UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

DETECTAREA TRANZACŢIILOR BANCARE FRAUDULOASE UTILIZÂND ÎNVĂŢARE AUTOMATĂ NESUPERVIZATĂ

Absolvent

Eftimie Petre-Laurenţiu

Coordonator științific

Conf. dr. Cristian Rusu

București, iunie 2024

Rezumat

Detecţia anomaliilor este un subiect ce apare frecvent în cercetarea din domeniul învăţării automate. Acest fapt se datorează multitudinii de aplicaţii într-o gamă largă de industrii.

Totuşi, găsirea unei tehnici bune de rezolvare a problemei este dificilă în cele mai multe cazuri, întrucât este costisitor sau aproape imposibil să obţii date despre evenimentele rare ce merită semnalate.

În această lucrare, explorăm un set de date ce conţine informaţii despre tranzacţii bancare folosind 3 algoritmi de învăţare automată nesupervizată: "One Class SVM", "Gaussian Mixture Model" şi "Kernel Density Estimation". Performanţa acestora din urmă este comparată prin metricile: "accuracy", "precision", "recall" şi "f1 score".

Obiectivul este detectarea tranzacţiilor frauduloase.

Abstract

Anomaly detection is a topic appearing frequently in machine learning research. This is due to its many applications in various industrial settings.

However, finding a good algorithm to solve this problem is difficult in most cases, since obtaining data about rare events that are worth reporting is costly or almost impossible.

In this paper, we analyze a dataset containing information about bank transactions using 3 unsupervised machine learning algorithms: "One Class SVM", "Gaussian Mixture Model" and "Kernel Density Estimation". Their performance is compared using the following metrics: "accuracy", "precision", "recall" and "f1 score".

Our objective is detecting fraudulent transactions.

Cuprins

Capitolul 1

Introducere

1.1 Inteligenţa artificială

Este un domeniu de studiu din Informatică, discutat încă din prima jumătate a secolului al XX-lea, dar care a devenit popular abia la finalul secolului menţionat anterior.

Aceasta are ca scop simularea inteligenţei umane folosind diverşi algoritmi şi structuri de date. Există mai multe arii de cercetare în acest domeniu, precum: raţionamentul, reprezentarea cunoştinţelor, planificarea, procesarea limbajului natural, cât şi învăţarea automată care va fi prezentată detaliat în această lucrare. Totuşi, obiectivul pe termen lung al cercetătorilor este de a simula inteligenţa generală, nu doar părţi specifice din aceasta.

Popularitatea acesteia a început să crească odată cu mărirea spaţiului de stocare, a puterii de procesare, cât şi cu faptul că sistemele de calcul au devenit mai accesibile.

De asemenea, deşi algoritmii existenţi nu s-au mai îmbunătăţit semnificativ, accesul la o cantitate enormă de date în ziua de astăzi, anume fenomenul "Big Data", a oferit un sprijin considerabil pentru performanţa învăţării inteligenţei artificiale[**ai-history**].

În acest domeniu, datele colectate, cât şi calitatea lor au un rol extrem de important. Oricât de bun ar fi algoritmul propus, în lipsa unor date reprezentative pentru problema ce trebuie rezolvată, performanţa va fi îndoielnică. Putem spune că datele sunt cele care introduc particularităţile din lumea reală în spaţiul teoretic al inteligenţei artificiale.

Prin urmare, se observă că aplicabilitatea inteligenţei artificiale a fost împiedicată în principal de considerente practice, nu neapărat teoretice.

Printre succesele recente din domeniu se remarcă AlphaFold, un model inventat de DeepMind care prezice structura proteinelor din secvenţele de amino acizi, GPT-3 al OpenAI ce poate genera text care să răspundă la întrebările utilizatorilor aproape ca o fiinţă umană, dar şi algoritmii pentru maşini autonome dezvoltaţi de cei de la Tesla.

Utilizarea inteligenţei artificiale poate reduce costurile asociate sarcinilor ce ar necesita efortul unei persoane, dar de multe ori apare şi problema eticii, precum în cazul înlocuirii

recrutorilor cu un program ce selectează candidaţii pentru un loc de muncă în funcţie de anumite criterii care nu ar fi întocmai corespunzătoare.

Automatizarea detecţiei anomaliior reprezintă una din posibilele aplicaţii ale inteligenţei artificiale, subiect ce va fi abordat în această lucrare.

1.2 Definiţia anomaliilor

O **anomalie** este o entitate ce diferă semnificativ de restul enităţilor din setul de date. Definiţia lui Hawkins este urmatoarea[**hawkins1980identification**]: *"O anomalie este o observaţie ce deviază atât de mult faţă de restul observaţiilor, încât să creeze suspiciunea că a fost generată de un mecanism diferit"*.

Datele sunt colectate prin observarea unor procese, de preferat din viaţa reală, precum funcţionarea unui motor sau traficul pe o reţea de calculatoare. În majoritatea timpului, procesele generează date ce corespund unei derulări normale, dar în anumite cazuri acestea se comportă anormal şi astfel apar anomaliile.

"Normalitatea" observaţiilor este definită de oameni. Algoritmii de învăţare automată pot să semnaleze potenţialele abateri, dar în final, decizia rămâne a omului.

De asemenea, caracteristicile anomaliilor depind de domeniul unde acestea apar. Ce este considerat anomalie în domeniul medical, va diferi aproape complet de domeniul finanțelor, spre exemplu.

Mai mult, unele observaţii nici măcar nu pot reprezenta comportament anormal în afara unui context bine definit sau fără a apărea în prezența altor observaţii care adunate ar forma o anomalie. Un exemplu de comportament ce poate fi observat în traficul de date pe o rețea al unei organizații ar fi conectarea de la distanță pe un anumit calculator. Acest lucru se întâmplă regulat și nu poate ridica nicio suspiciune de cele mai multe ori. Dar în combinație cu o oră târzie de accesare, când probabil angajații nu mai sunt la lucru, sau cu alte comenzi care sunt folosite succesiv pentru a comite un atac cibernetic, o observaţie ce în general este complet normală, acum prezintă caracteristici anormale.

Prin urmare, se observă că decizia de a eticheta o observaţie ca fiind anomalie sau nu este una dificilă chiar şi pentru un om, iar în combinație cu faptul că obţinerea de date ce ar putea fi anormale este de cele mai multe ori costisitoare, face ca evaluarea sistemelor de detecţie să devină una problematică.

1.3 Avantajele învăţării automate

Când sarcinile ce trebuie îndeplinite periodic, necesare bunei funcţionări a unei companii spre exemplu, încep să ocupe o mare parte din timpul oamenilor, apare nevoia automatizării.

În general, soluţiile ce includ stabilirea unui set de reguli sunt de preferat, datorită simplităţii lor. Un exemplu banal ar fi verificarea siguranţei unei parole introduse de utilizator la crearea unui cont pe o platformă web. Este suficient să verificăm dacă parola are un număr minim precizat de caractere sau dacă conţine anumite simboluri ce reduc vulnerabilitatea în faţa unui atac cibernetic.

Din păcate, multe din problemele cu care ne confruntăm nu au soluţii simple. Aici intervine învăţarea automată ce încearcă să aproximeze posibilul set de reguli, prea complex pentru a fi definit, ce ar putea rezolva problema. Am putea spune că această tehnică "învaţă" setul de reguli din datele puse la dispoziţie, de unde şi denumirea. Evident, o aproximare nu are cum să fie perfectă, dar este suficient de bună încât să scutească oamenii de o mare parte din treabă. În cazul tranzacţiilor bancare, ce probabil sunt efectuate în număr de sute de milioane pe zi, nu este greu de observat că verificarea manuală de către oameni este imposibilă, iar găsirea unui set simplu de reguli care să le caracterizeze din nou este o problemă dificilă. Totuşi, învăţarea automată ne poate ajuta să minimizăm fraudele ce trec nedetectate de la milioane la câteva zeci pe zi.

1.4 Caracterizarea anomaliilor nu este viabilă

Un exemplu ce ilustrează de ce obţinerea de informaţii despre anomalii este costisitoare ar putea fi activitatea unui motor defect.

În acest caz, avem 2 variante:

- Una dintre ele este să **simulăm** într-un fel sau altul comportamentul unui motor defectuos, dar se observă că această operaţiune este greu de realizat şi chiar dacă am reuşi să o ducem la capăt, datele colectate nu ar fi autentice şi ar putea duce la o reprezentare incorectă a anomaliilor.

- A doua variantă implică **sabotarea intenţionată** a motorului pentru a obţine datele, dar acest lucru produce un cost prea mare de cele mai multe ori.

Este evident că definirea anomaliei în acest context nu este viabilă, iar cazul nu este singular. Tranzacţiile frauduloase ce constituie tema acestei lucrări suferă de aceleaşi probleme, ba mai mult, nici măcar nu avem opţiunea de a sabota intenţionat derularea normală a tranzacţiilor. De asemenea, dacă anomaliile apar ca urmare a unor acțiuni cu caracter maliţios, atunci adversarii responsabili pentru acestea vor încerca să evite mecanismele existente de detecţie prin schimbarea frecventă a metodelor folosite. Prin urmare, ce era considerat comportament normal înainte, acum poate indica prezența unui atac.

În schimb, este mult mai uşor şi cel mai probabil implică un cost aproape inexistent, să urmărim activitatea unui motor în stare bună de funcţionare şi să folosim datele respective pentru a defini ce înseamnă o observaţie normală.

1.5 Aplicaţii ale detecţiei anomaliilor

În aproape toate domeniile apare problema semnalării unor fenomene sau evenimente ieşite din comun ce necesită atenţia unei persoane pentru analiză. Mai jos prezentăm doar câteva din cazurile unde detecţia anomaliilor este necesară, cât şi ce ar reprezenta o anomalie pentru fiecare:

- **Fraude financiare**: ne dorim să semnalăm comportamentul ciudat observat într-o serie de tranzacţii, precum activitatea generată de o persoana care foloseşte cardul de credit al altei persoane în mod neautorizat. Această aplicaţie constituie şi obiectul lucrării noastre[**financial-fraud**].
- **Intruziune în reţele de calculatoare**: vrem să semnalăm activitatea neobişnuită ce poate indica un potenţial atac cibernetic sau accesul neautorizat al unui terţ maliţios[**network-traffic**].
- **Controlul calităţii în manufactură**: suntem interesaţi să monitorizăm procesul de producţie pentru a raporta eventualele defecţiuni ce ar afecta calitatea produsului[**quality-control**].
- **Domeniul medical**: vrem să identificăm anomalii în analizele de sânge, semnele vitale sau imaginile medicale ale pacientului pentru a depista şi preveni anumite afecţiuni[**medical-images**].
- **Reţele sociale**: ne dorim să raportăm automat conţinutul postat ce poate include remarci jignitoare, precum comentariile rasiste sau xenofobe, astfel fiind de ajutor în moderarea materialului încărcat pe platforme[**social-media**].

1.6 Anomalii în tranzacţii bancare

Tranzacţiile fraudulente pot produce pagube financiare majore atât instituțiilor responsabile de operaţiunile economice efectuate prin intermediul lor, cât și celor care apelează la respectivele servicii.

Pe lângă prejudiciul material adus, fraudele nedetectate deteriorează și reputația băncilor ce cad victimă acestui fenomen, iar pe termen lung acestea pot duce la pierderea potenţialilor clienți.

De asemenea, în urma unor operaţiuni frauduloase ce nu au fost prevenite, apar și tulburări ale activității economice desfășurate, deoarece trebuie investite resurse și timp pentru investigarea și soluţionarea problemei. Alte costuri suplimentare ce mai pot apărea reprezintă procese juridice îndreptate împotriva instituției financiare afectate, cât și sancțiuni pentru nerespectarea normelor de siguranţă.

Numai Marea Britanie a înregistrat pierderi de peste 1.2 miliarde de lire sterline în 2022 cu aproape 80% din fraude originând din mediul online, conform *UK Finance* [**uk-finance**].

