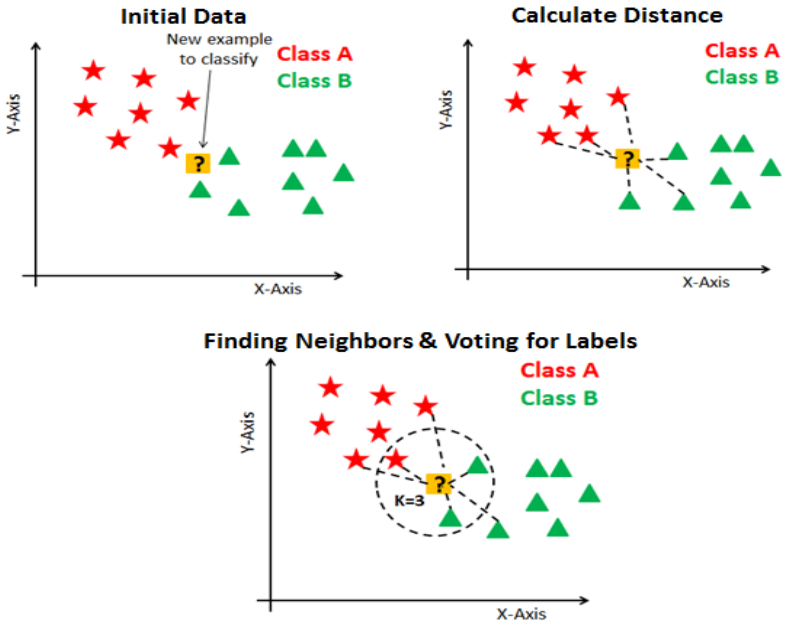

Рисунок 2 – Метод K-найближчих сусідів [16]

Для задачі фільтрації спаму алгоритм K-NN порівнює її з існуючими програмами, позначеними як «зловмисне програмне

забезпечення» або «не шкідливе програмне забезпечення», і передбачає її мітку на основі найчастішого класу серед найближчих сусідів.

Розглянемо використання k-NN у сфері кібербезпеки для задачі виявлення шкідливих програм.

Скажімо, у нас є функції, отримані з набору програм, де кожна програма позначена як «зловмисне програмне забезпечення» або «не шкідливе програмне забезпечення». Функції можуть складатися з числових, категоріальних і двійкових значень. Ось невеликий приклад набору функцій [16]:

– кількість підозрілих викликів API (числові дані);
– тип операційної системи (Windows/Mac/Linux) (категорійні дані);
– наявність відомих рядків зловмисного програмного забезпечення (двійкові);
– методи обфускації коду (категорійні);
– час виконання (числові).

Вибір значення ***k*** може значно вплинути на точність і поведінку алгоритму K-NN:

– Низький *k* (наприклад, *k* =1) : Алгоритм фокусується лише на найближчому сусіді. Це може зробити його чутливим до шуму та викидів, що призведе до неправильної класифікації. Під час виявлення зловмисного програмного забезпечення це може призвести до неправильної класифікації на основі одного крайнього випадку.

– Високий *k* (наприклад, *k* =20) : алгоритм враховує більше сусідів під час прогнозування, що може згладити прогнози, але може призвести до надмірного узагальнення. Якщо значення зависоке, модель може не помітити важливі локальні шаблони в даних зловмисного програмного забезпечення (рис.3).

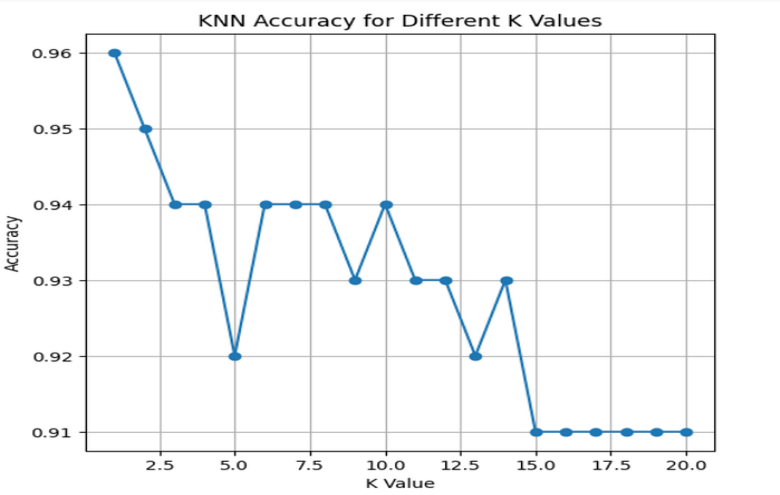

Рисунок 3 – Вплив значення k на точність алгоритму [16]

Переваги алгоритму k-NN:

– простота: k-NN легко реалізувати через те, наскільки він простий і точний. Як такий, він часто є одним із перших класифікаторів, які досліджує дані;

– можливість адаптації: щойно до набору даних додаються нові навчальні зразки, алгоритм k-NN коригує свої прогнози, щоб включити нові навчальні дані;

– легко програмується: k-NN вимагає лише кількох гіперпараметрів — значення ak і метрики відстані. Це робить його досить нескладним алгоритмом;

– алгоритм k-NN не потребує часу на навчання, оскільки він зберігає навчальні дані, а його обчислювальна потужність використовується лише під час прогнозування.

Проблеми та обмеження використання алгоритму k-NN, частково через його простоту:

– важко масштабувати: оскільки k-NN займає багато пам’яті та сховища даних, це призводить до витрат, пов’язаних із зберіганням; це означає, що алгоритм потребує обчислень, що, у свою чергу, потребує ресурсів;

– прокляття розмірності: це відноситься до явища, яке виникає в інформатиці, коли фіксований набір навчальних прикладів кидається викликом зростаючої кількості вимірів і невід’ємного збільшення значень функцій у цих вимірах. Іншими словами, навчальні дані моделі не можуть встигати за еволюцією розмірності гіперпростору. Це означає, що передбачення стають менш точними, оскільки відстань між точкою запиту та подібними точками стає ширшою — в інших вимірах;

– перенавчання: значення k, як показано раніше, впливатиме на поведінку алгоритму. Особливо це може статися, коли значення k занадто низьке. Нижчі значення k можуть перевищувати дані, тоді як вищі значення k «згладять» прогнозовані значення, оскільки алгоритм усереднює значення на більшій площі [17].

1.2.4 Алгоритм Support Vector Machine

Метод опорних векторів (Support Vector Machine, SVM) – це алгоритм контрольованого машинного навчання (з учителем), який використовується для задач класифікації та регресії. Основна мета методу – знайти найкращу межу, відому як гіперплощина, яка розділяє різні класи в даних [18, 19]. Наприклад, коли потрібно виконати бінарну класифікацію щодо спаму або не спаму [20-22]. Чим більша відмінність між двома класами, тим краще модель працює з новими та раніше невідомими даними (рис.4).

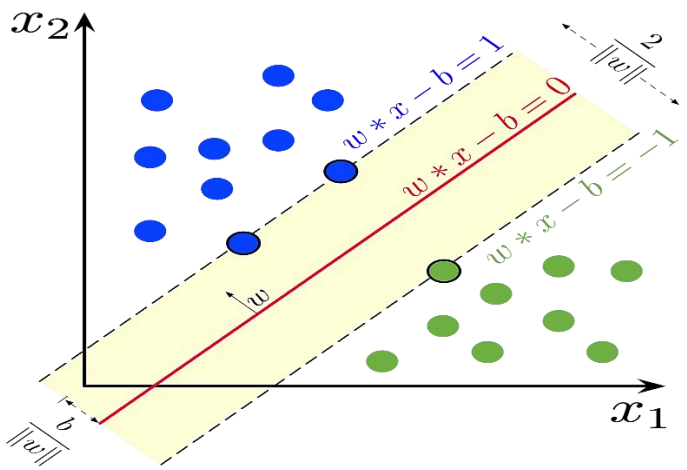

Рисунок 4 – Ілюстрація ідеї методу опорних векторів [19]

Гіперплощина (*червоним кольором*): це межа прийняття рішення, представлена рівнянням $w \cdot x - b = 0$. Вона розділяє два класи (зелена та синя точки). Алгоритм SVM шукає гіперплощину, яка найкраще розділяє дані, максимізуючи запас.

