Київський національний університет імені Тараса Шевченка

Звіт
з дисципліни
«Штучний Інтелект»
до лабораторної роботи
«Ідентифікатор спаму»

ПОСТАНОВКА ЗАДАЧІ

Побудувати ідентифікатор спаму, використовуючи моделі машинного навчання (**Naive Bayes classifier, K-Nearest Neighbors algorithm, Decision Tree learning, Support Vector Machine (SVM), Random Forest**) за допомогою бібліотеки scikit-learn.

Зробити висновки та порівняльну характеристику наведених методів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Naive Bayes classifier (Наївний баєсів класифікатор)

Наївний баєсів класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному) незалежності змінних.

У теорії ймовірностей дві випадкові події називаються **незалежними**, якщо настання однієї з них не змінює вірогідність настання іншої. Аналогічно, дві випадкові величини називають **незалежними** якщо значення однієї з них не впливає на розподіл значень іншої.

Вважатимемо, що дано фіксований ймовірнісний простір $(\Omega,\mathbb{F},\mathrm{P})$.

Означення. Дві події $A, B \in \mathbb{F}$ називають **незалежними**, якщо

$$\mathrm{P}(A \cap B) = \mathrm{P}(A) \cdot \mathrm{P}(B).$$

Зауваження. В тому випадку, якщо ймовірність однієї події, скажемо B ненульова, тобто $P(B)>0$, визначення незалежності еквівалентне:

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

тобто умовна ймовірність події A за умови B дорівнює безумовній вірогідності події A.

Теорема Баєса. Теорема Баєса задається математично наступним рівнянням

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

де A та B є подіями.

$\mathrm{P}(A)$ — апріорна ймовірність гіпотези A.

$\mathrm{P}(A|B)$ — ймовірність гіпотези A, якщо настала подія B (апостеріорна ймовірність).

$\mathrm{P}(B|A)$ — ймовірність настання події B, якщо гіпотеза A істина.

$\mathrm{P}(B)$ — повна ймовірність настання події B.

Тобто, якщо на основі значень змінних можна однозначно визначити, до якого класу належить спостереження, байєсів класифікатор повідомить ймовірність приналежності до цього класу.

У проміжних же випадках, коли спостереження може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде вектор, компоненти якого є ймовірностями приналежності до того чи іншого класу.

Баєсова фільтрація спаму

Баєсова фільтрація спаму (англ. *Naive Bayes spam filtering*) — метод для фільтрації спаму, заснований на застосуванні наївного баєсова класифікатора, що спирається на пряме використання теореми Баєса.

Під час навчання фільтру для кожного слова в тексті вираховують та зберігають його **«вагу»** — оцінку ймовірності того, що текст із цим словом — спам. У найпростішому випадку як оцінку використовують частоту: «появ в спамі / появ всього». У складніших випадках можлива попередня обробка тексту: приведення слів до початкової форми, видалення службових слів, обчислення «ваги» для цілих фраз, транслітерація тощо.

Під час перевірки нового тексту ймовірність «спаму» обчислюють за вказаною вище формулою для множини гіпотез. В даному випадку «гіпотези» — це слова, і для кожного слова «достовірність гіпотези»

$$\mathrm{P}(A_i) = \frac{N_{word_i}}{N_{words\ total}} \text{ — частка цього слова в тексті,}$$

а «залежність події від гіпотези» $\mathrm{P}(B|A_i)$ — обчислена раніше «вага» слова.

Тобто «вага» тексту в даному випадку — усереднена «вага» всіх його слів.

Віднесення тексту до «спаму» чи «не-спаму» проводиться в залежності від того, чи перевищує його «вага» якусь планку, задану користувачем (зазвичай беруть 60-80%). Після ухвалення рішення стосовно тексту в базі даних оновлюються «ваги» для слів, що входять до його складу.

Поштові байєсовські фільтри грунтуються на теоремі Байеса. Теорема Байєса використовується кілька разів в контексті спаму:

- в перший раз, щоб обчислити ймовірність, що повідомлення — спам, знаючи, що дане слово з’являється в цьому повідомленні;
- вдруге, щоб обчислити ймовірність, що повідомлення — спам, враховуючи всі його слова (або відповідні їх підмножини);

• іноді в третій раз, коли зустрічаються повідомлення з рідкісними словами.

Обчислення ймовірності того, що повідомлення, що містить дане слово, є спамом

Припустимо, що підозрюване повідомлення містить слово «Replica». Більшість людей, які звикли отримувати електронного листа, знає, що це повідомлення, швидше за все, буде спамом, а точніше — пропозицією продати підроблені копії годинників відомих брендів. Програма виявлення спаму, однак, не «знає» такі факти; все, що вона може зробити — обчислити ймовірності.

Формула, яка використовується програмним забезпеченням, щоб визначити це, отримана з теореми Байеса і формули повної ймовірності:

$$\Pr(S|W) = \frac{\Pr(W|S) \bullet \Pr(S)}{\Pr(W)} = \frac{\Pr(W|S) \bullet \Pr(S)}{\Pr(W|S) \bullet \Pr(S) + \Pr(W|H) \bullet \Pr(H)},$$

де

- $\Pr(S|W)$ — умовна ймовірність того, що **повідомлення-спам**, за умови, що слово «Replica» знаходиться в ньому;
- $\Pr(S)$ — повна ймовірність того, що довільне **повідомлення-спам**;
- $\Pr(W|S)$ — умовна ймовірність того, що слово «replica» з'являється в повідомленнях, якщо вони **є спамом**;
- $\Pr(H)$ — повна ймовірність того, що довільне повідомлення **не спам** (тобто «ham»);
- $\Pr(W|H)$ — умовна ймовірність того, що слово «replica» з'являється в повідомленнях, якщо вони є **не спам** («ham»).

