Analýza ML metód a algoritmov pre detekciu spam emailových správ

Projekt INNOvative SOC/INNOSOC

Obsah

Úvod ... 3
Experimenty a výsledky ... 4
Datasety ... 4
SpamAssassin ... 4
SMS Spam ... 5
Použité algoritmy a metódy ... 6
Rozhodovacie stromy ... 6
SVM ... 8
Naive Bayes ... 9
Logistická regresia ... 11
XGBoost ... 12
Nastavenia a parametre zvolených metód ... 14
Rozhodovacie stromy ... 14
SVM ... 14
XGBoost ... 15
Logistická regresia ... 16
Naive Bayes ... 16
Vyhodnocovanie experimentov ... 16
Konfúzna matica ... 17
Accuracy ... 18
Precision ... 18
Recall ... 18
F1-Score ... 19
Výsledky ... 19
SpamAssassin ... 19
SMS Spam ... 19
Zhrnutie ... 19

Úvod

Väčšina bezpečnostných hrozieb používa e-mail ako vektor útoku. Keďže objem takto prenášanej prevádzky je obzvlášť veľký, je potrebné použiť automatizované postupy detekcie, ktoré využívajú algoritmy strojového učenia (ML) a umelej inteligencie (AI). Aby sme plne pochopili silu algoritmov ML/AI aplikovaných na detekciu spamu, je potrebné si najprv ujasniť predstavy, na akých úlohách by sa malo vykonávať filtrovanie mailových správ typu spam.

Pre lepšie pochopenie úloh, vykonávaných spamovým filtrom, možno uviesť nasledovný príklad. Predstavte si, že oddeľujete e-maily, ktoré dostávame, a kategorizujete ich na základe prítomnosti alebo neprítomnosti konkrétnych kľúčových slov, ktoré sa vyskytujú v texte e-mailov s určitou frekvenciou. Na tento účel možno mohli uviesť všetky správy, ktoré dostávame do doručenej pošty, v tabuľke. Ako však postupovať pri klasifikácii dostávaných správ do kategórií HAM (legitímna správa) alebo SPAM?

Ako už bolo spomenuté, bude sa hľadať počet výskytov podozrivých kľúčových slov v texte e-mailových správ. Jednotlivým správam identifikovaným ako spam sa potom pridelí skóre na základe počtu výskytov identifikovaných kľúčových slov. Toto skóre ďalej poskytne referenciu pre klasifikáciu následných e-mailových správ.

Určí sa prahová hodnotu, vďaka ktorej bude možné oddeliť emailové správy typu spam. Ak vypočítané skóre prekročí stanovenú prahovú hodnotu, e-mail bude automaticky klasifikovaný ako spam - v opačnom prípade bude správa prijatá ako legitímna a teda klasifikovaná ako ham. Táto prahová hodnota (ako aj pridelené skóre) sa bude neustále určovať, aby sa zohľadnila nová séria spamových správ, s ktorými sa v budúcnosti možno stretnúť.

Už z abstraktného popisu algoritmu detekcie spamu si možno všimnúť niektoré dôležité vlastnosti, ktoré je potrebné mať na pamäti: je nutné pristúpiť k identifikácii určitého počtu podozrivých kľúčových slov, ktoré umožňujú klasifikovať správy ako potenciálne spamové e-maily, pričom každému e-mailu sa priradí skóre na základe počtu výskytov identifikovaných kľúčových slov.

Je potrebné nastaviť určitú prahovú hodnotu pre skóre priradené jednotlivým e-mailom, nad ktorou budú e-maily automaticky klasifikované ako spam. Takisto je dôležité správne zvážiť význam kľúčových slov prítomných v texte e-mailov, aby bolo možné adekvátne reprezentovať mieru pravdepodobnosti, že správa, ktorá obsahuje dané kľúčové slová, predstavuje spam (kľúčové slová by v podstate mohli byť dokonca neškodné, ale spolu s väčšou pravdepodobnosťou predstavujú nevyžiadanú poštu).

Je nutné vziať do úvahy, že odosielatelia nevyžiadanej pošty si dobre uvedomujú snahu filtrovať nechcené správy, a preto sa budú snažiť čo najlepšie prijať nové stratégie, aby

spamové filtre oklamali. To sa premieta do procesu nepretržitého a iteratívneho učenia, ktorý sa dá dobre implementovať pomocou algoritmov ML/AI.

Z uvedeného je jasné, že nie je náhoda, že detekcia spamu predstavuje prvý test pri zavádzaní ML/AI v oblasti kybernetickej bezpečnosti. Prvé riešenia na detekciu spamu v praxi využívali statické pravidlá, ktoré používali regulárne výrazy na identifikáciu preddefinovaných vzorov podozrivých slov v texte e-mailu.

Tieto statické pravidlá sa rýchlo ukázali ako neúčinné v dôsledku stále nových stratégií klamania implementovaných spamermi na oklamanie antispamových filtrov. Preto bolo potrebné prijať dynamický prístup, ktorý by umožnil spamovému filtru učiť sa na základe neustálych inovácií zavádzaných spamermi a zároveň využívať rozhodnutia používateľa pri klasifikácii jeho e-mailov. Týmto spôsobom bolo možné efektívne zvládnuť explozívne šírenie fenoménu spamu.