Опорні вектори: це точки, найближчі до гіперплощини з обох класів, виділені на граничних лініях $w \cdot x - b = 1$ та $w \cdot x - b = -1$. На рисунку це точки на пунктирних лініях, що представляють межу. Опорні вектори безпосередньо впливають на положення гіперплощини.

Запас (*жовтим кольором*): Запас — це відстань між опорними векторами та гіперплощиною. Метою методу опорних векторів є максимізація цього запасу, забезпечення того, щоб гіперплощина якомога чіткіше розділяла класи. На рисунку запас — це область між пунктирними лініями $w \cdot x - b = 1$ та $w \cdot x - b = -1$.

Вектор ваги *w*: Стрілка з позначкою *w* позначає вектор ваги, перпендикулярний до гіперплощини. Напрямок цього вектора вказує на орієнтацію гіперплощини, а його величина визначає крутість нахилу межі розділу.

Кожен із цих компонентів допомагає алгоритму SVM класифікувати точки даних за різними класами, забезпечуючи оптимальне розділення з максимальним запасом.

Переваги SVM-класифікаторів:

– висока точність: SVM пропонує чудову точність і добре працює з високорозмірними даними;

– нелінійні можливості: використання функцій ядра, таких як RBF та поліноміальний SVM, ефективно обробляє нелінійні залежності;

– стійкість до випадаючих значень: функція м’якої межі дозволяє SVM ігнорувати випадки, підвищуючи надійність виявлення спаму та аномалій;

– підтримка бінарних та багатокласових систем: SVM ефективний як для бінарної, так і для багатокласової класифікації, що підходить для застосувань у класифікації тексту.

– ефективне використання пам'яті: використовується підмножина навчальних точок, що зменшує споживання пам'яті.

Недоліки SVM-класифікаторів:

– високий час навчання: SVM може бути обчислювально ресурсоємним, що робить його непридатним для великих наборів даних;

– погана продуктивність із перекриваючими класами: SVM має проблеми, коли класи суттєво перекриваються;

– складність налаштування параметрів: вибір правильного ядра та налаштування параметрів, що впливає на точність алгоритму SVM;

– чутливість до шуму: SVM має проблеми з шумними наборами даних та перекриванням класів, що обмежує ефективність у реальних сценаріях;

– обмежена інтерпретація: складність гіперплощини у вищих вимірах робить SVM менш інтерпретованою, ніж інші моделі;

– чутливість масштабування ознак: правильне масштабування ознак є важливим, інакше моделі SVM можуть працювати погано.

1.2.5 Алгоритм Random Forest

Алгоритм випадкового лісу (Random Forest) створює ансамбль з кількох дерев рішень для досягнення єдиного, точнішого прогнозу або результату [23].

Алгоритми випадкового лісу мають три основні гіперпараметри, які необхідно встановити перед навчанням. До них належать розмір вузла, кількість дерев і кількість вибіркових ознак. Звідси класифікатор випадкового лісу можна використовувати для вирішення задач регресії або класифікації.

Робота алгоритму випадкового лісу (рис.5):

1. Створення багатьох дерев рішень: алгоритм створює багато дерев рішень, кожне з яких використовує випадкову частину даних. Тому кожне дерево дещо відрізняється.

2. Вибір випадкових ознак: під час побудови кожного дерева не розглядаються всі ознаки (стовпці) одночасно. Вибір випадковим чином передбачає вибір кількох ознак для розподілу даних. Це допомагає деревам відрізнятися одне від одного.

3. Кожне дерево робить прогноз: кожне дерево дає власну відповідь або прогноз на основі того, що воно дізналося зі своєї частини даних.

4. Об'єднання прогнозів: для класифікації обираємо категорію за голосуванням більшістю, отже, остаточна відповідь – це та, з якою «погоджується» більшість дерев.

Для регресії прогнозуємо число, оскільки кінцева відповідь є середнім значенням усіх прогнозів дерев.

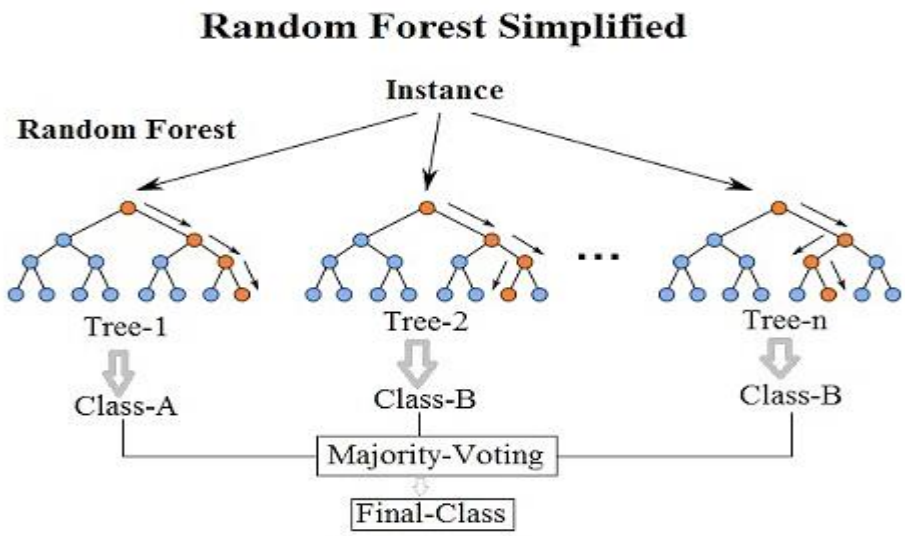

Рисунок 5 – Ілюстрація ідеї алгоритму випадкового лісу [24]

Переваги моделі випадкового лісу

– універсальність: можна використовувати як для задач регресії, так і для задач класифікації, а також легко побачити відносну важливість, яку він надає вхідним ознакам;

– легкі для розуміння гіперпараметри: гіперпараметри за замовчуванням, які алгоритм використовує, часто дають хороший результат прогнозування; розуміння гіперпараметрів досить просте, і їх також не так багато;

– запобігає перенавчанню моделі: одна з найбільших проблем машинного навчання — це перенавчання, але здебільшого цього не станеться завдяки класифікатору випадкового лісу; якщо в лісі достатньо дерев, класифікатор не перенавчатиме модель.

Недоліки моделі випадкового лісу:

– вища точність уповільнює модель: у випадкових лісах для точніших прогнозів потрібна більша кількість дерев, що може збільшити використання пам'яті та уповільнити модель. Хоча алгоритм загалом достатньо швидкий для більшості реальних застосувань, додавання занадто великої кількості дерев може зробити його занадто повільним для прогнозів у реальному часі;

– алгоритми випадкового лісу зазвичай швидко навчаються, але досить повільно генерують прогнози після навчання, що робить їх менш ефективними в ситуаціях, коли продуктивність під час виконання є критично важливою. У таких випадках може бути перевага альтернативним підходам;

– неможливо описати зв'язки всередині даних: випадковий ліс – це інструмент прогнозного моделювання, а не описового . Це означає, що він розроблений для прогнозування на основі закономірностей у даних, а не для пояснення взаємозв'язку між змінними. Якщо треба зрозуміти, як пов'язані різні фактори, інші підходи будуть кращими.

