Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

курсов проект на тема:

Засичане на измами чрез автоенкодери

Студент: **Ростислав Милославов Стоянов** ф-н: **26321**
Специалност: Изкуствен интелект, Курс: 1-ви
Учебна година: 2020/21
Преподаватели: проф. Иван Койчев

Декларация за липса на плагиатство

- Плагиатство е да използваш, идеи, мнение или работа на друг, като претендираш, че са твои. Това е форма на преписване.
- Тази курсова работа е моя, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
- Разбирам, че ако се установи плагиатство в работата ми ще получа оценка "Слаб".

25 юни 2021 г. Подпис на студента:

Съдържание

Увод

Измамата е целенасочено използване на заблуда с цел осигуряване на неправомерни и нечестни облаги. Съществуват различни видове измама, които се различават освен по метода, по който се осъществява измамата, така и по областта в която се провежда измамата и крайната й цел. Така например, различни видове измами спрямо крайната им цел, са: застрахователна измама, данъчна измама, измами с кредитни/дебитни карти, крадене на самоличност и други. Загубите вследствие на измами са големи – загубите от измами с откраднати самоличности през 2020 г. се изчисляват на 56 млрд.долара[5], а според друго, глобално, проучване, базирано на събрани над 5000 отговора, загубите на разпитаните за период от 48 месеца възлизат на над 42 млрд долара[8]. Това обяснява и големият интерес към разработването на методи за засичане на измами. Този проект се фокусира върху под-задачата за засичане на измами с кредитни карти. По-конкретно, проектът цели да демонстрира възможни начини за използването на невронни мрежи базирани на автоенкодер *(autoencoder)* архитектурата в комбинация с други класификатори с цел откриване на измамни транзакции.

1 Засичане на измами с кредитни карти

Засичането на измами с кредитни карти е свързано с установяването на легитимността на дадена транзакция. Използването на ръчен труд за тази цел е неточно и не предоставя възможност за засичане на измами в реално време, което налага използването на различни техники от областта на машинното самообучение и извличането на знания от данни. Целта на една система, използваща такива методи, е не само да засича измами, но и да минимизира броя на погрешно категоризираните легитимни транзакции. В противен случай, пропускането на измамни транзакции би довело до загуби, а наличието на голям брой погрешно класифицирани легитимни транзакции би намалило доверието на клиентите на съответната организация.