Prin urmare, este nevoie de o soluţie care să poată preveni măcar o mare parte din tranzacţiile frauduloase dacă nu chiar pe toate și care să nu necesite verificarea manuală a unui om decât pentru un număr mic de evenimente.

1.7 Detecţia anomaliilor vs clasificare clasică

1.7.1 Greşeală comună

La prima vedere, aceste două probleme par sa coincidă şi ne determină să ne întrebăm de ce a mai apărut un nou domeniu de cercetare, anume detecţia anomaliilor, când deja avem la dispoziţie atâtea rezultate utile pentru problema clasificării. Din păcate, dacă nu cunoaştem numărul de clase ce pot apărea sau nu avem un eşantion reprezentativ pentru toate tipurile de clase, fapt ce se întâmplă deseori în practică, nu putem utiliza un clasificator clasic. De exemplu, în cazul tranzacţiilor bancare, nu putem şti cu certitudine care sunt toate tipurile de fraudă ce pot apărea şi nici nu avem un număr suficient de exemple pentru a defini măcar tipurile de fraudă pe care le-am identificat până acum. Prin urmare, este mult mai uşor să modelăm problema în jurul unei singure clase de referință, decât să încercăm definirea unei clase adiţionale ce de fapt este compusă dintr-o multitudine de subclase necunoscute.

De asemenea, problemele de clasificare de cele mai multe ori presupun accesul la etichetele asociate fiecărei observaţii, deci metodele supervizate sunt cele mai potrivite aici. Totuşi, și dacă setul de date este etichetat, lucru ce se întâmplă în cel mai bun caz, un clasificator clasic ar avea o performanță slabă pentru problema detecției anomaliilor din cauza dificultății modelării claselor, după cum am menționat și mai sus.

De cele mai multe ori nu vom avea etichetele datelor colectate, sau cel puțin nu un număr suficient de mare, pentru detecţia anomaliilor, așa că apare nevoia metodelor nesupervizate. În realitate, totuşi, este posibil ca în timp să acumulăm măcar câteva etichete, chiar dacă la început soluţia este pur nesupervizată, pe care le putem exploata folosind metode semi-supervizate care în general au o performanță mai bună decât cele menționate anterior.

1.7.2 Scopul detecţiei anomaliilor

Diferenţa constă în faptul că detecţia anomaliilor are la bază **o singură clasă de referință**. Observaţia fie aparţine acestei clase, fie aparţine oricărei alte clase care este diferită de aceasta din urmă. Multitudinea de "clase diferite" ne indică faptul că este greu

să definim ce înseamnă *"diferit de normal"*, dar este relativ uşor să definim ce înseamnă *"normal"*.

Un exemplu trivial ar putea fi să detectăm dacă animalul din imagine este un câine sau nu. În acest caz, normal înseamnă câine, iar anormal înseamnă orice alt animal care nu este câine. Se observă că este relativ uşor să caracterizăm conceptul de câine, în timp ce definirea conceptului de "diferit de câine" este complexă, fapt ce ar pune în dificultate un clasificator clasic care prin definiţie are nevoie de un număr cunoscut de clase cu observaţiile reprezentative aferente.

1.7.3 Scopul clasificării clasice

La clasificarea clasică, clasele ce trebuie identificate sunt bine definite şi de cele mai multe ori, trasăturile lor se suprapun în mai multe locuri. De asemenea, această problemă acordă o importanţă **egală** tuturor categoriilor implicate, pe când eşecul de a semnala o anomalie este în general mult mai dăunător faţă de detectarea unei observaţii normale ca fiind anormală.

Pentru a continua exemplul precedent cu imaginile cu animale, putem reformula problema ca de această dată să diferenţiem între pisici şi câini. Deja se vede că ambele clase sunt mult mai bine definite şi putem găsi atât trăsături comune, precum nasul, cât şi trăsături definitorii, precum mustăţile. Vrem să detectăm la fel de bine ambele animale, pe când într-o problemă clasică de detecţie a anomaliilor, precum identificarea tranzacţiilor frauduloase, vrem preponderent să nu ratăm evenimentele cu caracter maliţios.

1.8 Probleme conexe pentru detecţia anomaliilor

1.8.1 Outlier detection

Outlier detection are scopul de a identifica observaţii fie noi, fie previzibile, din setul de antrenare, care deviază de la comportamentul normal. **Outliers** pot fi valori extreme, erori cauzate de zgomot, posibil din instrumentele de măsură sau senzorii folosiţi pentru colectarea datelor, sau comportament anormal.

Această abordare este utilă atunci când avem un set de date **"poluat"** cu observaţii anormale şi ne dorim să extragem din el doar porţiunea ce conţine observaţii deviante. De asemenea, se poate folosi atât pe seturi adnotate, cât şi pe seturi fără etichete.

1.8.2 Novelty detection

Novelty detection are scopul de a identifica observaţii noi ce diferă semnificativ de datele de antrenare care au rolul de a caracteriza cât mai bine clasa normală. Din acest motiv, este foarte important ca setul de antrenare să nu conţină decât date normale pentru

a putea identifica caracteristicile clasei de referință, astfel modelând comportamentul normal.

Prin urmare, această abordare este utilă atunci cand avem un set de date **"curat"** (fără anomalii) şi ne aşteptăm ca observaţiile noi să fie rare şi să aibă trăsături distincte faţă de setul de antrenare. De asemenea, se presupune că datele sunt adnotate pentru a putea extrage punctele cu eticheta normală şi să le folosim la antrenare.

Novelty detection este problema pe care o vom studia detaliat în această lucrare.

1.8.3 Change detection

Change detection este folosit atunci când dorim să găsim anomalii într-o serie de timp, anume să analizăm dacă comportamentul seriei se schimbă semnificativ. De obicei, suntem interesați ori doar să aflăm dacă o schimbare a apărut, ori să aflăm pozițiile în timp unde schimbările respective au avut loc.

Aici se remarcă 2 categorii, **offline**, unde presupunem că avem la dispoziţie o serie de timp de lungime finită pe care o analizăm cu scopul de a detecta puncte anormale, și **online**, unde avem un flux de date din care vrem să extragem, în timp real, punctele problematice. Deși este probabil impropriu, putem considera pentru seriile de timp, intuitiv, că outlier detection este similar cu cazul offline din change detection, iar novelty detection este asemănător cu cel online din această problemă.

În această lucrare, nu vom aborda problema change detection.

Capitolul 2

Preliminarii

2.1 Tipuri de învăţare automată

Există numeroase tipuri de învăţare automată, aşa că le vom prezenta doar pe cele relevante lucrării noastre. De asemenea, notațiile pentru setul de date introduse în această secțiune vor fi folosite și în restul lucrării.

2.1.1 Supervizată

Formal, avem un set de date cu N elemente, sub formă de perechi:

$$S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \tag{2.1}$$

unde observaţiile $x_i \in \mathbb{R}^d$ și etichetele $y_i \in \mathbb{R}$ sau $y_i \in \{0, 1\}$. Problema se reduce la aproximarea cât mai bună a etichetei y_{N+1} pentru o observaţie nouă x_{N+1}.

Învăţarea **supervizată** poartă acest nume deoarece necesită supraveghere umană pentru a putea funcţiona. Un specialist în ştiinţa datelor trebuie să parcurgă fiecare observaţie din setul de date şi să îi atribuie o **etichetă** corespunzătoare. Acest lucru este evident dificil, având în vedere că seturile de date cu minim sute de mii de entităţi sunt des întâlnite.

Această metodă este folosită în probleme de clasificare sau de prezicere a unor fenomene.

2.1.2 Nesupervizată

Formal, avem un set de date cu N elemente:

$$S = \{x_1, x_2, \dots, x_N\} \tag{2.2}$$

unde observaţiile $x_i \in \mathbb{R}^d$, dar nu mai apar etichetele y_i. În lipsa etichetelor, problema

generalizată nu mai poate fi descrisă la fel de ușor.

Învăţarea **nesupervizată** este opusul celei supervizate, deci implică faptul că interacţiunea umană nu este necesară în pregătirea setului de date. Cu toate acestea, rezultatele trebuie să fie interpretate de o persoană pentru a fi relevante, întrucât nu mai avem etichete pe care să le folosim în evaluarea automată a performanţei.

Această metodă este folosită pentru a grupa datele în funcţie de similaritate, a înţelege relaţia dintre punctele din setul de date şi pentru a face o analiză iniţială a datelor.

Toţi algoritmii din această lucrare aparţin acestei metode de învăţare automată.

2.1.3 Semi-supervizată

Formal, avem un set de date cu N elemente, partiționat în 2 mulțimi cu N_1 elemente sub formă de perechi, respectiv N_2 elemente:

$$S_1 = \{(x_1, y_1), (x_2, y_2), \dots, (x_{N_1}, y_{N_1})\} \tag{2.3}$$

$$S_2 = \{x_1, x_2, \dots, x_{N_2}\} \tag{2.4}$$

unde observaţiile $x_i \in \mathbb{R}^d$ și etichetele $y_i \in \mathbb{R}$ sau $y_i \in \{0, 1\}$. La fel ca în cazul supervizat, problema este în general aproximarea cât mai bună a etichetei y_{N+1} pentru o observaţie nouă x_{N+1}.

Învăţarea **semi-supervizată** îmbină ambele paradigme prezentate anterior, astfel că necesită un număr mic de date adnotate, lucru ce este evident mai uşor de obţinut faţă de un întreg set, şi un număr mare de date fara etichetă.

Această metodă este folosită spre exemplu în modelele ce se antrenează singure, folosind un algoritm supervizat antrenat pe datele adnotate ce este apoi folosit pe datele fara etichetă pentru a obţine un nou set de date adnotat.

Deşi această metodă nu este folosită în lucrarea noastră, am inclus-o deoarece ideea de bază în novelty detection cu metode nesupervizate este similară. Nu avem nevoie de date adnotate la antrenare şi este suficient un set mic pentru testarea ulterioară a performanţei.

2.2 One Class SVM

2.2.1 Ideea algoritmului

Această metodă este inspirată din clasificatorul cu vectori suport. Ideea este sa găsim **un hiperplan cu margine maximă**, posibil într-un spaţiu cu mai multe dimensiuni decât cel iniţial, în funcţie de kernel, care să separe originea (se presupune că punctele sunt centrate) spaţiului de trăsături de restul punctelor din setul de date[**scholkopf2000support**].

Un alt mod de a privi algoritmul este găsirea celei mai **mici hipersfere** care să includă toate punctele din setul de date[**tax2004support**].

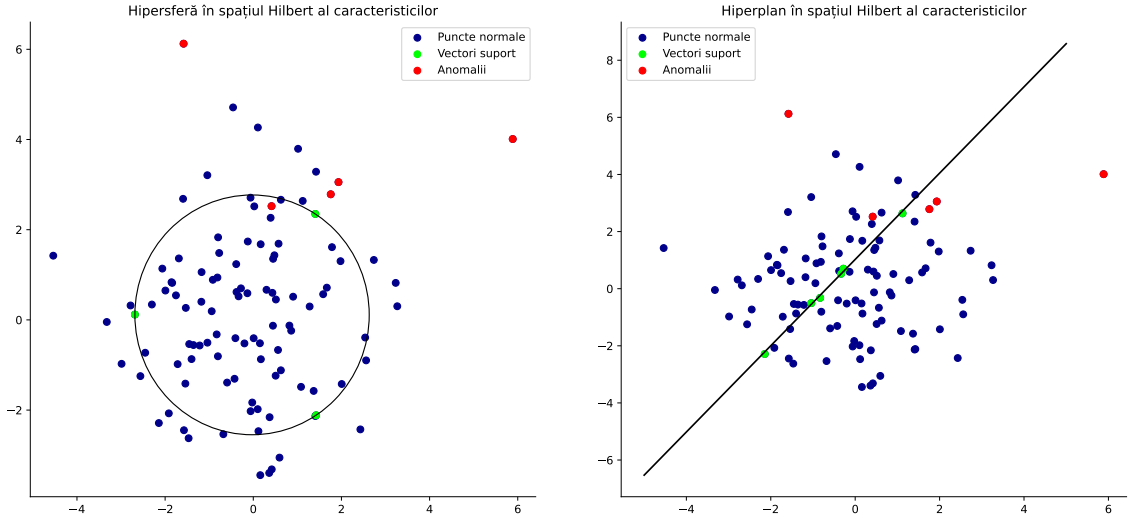

Figura 2.1: Varianta Schölkopf et al în dreapta şi varianta Tax et al în stânga

2.3 Caracteristici

One Class SVM ne ajută să transformăm un model de clasificare cu mai multe clase, anume SVM, într-unul cu o singură clasă, păstrând abilitatea de a putea introduce neliniarități cu ajutorul funcţiilor de kernel ce au fost studiate extensiv. Prin urmare, are majoritatea avantajelor celui din urmă, precum garanția optimului global datorită funcţiei convexe pe care o minimizăm, faptul că este relativ robust în spaţii cu multe dimensiuni și că poate fi folosit chiar și dacă numărul de caracteristici al punctelor din set este mai mare decât cardinalul setului.