Висновок. Виявлення спаму в електронній пошті за допомогою машинного навчання пропонує надійне рішення для постійної проблеми небажаних повідомлень. Очищаючи та впорядковуючи дані, створюючи корисні функції та будуючи «розумні» моделі, можна створювати ефективні фільтри, які захищають електронні листи [25].

1.3 Приклад виконання ІДЗ№1

Розглянемо приклад класифікації повідомлення як любительський (ham) та спамовий (spam).

Приклад взято з джерела https://www.kaggle.com/code/karnikakapoor/spam-or-ham-sms-classifier/notebook

1. Імпорт бібліотек Phyton (рис.6).

```
#Importing all the libraries to be used
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from matplotlib.colors import ListedColormap
from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_rep
ort, accuracy_score, f1_score
        from sklearn import metrics
```

Рисунок 6 – Код для імпорту бібліотек

2 Завантаження даних (рис.7). Дата сет можна завантажити за посиланням https://www.kaggle.com/code/karnikakapoor/spam-or-ham-sms-classifier/input

```
#Loading data
data = pd.read_csv("../input/sms-spam-collection-dataset/spam.csv")
data.info()
```

Рисунок 7 – Код для завантаження даних

Отримуємо опис завантажених даних, результат на рис.8.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

Рисунок 8 – Результат щодо завантаженого дата сету

Видалення зайвих стовпців у дата сеті (рис.9).

```
# Dropping the redundent looking collumns (for this project)
to_drop = ["Unnamed: 2","Unnamed: 3","Unnamed: 4"]
data = data.drop(data[to_drop], axis=1)
# Renaming the columns because I feel fancy today
data.rename(columns = {"v1":"Target", "v2":"Text"}, inplace = True)
data.head()
```

Рисунок 9 – Видалення зайвих стовпців

Результат показано на рис.10.

	Target	Text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Рисунок 10 – Фрагмент даних після видалення стовпців

Набір даних складається з 5574 повідомлень англійською мовою. Дані позначаються як любительські або спамові. Dataframe має три стовпці. Перший стовпець – номер строки, другий - «Target» вказує на клас повідомлення як ham або spam, а третій стовпець «Text» – це рядок тексту.

3. Дослідження даних (Data exploration).

3.1.Перевірка збалансованості даних (рис.11). Здійснюється підрахунок кількості ham та spam повідомлень у датасеті.

```
#Palette
cols= ["#E1F16B", "#E598D8"]
#first of all let us evaluate the target and find out if our data is imbalanced or not
plt.figure(figsize=(12,8))
fg = sns.countplot(x= data["Target"], palette= cols)
fg.set_title("Count Plot of Classes", color="#58508d")
fg.set_xlabel("Classes", color="#58508d")
fg.set_ylabel("Number of Data points", color="#58508d")
```

Рисунок 11 – Виявлення збалансованості даних

Отримуємо звіт щодо структури даних (рис.12).

```
Text(0, 0.5, 'Number of Data points')
```

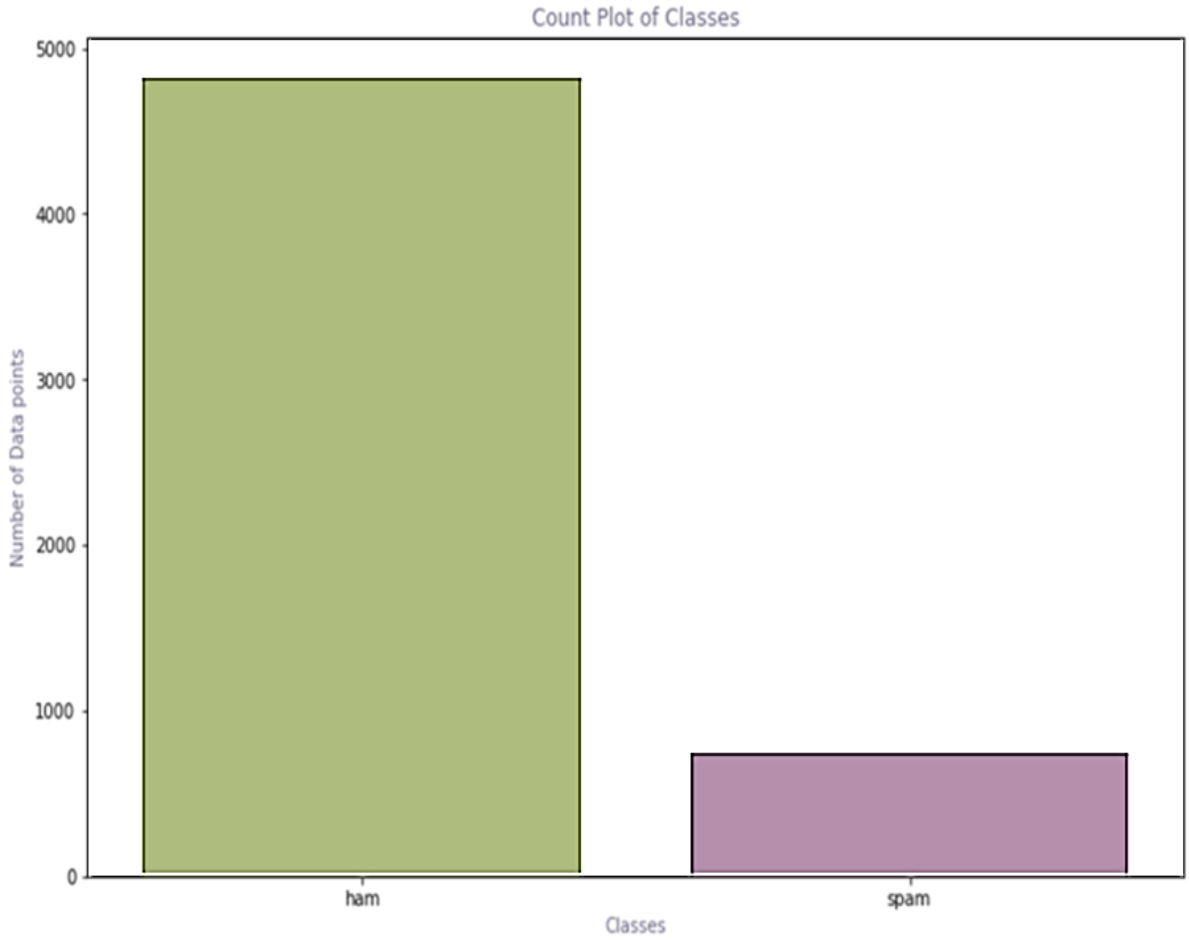

Рисунок 12 – Результати підрахунку кількості точок даних

На рисунку 12 наведено графік підрахунку кількості ham та spam, Робимо висновок щодо дисбалансу даних.

3.2. Інжиніринг ознак (Feature engineering). З метою дослідження даних створимо нові ознаки:

- No_of_Characters: кількість символів у текстовому повідомленні;
- No_of_Words: кількість слів у текстовому повідомленні;

– No_of_sentence: кількість речень у текстовому повідомленні.