Методите, използвани за засичане на измамни транзакции, могат да бъдат разделени

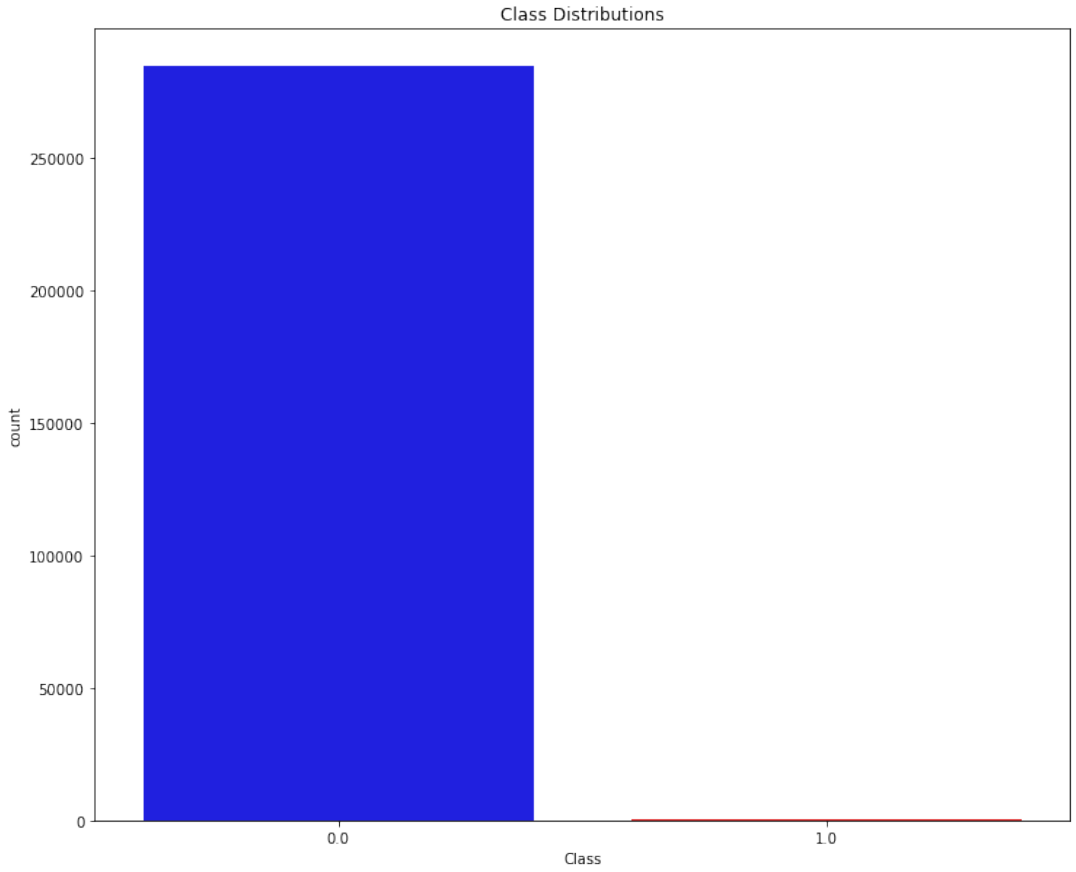
на два класа – методи от тип самообучение с учител, при които задачата се свежда до бинарен класификационен проблем, и методи от тип самообучение без учител, при които задачата се свежда до засичането на отклонения(*outliers*). Примери за използвани методи от машинното самообучение са логистична регресия, наивен Бейсов класификатор и к най-близки съседи[1], а автоенкодер мрежи са използвани в [9] и [6].

2 Проектиране

За целите на проекта са използвани данните от *Credit Card Fraud Detection*, които са свободно достъпни в платформата *Kaggle*[1]. Те са предоставени в csv файл с размер 144 мб и обем 284,807 транзакции. Всеки запис на транзакция се състои от 31 числа. Това са: индикатор за принадлежност към клас, стойност на транзакцията, време изминало между описваната и първата транзакция, както и 28 числа, които са различни характеристики на транзакциите, получени чрез метод на главните компоненти(МГК), на англ *Principal Component Analysis (PCA)*. При това те са силно небалансирани – само 492 от предоставените транзакции са измамни, което се равнява на 0.172% от всички записи (фиг.1).

Настоящият проект цели да провери хипотезата, че използването на *Denosing autoencoder (DAE)* върху началните данни преди те да бъдат подадени на класифициращ модел би подобрило качеството на класификацията. Извършва се проверка дали една такава мрежа (по-точно *encoder* частта от тренираната *DAE* мрежа) би могла да се използва за намаляване на размерността на данните и как това се отразява върху класификацията. Имбалансираността на входните данни може да се адресира чрез използването на различни алгоритми за *oversampling* (съкратено като *OS*). Такива са например алгоритмите *SMOTE[3], ADASYN[4], SVM-SMOTE[7], LoRAS[2]*. Прилагането на различни алгоритми проверява хипотезата, че крайният резултат, при наличието на силно небалансирано множество от данни, каквото е избраното, е силно зависим от използването на алгоритъм за *oversampling* и използването на различен алгоритъм може да доведе до съществена раз-

[1] https://www.kaggle.com/mlg-ulb/creditcardfraud



Фигура 1: Разпределение на данните по класове

лика в класификацията на измамни транзакции.

3 Реализация, експерименти

3.1 Използвани технологии, платформи и библиотеки

За реализация на проекта е избран езикът за програмиране Python[2], заради възможностите за лесно използване на множество полезни библиотеки. Основните използвани библиотеки са:

- Pandas[3] – за зареждане и предварителна обработка на данните;
- Seaborn[4]– за визуализация;

[2] https://www.python.org/
[3] https://pandas.pydata.org/
[4] https://seaborn.pydata.org/

- Scikit-learn[5] – за разбиване на входните данни, трениране на класификатори и оценяване на получените резултати;
- Pytorch[6] – за трениране на автоенкодери и класификатори;
- Imblearn[7] – за *oversampling* на входящите данни;
- Loras[8], съдържаща имплементация на алгоритъма за *oversampling* LoRAS.