Totuşi, acest model devine imposibil de folosit pentru seturi mari de date, complexitatea de timp lasă de dorit la antrenare, iar găsirea hiperparametrilor optimi este dificilă, atât din lipsa unor reguli stricte după care să ne ghidăm, dar și din faptul că o căutare exhaustivă ar dura prea mult din cauza complexității de timp ridicate la antrenare. De asemenea, rezultatele date de acest model nu pot fi ușor interpretate, deoarece ele reprezintă distanțe, nu probabilități.

2.3.1 Formularea matematică

Prima formulare ce implică un hiperplan de separare

$$\begin{aligned} \min_{w,\rho,\xi} \quad & \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ \text{cu constrângerea} \quad & \langle w, \phi(x_i)\rangle \geq \rho - \xi_i, \quad i = 1, 2, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{2.5}$$

- w este vectorul de pondere al hiperplanului
- ρ este termenul de influenţă
- ξ_i sunt variabilele de relaxare pentru încălcarea marginii
- $\phi(x_i)$ este funcţia de scufundare pentru x_i.
- n este numărul total de puncte
- ν este marginea superioară pentru ponderea de anomalii şi marginea inferioară pentru ponderea de vectori suport

împreună cu forma sa dual ce implică folosirea unei funcţii kernel pentru găsirea unui hiperplan de separare într-un spaţiu cu mai multe dimensiuni decât cel iniţial

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(x_i, x_j) \\ \text{cu constrângerea} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu n}, \quad i = 1, 2, \ldots, n \\ & \sum_{i=1}^{n}\alpha_i = 1 \end{aligned} \tag{2.6}$$

- α_i sunt variabilele duale asociate punctelor
- $K(x_i, x_j)$ este funcţia kernel

A doua formulare ce implică găsirea hipersferei minime

$$\begin{aligned} \min_{R,c,\xi} \quad & R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \\ \text{cu constrângerea} \quad & \|\phi(x_i) - c\|^2 \leq R^2 + \xi_i, \quad i = 1, 2, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{2.7}$$

- R este raza hipersferei
- c este centrul hipersferei

2.3.2 Tehnici de optimizare

Toate funcțiile prezentate mai sus sunt convexe, iar constrângerile lor sunt mulțimi convexe. Chiar și prima constrângere din formularea cu hipersferă care conține un termen la pătrat în dreapta inegalității poate fi redusă la o constrângere liniară introducând o nouă variabilă $R_2 = R^2$ cu constrângerea $R_2 >= 0$.

Deci știm că problemele de optimizare sunt convexe și astfel avem garanția unui unic minim global. Prin urmare, putem folosi tehnicile populare de optimizare convexă.

O metodă des întâlnită de ordinul întâi este coborârea pe gradient ce este avantajoasă atunci când setul de antrenare este mare, întrucât complexitatea algoritmului nu depinde de mărimea setului de date. Totuşi, această tehnică întâmpină dificultăţi la optimizarea funcţiilor cu constrângeri. Există modificări ale tehnicii de bază, precum coborârea pe gradient proiectată care poate lucra şi cu constrângeri liniare simple sau algoritmul Pegasos[**Pegasos**] care poate rezolva varianta primitivă a primei forme prezentate mai sus, dar pentru formulările complexe nu se poate aplica această metodă.

În majoritatea cazurilor, un alt kernel decât cel liniar va fi folosit, şi prin urmare vor apărea următoarele dificultăţi: calcularea valorilor funcţiei kernel este o operaţie costisitoare, calcularea matricei kernel este de cele mai multe ori inutilă, întrucât ne interesează doar calculele ce implică un vector suport, şi matricea kernel de cele mai multe ori nu va încăpea în memorie din cauza numărului excesiv de exemple de antrenare, având în vedere că dimensiunea ei este de $N \times N$[**SVM-solvers**].

Astfel, apare nevoia unor algoritmi specializaţi. Cel mai popular dintre ei este SMO[**SMO**] care foloseşte tehnici de programare pătratică şi care utilizează o descompunere în subprobleme mai mici ce pot fi rezolvate eficient şi care încap în memorie. O implementare a acestei metode se poate găsi în biblioteca LIBSVM[**LIBSVM**].

2.4 Gaussian Mixture Model

2.4.1 Ideea algoritmului

Este prea restrictiv să presupunem ca fiecare punct dintr-un set de date provine din aceeaşi distribuţie unimodală. Prin urmare, a apărut tehnica de bază **"Mixture model"** ce îşi propune să elimine presupunerea anterioară pentru a putea modela şi distribuţii multimodale care poate nu provin nici măcar din aceeaşi familie, utilizând o sumă ponderată de mai multe componente cu proprietăţi cunoscute. Totuşi, în practică, se folosesc componente din aceeaşi familie pentru modelarea distribuţiei doar că fiecare componentă are parametri diferiţi.

Algoritmul încearcă să estimeze funcţia densitate de probabilitate din care au fost generate datele folosind **mai multe distribuţii Gaussiene**. Astfel, putem modela distribuţii multimodale utilizând o distribuţie bine cunoscută. Parametrii necesari sunt mediile,

matricele de covarianţă și ponderile fiecărei componente.

Metoda este similară cu **k-means**, întrucât ambele folosesc măsuri de similaritate pentru a crea grupuri ce modelează datele. Totuşi, față de k-means, Gaussian Mixture Model permite unui punct să aparțină mai multor grupuri, oferind probabilități pentru fiecare, și poate modela chiar și seturi de date complexe cu margini de decizie neliniare.

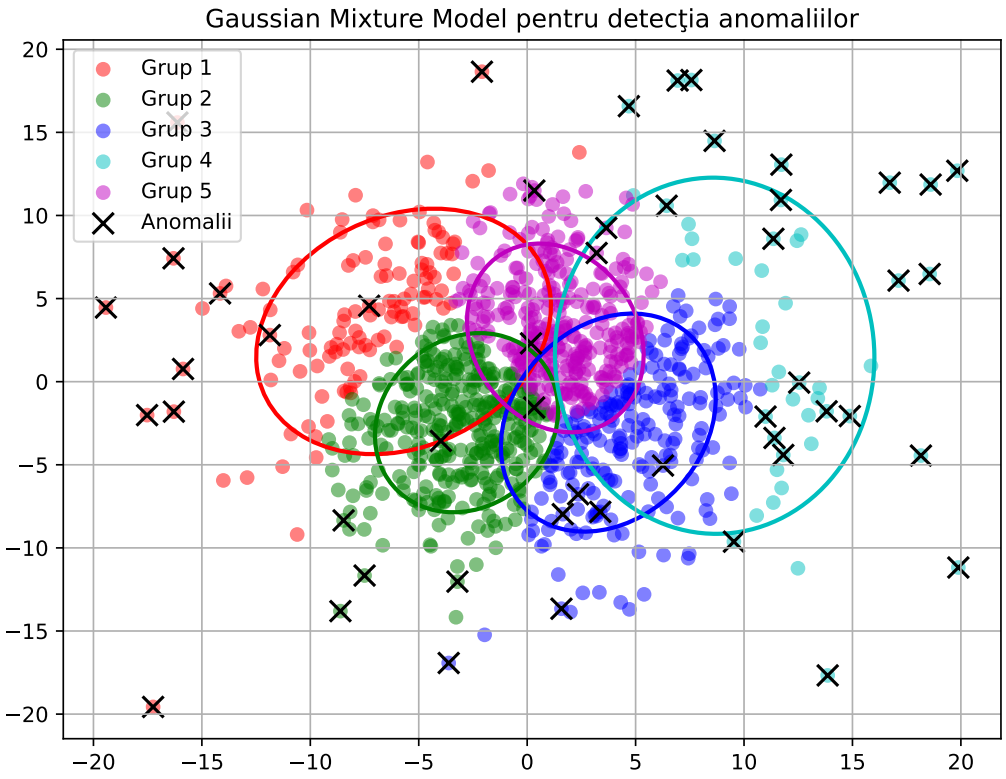

Figura 2.2: Detecţia anomaliilor cu 5 componente Gaussiene

2.5 Caracteristici

Gaussian Mixture Model este util pentru modelarea distribuţiilor complexe și se descurcă bine chiar și când setul de date este compus din mulțimi suprapuse de puncte, întrucât află pentru fiecare componentă probabilitatea ca o observaţie să fie membra acesteia, deci chiar dacă un punct se află în mai multe mulțimi, nu va fi repartizat doar în una singură. De asemenea, timpul de antrenare este redus chiar și pentru seturi mari de date datorită tehnicii iterative de optimizare utilizată.

Din păcate, numărul de componente nu este ușor de ales pentru a obține un model optim, iar performanţa algoritmului depinde foarte mult de inițializarea parametrilor inițiali, ceea ce poate duce la imposibilitatea convergenței către optim. De asemenea,

se presupune că datele pot fi modelate de o combinație de distribuţii Gaussiane, dar dacă acest lucru nu este adevărat, algoritmul nu va oferi rezultate bune.

Pentru reducerea numărului de parametri ce trebuie învățați, se poate face presupunerea că caracteristicile datelor sunt independente unele de altele. Acest lucru este echivalent cu presupunerea că valorile ce nu se află pe diagonala principală a matricelor de covarianţă pot fi considerate 0, dar în practică, datele pot fi corelate de-a lungul diferitelor dimensiuni, ceea ce ar conduce la învăţarea unui număr de d^2 parametri pentru fiecare componentă, unde d este dimensiunea unui punct. Acest lucru ar putea duce la overfitting când setul de date este mic[**aggarwal2017outlier**]. Un alt dezavantaj este vulnerabilitatea la blestemul dimensionalității.

2.5.1 Formularea matematică

Funcţia densitate de probabilitate estimată este dată de

$$\mathrm{p}(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{2.8}$$

- K este numărul de componente Gaussiene
- $\mathcal{N}(x|\mu_i, \Sigma_i)$ este distribuţia Gaussiană cu medie μ_i şi matrice de covarianţă Σ_i
- ϕ_i este ponderea componentei i

Totuşi, pentru a detecta anomalii avem nevoie de

$$p'(x) = 1 - \prod_{i=1}^{K} (1 - p_k(x)) \tag{2.9}$$

- $p_k(x)$ este probabilitatea ca punctul x să aparțină componentei k

Această formulă ne indică probabilitatea ca un punct să fi fost generat de oricare dintre componentele Gaussiene implicate. Deci, o valoare cât mai mică sugerează o şansă mare ca punctul să fie anomalie.

2.5.2 Tehnici de optimizare

Pentru Gaussian Mixture Model nu se poate găsi nicio soluție în formă închisă din cauza faptului că valorile optime ale parametrilor depind de probabilitatea datelor de a aparține fiecărei componente, dar în același timp, calcularea acestor probabilități depinde de parametrii optimi.

Prin umare, este nevoie de o soluție iterativă care aproximează alternativ valorile căutate mai sus. Cea mai populară metodă este algoritmul **Expectation-Maximization**

(EM). Acesta este compus din 2 pași care sunt repetați până când eroarea față de soluția optimă scade sub un anumit prag. La pasul E, folosim valorile actuale ale parametrilor pentru a calcula probabilitatea că un punct a fost generat de o anumită componentă, făcând acest lucru pentru toate punctele și toate componentele. La pasul M, utilizăm probabilităţile calculate la pasul E pentru a estima valorile parametrilor optimi cu metoda **Maximum Likelihood Estimation** care caută să maximizeze probabilitatea de observare a datelor sub un anumit model statistic.