Додамо стовпці з кількістю символів, слів та речень у кожному повідомленні (рис.13). Цей процес має назву токенизації.

```
#Adding a column of numbers of charachters,words and sentences in each msg
data["No_of_Characters"] = data["Text"].apply(len)
data["No_of_Words"]=data.apply(lambda row: nltk.word_tokenize(row["Text"]), axis=1).apply(len)
data["No_of_sentence"]=data.apply(lambda row: nltk.sent_tokenize(row["Text"]), axis=1).apply(len)

data.describe().T

#PS. At this step, I tokenised the words and sentences and used the length of the same.
#More on Tokenizing later in the notebook.
```

Рисунок 13 – Код для Feature engineering датасету

Токенізація це процес розділення фрази, речення, абзацу, одного або кількох текстових документів на менші одиниці . Кожна з цих менших одиниць називається токеном. Ці токени можуть бути будь-чим — словом, підсловом або навіть символом [11, 25]. Результати Feature engineering датасету надано на рис.14.

	count	mean	std	min	25%	50%	75%	max
No_of_Characters	5572.0	80.058327	59.623937	2.0	36.0	61.0	121.0	910.0
No_of_Words	5572.0	18.502692	13.638372	1.0	9.0	15.0	27.0	219.0
No_of_sentence	5572.0	1.993001	1.503584	1.0	1.0	2.0	2.0	38.0

Рисунок 14 – Результат Feature engineering датасету

Візуалізуємо ці результати (рис.15).

```
plt.figure(figsize=(12,8))
fg = sns.pairplot(data=data, hue="Target",palette=cols)
plt.show(fg)
```

Рисунок 15 – Код для побудови графіків

Результати візуалізації датасету надано на рис.16. На графіках рис.16 можна побачити кілька викидів у класі ham. Оскільки вони по суті вказують на те саме, тобто довжину SMS, то далі треба видалити ці викиди.

```
<Figure size 864x576 with 0 Axes>
```

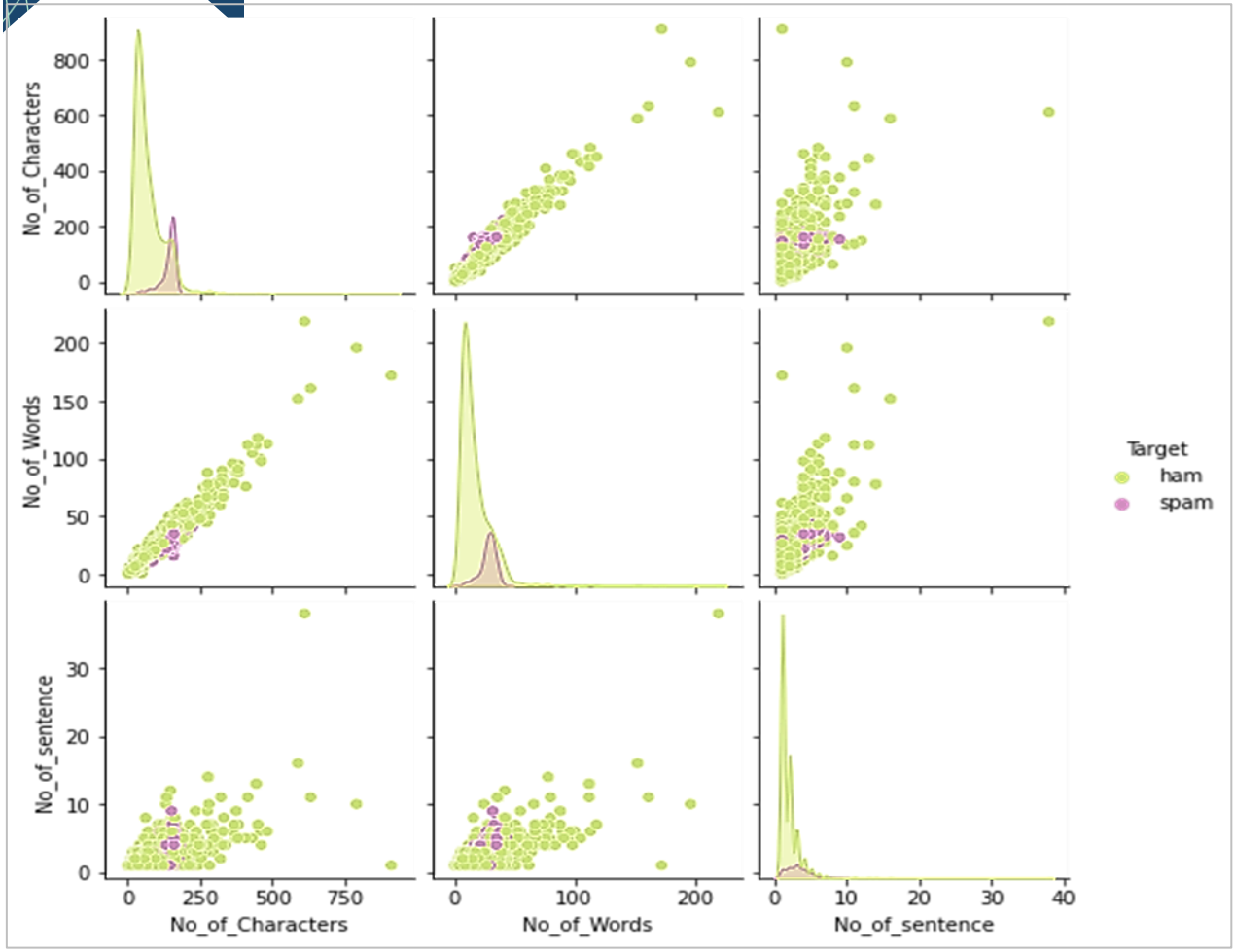

Рисунок 16 – Візуалізація датасету

3.3. Виявлення викидів (Outlier detection). Для видалення викидів використано код на рис.17 та результат його виконання.

```
#Dropping the outliers.
data = data[(data["No_of_Characters"]<350)]
data.shape

(5548, 5)
```

Рисунок 17 – Код для Outlier detection та результат

Візуальне представлення (рис.18) виправлених даних без викидів надано на рис.19.

```
plt.figure(figsize=(12,8))
fg = sns.pairplot(data=data, hue="Target",palette=cols)
plt.show(fg)
```

Рисунок 18 – Код для візуалізації виправленого датасету

```
<Figure size 864x576 with 0 Axes>
```