Experimenty a výsledky

Datasety

SpamAssassin

Dataset SpamAssassin (tiež nazývaná SpamAssassin Public Corpus) je kolekcia e-mailových správ, ktoré boli označené ako spam alebo legitímne správy (ham). Bola vytvorená projektom Apache SpamAssassin, čo je open-source anti-spamová platforma. Korpus obsahuje niekoľko tisíc e-mailových správ rozdelených do kategórií spam a ham. Celkovo obsahuje dataset 6 047 správ s pomerom spamu približne 31 %.

Spamové správy predstavujú nevyžiadané komerčné e-maily, pokusy o phishing a inú nechcenú komunikáciu, zatiaľ čo ham správy predstavujú legitímne osobné a profesionálne e-maily.

E-maily sú uložené ako surové textové súbory v štandardnom e-mailovom formáte (RFC 822), pričom sa zachovávajú hlavičky, obsah tela a metadáta. Dátová sada obsahuje rôzne typy e-mailov s rôznymi štýlmi formátovania, obsahom HTML, odkazmi na prílohy a rôznymi jazykmi (hoci prevažne angličtinou).

Z hľadiska obsahu sa jedná o reálne e-maily zhromaždené od skutočných používateľov, ktorí darovali svoje správy, vďaka čomu sú reprezentatívne pre skutočné vzorce spamu a hamu. V datasete sú zahrnuté celé hlavičky e-mailov, ktoré obsahujú cenné metadáta, ako sú informácie o odosielateľovi, podrobnosti o smerovaní, časové pečiatky a informácie o type obsahu.

Najčastejšie je tento dataset používaný na trénovanie a hodnotenie algoritmov klasifikácie spamu, porovnávanie modelov strojového učenia pre klasifikáciu textu, výskum v oblasti spracovania prirodzeného jazyka a vyhľadávania informácií alebo

testovanie techník extrakcie prvkov pre filtrovanie e-mailov. Dataset bol v priebehu rokov vydaný vo viacerých verziách, pričom verejný korpus SpamAssassin je voľne dostupný na výskumné účely.

SMS Spam

Kolekcia SMS spamu je súbor SMS správ označených ako SMS správy, ktoré boli zozbierané na výskum SMS spamu. Tento dataset je široko používaný v strojovom učení a spracovaní prirodzeného jazyka na trénovanie systémov detekcie spamu v SMS správach. Existujú 2 verzie tejto databázy, najpoužívanejšou je v SMS je „SMS Spam Collection v.1“, ktorá sa stala štandardom v tejto oblasti. Obsahuje sadu SMS správ v angličtine s 5 574 správami, označenými ako ham (legitímne) alebo spam.

Súbory obsahujú jednu správu na riadok. Každý riadok sa skladá z dvoch stĺpcov: v1 obsahuje označenie (ham alebo spam) a v2 obsahuje surový text. Korpus bol zozbieraný z bezplatných alebo pre výskum bezplatných zdrojov na internete:

- Kolekcia 425 SMS spamových správ bola manuálne extrahovaná z webovej stránky Grumbletext. Ide o britské fórum, na ktorom používatelia mobilných telefónov verejne uvádzajú tvrdenia o SMS spamových správach, pričom väčšina z nich nenahlási samotnú prijatú spamovú správu. Identifikácia textu spamových správ v tvrdeniach je veľmi náročná a časovo náročná úloha a zahŕňala starostlivé skenovanie stoviek webových stránok.
- Podmnožina 3 375 náhodne vybraných SMS správ z NUS SMS Corpus (NSC), čo je dataset s približne 10 000 legitímnymi správami zozbieranými na výskum na Katedre informatiky Národnej univerzity v Singapure. Správy pochádzajú prevažne od Singapurčanov a prevažne od študentov navštevujúcich univerzitu. Tieto správy boli zozbierané od dobrovoľníkov, ktorí boli informovaní o tom, že ich príspevky budú verejne dostupné.
- 450 SMS správ bolo zozbieraných z dizertačnej práce Caroline Tag.
- Nakoniec je začlenených SMS Spam Corpus v.0.1 Big. Obsahuje 1 002 legitímnych SMS správ a 322 spamových správ.

Najčastejšie býva tento dataset používaný na nasledujúce úlohy:

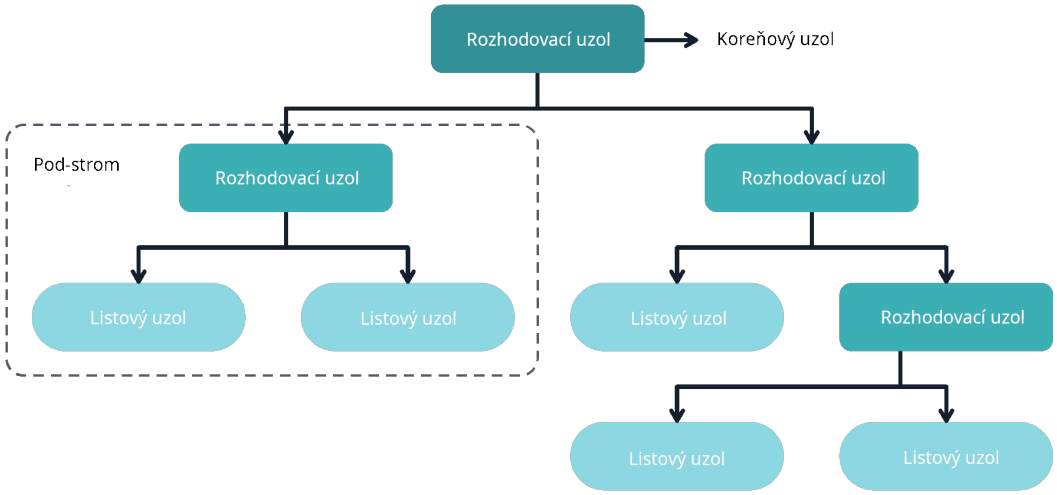
- Klasifikácia binárneho textu
- Tréning správy príznakov
- Manipulácia s nevyváženými dátami
- Techniky spracovania prirodzeného jazyka (NLP)
- Vyhodnotenie a porovnávanie modelov

Tento dataset si udržiava popularitu pre vzdelávacie účely, výskumné práce a ako základ pre vývoj algoritmov na detekciu spamu.