Спамовість слова

Недавні статистичні дослідження показали, що на сьогоднішній день ймовірність будь-якого повідомлення бути спамом становить щонайменше 80%: $\Pr(S) = 0.8;\quad \Pr(H) = 0.2$.

Однак більшість байесовских програм виявлення спаму роблять припущення про відсутність апріорних переваг у повідомлення бути «spam», а не «ham», і вважає, що у обох випадків є рівні ймовірності 50%: $\Pr(S) = 0.5;\quad \Pr(H) = 0.5$.

Про фільтри, які використовують цю гіпотезу, кажуть як про фільтри «без упереджень». Це означає, що у них немає ніякого упередження щодо вхідних повідомлень електронної пошти. Дане припущення дозволяє спрощувати загальну формулу до:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)}.$$

Значення $\Pr(S|W)$ — називають «спамовістю» слова W; при цьому число $\Pr(W|S)$, що використовується в наведеній вище формулі, наближено дорівнює відносній частоті повідомлень, які містять слово W в повідомленнях, ідентифікованих як спам під час фази навчання, тобто:

$$\Pr(W_i|S) = \frac{count\left(M : W_i \in M, M \in S\right)}{\sum_j count\left(M : W_j \in M, M \in S\right)}.$$

Так само $\Pr(W|H)$ наближено дорівнює відносній частоті повідомлень, що містять слово W в повідомленнях, ідентифікованих як «ham» під час фази навчання.

$$\Pr(W_i|H) = \frac{count\left(M : W_i \in M, M \in H\right)}{\sum_j count\left(M : W_j \in M, M \in H\right)}.$$

Для того, щоб ці наближення мали сенс, набір навчальних повідомлень повинен бути великим. Також бажано, щоб набір навчальних повідомлень відповідав 50% гіпотезі про перерозподіл між спамом і «ham», тобто що набори повідомлень «spam» і «ham» мали один і той же розмір.

Звичайно, визначення, чи є повідомлення «spam» або «ham», що базується тільки на присутності лише одного певного слова, схильне до помилок. Саме тому байєсовські фільтри спаму намагаються розглянути кілька слів і комбінувати їх спамовість, щоб визначити повну ймовірність того, що повідомлення є спамом.

K-Nearest Neighbors algorithm (Метод *k*-найближчих сусідів)

Метод k-найближчих сусідів (англ. *K-nearest neighbors algorithm, k-NN*) — метричний алгоритм для автоматичної класифікації об'єктів або регресії.

- У разі використання методу для класифікації об'єкт присвоюється того класу, який є найбільш поширеним серед *k* сусідів даного елемента, класи яких вже відомі.
- У разі використання методу для регресії, об'єкту присвоюється середнє значення по *k* найближчим до нього об'єктів, значення яких вже відомі.

Алгоритм може бути застосований до вибірок з великою кількістю атрибутів (багатовимірних). Для цього перед застосуванням потрібно визначити функцію дистанції. Класичний варіант визначення дистанції — дистанція в евклідовому просторі.

Нормалізація

Однак різні атрибути можуть мати різний діапазон представлених значень у вибірці (наприклад, атрибут A представлений в діапазоні від 0.1 до 0.5, а

атрибут Б представлений в діапазоні від 1000 до 5000), то значення дистанції можуть сильно залежати від атрибутів з великими діапазонами. Тому дані зазвичай підлягають нормалізації. При кластерному аналізі є два основних способи нормалізації даних:

- **Міні-макс нормалізація:**

$$x' = \frac{\left(x - MIN[x]\right)}{\left(MAX[x] - MIN[x]\right)}$$

В цьому випадку всі значення будуть лежати в діапазоні від 0 до 1. Дискретні бінарні значення визначаються як 0 і 1.

- **Z-нормалізація:**

$$x' = \frac{\left(x - M[x]\right)}{\sigma[x]},$$

де σ - середньоквадратичне відхилення. У цьому випадку більшість значень потрапить в діапазон (-3σ; 3σ).

Визначення класу

При такому способі до уваги береться не тільки кількість потрапили в область певних класів, а й їх віддаленість від нового значення.

Для кожного класу *j* визначається оцінка близькості:

$$Q_j = \sum_{i=1}^{n} \frac{1}{d\left(x, a_i\right)^2},$$

де *d (x, a)* — дистанція від нового значення *x* до об'єкта *a*.

У якого класу вище значення близькості, той клас і присвоюється новому об’єкту.

Decision Tree learning (Дерева рішень)

Дерева рішень використовуються як передбачувальні моделі, що відображають знання про об'єкт (представлені гілками) у множину рішень.

Структура дерева являє собою «листя» та «гілки». На ребрах («гілках») дерева рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах — атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної на основі декількох змінних на вході.

Кожен лист представляє собою значення цільової змінної, зміненої в ході руху від кореня по листу. Кожен внутрішній вузол відповідає одній з вхідних змінних. Дерево може бути також «вивчено» поділом вихідних наборів змінних на підмножини, засновані на тестуванні значень атрибутів. Це процес, який повторюється на кожному з отриманих підмножин. Рекурсія завершується тоді, коли підмножина в вузлі має ті ж значення цільової змінної, таким чином, воно не додає цінності для пророкувань. Процес, що йде «зверху вниз», індукція дерев рішень (TDIDT), є прикладом поглинає «жодного» алгоритму, і на сьогоднішній день є найбільш поширеною стратегією дерев рішень для даних, але це не єдина можлива стратегія.

Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, до цього класу належать задачі чисельного прогнозування (передбачення значень цільової змінної).

Нехай нам задано деяку навчальну множину T, що містить об’єкти (приклади), кожен з яких характеризується m атрибутами (атрибутами), причому один з них вказує на приналежність об'єкта до певного класу.

Нехай через $\{C_1, C_2, \ldots, C_k\}$ позначені класи (значення мітки класу), тоді існують 3 ситуації:

1. множина T містить один або більше прикладів, що відносяться до одного класу C_k. Тоді дерево рішень для T — це лист, який визначає клас C_k;

2. множина T не містить жодного прикладу, тобто порожня множина. Тоді це знову лист, і клас, асоційований з листом, вибирається з іншого безлічі відмінного від T, скажімо, з множини, асоційованого з батьком;

3. множина T містить приклади, відносяться до різних класів. В цьому випадку слід розбити множину T на деякі підмножини. Для цього вибирається одина з ознак, що має два і більше відмінних один від одного значень $O_1, O_2, \ldots, O_k$. T розбивається на підмножини $T_1, T_2, \ldots, T_k$, де кожна підмножина T_i містить всі приклади, що мають значення O_i для вибраної ознаки. Це процедура буде рекурсивно тривати до тих пір, поки кінцевf множина не буде складатися із прикладів, що відносяться до одного і того ж класу.

Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову (перевірку), яка б розбивала множину, асоційовану з цим вузлом на підмножини. В якості такої перевірки повинен бути обраний один з атрибутів. Загальне правило для вибору атрибута можна сформулювати наступним чином: обраний атрибут повинен розбити множину так, щоб одержувані в результаті підмножини складалися з об'єктів, що належать до одного класу, або були максимально наближені до цього, тобто кількість об'єктів з інших класів ("домішок") в кожному з цих множин було якомога менше.

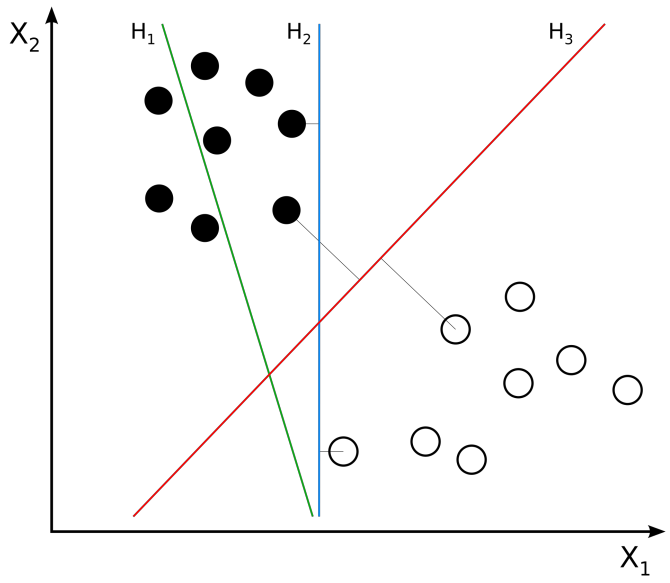
Support Vector Machine (SVM) (Метод опорних векторів)

Метод опорних векторів — це метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням з

пов'язаними алгоритмами навчання, які називаються **опо́рно-ве́кторними маши́нами.**

Для заданого набору тренувальних зразків, кожен із яких відмічено як належний до однієї чи іншої з двох категорій, алгоритм тренування ОВМ будує модель, яка відносить нові зразки до однієї чи іншої категорії, роблячи це ймовірнісним бінарним лінійним класифікатором. Модель ОВМ є представленням зразків як точок у просторі, відображених таким чином, що зразки з окремих категорій розділено чистою прогалиною, яка є щонайширшою. Нові зразки тоді відображуються до цього ж простору, й робиться передбачення про їхню належність до категорії на основі того, на який бік прогалини вони потрапляють.

В машинному навчанні поширеною задачею є класифікація даних. Розгляньмо деякі задані точки даних, кожна з яких належить до одного з двох класів, а метою є вирішувати, в якому класі буде *нова* точка даних. У випадку опорно-векторних машин точку даних розглядають як ρ-вимірний вектор (список ρ чисел), і хочуть дізнатися, чи можливо розділити такі точки $(\rho-1)$-вимірною гіперплощиною. Це називається **лінійним класифікатором**.



H_1 не розділяє ці класи. H_2 розділяє, але лише з невеликим розділенням. H_3 розділяє їх із максимальним розділенням.

Існує багато гіперплощин, які могли би розділяти одні й ті ж дані. Одним із варіантів розумного вибору найкращої гіперплощини є такий, який пропонує найбільший проміжок, або розділення (англ. *margin*) між двома класами. Тож ми обираємо гіперплощину таким чином, щоби відстань від неї до найближчих точок даних з кожного боку була максимальною. Така гіперплощина, якщо вона існує, відома як **максимально розділова гіперплощина** (англ. *maximum-margin hyperplane*), а лінійний класифікатор, що вона його визначає, — як максимально розділовий класифікатор (англ. *maximum margin classifier*); або, рівнозначно, як перцептрон оптимальної стабільності.