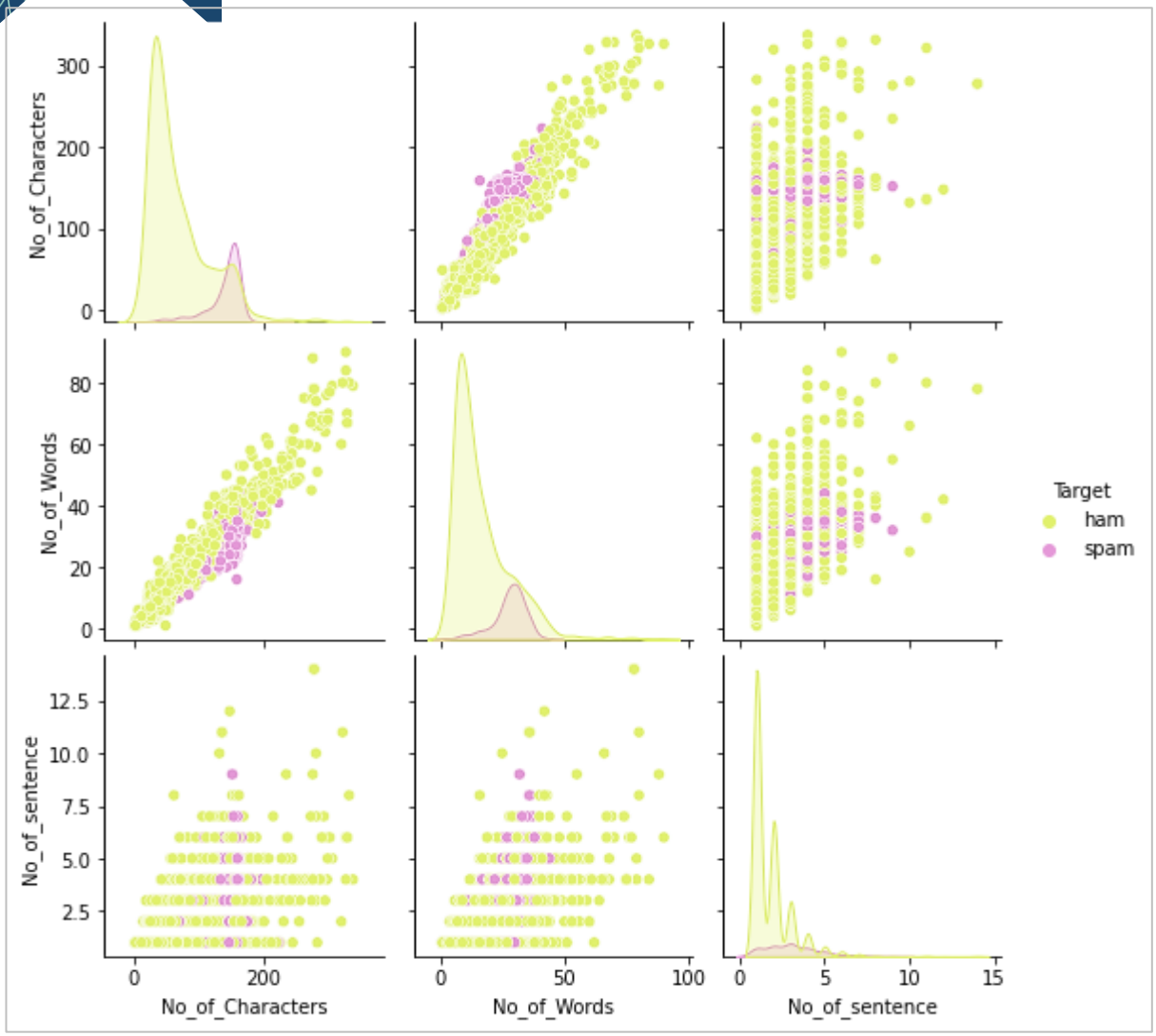

Рисунок 19 – Візуалізація виправлених даних датасету

4. Попередня обробка даних.

4.1. Очищення тексту. Процес очищення даних NLP (Neuro-Linguistic Programming) має вирішальне значення. Обробка природної мови (NLP) – галузь інформатики, яка дозволяє машинам розуміти, інтерпретувати та генерувати людську мову.

Комп'ютер не розуміє текст, для комп'ютера це просто кластер символів. Для подальшої обробки даних потрібно зробити дані чистішими.

На першому кроці вилучаються лише букви, при цьому видаляються знаки пунктуації та цифри.

На наступному кроці треба перетворити усі символи на малі літери.

Цей текст потім буде використано у подальшій обробці.

Спочатку представимо зразок текстів перед чищенням (рис.20, рис.21).

```
#Lets have a look at a sample of texts before cleaning
print("\033[1m\u001b[45;1m The First 5 Texts:\033[0m",*data["Text"][:5], sep = "\n"
```

Рисунок 20 – Вибір тексту

```
The First 5 Texts:
Go until jurong point, crazy.. Available only in bugis n great world la e
buffet... Cine there got amore wat...
Ok lar... Joking wif u oni...
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text F
A to 87121 to receive entry question(std txt rate)T&C's apply 08452810075o
ver18's
U dun say so early hor... U c already then say...
Nah I don't think he goes to usf, he lives around here though
```

Рисунок 21 – Вибраний текст

Визначимо функції для очищення тексту - заміна всіх неалфавітних символів пробілом (рис.22), та застосуємо її для обраного тексту. На рис.23 надано текст після очищення.

```
# Defining a function to clean up the text
def Clean(Text):
    sms = re.sub('[^a-zA-Z]', ' ', Text) #Replacing all non-alphabetic characters with a
space
    sms = sms.lower() #converting to lowecase
    sms = sms.split()
    sms = ' '.join(sms)
    return sms

data["Clean_Text"] = data["Text"].apply(Clean)
#Lets have a look at a sample of texts after cleaning
print("\033[1m\u001b[45;1m The First 5 Texts after cleaning:\033[0m",*data["Clean_Te
xt"][:5], sep = "\n")
```

Рисунок 22 – Код для очищення тексту

```
The First 5 Texts after cleaning:
go until jurong point crazy available only in bugis n great worl
d la e buffet cine there got amore wat
ok lar joking wif u oni
free entry in a wkly comp to win fa cup final tkts st may text f
a to to receive entry question std txt rate t c s apply over s
u dun say so early hor u c already then say
nah i don t think he goes to usf he lives around here though
```

Рисунок 23 – Текст після очищення

4.2. Токенізація. Токенізація розбиває складні дані на менші одиниці, які називаються ***токенами***. Це можна зробити, розділивши абзаци на речення, а речення на слова. На цьому кроці розбиваємо очищені дані *Clean_Text* на слова (рис. 24) та отримуємо новий датасет – нові слова (рис.25).

```
data["Tokenize_Text"]=data.apply(lambda row: nltk.word_tokenize(row["Clean_Text"]),
axis=1)

print("\033[1m\u001b[45;1m The First 5 Texts after Tokenizing:\033[0m",*data["Token
ize_Text"][:5], sep = "\n")
```

Рисунок 24 – Код для токенізації

```
The First 5 Texts after Tokenizing:
['go', 'until', 'jurong', 'point', 'crazy', 'available', 'only', 'in', 'bu
gis', 'n', 'great', 'world', 'la', 'e', 'buffet', 'cine', 'there', 'got',
'amore', 'wat']
['ok', 'lar', 'joking', 'wif', 'u', 'oni']
['free', 'entry', 'in', 'a', 'wkly', 'comp', 'to', 'win', 'fa', 'cup', 'fi
nal', 'tkts', 'st', 'may', 'text', 'fa', 'to', 'to', 'receive', 'entry', '
question', 'std', 'txt', 'rate', 't', 'c', 's', 'apply', 'over', 's']
['u', 'dun', 'say', 'so', 'early', 'hor', 'u', 'c', 'already', 'then', 'sa
y']
['nah', 'i', 'don', 't', 'think', 'he', 'goes', 'to', 'usf', 'he', 'lives'
, 'around', 'here', 'though']
```

Рисунок 25 – Текст після токенізації

4.3. Видалення стоп-слова. Стоп-слова – це слова, які часто зустрічаються (наприклад, *few, is, an* тощо). Ці слова зберігають значення в структурі речення, але не роблять значного внеску в обробку мови в NLP. З метою усунення надмірності у обробці їх можна видалити. У бібліотеці NLTK є набір стоп-слів за замовчуванням, які будемо видаляти (рис.26). Результат надано на рис.27.

```
# Removing the stopwords function
def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    filtered_text = [word for word in text if word not in stop_words]
    return filtered_text

data["Nostopword_Text"] = data["Tokenize_Text"].apply(remove_stopwords)

print("\033[1m\u001b[45;1m The First 5 Texts after removing the stopwords:\033[0m",
*data["Nostopword_Text"][:5], sep = "\n")
```

Рисунок 26 – Код для видалення стоп-слів

```
The First 5 Texts after removing the stopwords:
['go', 'jurong', 'point', 'crazy', 'available', 'bugis', 'n', 'great',
world', 'la', 'e', 'buffet', 'cine', 'got', 'amore', 'wat']
['ok', 'lar', 'joking', 'wif', 'u', 'oni']
['free', 'entry', 'wkly', 'comp', 'win', 'fa', 'cup', 'final', 'tkts',
st', 'may', 'text', 'fa', 'receive', 'entry', 'question', 'std', 'txt',
rate', 'c', 'apply']
['u', 'dun', 'say', 'early', 'hor', 'u', 'c', 'already', 'say']
['nah', 'think', 'goes', 'usf', 'lives', 'around', 'though']
```