Použité algoritmy a metódy

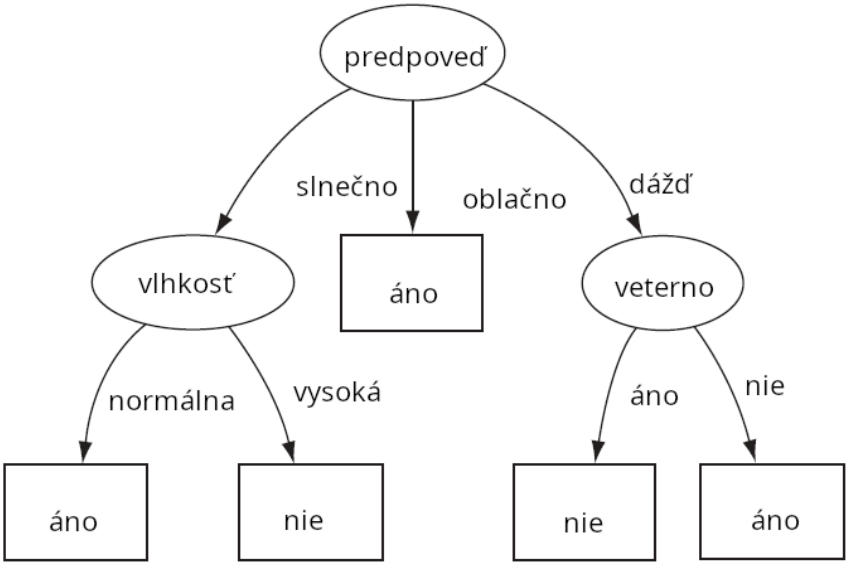
Rozhodovacie stromy

Rozhodovací strom je neparametrický algoritmus, ktorý sa používa na klasifikačné aj regresné úlohy. Má hierarchickú stromovú štruktúru, ktorá pozostáva z koreňového uzla, vetiev, vnútorných uzlov a listových uzlov. Ako možno vidieť z nižšie uvedeného diagramu, rozhodovací strom začína koreňovým uzlom, ktorý nemá žiadne prichádzajúce vetvy.



Obr. 1: Rozhodovací strom

Odchádzajúce vetvy z koreňového uzla sa potom privádzajú do interných uzlov, známych aj ako rozhodovacie uzly. Na základe dostupných funkcií oba typy uzlov vykonávajú hodnotenia s cieľom vytvoriť homogénne podmnožiny, ktoré sú označené listovými uzlami alebo koncovými uzlami. Listové uzly predstavujú všetky možné výsledky v rámci súboru údajov. Ako príklad si možno predstaviť, že ste sa snažíme posúdiť, či by sme mali alebo nemali ísť hrať golf, môžeme na výber použiť nasledujúce pravidlá rozhodovania:



Obr. 2: Rozhodovací strom pre dataset Golf

Tento typ štruktúry vývojového diagramu tiež vytvára ľahko stráviteľnú reprezentáciu rozhodovania, čo umožňuje rôznym skupinám v celej organizácii lepšie pochopiť, prečo bolo rozhodnutie prijaté. Učenie rozhodovacieho stromu využíva stratégiu rozdeľuj a panuj tým, že vykonáva chamtivé vyhľadávanie s cieľom identifikovať optimálne body rozdelenia v strome. Tento proces rozdelenia sa potom opakuje zhora nadol, rekurzívnym spôsobom, kým všetky alebo väčšina záznamov nie sú klasifikované pod špecifickými štítkami tried. To, či sú všetky dátové body klasifikované ako homogénne množiny, do značnej miery závisí od zložitosti rozhodovacieho stromu. Menšie stromy sú ľahšie schopné dosiahnuť čisté listové uzly – t. j. dátové body v jednej triede. Ako však strom rastie, je čoraz ťažšie udržať túto čistotu a zvyčajne to vedie k tomu, že do daného podstromu spadá príliš málo údajov. Keď k tomu dôjde, je tento jav známy ako fragmentácia údajov a často to môže viesť k nadmernému prispôsobeniu. Výsledkom je, že rozhodovacie stromy uprednostňujú malé stromy, čo je v súlade so zásadou úspornosti podľa princípu „Occamova britva“, čo znamená, že "entity by sa nemali množiť pokiaľ to nie je nevyhnutné". Inak povedané, rozhodovacie stromy by mali zvyšovať svoju zložitosť iba v prípade potreby, pretože najjednoduchšie vysvetlenie je často najlepšie. Aby sa znížila zložitosť a zabránilo sa nadmernému prispôsobeniu, zvyčajne sa používa prerezávanie; Toto je proces, ktorý odstraňuje vetvy, ktoré sa rozdeľujú na prvkoch s nízkou dôležitosťou. Prispôsobenie modelu sa potom môže vyhodnotiť prostredníctvom procesu krížovej validácie.