Опорно-векторна машина будує гіперплощину, або набір гіперплощин у просторі високої або нескінченної вимірності, які можна використовувати для класифікації, регресії та інших задач. Інтуїтивно, добре розділення досягається гіперплощиною, яка має найбільшу відстань до найближчих точок тренувальних даних будь-якого з класів (так зване функційне розділення), оскільки в загальному випадку що більшим є розділення, то нижчою є похибка узагальнення класифікатора.

В той час, як первинну задачу може бути сформульовано у скінченновимірному просторі, часто трапляється так, що множини, які треба розрізнювати, не є лінійно роздільними в ньому. З цієї причини було запропоновано відображувати первинний скінченновимірний простір до простору набагато вищої вимірності.

Для збереження помірного обчислювального навантаження, відображення, які використовуються методом опорних векторів, розробляють такими, щоби забезпечувати можливість простого обчислення скалярних добутків у термінах змінних первинного простору, визначаючи їх у термінах ядрових функцій $k(x,y)$, що їх обирають відповідно до задачі. Гіперплощини в просторі вищої вимірності визначаються як геометричне місце точок, чиї скалярні добутки з вектором у цьому просторі є сталими. Вектори, які визначають гіперплощини, можуть обиратися як лінійні комбінації з параметрами α_i відображень векторів ознак x_i, які трапляються в базі даних. За такого вибору гіперплощини, точки x

простору ознак, які відображаються на гіперплощину, визначаються відношенням $\sum_i \alpha_i k(x_i, x) = const$. Зауважте, що якщо $k(x, y)$ стає малою з віддаленням y від x , то кожен член цієї суми вимірює ступінь близькості пробної точки x до відповідних основних точок даних x_i . Таким чином, наведена вище сума ядер може використовуватися для вимірювання відносної близькості кожної пробної точки до точок даних, які походять з однієї або іншої з множин, які треба розрізнювати. Множина точок x, відображена на будь-яку гіперплощину, може бути в результаті доволі вигнутою, уможливлюючи набагато складніше розрізнювання між множинами, які взагалі не є опуклими в первинному просторі.

Random Forest

Random forest (англ. *випадковий ліс*) — ансамблевий метод машинного навчання для класифікації, регресії та інших завдань, які оперують за допомогою побудови численних дерев прийняття рішень під час тренування моделі і продукують моду для класів (класифікацій) або усереднений прогноз (регресія) побудованих дерев. Недоліком є схильність до перенавчання.

Нехай навчальна вибірка складається з N прикладів, розмірність простору ознак дорівнює M, і заданий параметр m (в задачах класифікації зазвичай $m \approx \sqrt{M}$).

Усі дерева комітету будуються незалежно один від одного за такою процедурою:

1. Згенеруємо випадкову підвибірку **з повторенням** розміром n з навчальної вибірки. (Таким чином, деякі приклади потраплять в неї кілька разів, а приблизно $N/3$ прикладів не ввійдуть у неї взагалі)

2. Побудуємо дерево рішень, яке класифікує приклади даної підвибірки, причому в ході створення чергового вузла дерева будемо вибирати ознаку, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних. Вибір найкращого з цих m ознак може здійснюватися різними способами.
3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі відсікання.

Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить об'єкт, який класифікується до одного з класів, і перемагає клас, за який проголосувало найбільше число дерев.

Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. У разі її відсутності, мінімізується оцінка помилки out-of-bag: частка прикладів навчальної вибірки, неправильно класифікованих комітетом, якщо не враховувати голоси дерев на прикладах, що входять в їх власну навчальну підвибірку.

ПРАКТИЧНА ЧАСТИНА

Будемо використовувати готовий набір даних SMS повідомлень, що містить як повідомлення зі спамом так і без — «spam.csv».

Попередня обробка набору даних

Зчитуємо дані:

```
df = pd.read_csv("/Users/elizabethlorelei/Documents/Study/Магистратура/NLP/spam.csv", encoding = 'latin-1')
```

Виведемо дані, що містяться у нашому наборі даних:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Важливо відзначити різницю між обробкою нульових значень (а саме, числа з рухомою комою (NaN)) для категоріальних та числових даних, оскільки трактування результатів буде залежати від типу наявних даних. Оскільки даний набір даних включає в себе колонки, що містять NaN, то ми видалимо ці колонки та розділимо на лейбли та повідомлення:

	Label	Message
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...

Враховуючи те, що методи, які ми застосовуємо, а саме Random forest (випадковий ліс), дерева рішень, метод *k*-найближчих сусідів, наївний баєсів класифікатор та метод опорних векторів (SVM), для класифікації базуються на **способі навчання з учителем**, або контрольоване навчання (англ. *Supervised learning*), то нам потрібно зробити розрізнення між даними між функціями та лейблами для кожного спостереження:

```
0      ham
1      ham
2     spam
3      ham
4      ham
Name: Label, dtype: object
```

Візуалізація даних

Оскільки наш набір даних містить реальні повідомлення, то текст містить **стоп-слова** або **шумові слова** — термін з теорії пошуку інформації за ключовими словами. Тобто це слова, які не несуть смислового навантаження, тому їх користь та роль для пошуку не суттєва.

Стоп-слова діляться на:

- загальні
- залежні

До **загальних** можна віднести прийменники, суфікси, дієприкметники, вигуки, цифри тощо. Загальні шумові слова завжди виключаються з пошукового запиту (за винятком пошуку за строгою відповідністю пошукової фрази). Вважається, що кожне з загальних стоп-слів є майже в усіх документах колекції.

До типових загальних шумових слів належать:

- цифри: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (один, два, три, чотири, п'ять, шість, сім, вісім, дев'ять, нуль).