Numărul de componente poate fi ales ori folosind anumite criterii, precum **Bayesian information criterion** și **Akaike information criterion**, ori folosind cunoștințe din domeniul problemei.

Pentru inițializarea parametrilor se poate folosi algoritmul k-means. Acesta ne va da direct valori inițiale pentru mediile grupurilor. Pentru matricele de covarianţă putem folosi covarianţele calculate pentru punctele din fiecare grup, iar pentru ponderi vom folosi proporția de puncte din setul de date aparținând fiecărui grup[**EM-GMM-INIT**].

2.6 Kernel Density Estimation

2.6.1 Ideea algoritmului

Precum Gaussian Mixture Model, algoritmul încearcă să estimeze funcţia densitate de probabilitate din care au fost generate datele.

Tehnica de bază utilizează un kernel funcţie de distribuţie probabilitate pe care "îl plasăm" la fiecare punct din setul de date, căruia îi oferim o pondere de $\frac{1}{n}$, unde n este numărul de observaţii. Apoi, distribuţia adevărată este aproximată prin adunarea tuturor rezultatelor precedente. Procesul este similar aflării ariei de sub graficul unei funcţii cu ajutorul unei integrale ce apare dintr-o sumă de valori ale funcţiei într-o infinitate de puncte.

Altă tehnică similară de estimare a funcţiei densitate de probabilitate este cea a **histogramelor** care se bazează pe discretizarea datelor în coșuri de mărime fixă. Totuși, acest lucru irosește informația despre locațiile individuale ale punctelor, înlocuindu-le cu intervale ce corespund mai multor puncte. Astfel, graficul funcţiei va deveni discontinuu și constant pe fiecare interval. Kernel Density Estimation produce un grafic neted de cele mai multe ori și oferă o reprezentare mai bună a distribuției unei variabile continue[**KDE_paper**].

Un parametru numit **"lăţime de bandă"** influenţează netezimea distribuţiei rezultate. Cel mai des utilizat kernel este cel Gaussian şi pe acesta îl vom folosi şi noi.

Aici, pentru fiecare punct generăm o distribuţie Gaussiană cu **media** egală cu punctul respectiv şi **deviaţie** egală cu "lăţimea de bandă". Apoi, adunăm toate distribuţiile obţinute mai sus si le împărţim la numărul total de puncte.

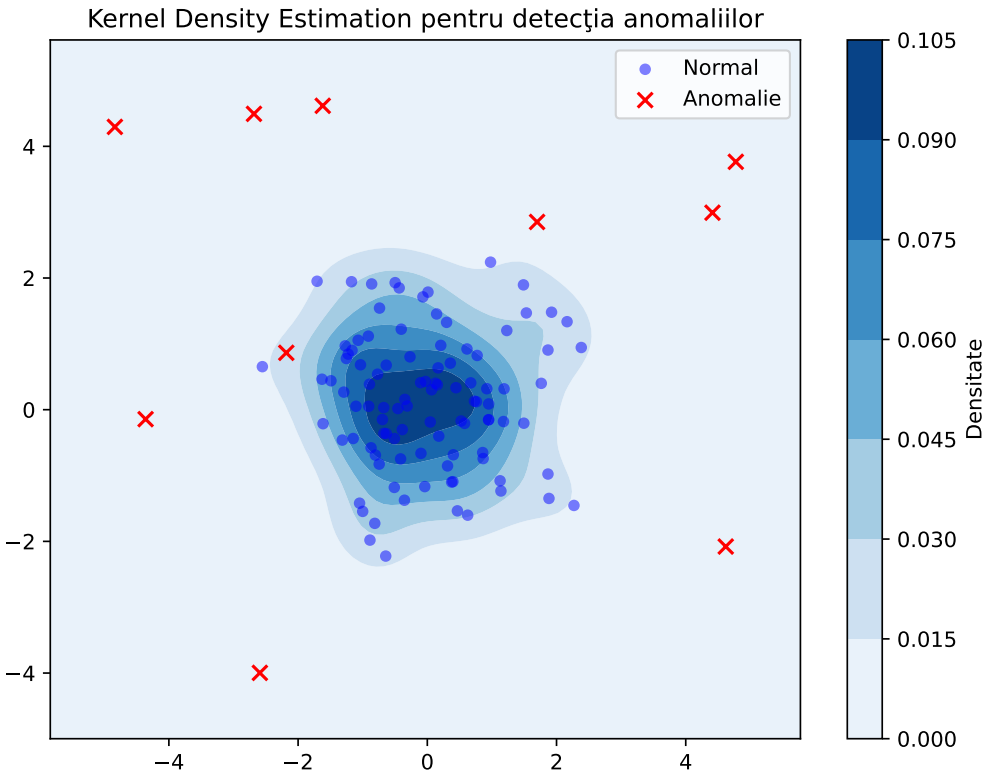

Figura 2.3: Anomaliile sunt reprezentate de punctele roşii

2.7 Caracteristici

Kernel Density Estimation are avantajul de a fi o metoda neparametrică și care nu face nicio presupunere asupra distribuţiei reale a datelor, ceea ce îl face flexibil pentru diverse tipuri de date. De asemenea, rezultatele sunt ușor de interpretat acestea fiind probabilități și beneficiem de informație locală despre densitatea datelor în jurul fiecărui punct din set, ceea ce ne ajută la găsirea regiunilor cu densitate ridicată sau scăzută.

Dezavantajele sunt găsirea dificilă a hiperparametrului optim pentru lăţime de bandă, complexitatea de timp crescută, în special pentru seturi de date mari, vulnerabilitatea la blestemul dimensionalității, dar și tendința de a atribui o densitate ridicată punctelor din marginea setului de date, chiar dacă distribuţia nu se mai întinde în direcția respectivă, iar acest lucru poate duce la netezirea excesivă a marginilor distribuţiei.

2.7.1 Formularea matematică

Densitatea estimată de kernel într-un punct x este dată de

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.10}$$

- n este numărul total de puncte
- h este lăţimea de bandă
- $K(u)$ este funcţia kernel

2.7.2 Tehnici de optimizare

Kernel Density Estimation este un model din familia celor care se bazează pe utilizarea unui număr finit dintre cei mai apropiaţi vecini ai unui punct pentru a lua o decizie. Pentru a cuantifica cât de apropiate sunt 2 puncte, se pot folosi oricare dintre metricile populare, precum distanța euclidiană, sau funcţiile kernel în cazul acestui model, spre exemplu.

Apartenenţa la această familie de algoritmi implică nevoia de a memora setul de date pe care este antrenat, etapa de învăţare, practic, nefiind existentă. Totuşi, o abordare cu forţă brută ar necesita calcularea funcţiei kernel între toate perechile de puncte memorate şi cele pentru care dorim să aflăm densitatea de probabilitate, adică o complexitate de timp de $O(NM)$, unde N este numărul de puncte de antrenare şi M este numărul de puncte evaluate.

O îmbunătăţire ar putea fi utilizarea unei structuri de date ce oferă căutari rapide ale punctelor într-un anumit spaţiu, precum **KD Trees** şi **Ball trees**, aceasta din urmă fiind adesea utilizată pentru date cu dimensiuni $d < 20$. Pentru seturi de date mici sau pentru cazurile în care ne dorim să evaluăm un număr M mic de puncte, este mai util să folosim forţa brută, întrucât crearea arborilor din KD Trees şi Ball trees poate dura mai mult decât simpla calculare a funcţiei kernel între toate perechile. În restul cazurilor însă, timpul de preprocesare devine amortizat de evaluările ce se pot face în timp $O(M \log(N))$. Implementări pentru aceste structuri de date se găsesc în biblioteca scikit-learn[**scikit-learn**].

Pentru acest model nu este prea relevantă funcţia kernel aleasă, hiperparametrul h fiind cel care influenţează mai mult performanţa.

Metodele prezentate mai sus pot estima funcţia de densitate probabilitate doar pentru un singur h. Dacă dorim să testăm performanţa pe mai multe valori h, trebuie să rulăm acei algoritmi de mai multe ori. Acest lucru va duce la o perioadă mult prea mare de căutare a hiperparametrului.

Astfel, a apărut şi o variantă generalizată a modelelor ce foloseau un h fix, menită să scadă timpul de căutare a valorii optime pentru lăţimea de bandă[**Rapid-KDE**].

2.8 Metrici de performanţă

Setul de date prezentat în această lucrare este adnotat în întregime, aşa că putem folosi aceleaşi tehnici de evaluare a performanţei utilizate pentru clasificarea binară.

Prin urmare, are sens să folosim următoarele noţiuni ce vor fi utile pentru descrierea metricilor de performanţă prezentate mai jos:

- **TP** (*true positive*) - reprezintă observaţiile din clasa **pozitivă** ce au fost clasificate **corect**
- **TN** (*true negative*) - reprezintă observaţiile din clasa **negativă** ce au fost clasificate **corect**
- **FP** (*false positive*) - reprezintă observaţiile din clasa **pozitivă** ce au fost clasificate **greşit**
- **FN** (*false negative*) - reprezintă observaţiile din clasa **negativă** ce au fost clasificate **greşit**

În cazul nostru, clasa **pozitivă** este reprezentată de **anomalii**, iar clasa **negativă** este reprezentată de datele **normale**.

Metricile descrise în această lucrare reprezintă o alegere personală pe care o facem, astfel încât să reflecte cât mai bine nevoile problemei expuse şi nicidecum nu reprezintă singura sau cea mai bună cale de a evalua performanţa algoritmilor. Putem pune în paralelă cu teorema **"No Free Lunch"** care ne spune că nu există un model care să fie cel mai bun în toate situaţiile, ci că utilitatea acestuia depinde strict de context. La fel este şi în cazul măsurilor de evaluare, iar acest fapt face găsirea uneltei potrivite pentru problemă una nu tocmai simplă, ba chiar poate necesita un timp considerabil de gândire.

2.8.1 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

Această metrică ne indică câte **clasificări făcute de model au fost corecte din totalul de puncte care trebuie clasificate**.

2.8.2 Precision

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.12}$$

Precision ne indică **capacitatea modelului de a nu produce fals pozitive**, în cazul nostru, de a nu raporta o valoare normală ca fiind anomalie.

2.8.3 Recall

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.13}$$

Recall ne indică **capacitatea modelului de a identifica toate observaţiile pozitive**, în cazul nostru, de a detecta toate anomaliile.

2.8.4 F1 score

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.14}$$

F1 score reprezintă **media armonică dintre precision şi recall**. Prin urmare, f1 score va tinde către valoarea mai mică dintre aceste 2 metrici. Pentru a o maximiza, ar trebui sa avem o valoare mare atât pentru precision, cât şi pentru recall, fapt ce ar duce la un model ideal.

2.8.5 AUC şi ROC Curve

ROC Curve ne ajută să evaluăm calitatea modelului prin reprezentarea grafică a ratei de fals pozitiv pe axa X şi a ratei de adevărat pozitiv pe axa Y. **Punctul ideal al graficului se află în colţul din stânga sus** pentru ca ne dorim o rată de fals pozitiv egală cu 0 şi o rată de adevărat pozitiv egală cu 1. Prin urmare, ne dorim sa maximizăm rata de adevărat pozitiv şi de a minimiza rata de fals pozitiv.

Pentru a crea graficul, avem nevoie de **probabilităţile sau valorile de încredere** pentru fiecare observaţie din setul de testare, generate de funcţia de decizie a modelului respectiv. Punctele de pe grafic pot fi văzute precum clasificatoare separate ce diferă prin **pragul** aplicat funcţiei de decizie. Prin urmare, dacă dorim să ilustrăm ROC Curve, avem nevoie de un algoritm ce are ca valori de ieşire scoruri care pot fi comparate. One Class SVM, prin definiţie, nu oferă astfel de rezultate, aşa că va trebui să îl tratăm în mod diferit.