Ďalším spôsobom, ako si môžu rozhodovacie stromy zachovať svoju presnosť, je vytvorenie súboru pomocou algoritmu náhodného lesa (Random Forest). Tento klasifikátor predpovedá presnejšie výsledky, najmä ak jednotlivé stromy navzájom nekorelujú. Spomedzi viacerých variant možno uviesť nasledujúce najčastejšie používané:

- ID3
- C4.5
- CART

SVM

Metóda podporných vektorov je klasifikačnou a regresnou metódou, ktorú navrhol v 90-tych rokoch V. Vapnik a ktorá v podstate určuje hraničnú krivku oddeľujúcu 2 rôzne triedy. Hlavnou myšlienkou je nájdenie takej nadroviny, ktorá by optimálne rozdelila v príznakovom priestore vstupné dáta. Pre danú množinu trénovacích dát patriacich do 2 tried $(x_1, y_1), \ldots, (x_l, y_l)$, kde $x_i \in R_n$ a $y_i \in \{-1, 1\}$ sa teda snažíme nájsť takú nadrovinu $wx + b = 0$, aby sme mohli separovať dáta. Zo všetkých nadrovín určených w a b je len jedna taká, ktorá maximalizuje vzdialenosť medzi najbližšími bodmi z týchto tried a nadrovinou - optimálna nadrovina a platí:

$$d(w,b) = \min_{\{x_i | y_i = 1\}} \frac{wx_i + b}{|w|} - \max_{\{x_i | y_i = -1\}} \frac{wx_i + b}{|w|}$$

a zodpovedajúca minimálna a maximálna hodnota je -1, resp. +1

$$d(w,b) = \left| \frac{1}{\mathbf{w}} - \frac{-1}{\mathbf{w}} \right| = \frac{2}{|\mathbf{w}|} .$$

Minimalizáciou výrazu $|w|^2 / 2$ v závislosti na podmienke $y_i (wx_i + b) \geq 1$ dostaneme pre daný pár nadrovín maximálnu vzdialenosť 2 tried. Riešenie optimalizačného problému SVM je dané sedlovým bodom (saddle point) Lagrangeovej funkcie:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(wx_i + b \right) - 1 \right],$$

kde α_i je Lagrangeov multiplikátor. Zatiaľ čo $L(w, b, \alpha)$ sa zmenšuje s w a b, derivácie $L(w, b, \alpha)$ sú pre všetky $\alpha_i \geq 0$ rovné nule. Riešenie je dané

$$\overline{w} = \sum_{i=1}^{l} \overline{\alpha}_i y_i x_i \qquad \overline{b} = -\frac{1}{2} \overline{w} \left[x_r + x_s \right],$$

kde x_r a x_s sú podporné vektory, ktoré patria do triedy $y_r = 1$, resp. $y_s = -1$. Dáta v pôvodnom príznakovom priestore nie sú separovateľné žiadnou lineárnou nadrovinou, ale môžu byť lineárne separovateľné v nelineárne mapovanom priestore príznakov, ktorý je odlišný od toho pôvodného a zvyčajne má vyššiu dimenziu. Toto nelineárne mapovanie $K(.,.)$ sa uskutočňuje pomocou jadrových funkcií (kernel functions):

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \overline{\alpha}_i y_i K(x_i, x) + b \right) \sum_{i=1}^{N} \alpha_i t_i K(x, x_i) + b,$$

kde yi sú cieľové hodnoty, $\sum_{i=1}^{l} \overline{\alpha}_i y_i = 0$ a $\overline{\alpha}_i > 0$. Jadro *K*(.,.) sa vytvára tak, aby spĺňalo špecifické vlastnosti v závislosti na danom probléme a možno napísať:

$$K(x, y) = \Phi(x)\Phi(y) ,$$

kde Φ(x) je mapovanie vstupného príznakového priestoru do nového priestoru s potenciálne nekonečnou dimenziou. Existujú 3 jadrové funkcie zabezpečujúce nelineárne mapovanie:

- polynomiálna $K(x, y) = (xy + 1)^z$ – parameter *z* predstavuje stupeň polynómu
- Gaussove radiálne bázové funkcie (Radial Basis Functions RBF) $K(x, y) = exp\{-[(x – y)^2 / 2\sigma^2]\}$ – kde parameter σ je štandardná odchýlka gaussovskej funkcie
- MLP funkcia $K(x, y) = tanh(scale(xy) - offset)$ – *scale* a *offset* sú dané parametre.

Naive Bayes

Naivný Bayesov klasifikátor patrí do kategórie pravdepodobnostných klasifikátorov, ktorá je založená na Bayesovej vete.

V teórii pravdepodobnosti rozlišujeme pre vzťah medzi náhodnými premennými 3 druhy pravdepodobností:

- Podmienená pravdepodobnosť
- Spojená pravdepodobnosť
- Marginálna pravdepodobnosť

Podmienená pravdepodobnosť je miera pravdepodobnosti jednej náhodnej premennej vzhľadom na hodnotu inej náhodnej premennej. Nech $\forall X, Y \in F$ a $P(Y) \neq 0$, potom

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}.$$

$P(X|Y)$ znamená podmienenú pravdepodobnosť *X* pri danej hodnote *Y*. Napríklad $P(X|Y = y)$ znamená podmienenú pravdepodobnosť X pri danej hodnote $Y = y$.