- окремо розташовані знаки пунктуації: . , = + /! "; :%? * ()
- окремо розташовані букви алфавіту: а, б, в, г, ґ, д, е, є, ж, з, и, і, ї, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ь, ю,я
- займенники, дієприкметники, прийменники, вигуки, суфікси і поєднання букв: без, більш, б, був, була, були, було, бути (окрім фразеологічних зворотів, таких як *«бути чи не бути»*), вам, вас, адже, весь, вздовж, замість, поза, вниз, внизу, всередині, під, навколо, от, все, завжди, все, всіх, ви, де, да, давай, давати, навіть, для, до і т. д.
- слова, які часто зустрічаються на web-сайтах: Інтернет, сайт, питання, відповіді, комп'ютери, прайс, замовлення та інші.
- нецензурна мова

Залежні стоп-слова залежать від пошукової фрази. Ідея полягає в тому, щоб по-різному враховувати відсутність звичайних слів із запиту і залежних стоп-слів із запиту в знайденому документі. Залежні стоп-слова відрізняються тим, що в пошуковому запиті їх слід враховувати тільки при наявності в шуканому документі значущих ключових слів.

Отже, ми видалимо всі шумові слова із нашого набору даних та розіб’ємо наш текст на токени, тобто проведемо лексичний аналіз повідомлень.

Лексичний розбір — це процес перетворення послідовності символів в послідовність токенів (груп символів що відповідають певним шаблонам), та визначення їх типів. Часто сканер є звичайною функцією що використовується парсером (синтаксичним аналізатором), для отримання наступного токена з потоку вхідних символів в процесі компіляції.

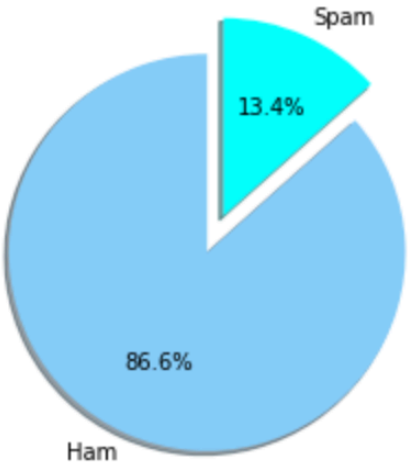
Позначка або **Токен** — це рядок літералів, впорядкованих відповідно до правил (наприклад, IDENTIFIER, NUMBER, COMMA). Процес утворення позначок з вихідного потоку називається **видобуванням позначок** (англ. *tokenization*) і розбирач впорядковує їх за відповідними типами.

Також виведемо найбільш вживані слова в спам повідомленнях та повідомленнях, що не містять спам.

Візуалізацію найбільш вживаних слів зробимо за допомогою Python бібліотек «Wordcloud»:

SPAM WORDS

У даному наборі даних міститься 13.4 % спамових повідомлень та 86.6 % — не є спамом.



Тобто наш набір даних не є рівномірним.

Розділення на тренувальну та тестову вибірку

Рандомним чином розділимо наш набір даних на тренувальну вибірку, на якій ми будемо тренувати нашу нейронну мережу, та тестову, на якій будемо тестувати.

```
(3733, 2)
(1839, 2)

The Trainset consists of 3733 records and 2 features

The Testset consists of 1839 records and 2 features
```

Тобто наша тренувальна вибірка містить 3733 записів та 2 класи (спам, не спам), а тестувальна — 1839 записи і також 2 класи.

Матриця помилок (Confusion Matrix)

У галузі проблем статистичної класифікації, матриця плутанини, також відома як **матриця помилок**, є специфічним макетом таблиці, яка дозволяє візуалізувати ефективність алгоритму, як правило, **способі навчання з учителем.** Кожен рядок матриці являє собою екземпляри у передбачуваному класі, тоді як кожен стовпець відображає екземпляри у фактичному класі (або навпаки). Ім'я пов'язане з тим, що це дає змогу легко з'ясувати, чи система заплутала два класи (тобто, як правило, неправильно позначаючи їх як інші).

Це спеціальний тип таблиць для непередбачених випадків, з двома параметрами ("актуальними" та "передбаченими") та однаковими наборами "класів" в обох вимірах (кожна комбінація розміру та класу є змінною в таблиці непередбачених ситуацій).

В аналітіці **таблиця плутанини** (іноді також називається **матрицею плутанини**) — це таблиця з двома рядками та двома стовпцями, в яких

повідомляється кількість помилкових спрацьовувань, помилкових негативів, справжніх позитивних і справжніх негативів. Це дозволяє проводити більш детальний аналіз, ніж лише частка правильних класифікацій (точність).

Точність не є надійною метрикою для реальної роботи класифікатора, оскільки це призведе до введення в оману результатів, якщо набір даних незбалансований (тобто коли кількість спостережень у різних класах дуже сильно відрізняється).

Остаточний стіл плутанини міститиме середні значення для всіх класів разом.

У медичній статистиці **хибно позитивне та хибно негативне спрацювання** — це поняття аналогічні до помилок першого та другого роду при перевірці статистичних гіпотез, де позитивний результат відповідає відхиленню нульової гіпотези, а негативний результат — відсутності достатніх підстав для того, щоб відхилити нульову гіпотезу. Термін часто вживається взаємнозамінно з поняттями помилок, але існують відмінності в деталях й інтерпретаціях.

Хибно позитивна помилка, часто називають «**помилковим спрацюванням**» або «**помилковою тривогою**», це результат, що вказує на існування певної умови, коли це не так.

Хибно негативна помилка — результат тесту, котрий полягає в тому, що умова не виконується, у той час як, насправді, це не так (тобто помилково, ніяких аномалій відхилення від норми не виявлено).