3.2 Провеждане на експерименти

Първоначално се извършва предварителна обработка на данните, която включва:

- Разделяне на данните на три множества – тренировъчно, валидиращо и тестово в съотношение 70/10/20, като за всяко множество се запазва пропорцията легитимни/измамни транзакции от входното(общо) множество от данни;
- Нормализиране на стойностите на атрибутите Amount и Time с цел по-бързо трениране, избягване на локални минимуми, както и поставянето им в същия интервал като стойностите на останалите атрибути, получени чрез МГК. Нормализирането се постига чрез изваждане на средната стойност и делене на стандартното отклонение.

След извършената обработка, с помощта на всеки един от избраните алгоритми за *oversampling(SMOTE, ADASYN, SVMSMOTE, LoRAS)*, от тренировъчното множество се създава по едно ново множество, което съответства на прилагането на даден алгоритъм върху първоначалното тренировъчно множество. Върху всяко така създадено множество, включително и едно множество върху, което не е приложен алгоритъм за *oversampling* се тренира *denoising autoencoder*, като при трениране върху данните се добавя шум, който е извадка от нормално разпределение $\mathcal{N}(0, 1)$. Шум се добавя, от една страна, за да попречи на мрежата да научи функцията за идентитет, а от друга страна се цели разширяване на възможностите на автоенкодера да премахва шум във входните данни. За целта

[5] https://scikit-learn.org/stable/
[6] https://pytorch.org/
[7] https://imbalanced-learn.org/stable/
[8] https://github.com/sbi-rostock/LoRAS

към всеки входен вектор x се прибавя извадка от нормалното разпределение с тежест α, където α е параметър, определян от потребителя. Така зашумените вектори $\overline{x}$ имат вид $\overline{x} = x + \alpha n$, където n е извадка от $\mathcal{N}(0, 1)$. Архитектурата на автоенкодер мрежите е подобна на тази представена в [9], като разликата е в броя слоеве и наличието на сигмоидна активационна функция между слоевете. Тренираните автоенкодер мрежи се използват за последващо трениране на следните два класификатора:

- Първият класификатор се тренира върху данни преминали през цяла автоенкодер мрежа, т.е. входните данни първо преминават през вече тренирания автоенкодер и след това захранват класификатора. От своя страна, класификатора представлява невронна мрежа, изградена от няколко силно свързани слоя с активационни функции от тип *ReLu* между тях.
- Вторият класификатор е аналогичен, но използва само *encoder* частта от текущия автоенкодер за извличане на латентна репрезентация на входните данни. Дължината на векторите преминали през енкодера е значително по-малка от тази на оригиналните данни: получените вектори имат дължина 15, в сравнение с оригиналната дължина от 30. Класификаторът, от този тип, представлява два силно свързани слоя с ReLu активационна функция между тях.

Поради силната небалансираност на данните, за метрики за оценка се използват прецизност *(precision)*, възврат *(recall)* и F1-оценка *(F1-score)*. Резултатите получени при тестването на тренираните класификатори са илюстрирани в табл.1. Всяка клетка описва метриките за оценка след тестване на съответния класификатор с конкретно приложен метод за *oversampling*, като резултатите са представени във формат прецизност/възврат/F1-оценка. Удебелени са най-добрите стойности за съответната метрика за всеки ред.

Таблица 1: Резултати от тестването на различните възможности за прилагане на *autoencoder* архитектурата за решение на задачата

Classifier	No OS	SMOTE	ADASYN	SVMSMOTE	LoRAS
Full autoencoder	**.8**/.8/**.8**	.24/.83/.37	.18/.84/.3	.31/**.86**/.47	.79/.77/.78
Encoder part only	**.86**/.78/**.71**	.04/.93/.08	.03/**.95**/.05	.07/.87/.15	.53/.8/.64

Оказва се, че използването на някои от методите за *oversampling* могат да имат отрицателно влияние върху тренирания модел – въпреки че възвратата се увеличава, прецизността намалява, т.е. класификатора намира повече от измамните транзакции, но грешно класифицира транзакции като принадлежащи на неподходящи класове. Причина за това може да е някой от недостатъците на *SMOTE/ADASYN*, например прекаленото генерализиране на минорния клас или създаването на много синтетични точки в район, който е отдалечен от минорния клас (там, където се намират *outlier-ume*). Най-балансирани резултати дават класификаторите тренирани при липса на *oversampling*, както и тези тренирани върху данни, нормализирани спрямо класове с алгоритъма LoRAS, като разликата в случая с първия класификатор е минимална.