Рисунок 27 – Текст після видалення стоп-слів

4.4. Лемматизація (lemmatization). Створення кореня (лемматизація) - це процес отримання кореневої форми слова. Основа або корінь – це частина, до якої додаються флективні афікси. Основа слова утворюється шляхом видалення префікса чи суфікса слова. Це сходить до етимології слова. Мови розвиваються з часом. Багато різних мов розгалужуються одна в одну; наприклад, англійська є похідною від латини. Отже, утворення кореня слова повертає його до кореня. Лемматизація також перетворює слово на його кореневу форму. Однак різниця полягає у тому, що лемматизація гарантує приналежність кореня слова до мови, у цьому випадку це англійська. Якщо використовуємо лематизацію (рис. 28), то вихід буде англійською мовою (рис.28).

```
lemmatizer = WordNetLemmatizer()
# lemmatize string
def lemmatize_word(text):
        #word_tokens = word_tokenize(text)
        # provide context i.e. part-of-speech
        lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in text]
        return lemmas

data["Lemmatized_Text"] = data["Nostopword_Text"].apply(lemmatize_word)
print("\033[1m\u001b[45;1m The First 5 Texts after lemitization:\033[0m",*data["Lem
matized_Text"][:5], sep = "\n")
```

Рисунок 28 – Код для лемматизації

```
The First 5 Texts after lemitization:
['go', 'jurong', 'point', 'crazy', 'available', 'bugis', 'n', 'great',
'world', 'la', 'e', 'buffet', 'cine', 'get', 'amore', 'wat']
['ok', 'lar', 'joke', 'wif', 'u', 'oni']
['free', 'entry', 'wkly', 'comp', 'win', 'fa', 'cup', 'final', 'tkts',
'st', 'may', 'text', 'fa', 'receive', 'entry', 'question', 'std', 'txt
', 'rate', 'c', 'apply']
['u', 'dun', 'say', 'early', 'hor', 'u', 'c', 'already', 'say']
['nah', 'think', 'go', 'usf', 'live', 'around', 'though']
```

Рисунок 29 – Текст після лемматизації

5. Векторизація. Векторизація в NLP перетворює текст на числові представлення (вектори), які можуть обробляти моделі машинного навчання, що дозволяє виконувати такі завдання, як класифікація та аналіз настроїв До поширених методів належать Bag-of-Words (BoW), який підраховує кількість слів; TF-IDF, який зважує частоту слова порівняно з його присутністю в загальному корпусі; а також Word Embeddings (наприклад, Word2Vec, GloVe) та моделі трансформаторів, такі як BERT , які фіксують семантичне значення за допомогою щільних, контекстно-залежних представлень

TF-IDF у NLP означає Term Frequency – Inverse document frequency. У NLP очищені дані необхідно перетворити на числовий формат, де кожне слово представлене матрицею. Це також відоме як вбудовування слів або векторизація слів.

Term Frequency (TF) = (Частота терміну в документі)/(Загальна кількість термінів у документах)

Inverse Document Frequency (IDF) = log((загальна кількість документів)/(кількість документів з терміном t))

У цьому прикладі використано TfidfVectorizer() для векторизації попередньо оброблених даних

Кроки векторизації:

- створення корпусу лематизованого тексту;
- перетворення корпусу на векторну форму;
- кодування міток класів у Target.

Примітка: досі ми вистежували стовпці у датасеті, далі будемо працювати з рядками.

Код для створення корпусу текстових об'єктів для подальшого кодування у векторизовану форму наведено на рис.30, а результати – на рис.31.

```
#Creating a corpus of text feature to encode further into vectorized form
corpus= []
for i in data["Lemmatized_Text"]:
      msg = ''.join([row for row in i])
      corpus.append(msg)

corpus[:5]
print("\033[1m\u001b[45;1m The First 5 lines in corpus :\033[0m",*corpus[:5], sep =
"\n")
```

Рисунок 30 – Код для створення текстових об'єктів

```
The First 5 lines in corpus :
go jurong point crazy available bugis n great world la e buffet
cine get amore wat
ok lar joke wif u oni
free entry wkly comp win fa cup final tkts st may text fa receiv
e entry question std txt rate c apply
u dun say early hor u c already say
nah think go usf live around though
```

Рисунок 31 – Текстові об’єкти для векторізації

Код для перетворення текстових даних на числа наведено на рис.32, а результат буде такий: dtype('float64').

```
#Changing text data in to numbers.
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(corpus).toarray()
#Let's have a look at our feature
X.dtype
```

Рисунок 32 – Код для перетворення текстових даних на числа

Для перетворення категоріальних міток у числові представлення у Python використовується утиліта LabelEncoder з модуля sklearn.preprocessing (рис.33).

```
#Label encode the Target and use it as y
label_encoder = LabelEncoder()
data["Target"] = label_encoder.fit_transform(data["Target"])
```

Рисунок 33 – Код для перетворення категоріальних міток у числове представлення

6. Побудова моделі.

6.1. Налаштування ознак та мітки (*Target*) як X та Y (рис.33).

```
#Setting values for labels and feature as y and X(we already did X in vectorizing...)
y = data["Target"]
```

Рисунок 34 – Код для визначення *Target*

6.2. Поділ тестового та навчального наборів (рис.35).

```
# Splitting the testing and training sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Рисунок 35 – Код для визначення навчального та тестового датасету

Функція *train_test_split* в Python є ключовим інструментом машинного навчання, який в основному використовується для

розділення набору даних на дві підмножини: навчальний набір та тестовий набір. Ця функція є частиною модуля sklearn.model_selection в бібліотеці Scikit-learn.

6.3. Створення моделі конвеєра для чотирьох різних класифікаторів. Налаштування усіх моделей до навчальних даних (рис.36):

1) Naïve Bayes;
2) RandomForestClassifier;
3) KNeighborsClassifier;
4) Support Vector Machines.

```
#Testing on the following classifiers
classifiers  =  [MultinomialNB(),
                 RandomForestClassifier(),
                 KNeighborsClassifier(),
                 SVC()]
for  cls  in  classifiers:
      cls.fit(X_train,  y_train)

# Dictionary of pipelines and model types for ease of reference
pipe_dict  =  {0:  "NaiveBayes",  1:  "RandomForest",  2:  "KNeighbours",3:  "SVC"}
```

Рисунок 36 – Код застосування чотирьох класифікаторів

6.4. Отримання перехресної перевірки на навчальному наборі для усіх моделей для досягнення необхідної точності (рис.37).

```
# Cossvalidation
for  i,  model  in  enumerate(classifiers):
      cv_score  =  cross_val_score(model,  X_train,y_train,scoring="accuracy",
cv=10)
      print("%s: %f "  %  (pipe_dict[i],  cv_score.mean()))
```

Рисунок 37 – Код перехресної перевірки

Функція cross_val_score в Python з модуля sklearn.model_selection є зручним інструментом для виконання перехресної перевірки K-Fold моделі машинного навчання. Вона забезпечує надійну оцінку продуктивності моделі шляхом навчання та оцінки моделі на кількох підмножинах даних.

Призначення: cross_val_score має на меті оцінити, наскільки добре модель узагальнюється на невидимі дані, зменшуючи ризик перенавчання та забезпечуючи надійніший показник продуктивності, ніж одинарний поділ навчального тесту. Результати перевірки наведено на рис.38.