Funcţia de decizie este cea care atribuie un scor pentru un punct din set cu scopul de a indica nivelul de normalitate sau de anomalie al acestuia. Generarea etichetei se face apoi folosind un prag în cazul probabilităţilor, precum Gaussian Mixture Model şi Kernel Density Estimation, sau efectiv reducând valorile pozitive la $+1$ şi pe cele negative la -1, precum One Class SVM.

Totuşi, ROC Curve are caracter vizual şi nu ne oferă o măsură concretă a performanţei. De aceea, avem nevoie de **AUC**, valoare ce reprezintă aria de sub grafic. Cu cât aria este mai mare, cu atât modelul este mai bun.

2.8.6 Micro average vs Macro average

Aceste 2 tehnici pot fi folosite și în cazul clasificării cu 2 clase, dar ele au apărut în principal pentru a servi ca generalizare a metricilor clasice de performanţă care de unele singure tratează doar cazul **One-versus-All**, anume când considerăm că există doar 2 clase, una care conține clasa țintă și una care conține toate celelalte clase diferite de cea din urmă. Astfel, putem combina măsurile de evaluare deja existente pentru a obține rezultate relevante în cazul general.

Nu există o singură metodă de evaluare a modelelor care să fie potrivită în toate cazurile. În schimb, metodele sunt alese astfel încât să reflecte cât mai bine nevoile problemei.

Macro average pentru o măsură de evaluare are forma:

$$B_{macro} = \frac{1}{q}\sum_{\lambda=1}^{q} B(tp_\lambda, tn_\lambda, fp_\lambda, fn_\lambda) \tag{2.15}$$

Micro average pentru o măsură de evaluare are forma:

$$B_{micro} = B(\sum_{\lambda=1}^{q} tp_\lambda, \sum_{\lambda=1}^{q} tn_\lambda, \sum_{\lambda=1}^{q} fp_\lambda, \sum_{\lambda=1}^{q} fn_\lambda) \tag{2.16}$$

$$L = \{\lambda_j : j = 1, ..., q\}$$

este setul tuturor etichetelor asociate claselor, iar

$$B(tp, tn, fp, fn)$$

este o măsură de evaluare binară bazată pe noţiunile introduse mai sus [**Asch2013MacroandME**]. În cazul nostru, $\lambda = 2$.

Diferenţa între cele 2 metode este că macro average acordă o importanţă **egală fiecărei clase**, pe când micro average acordă o importanţă **egală fiecărei observaţii**. Prin urmare, varianta micro favorizează clasa **majoritară** la calcularea scorului, în timp ce varianta macro favorizează clasa **minoritară**.

Ilustrăm aceste diferenţe printr-un exemplu ce are ca măsură de evaluare binară **accuracy**, şi care arată cum metodele acoperă nevoi diferite.

Ponderea claselor este extrem de neechilibrată în setul nostru de date, anomaliile reprezentând doar 0.017% din total. Prin urmare, dacă dorim să maximizăm micro average accuracy, putem alege un model trivial ce mereu prezice clasa majoritară. Am obţine un accuracy de peste 99.9% cu un minim de efort!

Este impresionant, dar modelul de mai sus este practic inutil pentru problema noastră. Toate anomaliile ar trece nedetectate. În schimb, dacă am evalua acelaşi model folosind varianta macro average, am obţine un accuracy de doar 50%. Modelul este acum inutil,

întrucât suntem interesaţi să detectăm anomaliile, nu doar să punem eticheta corectă pe cât mai multe observaţii indiferent de clasă.

Astfel, vom folosi varianta macro average pentru accuracy, iar pentru precision, recall şi f1 score, vom calcula rezultatul folosind clasa minoritară ca referinţă. Decizia este influenţată de faptul că vrem să urmărim performanţa modelului pe detectarea anomaliilor în special, cu ajutorul precision şi recall, dar în acelaşi timp vrem să obţinem un echilibru între fals pozitive şi adevărat pozitive, utilizând accuracy împreună cu f1 score, metode ce iau în calcul performanţa per total pe ambele clase. Este important să detectăm cât mai multe fraude, dar în acelaşi timp nu ne dorim să semnalăm un număr prea mare de tranzacţii ca fiind problematice deoarece modelul ar deveni un inconvenient.

Capitolul 3

Explorarea setului de date

3.1 Descriere

Acest set de date conţine informaţii despre 284,807 de tranzacţii dintre care 492 sunt **frauduloase**. Conţine doar variabile numerice.

Din motive de confidenţialitate, dimensiunea trăsăturilor originale a fost redusă folosind **Principal Component Analysis** (PCA) la 28 de trăsături denumite "V1-V28" şi încă 2 trăsături ce nu au fost transformate, anume suma de bani a tranzacţiei şi timpul relativ, începând de la 0, când aceasta a fost făcută. De asemenea, pentru fiecare tranzacţie avem eticheta 0 sau 1, daca este normală sau, respectiv, frauduloasă.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	0
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	0

284807 rows × 31 columns

Figura 3.1: Cele 284,807 de tranzacţii

3.2 Matricea de corelaţie

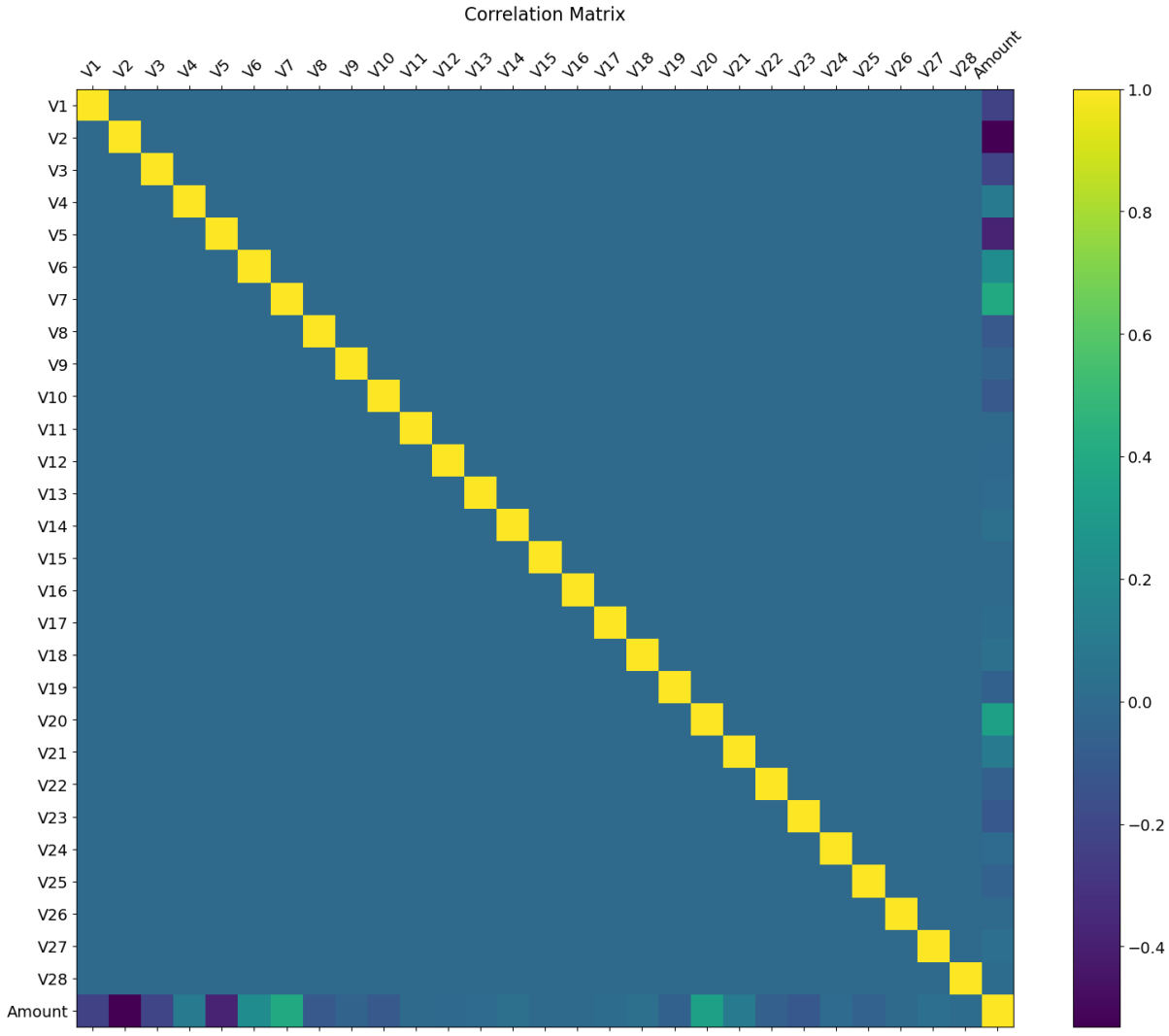

Figura 3.2: Matricea de corelaţie

Se observă lipsa corelaţiei între variabilele anonime. Este de aşteptat, totuşi, având în vedere că variabilele au fost obţinute prin PCA, care din definiţie generează componente **necorelate liniar**.

Singura corelaţie apare între variabila "Amount" şi restul variabilelor. Corelaţiile cu o magnitudine semnificativă apar între amount şi V6, V7, V20, cu valoare pozitivă, şi între amount şi V1, V2, V5, cu valoare negativă.

3.3 Preprocesarea datelor

3.3.1 Eliminăm trăsăturile inutile

În viaţa reală, data şi ora tranzacţiei ne pot ajuta să depistăm un comportament ciudat. Totuşi, aceste informaţii sunt relevante numai atunci când am monitorizat pe o perioada de măcar câteva zile activitatea din contul/cardul respectiv pentru a stabili ce reprezintă un comportament "normal".

În cazul nostru, nu avem la dispoziţie nici măcar data şi ora exactă. Prin urmare, vom elimina timpul din fiecare observaţie.

De asemenea, eliminăm etichetele. Acestea sunt utile numai la testare pentru a analiza performanţa şi la alegerea setului de antrenare. Metodele utilizate în această lucrare sunt exclusiv **nesupervizate**, deci nu avem nevoie de etichete la procesul de antrenare.

3.3.2 Validarea încrucişată

Împărţim setul de date, salvând 75% pentru antrenare. Din **setul de antrenare**, extragem toate datele cu eticheta de anomalie şi le mutăm în porţiunea de 25% rămasă de la pasul anterior. Apoi, ultima porţiune este împărţită la rândul ei în jumătăţi egale ce vor servi ca set de validare, respectiv testare.

Facem acest lucru deoarece algoritmii prezentaţi "învaţă" structura datelor normale şi, ideal, nu am vrea niciun fel de anomalie în setul de antrenare. În practică, este o şansă destul de mare ca setul de antrenare să fie **"contaminat"** cu un procent mic de anomalii, fiind dificil să garantăm **"puritatea"** datelor. Totuşi, am decis să presupunem că nu există nicio eroare în datele noastre pentru a păstra un număr mai mare de anomalii în setul de validare şi cel de testare, pe care să evaluăm performanţa modelelor.

De asemenea, o tehnică des utilizată în împarţirea setului de date pentru metode supervizate este **stratificarea** ce ne conferă o proporţie a claselor, aproximativ egală cu cea din setul iniţial, în porţiunile de antrenare, validare şi testare. Acest lucru este important pentru a ne asigura că fiecare clasă este reprezentată corespunzător în eşantionul nostru. În cazul detecţiei anomaliilor, o vom ignora deoarece avem doar o singură clasă de referinţă, anume cea normală, la antrenare. Datele etichetate ca fiind anomalii sunt utile doar la testare.

Setul de validare va fi folosit pentru alegerea hiperparametrilor optimi, în timp ce **setul de testare** va fi folosit la final pentru a face o comparaţie imparţială a performanţei modelelor.