Хибно негативна помилка — помилка другого роду, що виникає під час перевірки однієї гіпотези, коли результат тесту помилково вказує на те, що дана гіпотеза неправильна.

Матриця помилок заповнюється статистичними результатами проведеної класифікації n об'єктів при наявності K класів. Кожен рядок нумерується індексом i, а кожен стовпчик нумерується індексом j. При цьому $i, j = 1,2,...,K$. Елемент матриці помилок nij відображає число об'єктів, що помилково були віднесені при класифікації до класу i, хоча в дійсності вони належать класу j.

Матриця помилок вказує на те, як співвідносяться значення збіжних класів, отримані з різних джерел. Як джерела можуть бути дані, які підлягають перевірці, та опорні (еталонні) дані, отримані з більш надійного джерела даних. Також під час інтерпретації результатів припускається, що результат, який перевіряється, є неточним, а перевірочні дані відображають реальну ситуацію. У випадку, коли перевірочні дані також є неточними, ми вже не можемо говорити про "помилку", а слід говорити про "різницю" між двома наборами даних.

Матриця помилок містить назви класів легенди класифікації даних, які підлягають перевірці, та класи легенди даних, що використовуються для перевірки.

Головна діагональ матриці вказує на ті випадки, де отримані розрахункові класи та реальні дані співпадають (правильна класифікація). Сума значень діагональних елементів вказує на загальну кількість правильно класифікованих пікселів. Відношення цієї кількості правильно класифікованих пікселів до загальної кількості пікселів у матриці називається загальною точністю класифікації (overall accuracy) і виражається у відсотках.

Для визначення точності певного розрахункового класу необхідно розділити кіль- кість правильно класифікованих пікселів цього класу на загальну кількість пікселів у ньо- му згідно з перевіреними даними. Даний показник називається точністю виробника (pro- ducer's accuracy). Точність виробника показує, наскільки добре результат класифікації для цього класу співпадає з даними, що були перевірені.

Також ми можемо обчислити аналогічний показник для реального класу (завіркових даних), якщо розділити кількість правильно класифікованих пікселів класу на загальну кі- лькість пікселів у ньому згідно з даними, що підлягають перевірці. Цей показник назива- ється точністю користувача, оскільки він показує, наскільки ймовірно, що даний клас збі- гається з результатами класифікації. Недіагональні елементи вказують на випадки

розбіж- ності між розрахунковими та реальними класами (помилки класифікації) .

Структура матриці

$$n_{i+} = \sum_{j=1}^{K} n_{ij}; \quad n_{+j} = \sum_{i=1}^{K} n_{ij}.$$

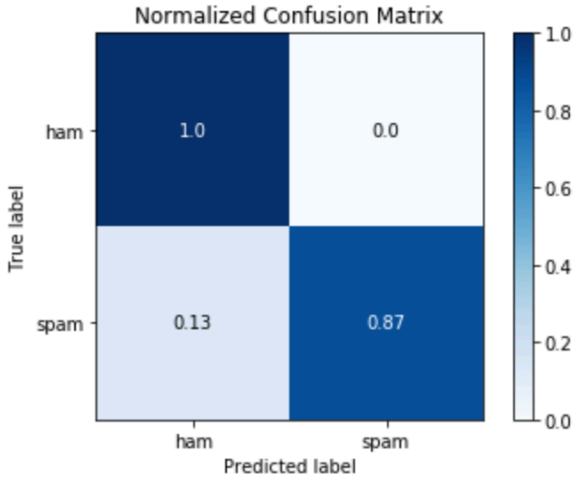
Наївний Баєсів класифікатор

В даному методі використовується згладжування Лапласа.

Точність наївного Баєсового класифікатора — 97.87928221859707%.

Матриця помилок:

$$\begin{bmatrix} 1581 & 6 \\ 33 & 219 \end{bmatrix}$$



З матриці помилок видно, що наївний Баєсовий класифікатор точно визначає не спам (1.0) та гарно визначає спам (на 0.87), проте є невелика похибка (0.13).

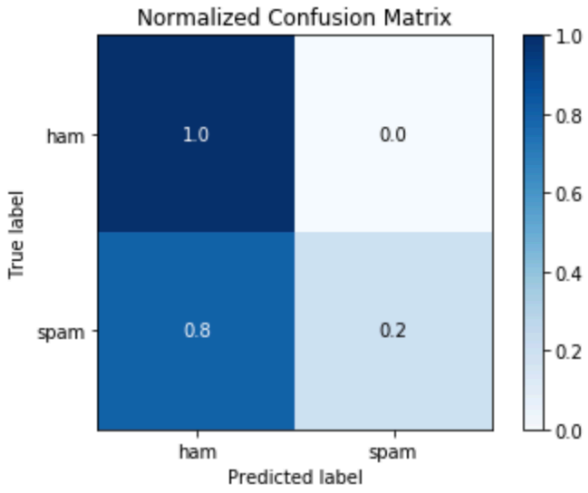
K-Nearest Neighbors algorithm (Метод *k*-найближчих сусідів)

В даному методі використовується Мінковського (метрика на Евклідовому просторі, яка є узагальненням Евклідового простору та Мангетенської відстані.)

Точність методу *k*-найближчих сусідів — 89.07014681892332%.