Koncept podmieneného rozdelenia možno rozšíriť na meranie pravdepodobnosti jednej náhodnej premennej vzhľadom na viacero ďalších náhodných premenných. Napríklad majme $Y = y$ a $Z = z$, podmienená pravdepodobnosť $X = x$ sa označuje ako

$$P(X|Y, Z) = \frac{P(X \cap Y \cap Z)}{P(Y \cap Z)}.$$

Podmienená pravdepodobnosť je jednou z dôležitých metód v teórii pravdepodobnosti pre inferenciu alebo učenie sa v neistote.

Spojená pravdepodobnosť sa vzťahuje na pravdepodobnosť dvoch alebo viacerých náhodných premenných. Spojenú (niekedy nazývanú aj združenú) pravdepodobnosť dvoch náhodných premenných možno vyjadriť ako $P(X, Y)$ a $P(X, Y) = P(X \cap Y)$. Z prvej rovnice pre podmienenú pravdepodobnosť je známe, že pre nasledovné spojené rozdelenie platí:

$$P(X, Y) = P(X|Y)\, P(Y)\,.$$

Podľa komutatívneho zákona teórie pravdepodobnosti možno túto rovnicu zapísať aj ako:

$$P(X, Y) = P(Y|X)\, P(X)\,.$$

Vyššie uvedené rovnice sú známe aj ako pravidlá súčinu. Rozšírením vyššie uvedeného spojeného rozdelenia na n náhodných premenných je možné vytvoriť nasledujúce reťazové pravidlo:

$$\begin{aligned} P(X_1, \ldots, X_n) &= P(X_1)\, P(X_2|X_1)\, P(X_3|X_2, X_1) \cdots P(X_n|X_{n-1}, \ldots, X_1) \\ &= P(X_1) \prod_{i=2}^{n} P(X_i|X_{i-1}, \ldots, X_1). \end{aligned}$$

Reťazové pravidlo sa často používa na spojenú pravdepodobnosť viacerých náhodných premenných, najmä ak existuje vlastnosť podmienenej nezávislosti medzi niektorými premennými.

Marginálna pravdepodobnosť sa vzťahuje na rozdelenie pravdepodobnosti samotnej náhodnej premennej. Vzhľadom na spojené rozdelenie môžeme pre náhodné premenné X a Y vypočítať buď marginálne rozdelenie X alebo Y. Pre nájdenie marginálneho rozdelenia jednej náhodnej premennej je potrebné sčítať všetky rozdelenia inej náhodnej premennej. To možno formálne vyjadriť ako:

$$P(X) = \sum_{y \in \mathrm{Val}(Y)} P(X, Y = y)\,.$$

Uvedená rovnica je známa aj ako pravidlo súčtu.

Marginálne rozdelenie sa tak nazýva kvôli tomuto uhlu pohľadu: ak spočítame všetky položky v riadku alebo stĺpci spojeného rozdelenia a výsledok umiestnime na poslednú stranu, je to pravdepodobnosť, že náhodná premenná nadobudne túto hodnotu, a táto posledná strana sa nazýva okraj (v angl. jazyku margin => názov marginálny).

Samozrejme, toto stanovisko platí len vtedy, keď spoločné rozdelenie zahŕňa dve náhodné premenné.

Klasifikátor Bayesovho teorému založený na silnej nezávislosti medzi príznakmi je zjednodušením podmienok teorému a preto získal prívlastok „naivný“.

Daný dátový vektor $\boldsymbol{x} = (x_1, \ldots, \mathbf{x}_n) \in X$, ktorý sa má klasifikovať vo vstupnom príznakovom priestore, kde výstupným priestorom je množina kategórií $C = \{c_j \mid j = 1, \ldots, k\}$, naivná Bayesova klasifikácia je podmienená pravdepodobnosť kategórie c_j pre daný dátový vektor $\boldsymbol{x}$, t.j. $P(c_j \mid \boldsymbol{x})$. Podľa Bayesovej vety existuje nasledujúca rovnica:

$$P(c_j|x) = \frac{P(x|c_j)\,P(c_j)}{P(x)} = \frac{P(x|c_j)\,P(c_j)}{\sum_k P(x|c_j)P(c_j)}.$$

Podľa reťazového pravidla spojeného rozdelenia pravdepodobnosti získame nasledovné:

$$P(c_j, x) = P(c_j, x_1, \ldots, x_n) = P(c_j) \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, , c_j).$$

$$P(c_j, x_1, \ldots, x_n) = P(c_j) \prod_{i=1}^{n} P(x_i|c_j).$$

$$P(c_j|x_1, \ldots, x_n) \propto P(c_j) \prod_{i=1}^{n} P(x_i|c_j).$$

$$P(c_j|x_1, \ldots, x_n) = \frac{1}{Z} P(c_j) \prod_{i=1}^{n} P(x_i|c_j),$$

Logistická regresia

Logistická regresia, tiež známa ako logistická regresná analýza, je metóda štatistickej analýzy, ktorá predpovedá zodpovedajúce binárne výsledky na základe vzťahu medzi nezávislou premennou a závislou premennou, a preto ju možno použiť na klasifikáciu dvoch tried. Klasifikácia je v podstate regresný problém s diskrétnym výstupom - algoritmus logistickej regresie odráža túto povahu.