```
NaiveBayes: 0.967552
RandomForest: 0.974537
KNeighbours: 0.911450
SVC: 0.974086
```

Рисунок 38 – Результати перевірки результатів класифікації

7. Оцінка моделі

У результаті тестування моделей на тестовому датасеті:

1) отримується звіт про точність;
2) будуються матриці плутанини.

Для звітності про точність можна реалізувати код, наведений на рис. 39.

```
# Model Evaluation
# creating lists of varios scores
precision =[]
recall =[]
f1_score = []
trainset_accuracy = []
testset_accuracy = []

for i in classifiers:
    pred_train = i.predict(X_train)
    pred_test = i.predict(X_test)
    prec = metrics.precision_score(y_test, pred_test)
    recal = metrics.recall_score(y_test, pred_test)
    f1_s = metrics.f1_score(y_test, pred_test)
    train_accuracy = model.score(X_train,y_train)
    test_accuracy = model.score(X_test,y_test)

    #Appending scores
    precision.append(prec)
    recall.append(recal)
    f1_score.append(f1_s)
    trainset_accuracy.append(train_accuracy)
    testset_accuracy.append(test_accuracy)
```

```
# initialise data of lists.
data = {'Precision':precision,
'Recall':recall,
'F1score':f1_score,
'Accuracy on Testset':testset_accuracy,
'Accuracy on Trainset':trainset_accuracy}
# Creates pandas DataFrame.
Results = pd.DataFrame(data, index =["NaiveBayes", "RandomForest", "KNeighbours",
"SVC"])
```

Рисунок 39 – Код для оцінювання моделей

У Python метод *.predict()* зазвичай використовується з навченими моделями машинного навчання для створення прогнозів на основі нових, невідомих даних.

У Python метрика precision_score — це метрика класифікації, яка використовується для оцінки продуктивності моделі класифікації, зокрема в бінарних або багатокласових завданнях класифікації. Вона доступна в sklearn.metricsмодулі бібліотеки scikit-learn.

Функція *f1_score* з *sklearn.metrics* обчислює F1-оцінку , яка представляє собою гармонійне середнє точності та повноти, для завдань класифікації.

F1-оцінка – це показник продуктивності, який використовується в машинному навчанні для оцінки ефективності моделі класифікації на наборі даних, особливо коли класи незбалансовані, тобто один клас з'являється набагато частіше, ніж інший. Це гармонійне середнє значення ***точності*** (Precision) та ***повноти*** *(Recall)*, яке об'єднує обидва показники в одне значення, що збалансовує їхню важливість.

Точність (Precision) - це частки правильних позитивних прогнозів (істинно позитивних результатів) від усіх позитивних прогнозів, зроблених моделлю (істинно позитивні результати + хибно позитивні результати).

Це міра точності позитивних прогнозів. Формула для точності така:

Precision = True Positives / (True Positives + False Positives)

Показник Повнота *(Recall)*, який також відомий як чутливість або позитивний відсоток, і він вимірює частку фактичних позитивних випадків, правильно ідентифікованих моделлю.

Це відношення числа істинно позитивних випадків до загального числа фактичних позитивних випадків (справді позитивні випадки + помилково негативні випадки). Формула для *Recall*:

Recall = True Positives / (True Positives + False Negatives)

Показник F1Score поєднує точність та повноту, використовуючи таку формулу:

F1 Score=2×(Precision×Recall / (Precision +Recall).

Для виводу результатів можна застосувати код на рис. 40 та 41.

```
cmap2 = ListedColormap(["#E2CCFF","#E598D8"])
Results.style.background_gradient(cmap=cmap2)
```

Рисунок 40 – Код для виводу оцінок моделей

```
cmap = ListedColormap(["#E1F16B", "#E598D8"])
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