3.3.3 Scalare

Având în vedere că nu cunoaştem dimensiunile, unităţile de măsură şi nici numele variabilelor aleatoare asociate punctelor, este indicat să scalăm datele. Uneori poate fi util să lăsăm punctele aşa cum sunt, dar aceasta ar necesita o decizie informată pe baza importanţei variabilelor în calcularea distanţei între puncte sau pe baza relevanţei unităţilor variabilelor, de exemplu.

În lipsa acestori factori, vom standardiza datele, adică **vom scădea media şi vom împărţi la deviaţia standard**, operaţie aplicată fiecărui punct. Aceasta are rolul de a centra variabilele şi de a obţine varianţă unitară pe fiecare variabilă în parte. Media şi deviaţia standard sunt obţinute din setul de antrenare. Le folosim pe acestea să transformăm setul de validare, cât şi pe cel de testare. Astfel, fiecare trăsătură x_i va fi inlocuită de o noua trăsătură x_i':

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}$$

unde μ_i este media:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)}$$

şi σ_i este deviaţia standard:

$$\sigma_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(x_i^{(j)} - \mu_i)^2}$$

calculate luând trăsăturile x_i din toate cele m observaţii din setul de antrenare.

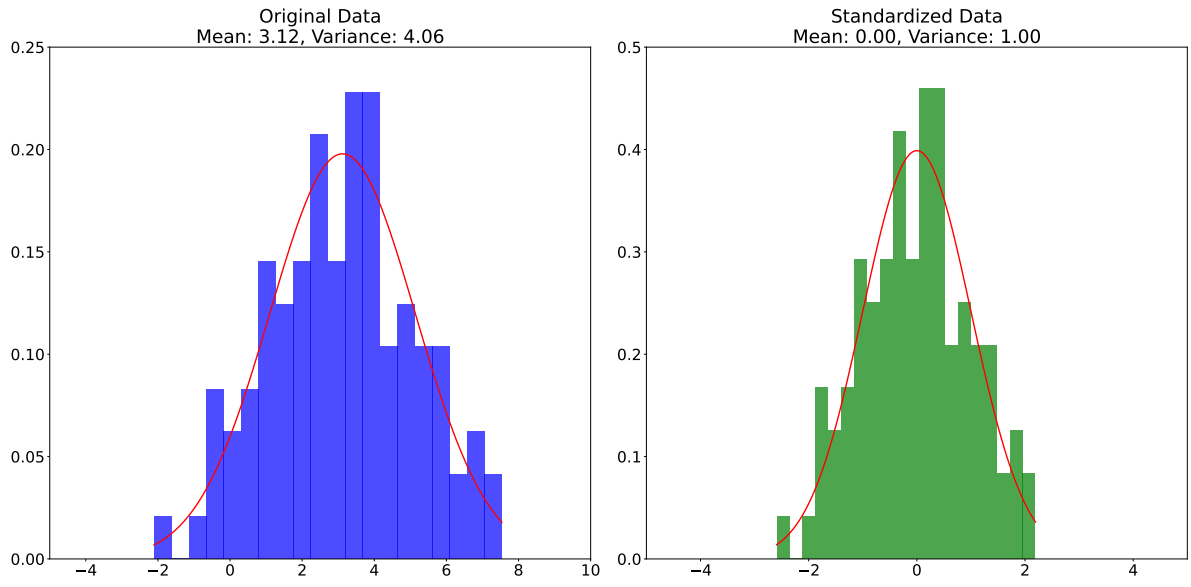

Figura 3.3: Media devine 0, varianţa devine 1

Capitolul 4

Evaluarea modelelor

Căutarea **hiperparametrilor optimi** se va realiza pentru fiecare model în parte folosind setul de validare. La final, cele mai bune modele găsite sunt evaluate într-un mod imparţial pe setul de testare.

4.1 Modele nesupervizate

4.1.1 One Class SVM

Pentru a exploata capabilităţile de modelare a unei margini de decizie neliniare a SVM-urilor, avem nevoie de funcţii kernel care să scufunde punctele din setul de date într-un spaţiu cu mai multe dimensiuni unde să putem găsi mai uşor un hiperplan de separare.

- $K(x, y) = x^T y$ - **Liniar**
- $K(x, y) = (\gamma x^T y + c)^d$ - **Polinomial**
- $K(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$ - **Gaussian**
- $K(x, y) = \tanh(\gamma x^T y + c)$ - **Sigmoid**

Un caz particular este kernel-ul **liniar** care păstrează punctele în spaţiul iniţial şi încearcă să găsească un hiperplan de separare optim la fel ca în cazul clasic fără kernel al SVM-ului. Acest kernel este nepotrivit în cele mai multe cazuri, mai puţin când datele sunt aproape liniar separabile, deci nu ne aşteptăm să performeze excepţional.

Totuşi, putem exploata eficienţa kernel-ului liniar folosind tehnica de optimizare **Stochastic Gradient Descent** (SGD) împreună cu metoda de aproximare **Nystroem** pentru a aplica o transformare neliniară asupra datelor de intrare şi apoi să găsim o margine de decizie liniară în noul spaţiu.

SGD este o metodă iterativă relativ simplă ce nu necesită un număr la fel de mare de calcule şi nici la fel de multă memorie precum metodele clasice de rezolvare a unui sistem de ecuaţii folosind algebră liniară. De aceea, este potrivită atunci când mărimea setului de date trece de ordinul sutelor de mii, în ciuda faptului că sacrificăm acurateţea ponderilor estimate.

Metoda Nystroem aproximează matricea kernel pentru o anumită funcţie kernel dată, folosind tehnica aproximării cu **matrice de rang scăzut** unde o fracţiune din punctele setului de antrenare este folosită ca bază vectorială pentru kernelul respectiv. Evident, calculul va fi mult mai rapid folosind o matrice de dimensiune redusă. Am comparat performanţa acestei metode pentru diverse funcţii kernel şi pentru diverse fracţiuni de puncte din setul de antrenare, care erau folosite de algoritm. Numărul de puncte a fost selectat astfel încât să ocupe un procent, dat ca parametru, din memoria RAM. Prin urmare, rezultatele pot să difere în funcţie de capacitatea memoriei pe care o avem la dispoziţie.

Pentru comparaţie am inclus şi funcţiile **polinomială** şi **sigmoidă**. Cu cea din urmă am obţinut rezultatele cele mai bune în comparaţie cu celelalte funcţii, dar nu mai bune decât kernelul Gaussian pe care îl vom folosi şi care este şi decizia des întâlnită în practică. De asemenea, atât kernel-ul liniar pentru valori mai mari ale lui ν, cât şi cel polinomial începând cu gradul 7 nu terminau antrenarea nici măcar după 12 ore, aşa că am abandonat căutarea hiperparametrilor pentru mai mult de atât, mai ales că acurateţea prezicerilor devenea din ce în ce mai îndoielnică.

d	γ	ν	Accuracy	Recall	Precision	F1 Score
7	0.001	0.007	0.474	0.947	0.006	0.0129
5	0.001	0.02	0.469	0.930	0.006	0.0128
7	1	0.0001	0.502	0.0243	0.008	0.0124
7	2	0.0001	0.502	0.024	0.008	0.0124

Tabela 4.1: Cele mai bune rezultate obţinute pentru kernel-ul polinomial

γ	ν	Accuracy	Recall	Precision	F1 Score
0.03	0.005	0.804	0.613	0.476	0.536
0.02	0.005	0.792	0.589	0.448	0.509
0.03	0.001	0.668	0.337	0.674	0.449
0.02	0.007	0.756	0.520	0.347	0.416

Tabela 4.2: Cele mai bune rezultate obţinute pentru kernel-ul sigmoid

ν	Accuracy	Recall	Precision	F1 Score
0.9	0.177	0.300	0.002	0.004
0.001	0.412	0.0813	0.002	0.004

Tabela 4.3: Cele mai bune rezultate obţinute pentru kernel-ul liniar

γ	ν	Accuracy	Recall	Precision	F1 Score
0.2	0.5	0.7405	0.9796	0.0134	0.0264
0.03	0.5	0.7402	0.9837	0.0133	0.0263

Tabela 4.4: Cele mai bune rezultate obţinute pentru kernel-ul liniar cu SGD şi Nystroem

Utilizăm kernelul Gaussian, deci printre hiperparametrii optimi pe care îi căutăm se va regăsi şi γ. Acest parametru influenţează **aria zonei de influenţă a fiecărui vector suport**. O valoare prea mare ar cauza ca zona să includă numai vectorul suport şi nimic altceva, ceea ce ar duce la o **varianţă crescută** a modelului. La polul opus, o valoare prea mică ar cauza ca zona sa includă toate punctele din setul de date, ceea ce ar duce la un **bias crescut**.

Parametrul ν este similar parametrului C din **Soft-Margin SVM**, cel din urmă fiind creat cu scopul de a rezolva problemele asociate parametrului C, anume că putea lua orice valoare pozitivă şi nu avea o interpretare directă. ν se află în intervalul $(0, 1]$ şi este interpretat ca marginea superioară a ponderii de anomalii şi marginea inferioară a ponderii de vectori suport. Prin urmare, ν controlează mărimea **frontierei din jurul datelor normale** a modelului, unde o frontieră mai mică este asociată cu o varianţă crescută, în timp ce o frontiera mai mare este asociată cu un bias crescut.

Folosim metoda **Grid Search** pentru a găsi parametrii favorabili. Afişăm doar valorile parametrilor pentru care obţinem rezultate semnificative.

γ	ν	Accuracy	Recall	Precision	F1 Score
0.01	0.0001	0.822	0.646	0.676	0.661
0.01	0.001	0.822	0.646	0.676	0.661
0.01	0.005	0.875	0.756	0.531	0.624
0.02	0.0001	0.877	0.760	0.526	0.622
0.02	0.001	0.877	0.760	0.523	0.620

Tabela 4.5: Cele mai bune rezultate obţinute pentru kernel-ul Gaussian

Se observă că γ este parametrul care aduce schimbările drastice în valorile metricilor, pe când ν doar creşte sau scade relativ puţin aceste valori.

Urmărim f1 score, aceasta fiind metrica ce evaluează modelul oarecum echilibrat, în contrast cu precision şi recall care favorează minimizarea fals pozitivelor, respectiv a fals

negativelor. Astfel, alegem $\gamma = 0.01$ şi $\nu = 0.0001$ pentru modelul final.

4.1.2 Gaussian Mixture Model

n	q	Accuracy	Recall	Precision	F1 Score
4	0.01	0.920	0.845	0.581	0.688
3	0.01	0.918	0.841	0.578	0.685
2	0.01	0.918	0.841	0.578	0.685
1	0.01	0.914	0.833	0.572	0.678
5	0.01	0.912	0.829	0.569	0.675

Tabela 4.6: Cele mai bune rezultate obţinute pentru Gaussian Mixture Model

Pentru acest model, hiperparametrul optim căutat este numărul de componente Gaussiene n. Vom încerca pe rând fiecare valoare din $\{1, 2, 3, ..., 16, 17\}$. De asemenea, pentru că modelul ne va oferi probabilitățile de apartenență a unui punct pentru fiecare componentă, vom avea nevoie și de un prag pentru a decide dacă punctul este sau nu anomalie. Ne vom folosi de cuantile calculate pe setul de validare pentru a găsi pragul optim.

Pentru $n = 1$ putem afla parametrii foarte eficient folosind **Maximum Likelihood Estimation**, întrucât problema se reduce la aflarea mediei şi a matricei de covarianţă pentru o distribuţie Gaussiană.

Pentru $n > 1$ vom folosi **Expectation Maximization** pentru a găsi parametrii optimi, având în vedere ca numărul de componente este precizat de la început.

Gaussian Mixture Model încearcă să estimeze o distribuţie posibil multimodală folosind mai multe componente Gaussiene. Prin urmare, numărul optim de componente ne indică intuitiv numărul de **moduri** pe care le are distribuţia ce a generat setul de date. Se observă că modelul se descurcă destul de bine chiar şi cu o singură componentă. Aceasta ne indică faptul ca distribuţia ce a generat setul de date este similară cu una Gaussiană.