Nech X označuje vstupný priestor klasifikácie a $C = \{0, 1\}$ označuje výstupný priestor dvojtriednej klasifikácie. Pre vstupné dáta $\boldsymbol{x} \in X$ je logistická regresia $P(c|\boldsymbol{x})$ pre predikciu ich kategórie ako $c \in C$ daná nasledovne:

$$P(c|x) = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)},$$

kde $\sigma(\cdot)$ sa označuje ako logistická funkcia, známa aj ako sigmoidná funkcia, ktorá sa tiež používa ako jedna z aktivačných funkcií v umelých neurónových sieťach.

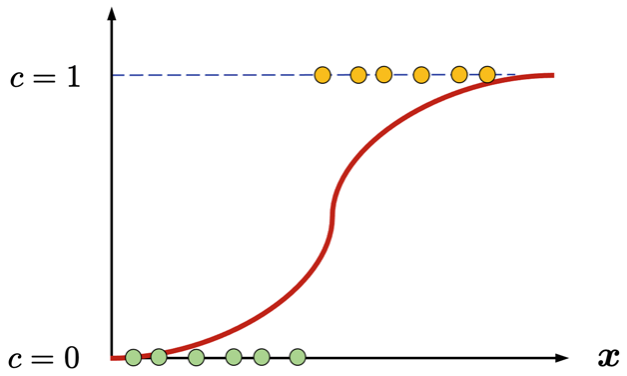
Logistická regresia má nasledujúcu vlastnosť, t. j. $P(c = 0|\boldsymbol{x}) = 1 - P(c = 1|\boldsymbol{x})$. Je to preto, že rozdelenie pravdepodobnosti dvojtriednej klasifikácie by malo spĺňať podmienku $P(c = 0|\boldsymbol{x}) + P(c = 1|\boldsymbol{x}) = 1$. Tiež nech $z = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b$, z čoho sa získa nasledovné:

$$P(c = 1|x) = \sigma\left(w^{\mathsf{T}}x - b\right) = \frac{1}{1 + \exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)},$$

$$P(c = 0|x) = 1 - \sigma\left(w^{\mathsf{T}}x - b\right) = 1 - \frac{1}{1 + \exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}$$

$$= \frac{\exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}{1 + \exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}.$$

Schematický diagram algoritmu klasifikácie logistickej regresie je znázornený na nasledovnom obrázku.



Obr. 3: Princíp algoritmu klasifikácie logistickej regresie

Vyššie uvedená metóda logistickej regresie je použiteľná len pre klasifikáciu dvoch tried. Neskôr sa objavili niektoré rozšírené verzie logistickej regresie pre viactriedne klasifikačné problémy, vrátane multi-normálnej logistickej regresie, usporiadanej logistickej regresie a podmienenej logistickej regresie.

XGBoost

Algoritmus XGBoost (Extreme Gradient Boosting) je jednou z najefektívnejších implementácií rozhodovacích stromov s podporou gradientu a bol vybraný ako jeden z najlepších algoritmov strojového učenia používaných v súťažiach Kaggle. Algoritmus XGBoost, špeciálne navrhnutý na optimalizáciu využitia pamäte a využitie hardvérového výpočtového výkonu, skracuje čas vykonávania pri súčasnej zvýšenej efektívnosti v porovnaní s mnohými algoritmami strojového učenia a dokonca aj modelmi hlbokého učenia. Hlavnou myšlienkou boostingu je postupne budovať pod-stromy z pôvodného stromu tak, aby každý nasledujúci strom znížil chyby (reziduá) stromu predchádzajúceho.

Týmto spôsobom budú nové pod-stromy aktualizovať predchádzajúce reziduá, aby sa znížila chybovosť účelovej funkcie (cost function).

Zoberme si súbor dát vyjadrený takto:

$$D = \{(x_i, y_i) \text{ kde } x_i \in R_n; y_i \in R)\},$$

kde n predstavuje dimenziu príznakov x_i a y_i je zodpovedajúca odpoveď alebo trieda. Pre i-ty prvok množiny dát definujeme predikovanú hodnotu $\hat{y}_i$ nasledovne:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F,$$

kde f_k označuje nezávislý strom v priestore regresných stromov F a $f_k(x_i)$ vyjadruje predikované skóre pre i-tu vzorku a k-ty strom. Objektívna funkcia algoritmu XGBoost, označovaná ako L, je daná nasledovne:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k).$$

Minimalizáciou objektívnej funkcie L je možné natrénovať funkcie f_k regresného stromu. Chybová funkcia $l(y_i, \hat{y}_i)$ vyhodnocuje rozdiel medzi predikciou $\hat{y}_i$ a skutočnou hodnotou y_i. Člen Ω sa používa na zabránenie problému s pretrénovaním penalizáciou zložitosti modelu nasledovne:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2,$$

kde T je počet listov v strome, w predstavuje skóre (váhu) listu (napr. predpokladaná hodnota pre vzorky, ktoré skončia v tomto liste), γ je regularizačný faktor, ktorý vyjadruje náklady na zložitosť, ktoré odrádzajú od vytvárania stromu s príliš veľkým počtom listov a λ je regularizačný výraz L2 na váhach listov, ktorý sa ich snaží zmenšovať tak, aby bol model konzervatívnejší. Priama minimalizácia objektívnej funkcie je zložitá. XGBoost používa Taylorovo rozšírenie druhého rádu na jeho aproximáciu. Toto je kľúčový dôvod jeho výkonu, pretože používa prvú aj druhú deriváciu (gradient a hessián). Nech $I_j = \{i|q(x_i) = j\}$ je inštancia množiny listov j s $q(x)$ fixnou štruktúrou. Optimálne váhy w_j listu j a zodpovedajúcu optimálnu hodnotu možno získať nasledujúcimi rovnicami:

$$w_j^* = -\frac{g_j}{h_j + \lambda},$$

$$\mathcal{L}^* = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \lambda T,$$

kde g_i a h_i sú prvá a druhá derivácia stratovej funkcie L. Stratovú funkciu L možno použiť ako skóre kvality stromovej štruktúry q. Čím je skóre menšie, tým je model lepší. Keďže nie je možné vyčísliť všetky stromové štruktúry, *greedy* algoritmus môže problém vyriešiť tak, že začne od jedného listu a iteratívne pridáva vetvy do stromu. Nech I_R a I_L sú množiny inštancií pravého a ľavého uzla po rozdelení. Za predpokladu $I = I_R \cup I_L$, potom sa po rozdelení redukcia straty vyjadrí nasledovne:

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}\right] - \gamma.$$

Tento vzťah sa zvyčajne používa v praxi na hodnotenie kandidátov na rozdelenie. Model XGBoost používa veľa jednoduchých stromov a skóre listových uzlov počas rozdelenia. Prvé tri členy rovnice predstavujú skóre ľavého, pravého a pôvodného listu. Okrem toho člen γ predstavuje regularizáciu na dodatočnom liste a bude sa používať v procese trénovania.

Nastavenia a parametre zvolených metód

Rozhodovacie stromy

- criterion 'gini'
- splitter 'best'
- max_depth None
- min_samples_split 2
- min_samples_leaf 1
- min_weight_fraction_leaf 0.0
- max_features None
- random_state None
- max_leaf_nodes None
- min_impurity_decrease 0.0
- class_weight 'balanced'
- ccp_alpha 0.0
- monotonic_cst None

SVM

- C 1.0
- kernel 'linear'
- degree 3
- gamma 'scale'
- coef0 0.0
- shrinking True

- probability True
- tol 0.001
- cache_size 200
- class_weight None
- verbose False
- max_iter -1
- decision_function_shape 'ovr'
- break_ties False
- random_state None

XGBoost

- objective 'binary:logistic'
- base_score None
- booster None
- callbacks None
- colsample_bylevel None
- colsample_bynode None
- colsample_bytree None
- device None
- early_stopping_rounds None
- enable_categorical False
- eval_metric None
- feature_types None
- feature_weights None
- gamma None
- grow_policy None
- importance_type None
- interaction_constraints None
- learning_rate None
- max_bin None
- max_cat_threshold None
- max_cat_to_onehot None
- max_delta_step None
- max_depth None
- max_leaves None
- min_child_weight None
- missing nan
- monotone_constraints None
- multi_strategy None

- n_estimators None
- n_jobs None
- num_parallel_tree None
- random_state None
- reg_alpha None
- reg_lambda None
- sampling_method None
- scale_pos_weight None
- subsample None
- tree_method None
- validate_parameters None
- verbosity None

Logistická regresia

- penalty 'l2'
- dual False
- tol 0.0001
- C 1.0
- fit_intercept True
- intercept_scaling 1
- class_weight None
- random_state None
- solver 'lbfgs'
- max_iter 100
- multi_class 'deprecated'
- verbose 0
- warm_start False
- n_jobs None
- l1_ratio None

Naive Bayes

- alpha 1.0
- force_alpha True
- binarize 0.0
- fit_prior True
- class_prior None

Vyhodnocovanie experimentov

Pre vyhodnocovanie experimentov boli použité nasledovné metriky.

Konfúzna matica

Konfúzna matica je jedným z najpopulárnejších spôsobov vyhodnocovania akéhokoľvek klasifikačného modelu. Hoci matica sama o sebe nie je metrikou, maticovú reprezentáciu možno použiť na definovanie rôznych metrík, z ktorých všetky sa stanú dôležitými v nejakom konkrétnom prípade alebo scenári. Konfúznu maticu je možné vytvoriť pre binárnu klasifikáciu, rovnako ako aj pre klasifikačný model s viacerými triedami.

Konfúzna matica sa vytvorí porovnaním predikovaného označenia triedy dátového bodu s jeho skutočným označením triedy. Toto porovnanie sa zopakuje pre celý súbor dát (dataset) a výsledky tohto porovnania sa zostavia v maticovom alebo tabuľkovom formáte. Táto výsledná matica predstavuje konfúznu maticu (obr. nižšie).

		Skutočná trieda	
		Kladná	Záporná
Predikovaná trieda	Kladná	Počet správne kladných (TP)	Počet nesprávne kladných (FP)
	Záporná	Počet nesprávne záporných (FN)	Počet správne záporných (TN)

Obr. 3 Konfúzna matica

Vo všeobecnosti máme zvyčajne 2 triedy – kladnú/pozitívnu a zápornú/negatívnu triedu. Na základe tejto štruktúry môžeme jasne vidieť štyri pojmy dôležitosti.

• True Positive (TP) – Počet správne kladných: Toto je suma celkového počtu príkladov (dátových bodov z datasetu) z pozitívnej triedy, kde sa skutočné označenie triedy rovnalo predpovedanému označeniu triedy, t. j. celkovým počtom prípadov, v ktorých sme správne predpovedali označenie pozitívnej triedy pomocou nášho modelu.