for cls, ax in zip(classifiers, axes.flatten()):
    plot_confusion_matrix(cls,
                          X_test,
                          y_test,
                          ax=ax,
                          cmap= cmap,
                          )
    ax.title.set_text(type(cls).__name__)
plt.tight_layout()
plt.show()
```

Рисунок 41 – Код для виводу матриці плутанини

Для відображення матриці плутанини в Python, зокрема для завдань класифікації машинного навчання, scikit-learn бібліотека зазвичай використовується разом з matplotlib візуалізацією.

Старі версії scikit-learn включали функцію plot_confusion_matrix (рис.40).

Рекомендовано у цій лабораторній використоввати функцію *ConfusionMatrixDisplay.from_estimator* або функцію *ConfusionMatrixDisplay.from_predictions, див. посилання* [https://scikit-learn.org/0.24/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html?highlight=confusionmatrixdisplay%20from_estimator].

У результаті на екран виводится матриця плутанини, аналітик аналізує її та обирає найкращий метод, керуючись обраними метриками якості.

1.4 Оформлення звіту з виконання ІДЗ № 1

Після виконання завдань ІДЗ №1 студентом складається звіт за результатами.

Звіт повинен містити:

1) титульний аркуш (приклад у додатку А);
2) мета та завдання ІДЗ №1;
3) опис задачі фільтрації спаму та короткий опис методів її розв’язання (методів класифікації);
4) скріни результатів реалізації розробленого ПЗ для фільтрації спаму з поясненнями;
5) аналіз отриманих результатів з обґрунтованим висновком щодо «найкращого» методу для фільтрації спаму;
6) висновки за результатами виконання ІДЗ №1 (що було вивчено та які отримані практичні навички).

2. ПОДАННЯ НА ОЦІНЮВАННЯ ТА ЗАХИСТ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

2.1 Оформлення, подання і захист індивідуального завдання

Після виконання ІДЗ№1 студент підготовлює звіт як есе у форматі *docx. Потім розміщує цей звіт у відповідному розділі дисципліни в Moodle у форматі, *docx або *pdf.

Звіт виконують державною мовою, викладається матеріал науковим стилем.

До тексту звіту висуваються такі вимоги: формат сторінки А4 (210х297мм) з полями: верхнє та нижнє поле– 2 см, праве – 1,5 см; ліве – 3 см, абзац – 1,27, шрифт Arial, 14 кегль, 1,5 інтервал.

Якщо є посилання у тексті на джерела, то вони зазначаються порядковим номером за текстом у квадратних дужках, наприклад, «... у роботі [1] ...». Список використаних джерел оформлюється за Національним стандартом України «Інформація та документація. Бібліографічна посилання. Загальні положення та правила складання. ДСТУ 8302:2015» .

Захист ІДЗ №1 здійснюється студентом на практичному занятті згідно з графіком контрольних точок, який надано у робочій програмі дисципліни «Машинне навчання у кібербезпеці» як вибіркова дисципліна ОПП «Комп’ютерні науки».

Оцінка за виконання ІДЗ №1 виставляється викладачем у відповідній активності системи Moodl, враховуючи поточну успішність студента.

2.2 Критерії оцінювання результатів виконання ІДЗ

Максимальна кількість балів, яку здобувач може отримати за ІДЗ №1 – 30 балів. Враховується виконання ІДЗ (максимальний бал 15) та захист його на практичному занятті (максимальний бал 15). Оскарження оцінки може бути здійснене на останньому практичному занятті модуля.

Критерії оцінювання:

– студент підготував роботу відповідно до поставленого завдання, в якій: правильно визначив задачу, методи, які необхідно застосувати для її розв’язання, обґрунтував своє бачення теоретичними концепціями або моделями, виконав необхідну розробку ПЗ, надав результати роботи ПЗ та провів їх аналіз у разі потреби, представив висновок або власне бачення розв’язання задачі і окреслив можливі перспективи використання отриманих результатів, обмеженість отриманого рішення, навів за необхідності обґрунтування використання інших методів; робота структурована, викладена діловим, науковим або діловим стилем (10 балів);

– робота містить комплексну, логічну і оригінальну пропозицію щодо використання методів розв'язання задачі індивідуального завдання аж до міждисциплінарного підходу; використання штучного інтелекту (ШІ) не забороняється, оскільки пропозиції відомих застосунків ШІ суттєво залежать від обміркованої постановки питання і уточнюючих питань; однак у разі, якщо відповідь, отримана з використанням ШІ, не є комплексною або не відповідає за стилем і викладеними позиціями іншим частинам роботи або завдання, містить очевидно неправдиву інформацію, то оцінка за цим критерієм знижується (5 балів);

– студент під час презентації / захисту свого індивідуального завдання та контрольної роботи у вигляді есе демонструє володіння термінологічним апаратом, методами машинного навчання, проєктування та розробки ПЗ, відповідає на запитання, здатний швидко адаптувати моделі, методи під зміни в умовах задачі (10 бали);

– студент дав пряму і релевантну відповідь на поставлене питання з використанням обґрунтованого посилання на теоретичний матеріал демонструє володіння термінологічним апаратом, методами проєктування ПЗ та методів машинного навчання, здатний адаптувати моделі під зміни завдання, у т.ч. у вигляді додаткових запитань / зміг стисло формалізувати вербально сутність задачі кібербезпеки, визначити ключові складові виконання практичної роботи, критерії якості отриманих результатів (5 бали).

Додаткові зауваження:

- студент може оскаржити отримані оцінки в порядку, передбаченому Положенням про організацію освітнього процесу (Нормативні документи : Polytechnic (metinvest.university)) та Положенням про політику та процедури врегулювання конфліктних ситуацій (Академічні політики : Polytechnic (metinvest.university));
- викладач не має права знижувати оцінку за індивідуальне завдання, якщо воно не було складено вчасно, однак у разі, якщо така робота була оцінена пізніше, ніж момент завершення теоретичного навчання у семестрі, то відповідна оцінка не враховується у рейтингу здобувачів освіти.

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Робоча програма навчальної дисципліни «Проєктування та розробка систем цифрового інтелекту» / уклад. В. В. Москаленко. Запоріжжя : ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2024. 13 с.
2. Положення про організацію освітнього процесу у ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА». URL: https://metinvest.university/data/file/14/86/148623526b844f999bad7c51257baa51.pdf. (дата звернення: 27.06.2025).
3. Положення про визнання у ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА» результатів навчання, набутих у неформальній/інформальній освіті. URL: https://metinvest.university/data/file/eb/ef/ebef5aa7e673444797bf73d46e33615b.pdf. (дата звернення: 27.06.2025).
4. Положення про академічну доброчесність здобувачів вищої освіти та науково-педагогічних працівників ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА». URL: https://metinvest.university/data/file/c1/c9/c1c998364cec4bdbb42478109c72e17c.pdf. (дата звернення: 27.06.2025).
5. Naive Bayes Classifiers. URL: https://www.geeksforgeeks.org/machine-learning/naive-bayes-classifiers/ (дата звернення: 27.06.2025).
6. Naive Bayes Classifier (NB). URL: https://medium.com/@akshayc123/naive-bayes-classifier-nb-7429a1bdb2c0 (дата звернення: 27.06.2025).
7. Naive Bayes. URL: https://scikit-learn.org/stable/modules/naive_bayes.html (дата звернення: 27.06.2025).
8. Bernoulli Naive Bayes, Explained: A Visual Guide with Code Examples for Beginners. URL: https://towardsdatascience.com/bernoulli-naive-bayes-explained-a-visual-guide-with-code-examples-for-beginners-aec39771ddd6/ (дата звернення: 27.06.2025).
9. Naive Bayes Classifier in Python. URL: https://www.kaggle.com/code/prashant111/naive-bayes-classifier-in-python (дата звернення: 27.06.2025).
10. Kaggle Best Notebooks — Topic wise (Data Science and Machine Learning) URL: https://armmoon4.medium.com/kaggle-best-notebooks-topic-wise-data-science-and-machine-learning-e475cccfee37 (дата звернення: 27.06.2025).
11. Build a Spam Filter. URL: https://new.pythonforengineers.com/blog/build-a-spam-filter/ (дата звернення: 27.06.2025).
12. Spam Filter using Naive Bayes Classifier URL: https://www.kaggle.com/code/jeffysonar/spam-filter-using-naive-bayes-classifier

13. Bayesian Spam Filter. URL: https://github.com/aashishsatya/Bayesian-Spam-Filter (дата звернення: 27.06.2025).
14. Detecting Spam Emails Using Tensorflow in Python. URL:https://www.geeksforgeeks.org/detecting-spam-emails-using-tensorflow-in-python/ (дата звернення: 27.06.2025).
15. Build a Spam Filter. URL: https://new.pythonforengineers.com/blog/build-a-spam-filter/ (дата звернення: 27.06.2025).
16. ML-4: K-Nearest Neighbors Algorithm in Cybersecurity: Detecting Malware. URL: https://medium.com/@amuldhungel01/ml-4-k-nearest-neighbors-algorithm-in-cybersecurity-detecting-malware-d559a7ed3822 (дата звернення: 27.06.2025).
17. What is kNN? URL: https://www.elastic.co/what-is/knn (дата звернення: 27.06.2025).
18. Support Vector Machine (SVM) Algorithm. URL: https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/ (дата звернення: 27.06.2025).
19. Support Vector Machines (SVM) URL: https://medium.com/@RobuRishabh/support-vector-machines-svm-27cd45b74fbb (дата звернення: 27.06.2025).
20. Spam Filtering using Support Vector Machine. URL: https://scispace.com/pdf/spam-filtering-using-support-vector-machine-54k3zfa7hm.pdf (дата звернення: 27.06.2025).
21. Cyber Security Approach in Web Application using SVM. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ff4dc7b8580220a9895efc59d27f2dc0269bfb5f#:~:text=SVM%20(Support%20Vector%20Machine)%20is,problem%20without%20loss%20of%20generality (дата звернення: 27.06.2025).
22. Improved Support Vector Machine for Cyber Attack Detection. URL: https://www.iaeng.org/publication/WCECS2011/WCECS2011_pp394-399.pdf (дата звернення: 27.06.2025).
23. Random Forest: A Complete Guide for Machine Learning. URL: https://builtin.com/data-science/random-forest-algorithm(дата звернення: 27.06.2025).
24. Silver BlogRandom Forests®, Explained. URL: https://www.kdnuggets.com/2017/10/random-forests-explained.html (дата звернення: 27.06.2025).
25. Email Spam Detection with Machine Learning: A Comprehensive Guide. URL: https://medium.com/@azimkhan8018/email-spam-detection-with-machine-learning-a-comprehensive-guide-b65c6936678b (дата звернення: 27.06.2025).

Додаток А. Приклад оформлення титульного аркушу

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
Факультет автоматизації виробництва та цифрових технологій
Кафедра цифрових технологій та проєктно-аналітичних рішень

ЗВІТ
З ІНДИВІДУАЛЬНОГО ЗАВДАННЯ №1

з дисципліни: «МАШИННЕ НАВЧАННЯ У КІБЕРБЕЗПЕЦІ»

на тему: «Використання методів Supervised learning для задачі фільтрації спаму»

Роботу виконав Студент групи _____
_____
Валентина
Роботу перевірив МОСКАЛЕНКО

ЗАПОРІЖЖЯ 2025

Навчально-методичне видання

Валентина Володимирівна Москаленко

МАШИННЕ НАВЧАННЯ У КІБЕРБЕЗПЕЦІ

методичні рекомендації
до виконання індивідуального завдання №1
«Використання методів Supervised learning для задачі фільтрації спаму»

Самостійне електронне мережеве видання

Публікується в авторський редакції