Тъй като резултатите от табл.1 показват, че, първият класификатор, т.е. този, който използва цяла автоенкдоер мрежа, дава по-балансирани резултати, се провеждат още тестове, които да определят влиянието на автоенкодера спрямо това на използвания алгоритъм за *oversampling*. За целта се използват класификаторите за логистична регресия (*LogisticRegression*) и случайна гора (*RandomForest*) от *scikit-learn* библиотеката, като се разглеждат различни прагове за класификация. Тези прагове се използват за определяне на силата на убеждение в принадлежността към измамния клас. Това означава че, ако за даден пример, вероятността за принадлежност към положителния клас с измами, върната ни от използвания класификатор, е по-висока от стойността на прага, можем да определим съответния пример за измамен. Резултатите за логистична регресия са представени в табл.2 и табл.3, а резултатите за случайна гора в табл.4 и табл.5. Форматът, в който са представени резултатите, и използваните метрики са идентични с тези използвани в табл.1, с единствената разлика, че тук са удебелени най-добрите стойности за всяка колона, т.е. за съответния *oversamlping* алгоритъм.

Таблица 2: Резултати за логистична регресия без прилагане на автоенкодер мрежа

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.02/**.97**/.03	.007/**1.**/.01	.07/**.89**/.12	.02/**.91**/.04
0.3	.03/.96/.06	.01/.1./.02	.05/.88/.15	.03/.89/.05
0.4	.05/.95/.09	.01/.98/.03	.1/.88/.18	.04/.89/.07
0.5	.06/.93/.12	.02/.97/.04	.12/.87/.22	.05/.87/.09
0.6	.09/.92/.16	.03/.95/.06	.14/.87/.24	.06/.87/.12
0.7	.12/.9/.21	.04/.94/.09	.16/.87/.27	.08/.84/.14
0.8	**.15**/.87/**.26**	**.07**/.93/**.12**	**.2**/.86/**.33**	**.1**/.83/..**18**

Таблица 3: Резултати за логистична регресия с прилагане на автоенкодер мрежа

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.006/**.97**/.01	.004/**.94**/.007	.03/**.92**/.06	.008/**.83**/.01
0.3	.007/.97/.02	.004/.94/.09	.05/.91/.09	.009/.82/.02
0.4	.01/.93/.02	.005/.94/.01	.06/.9/.11	.01/.82/.02
0.5	.01/.92/.02	.006/.94/.01	.07/.89/.13	.01/.82/.02
0.6	.02/.92/.03	.006/.94/.01	.08/.87/.15	.01/.82/.03
0.7	.02/.9/.04	.007/.93/.01	.1/.87/.19	.01/.81/.03
0.8	**.03**/.9/**.06**	**.008**/.93/**.02**	**.13**/.86/**.23**	**.02**/.79/**.03**

Таблица 4: Резултати за случайна гора без прилагане на автоенкодер мрежа

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.01/**.95**/.02	.002/**1.**/.003	.01/**.96**/.02	.01/**.96**/.03
0.3	.03/.92/.07	.006/1./.01	.04/.91/.07	.08/.88/.15
0.4	.08/.87/.15	.01/.95/.03	.06/.88/.12	.17/.86/.29
0.5	.17/.85/.29	.06/.9/.1	.1/.86/.18	.37/.84/.51
0.6	.47/.83/.6	.14/.88/.24	.2/.84/.32	.75/.79/.**77**
0.7	.76/.81/.78	.37/.83/.51	.46/.83/.59	.84/.71/.77
0.8	**.86**/.79/**.82**	**.79**/.76/**.77**	**.71**/.81/**.76**	**.86**/.64/.73