• False positive (FP) – Počet nesprávne kladných: Toto je počet celkových prípadov z negatívnej triedy, kde ich model nesprávne klasifikoval tým, že ich predpovedal ako pozitívne. Odtiaľ pochádza názov, falošne pozitívny.

• True Negative (FN) – Počet nesprávne záporných: Toto je počet celkového počtu príkladov z negatívnej triedy, kde sa skutočné označenie triedy rovnalo predpovedanému označeniu triedy, t. j. celkový počet prípadov, v ktorých sme správne predpovedali označenie negatívnej triedy pomocou nášho modelu.

• False negatívne (FN) – Počet nesprávne záporných: Toto je počet celkového počtu prípadov z pozitívnej triedy, kde ich náš model nesprávne klasifikoval tým, že ich predpovedal ako negatívne. Odtiaľ pochádza názov, falošne negatívny.

Accuracy

Táto metrika je jednou z najpopulárnejších spôsobov merania resp. vyhodnocovania efektívnosti ľubovoľného klasifikátora. Je definovaná ako celková presnosť alebo podiel správnych predpovedí modelu. Vzťah pre výpočet z konfúznej matice je nasledovný:

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision

Precision (presnosť), známa aj ako kladná prediktívna hodnota, je ďalšou metrikou, ktorú možno odvodiť z konfúznej matice. Táto metrika je definovaná ako počet predikcií, ktoré sú skutočne správne alebo relevantné zo všetkých predikcií založených na kladnej triede. Vzorec pre túto metriku je nasledovný:

$$P = \frac{TP}{TP + FP}$$

Model s vysokou hodnotou presnosti identifikuje vyšší podiel kladnej triedy v porovnaní s modelom s nižšou presnosťou. Táto metrika sa stáva dôležitou v prípadoch, keď sa viac zaujímame o nájdenie maximálneho počtu kladných tried, aj keď sa celková presnosť zníži.

Recall

Recall (úplnosť) je metrika modelu na identifikáciu percenta relevantných dátových bodov. Je definovaná ako počet inštancií kladnej triedy, ktoré boli správne predikované. Označuje sa aj ako miera pokrytia alebo citlivosť. Vzťah pre vypočítanie tejto metriky je nasledovný:

$$R = \frac{TP}{TP + FN}$$

Táto metrika sa stáva dôležitým ukazovateľom efektívnosti klasifikátora v scenároch, v ktorých chceme zachytiť čo najväčší počet prípadov konkrétnej triedy, aj keď to spôsobí zvýšenie falošných detekcií/predikcií. Ako príklad možno uviesť prípad bankového podvodu, model s vysokou hodnotou metriky recall nám poskytne vyšší počet prípadov potenciálnych podvodov. Rovnako to však zvýši vyvolanie poplachov pre väčšinu podozrivých prípadov.

F1-Score

Existujú prípady, v ktorých chceme dosiahnuť vyváženú optimalizáciu presnosti a úplnosti. F1-Score je metrika, ktorá je harmonickým priemerom metrík precision a recall a pomáha nám optimalizovať klasifikátor pre vyváženú presnosť a úplnosť predikcie. Vzorec pre výpočet F1-Score je nasledovný:

$$F = \frac{2.P.R}{P+R}$$

Výsledky

SpamAssassin

Metóda	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.9761	0. 9776	0.9774	0.9774	1.4
SVM	0.9928	0.9929	0.9928	0.9929	11.9
XGBoost	0.9928	0.9929	0.9928	0.9929	3.3
Log. Regression	0.9833	0.9833	0.9833	0.9833	1.2
Naive Bayes	0.8438	0.8624	0.8439	0.8330	1.3

SMS Spam

Metóda	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.7585	0.7624	0.7585	0.7604	0.001
SVM	0.7706	0.7722	0.7706	0.7714	0.005
XGBoost	0.7826	0.7757	0.7826	0.7791	0.001
Log. Regression	0.7887	0.7709	0.7888	0.7795	0.001
Naive Bayes	0.7785	0.7733	0.7785	0.7759	0.001

Zhrnutie

Na základe vykonaných experimentov možno zhodnotiť, že najvhodnejšia metóda pre detekciu emailových správ typu spam z hľadiska efektívnosti je metóda *XGBoost***.** Pre dataset „*SpamAssassin*“ dosahuje úspešnosťou detekcie spamu úroveň 99%. Táto hodnota je takmer identická pre metriky Precision aj Recall, takže je veľmi dobre pokrytá detekcia falošne pozitívnych aj falošne negatívnych detekcií správ typu spam. Pre tento dataset dosahuje takmer identické výsledky aj metóda *SVM*, avšak čas vyhodnocovania v prípade algoritmu *SVM* je približne 4x väčší oproti *XGBoost*. Napriek tomu sa jedná stále o niekoľko sekundový proces, takže možno nasadiť obe metódy pre real-time aplikácie. Pre dataset „*SMS Spam*“ úspešnosť detekcie spamu výrazne klesla na hodnotu ~78%. Najlepšie výsledky dosiahla metóda *Logistická regresia*, hoci rozdiel medzi metódami

XGBoost a *Logistickou regresiou* predstavuje len desatiny percent. Z hľadiska času potrebného pre predikciu dát, algoritmus *SVM* opäť vyžaduje najdlhší interval. Na základe vykonaných experimentov teda možno sumarizovať, že najvhodnejšou metódou na detekciu správ typu spam je algoritmus **XGBoost**.