Pentru acest model, pragul este foarte important. Dacă avem acelaşi prag, dar număr de componente diferite, varianţa rezultatelor nu este prea mare, dar dacă alegem greșit valoarea cuantilei, performanţa modelului are de suferit. Totuşi, dată simplitatea modelului, rezultatele sunt impresionante. Cu un timp de antrenare de sub câteva minute, obține rezultate mai bune decât One Class SVM pe setul de validare.

Alegem **numărul de componente** $n = 4$ pentru modelul final, chiar dacă am avut rezultate bune şi cu $n = 1$ deoarece ne dorim un model puţin mai complex decât o banală distribuţie Gaussiană. În practică, este o şansă mică să dăm peste un proces care să genereze date fix în acest mod. De asemenea, aici f1 score are valoarea cea mai mare şi după aceasta ne vom ghida, având în vedere că ne interesează performanţa modelului per total.

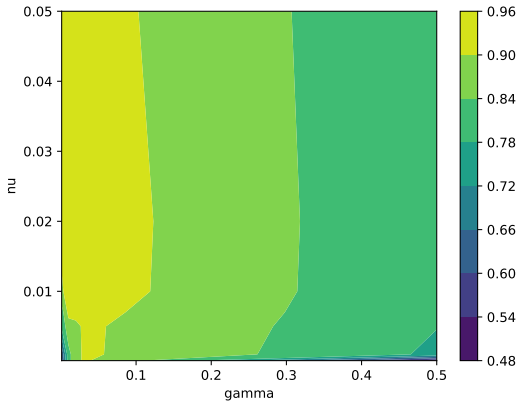

Figura 4.1: OCSVM Accuracy

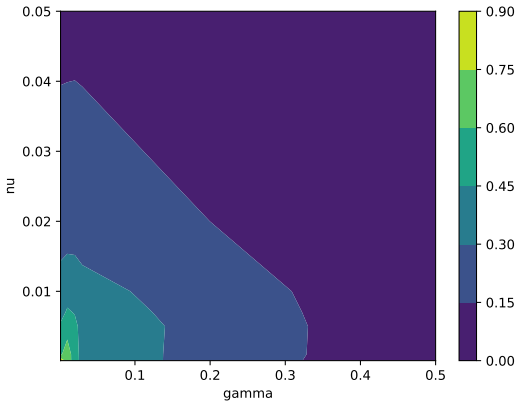

Figura 4.2: OCSVM Precision

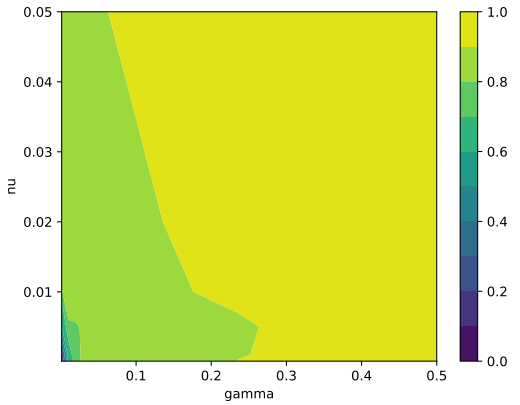

Figura 4.3: OCSVM Recall

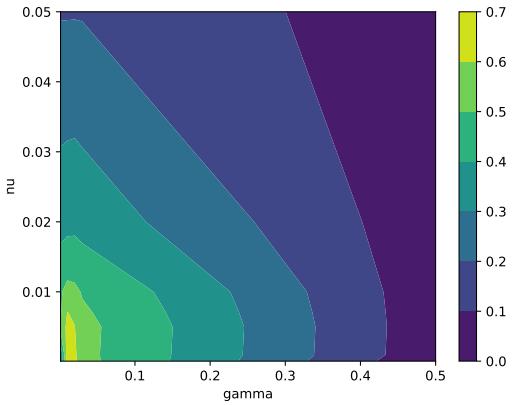

Figura 4.4: OCSVM F1 score

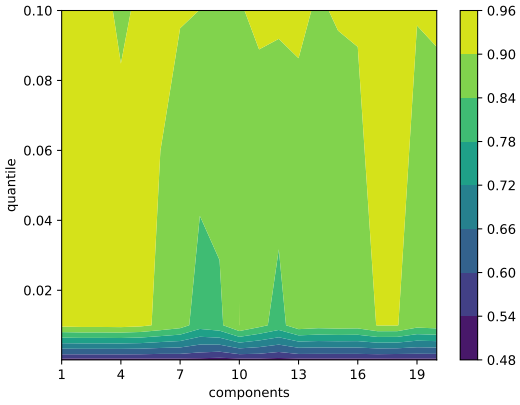

Figura 4.5: GMM Accuracy

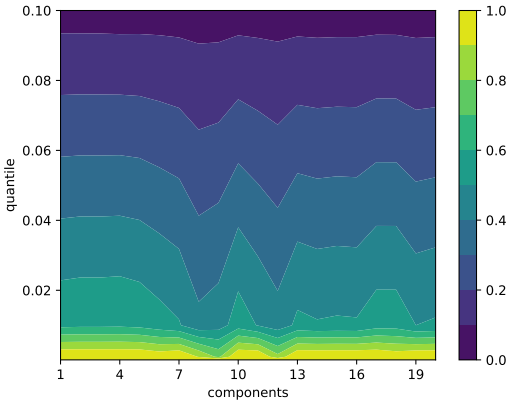

Figura 4.6: GMM Precision

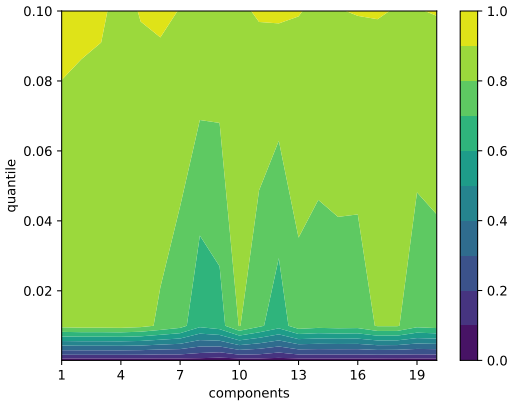

Figura 4.7: GMM Recall

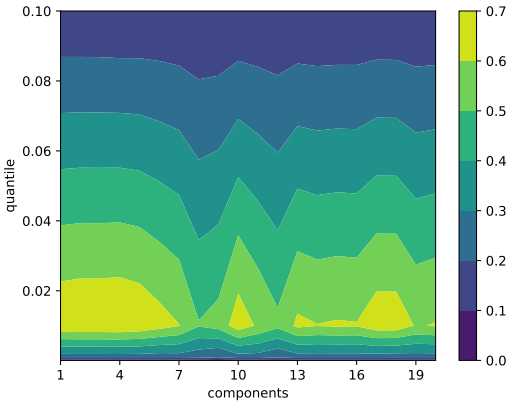

Figura 4.8: GMM F1 score

4.1.3 Kernel Density Estimation

Kernel	h	q	Accuracy	Recall	Precision	F1 Score
tophat	9	0.01	0.908	0.821	0.562	0.667
linear	9	0.01	0.881	0.768	0.527	0.625
parabolic	9	0.01	0.877	0.760	0.522	0.619
cosine	9	0.01	0.875	0.756	0.519	0.615
tophat	8	0.01	0.877	0.760	0.513	0.613

Tabela 4.7: Cele mai bune rezultate obţinute pentru Kernel Density Estimation

Am testat mai multe funcţii kernel, anume: **tophat**, **liniar**, **parabolic**, **cosinus**, **exponenţial** şi **gaussian**. Deşi kernelul Gaussian este cel mai des întâlnit în practică datorită numeroaselor proprietăţi utile pe care le deţine, aici se observă că nu obține rezultate satisfăcătoare, kernelul **tophat** fiind cel care performează cel mai bine. De asemenea, trebuie să impunem un prag după care să decidem daca un punct este sau

nu anomalie. Vom alege cuantile luate pe setul de validare, precum în cazul Gaussian Mixture Model.

Lăţimea de bandă este cea care stă la baza **bias-variance tradeoff** în acest model. Valorile prea mici implică **variance** mare, întrucât aria de sub grafic pentru fiecare punct este influenţată doar de punctele foarte apropiate de el, fapt ce duce la o distribuţie cu mulţi "ţepi". În schimb, valorile prea mari implică **bias** mare pentru că acum şi punctele aflate la distanţă mare joacă un rol important. În cel mai rău caz, o distribuţie multimodală ajunge sa fie estimată ca una unimodală din cauza netezimii graficului.

Se observă că după 1.0, valoarea lăţimii de banda nu mai aduce îmbunătăţiri semnificative. Acest lucru este ilustrat și pe grafic sub forma unui maxim local pentru f1 score. La fel ca în cazul Gaussian Mixture Model, cuantila folosită are un puternic impact asupra performanţei modelului.

Din păcate, după un timp de antrenare ce depășește 40 minute, timp ce se află între valorile pentru OCSVM și GMM, nu obținem rezultate mai bune decât niciunul dintre modelele precedente pe setul de validare.

Alegem **lăţimea de bandă** $bandwidth = 1.0$ pentru modelul final deoarece are cea mai bună valoare pentru f1 score. De asemenea, aceasta este şi cea mai mică valoare care aduce o performanţă decentă. O valoare prea mare a lăţimii de banda duce la o netezire excesivă a particularităţilor distribuţiei.

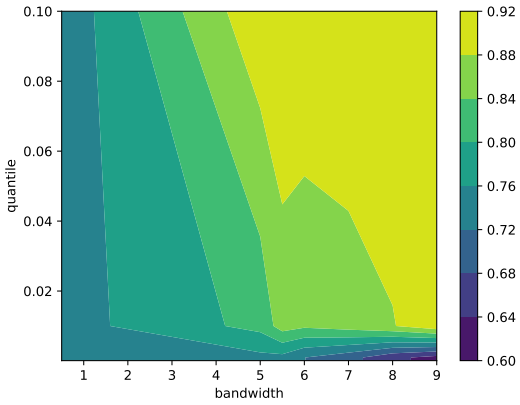

Figura 4.9: KDE Accuracy

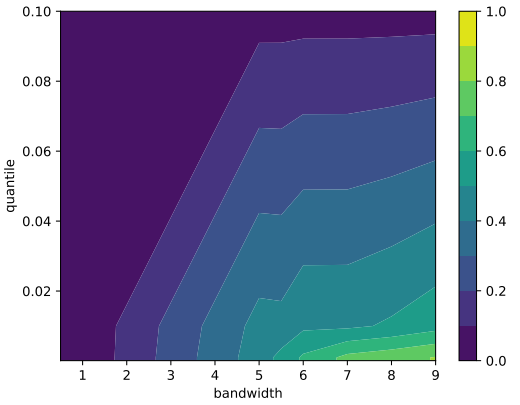

Figura 4.10: KDE Precision

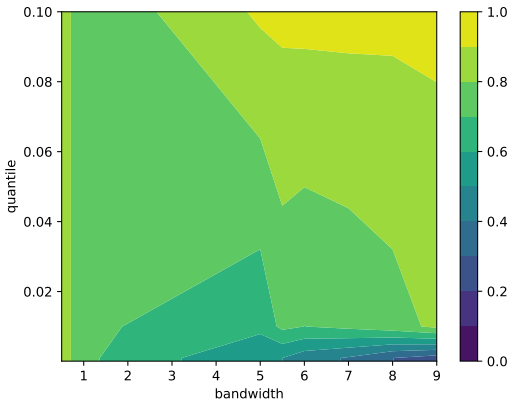

Figura 4.11: KDE Recall

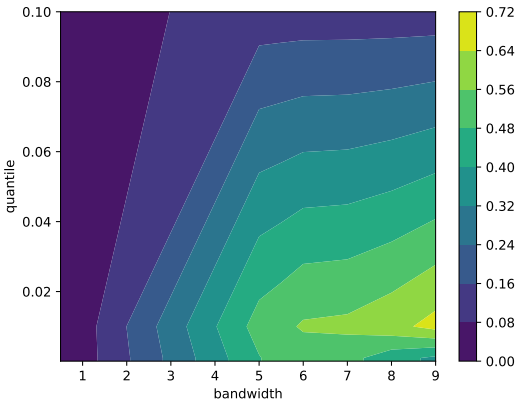

Figura 4.12: KDE F1 score

4.2 Modele supervizate

4.2.1 SVM Supervizat

Pentru a profita de faptul că avem la dispoziţie etichetele datelor, vom trata problema şi din punct de vedere supervizat. Gaussian Mixture Model şi Kernel Density Estimation nu pot fi aplicate decât într-un mediu nesupervizat, aşa că vom antrena doar SVM-ul, dar de data aceasta pentru clasificare binară. Astfel, putem observa performanţa algoritmilor prezentaţi anterior şi faţă de un algoritm supervizat care are de rezolvat o problemă relativ mai uşoară.