Таблица 5: Резултати за случайна гора с прилагане на автоенкодер мрежа

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.006/**.97**/.01	.001/**1.**/.003	.005/**.99**/.01	.007/..**95**/.01
0.3	.02/.95/.04	.005/.98/.01	.02/.93/.03	.02/.91/.05
0.4	.07/.88/.12	.01/.96/.02	.03/.9/.05	.11/.87/.2
0.5	.14/.87/.24	.04/.91/.08	.05/.89/.1	.27/.85/.41
0.6	.42/.83/.56	.1/.88/.18	.08/.86/.15	.69/.79/.73
0.7	.7/.81/.75	.26/.86/.4	.18/.85/.3	**.86**/.71/**.78**
0.8	**.84**/.79/**.81**	**.7**/.8/**.75**	**.5**/.8/**.61**	85/.58/.69

Преместването на праговете за класификация влияе върху оценката за прецизност и възврат – с повишаване на прага се увеличава прецизността на класификатора, но оценката за възврата намалява. Въпреки по-ниските стойности за възврата, класификаторите с по-висок праг са, в общия случай, по-балансирани, за което свидетелства по-високата F1-оценка. Най-висок възврат имат класификаторите използващи логистична регресия и алгоритъма ADASYN, но те не са балансирани и имат ниски стойности на прецизност. Моделите използващи случайна гора са по-балансирани от тези използващи логистична регресия за по-ниски стойности на праговете и имат по-висока F1-оценка, което се дължи на по-високата прецизност постигана от тези модели. Използването на автоенкодери позволява постигането на по-висок възврат, за сметка на по-нисък *F1-score*.

Заключение

Успешното засичане на измами може да има положително влияние върху множество бизнеси от различни сфери. Дефинираната хипотеза и извършените експерименти показват, как автоенкодер архитектурата може да се използва за откриване на измамни картови транзакции и какво влияние оказва използването на тази архитектура и различни видове техники за *oversampling*. Илюстрират се възможности за създаване на модели, които почти не изпускат измамни транзакции, за сметка на по-ниска точност при класификация и необходимост от използването на други техники след прилагане на тези модели. Съществуват, обаче, и други модели, които използват автоенкодер и подходящ алгоритъм за *oversampling* за постигане на по-добра и балансирана класификация.

Бъдещо развитие на проекта би могло да включва

- Изследване на възможностите на автоенкодерите за намаляване на размерността на данните;
- Използване на класификатори различни от приложените;
- Използване на методи за *undersampling* отделно или в комбинация с методи за *oversampling*.

Библиография

[1] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," in *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 2017, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/8123782

[2] S. Bej, N. Davtyan, M. Wolfien, M. Nassar, and O. Wolkenhauer, "Loras: An oversampling approach for imbalanced datasets," *Machine Learning*, vol. 110, no. 2, pp. 279–301, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10994-020-05913-4

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302

[4] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328. [Online]. Available: https://ieeexplore.ieee.org/document/4633969

[5] "2021 Identity Fraud Study: Shifting Angles," Javelin. [Online]. Available: https://www.javelinstrategy.com/content/2021-identity-fraud-report-shifting-angles-identity-fraud

[6] S. Misra, S. Thakur, M. Ghosh, and S. K. Saha, "An autoencoder based model for detecting fraudulent credit card transaction," *Procedia Computer Science*, vol. 167, pp. 254–262, 2020, international Conference on Computational Intelligence and Data Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920306840

[7] H. M. Nguyen, E. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *Int. J. Knowl. Eng. Soft Data Paradigms*, vol. 3, pp. 4–21, 2011. [Online]. Available: http://ousar.lib.okayama-u.ac.jp/files/public/1/19617/20160528004522391723/IWCIA2009_A1005.pdf

[8] “PwC’s Global Economic Crime and Fraud Survey 2020,” PWC. [Online]. Available: https://www.pwc.com/gx/en/services/forensics/economic-crime-survey.html

[9] J. Zou, J. Zhang, and P. Jiang, “Credit card fraud detection using autoencoder neural network,” *arXiv preprint arXiv:1908.11553*, 2019. [Online]. Available: https://arxiv.org/abs/1908.11553