Матриця помилок:

$$\begin{bmatrix} 1587 & 0 \\ 201 & 51 \end{bmatrix}$$

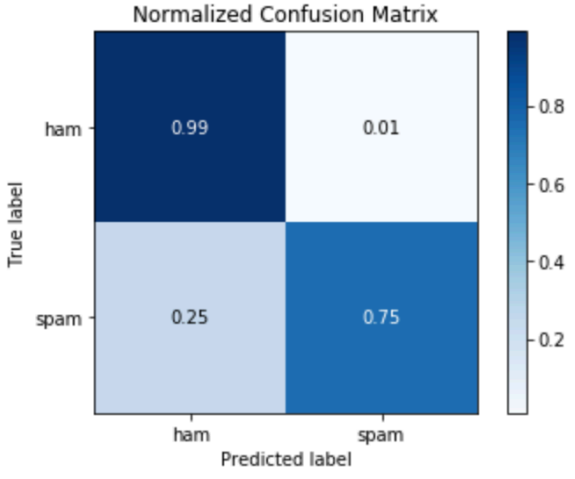


Метод *k*-найближчих сусідів точно визначає не спам (1.0) та погано визначає спам з точністю (0.2).

Decision Tree learning (Дерева рішень)

Точність дерева рішень — 95.86731919521479%.

Матриця помилок:



$$\begin{bmatrix} 1574 & 13 \\ 63 & 189 \end{bmatrix}$$

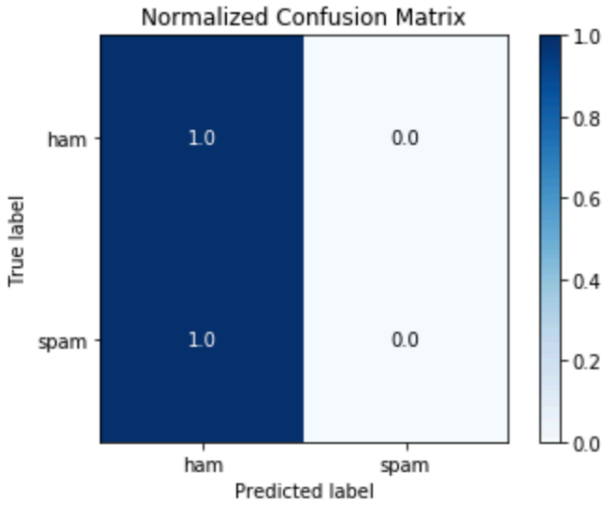
Дерево рішень майже точно визначає не спам (0.99) та досить гарно визначає спам з точністю (0.75), проте інколи визначає спам як не спам (0.25).

Support Vector Machine (SVM) (Метод опорних векторів)

Точність дерева рішень — 86.2969004893964%.

Матриця помилок:

$$\begin{bmatrix} 1587 & 0 \\ 252 & 0 \end{bmatrix}$$



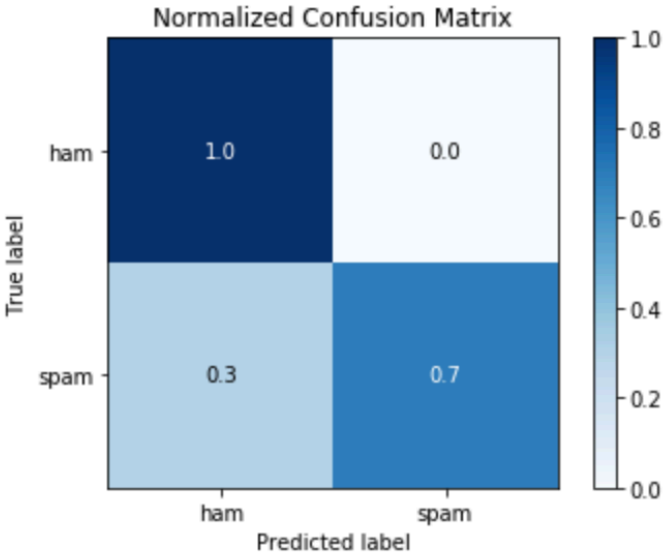
Метод опорних векторів точно визначає не спам (1.0), проте взагалі не визначає спам (0.0).

Random Forest

Точність Random Forest — 95.86731919521479%.

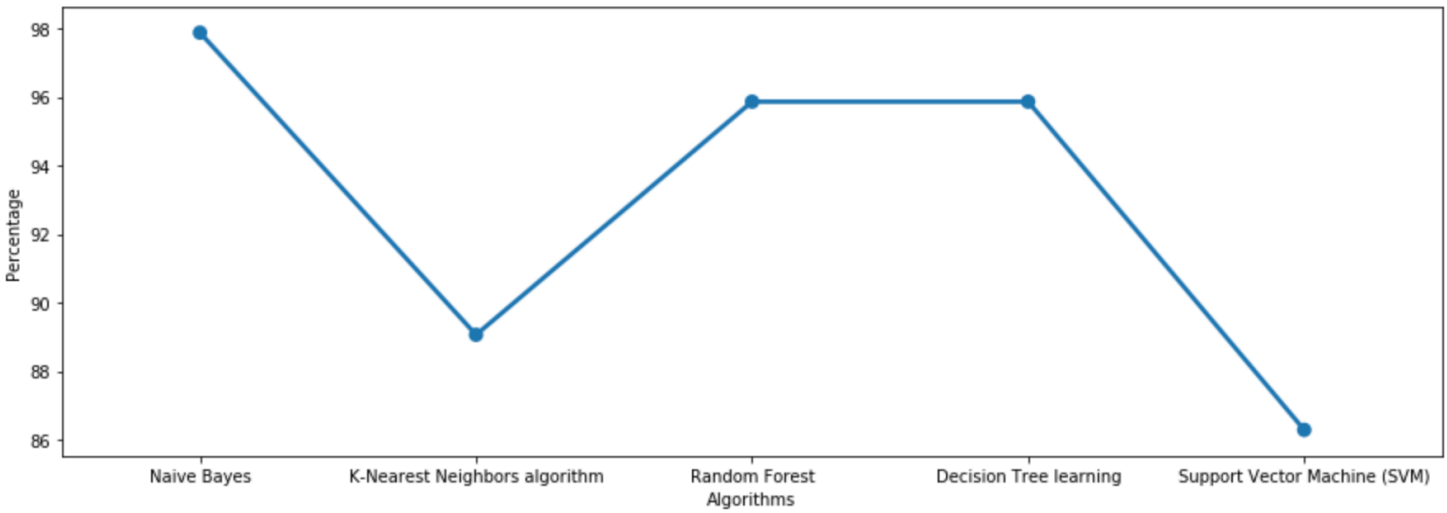
Матриця помилок:

$$\begin{bmatrix} 1587 & 0 \\ 76 & 176 \end{bmatrix}$$



Метод Random Forestв точно визначає не спам (1.0), і зн ганою точністю визначає спам (0.7).

ВИСНОВКИ



	Algorithms	Percentage
0	Naive Bayes	97.879282
1	K-Nearest Neighbors algorithm	89.070147
2	Random Forest	95.867319
3	Decision Tree learning	95.867319
4	Support Vector Machine (SVM)	86.296900

З отриманих даних ми бачимо, що найкраща точність у Наївного Баєсового класифікатору, потім у Decision Tree learning (Дерева рішень), а найгірший у Support Vector Machine (SVM) (Метод опорних векторів).

Метод Наївного Баєсового класифікатору простий (алгоритми елементарні), зручний (дозволяє обходитися без «чорних списків» і подібних штучних прийомів), ефективний (після навчання на досить великій вибірці відсікає до 95-97% спаму), причому в разі будь-яких помилок його можна донавчати. Загалом, є всі показання для його повсюдного використання, що і має місце на практиці — на його основі побудовані практично всі сучасні спам-фільтри.

Втім, у методу є і принциповий недолік: він базується на припущенні, що одні слова частіше зустрічаються в спам, а інші — в звичайних листах, і неефективний, якщо це припущення невірно. Втім, як показує практика, такий спам навіть людина не в змозі визначити «на око» - тільки прочитавши лист і зрозумівши його зміст.

В методі ***k*-найближчих сусідів** проблема вибору метрики — є найбільш складною з усіх проблем. У практичних завданнях класифікації рідко зустрічаються такі «ідеальні випадки», коли заздалегідь відома хороша функція. Якщо об'єкти описуються числовими векторами, часто беруть евклідову метрику. Цей вибір, як правило, нічим не обгрунтований - просто це перше, що спадає на думку. При цьому необхідно пам'ятати, що всі ознаки мають бути виміряні «в одному масштабі», а найкраще — отнормувати. В іншому випадку ознака з найбільшими числовими значеннями буде домінувати в метриці, інші ознаки, фактично, враховуватися не будуть.

Якщо ознак занадто багато, а відстань обчислюється як сума відхилень за окремими ознаками, то виникає проблема проблема розмірності. Суми великого числа відхилень з великою ймовірністю мають дуже близькі значення (відповідно до закону великих чисел). Виходить, що в просторі високої розмірності всі об'єкти приблизно однаково далекі один від одного; вибір k найближчих сусідів стає практично довільним.

Проблема вирішується шляхом відбору відносно невеликого числа інформативних ознак (features selection). В алгоритмах обчислення оцінок будується безліч різних наборів ознак (т.зв. опорних множин), для кожного будується своя функція близькості, потім по всіх функцій близькості проводиться голосування.

Перевагами методу **Random Forest** є:

- здатність ефективно обробляти дані з великим числом ознак і класів;
- нечутливість до масштабування (і взагалі до будь-яких монотонних перетворень) значень ознак;
- однаково добре обробляються як безперервні, так і дискретні ознаки;

- здатність працювати паралельно в багато потоків;
- масштабованість.

Проте він має і недоліки:

- Алгоритм схильний до перенавчання на деяких завданнях, особливо з великою кількістю шумів.
- Великий розмір отримуваних моделей. Потрібно $O(NK)$ пам'яті для зберігання моделі, де K — число дерев.

Метод опорних векторів може застосовуватися для розв'язання різноманітних практичних задач:

- є корисними для категоризації текстів та гіпертекстів, оскільки їхнє застосування може значно знижувати потребу в мічених тренувальних зразках як у стандартній індуктивній, так і в трансдуктивній постановках.
- Із застосуванням SVM може виконуватися й класифікація зображень. Експериментальні результати показують, що SVM можуть досягати значно вищої точності пошуку, ніж традиційні схеми уточнення запиту, всього лише після трьох-чотирьох раундів зворотного зв'язку про відповідність. Це є вірним і для систем сегментування зображень, включно з тими, які використовують видозмінену версію OBM, яка застосовує привілейований підхід, запропонований Вапником.[4][5]
- За допомогою SVM може здійснюватися розпізнавання рукописних символів.

SVM належить до сімейства узагальнених лінійних класифікаторів, і можуть бути інтерпретовані як розширення перцептрону. Їх також можна розглядати й як окремий випадок регуляризації Тихонова. Особливою властивістю є те, що вони одночасно мінімізують емпіричну похибку класифікації, й максимізують геометричне розділення; тому вони також відомі й як максимально розділові класифікатори.