O modificare pe care o vom face la modul de împărţire al setului de date este că de această dată stratificarea este luată în calcul, aşa că păstrăm ponderile claselor aproximativ la fel în toate cele 3 partiţii.

Şi aici, kernel-ul Gaussian este cel care oferă cele mai bune rezultate cu valori scăzute pentru γ, similar cu cazul OCSVM, dar parametrul ν este înlocuit de parametrul C care are rolul de regularizare în SVM-ul Soft-Margin.

Se observă o creştere substanţială a valorilor metricilor atunci când tratăm problema în mod supervizat, **F1 Score** depăşind 0.87 pe setul de validare, în timp ce OCSVM pe setul de validare abia atingea 0.12.

γ	C	Accuracy	Recall	Precision	F1 Score
0.01	0.1	0.7972	0.5946	0.8302	0.6929
0.01	0.5	0.9053	0.8108	0.8571	0.8333
0.01	1.0	0.9053	0.8108	0.8824	0.8451
0.01	2.0	0.9053	0.8108	0.8955	0.8511
0.01	3.0	0.9054	0.8108	0.9375	0.8696
0.01	4.0	0.9054	0.8108	0.9524	0.8759
0.02	0.1	0.7094	0.4189	0.7750	0.5439
0.02	0.5	0.8850	0.7703	0.8769	0.8201
0.02	1.0	0.8918	0.7838	0.9355	0.8529
0.02	2.0	0.8919	0.7838	0.9508	0.8593
0.02	3.0	0.8919	0.7838	0.9508	0.8593
0.02	4.0	0.8919	0.7838	0.9667	0.8657
0.03	0.1	0.6148	0.2297	0.8947	0.3656
0.03	0.5	0.8243	0.6486	0.9412	0.7680
0.03	1.0	0.8783	0.7568	0.9492	0.8421
0.03	2.0	0.8851	0.7703	0.9661	0.8571
0.03	3.0	0.8851	0.7703	0.9661	0.8571
0.03	4.0	0.8851	0.7703	0.9661	0.8571

Tabela 4.8: Grid Search pentru SVM

Capitolul 5

Compararea modelelor finale

În final, antrenăm fiecare tip de model cu hiperparametrii optimi găsiţi folosind setul de validare şi folosim setul de testare pentru a evalua imparţial modelele finale.

Model	Accuracy	Recall	Precision	F1 Score
OCSVM	0.8913	0.0577	0.8823	0.1080
KDE	0.8911	0.0575	0.8823	0.1078
GMM	0.865	0.735	0.505	0.599

Tabela 5.1: Performanţa fiecărui model pe setul de testare

Cele 2 modele care au o performanţă similară sunt Gaussian Mixture Model și One Class SVM, ultimul fiind foarte puţin mai slab decât cel din urmă, dar cu un timp de antrenare mult mai mare. Kernel Density Estimation este de departe un model nepotrivit pentru acest set de date, necesitând un timp crescut de antrenare pentru niște rezultate mediocre.

Precision este scăzut şi prin urmare şi f1 score este scăzut. În schimb, recall are o valoare ridicată pentru toate modelele. Acest fapt ne arată că detectăm aproape toate anomaliile, dar în acelaşi timp, semnalăm un număr mare de tranzacţii obişnuite ca fiind frauduloase. Acest lucru poate deveni un inconvenient pentru client în cazul în care tranzacţia este anulată ca urmare a raportării incorecte.

Gaussian Mixture Model este modelul cu valorile cele mai mari pentru toate metricile, chiar dacă nu este un model la fel de complex precum celelalte 2 tehnici nesupervizate. De asemenea, acesta a necesitat doar 3 minute pentru antrenare şi prezicere a etichetelor, dar a produs rezultate aproape la fel de bune precum One class SVM şi mult mai bune decât Kernel Density Estimation care au avut nevoie de mai mult de o oră şi jumătate, respectiv 40 minute, cel din urmă neavând o performanţă cu mult mai ridicată în raport cu complexitatea de timp.

Ilustrăm şi ROC Curve alături de metrica AUC pentru modelele finale. Cu cât graficul tinde să se lipească de colţul stânga sus, cu atât aria de sub grafic creşte rezultând într-

o performanţă mai bună. Linia punctată pe diagonală din fiecare diagramă reprezintă graficul produs de clasificatorul aleatoriu şi are rolul de a marca marginea inferioară a performanţei faţă de această metrică. De asemenea, este de menţionat că dacă graficul se află sub linia diagonală, atunci am trage concluzia că algoritmul nostru este inutil. Totuşi, putem inversa clasa negativă cu cea pozitivă şi astfel obţinem simetricul graficului faţă de diagonală care acum, evident, se află deasupra liniei. Prin urmare, distanţa între grafic şi linie este o măsură mai relevantă decât poziţia relativă. Se presupune că operaţia de aplicare a simetricului este folosită unde este cazul pentru diagramele de mai jos.

ROC Curve are nevoie ori de probabilităţi ori de valori ce exprimă încrederea pentru fiecare prezicere, fără aplicarea vreunui prag. Kernel Density Esimation şi Gaussian Mixture Model produc deja probabilităţi, pe când One Class SVM nu produce decât distanţa faţă de hiperplanul de separare. Totuşi, vom folosi un truc pentru a transforma rezultatele OCSVM în valori de încredere [**stackoverflow-auc**]. Înlocuim fiecare valoare cu valoarea respectivă scăzută din valoarea maximă dată de funcţia de decizie după cum urmează:

$$y_{score} = MAX - f(y)$$

unde

- y_{score} este valoarea folosită pentru ROC Curve
- MAX este $\max_{i=1}^{n} f(y_i)$ pentru n observaţii
- f este funcţia de decizie

Astfel, putem compara modelele finale folosind şi metrica AUC. Clasamentul este acelaşi şi aici, cu One Class SVM fiind cel mai performant, urmat de Kernel Density Estimation, iar Gaussian Mixture Model la coadă.

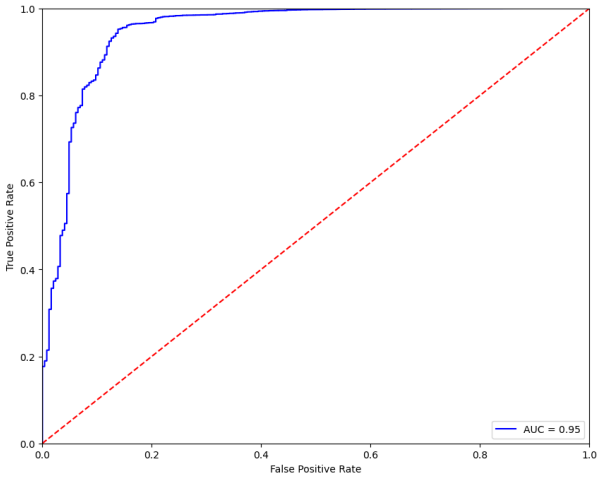

Figura 5.1: ROC Curve OCSVM

Figura 5.2: ROC Curve GMM

Figura 5.3: ROC Curve KDE

Capitolul 6

Concluzii

În această lucrare am ilustrat una din multitudinea de aplicaţii în industrie a detecţiei anomaliilor, anume identificarea tranzacţiilor frauduloase. Am explorat un set de date adnotat cu informaţii despre tranzacţii bancare şi am arătat cum putem evalua performanţa unor metode de învăţare automată nesupervizată folosind metrici din domeniul învăţării supervizate, mai specific, clasificarea binară.

Deşi tehnicile folosite aparţin învăţarii nesupervizate, am arătat cum putem exploata cunoştinţele din alte două categorii, anume semi-supervizată şi supervizată.

Prima a fost utilă pentru faptul că am antrenat algoritmi pe un set de date mare fără să utilizăm etichetele punctelor, dar apoi am folosit un set de date cu o mărime relativă mult mai mică pentru a analiza performanţa algoritmilor.

A doua a fost de folos pentru că ne-a oferit o multitudine de metrici care într-un context strict nesupervizat nu ar fi existat, fiind nevoie de inspecţia unui om pentru analiză.

Alegerea metricilor corespunzătoare este probabil una din cele mai dificile părţi atunci când căutăm o soluţie eficientă pentru problema dată, mai ales când proporţia de clase este neechilibrată şi acordăm o importanţă mai mare unei clase faţă de cealaltă, întrucât o alegere greşită ne poate induce în eroare cu privinţă la utilitatea reală a algoritmului, după cum am şi demonstrat.

De asemenea, am analizat trei algoritmi cu caracteristici diferite, dar cu scopuri similare, anume idenfiticarea anomaliilor.

One Class SVM se foloseşte de un hiperplan de separare pentru a împărţi spaţiul Hilbert într-o porţiune care cuprinde toate punctele normale şi una care cuprinde restul punctelor izolate. Astfel, putem spune că acesta face o clasificare dură a anomaliilor, întrucât ne indică doar daca o observaţie este normală sau nu.

La polul opus, avem Gaussian Mixture Model şi Kernel Density Estimation care încearcă să estimeze funcţia densitate de probabilitate din care au fost generate datele şi să atribuie fiecărei observaţii noi o valoare dată de această funcţie. Peste aceste probabilităţi se aplică un prag şi abia apoi obţinem etichetele corespunzătoare. Astfel, putem spune că acestea fac o clasificare slabă a anomaliilor, întrucât ne indică cât de probabil este ca

punctul să fie normal sau nu, dar nu ne indică concret clasa de care aparţine.

Deşi am întâmpinat probleme în ilustrarea graficului ROC Curve pentru algoritmul cu clasificare dură, am demonstrat cum putem transforma distanţele produse de funcţia de decizie în valori de încredere pentru a obţine o clasificare slabă artificială.

La final, am observat că modelele au performat mai mult sau mai puţin la fel, cele mai complexe având un dezavantaj când vine vorba de complexitatea de timp.

Cu toate acestea, lucrarea nu a abordat problema seriilor de timp, lucru des întâlnit în practică, dar care creează o direcţie de cercetare interesantă. Două cerinţe relevante pentru acest subiect ar fi prognoza şi clasificarea seriilor de timp.

Prima se referă la prezicerea evoluţiei în timp a seriei de date şi este folosită ca suport pentru unele modele ce detectează anomaliile folosindu-se de deviaţia prognozei faţă de valorile observate. Prognoza se bazează strict pe valorile punctelor din şir din contextul precedent şi pe modelul antrenat anterior. Deseori, se utilizează o fereastră glisantă de mărime unitară pentru a crea contextul şi a prezice câte un punct pe rând. Este similar cu abordarea clasică din domeniul Procesării Limbajului Natural, unde suntem interesaţi să prezicem cuvântul cel mai potrivit pentru continuarea unei propoziţii, spre exemplu. Aici, putem privi cuvintele din text ca fiind puncte dintr-o serie de timp.

A doua se referă la atribuirea unei categorii pentru diverse subşiruri din seriile de timp. Astfel, metoda poate fi folosită ori pentru post-procesarea rezultatelor obţinute după detecţia anomaliilor pentru a categoriza diferitele valori deviante, ori chiar în cadrul modelului cu rolul de a clasifica componentele din seriile de date în grupuri relevante, implict anomalii sau observaţii normale. Metoda ne aduce aminte de clasica problemă a clasificării unor puncte dintr-un set de date, întâlnită în contextul învăţării automate supervizate [**time-series**].