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".

21 май 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 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. Целта на този проект е да демонстрира как невронни мрежи, базирани на автоенкодер*(autoencoder)* архитектурата, могат да бъдат използвани за засичане на измами с кредитни карти.

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

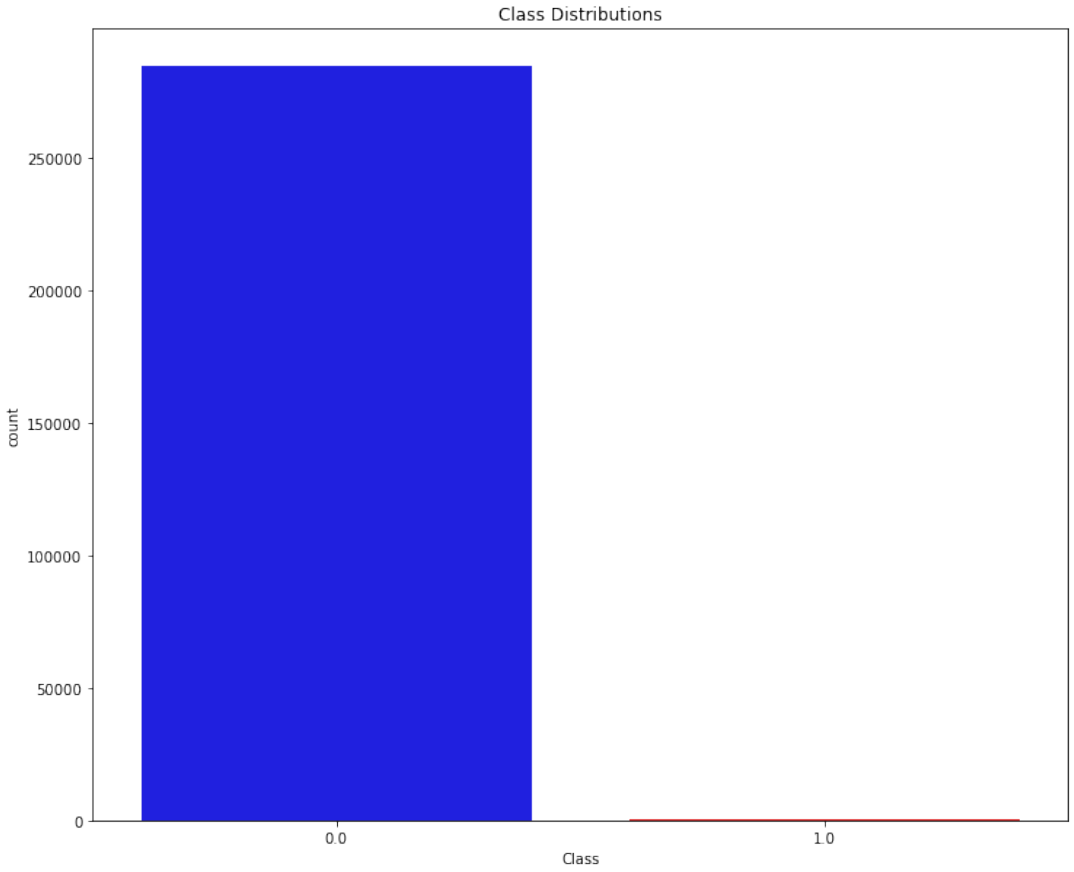
За целите на проекта са използвани данните от *Credit Card Fraud Detection*, които са свободно достъпни в платформата *Kaggle*[1]. Те са предоставени в csv файл с размер 144 мб и обем 284,807 транзакции. Всеки запис на транзакция се състои от 31 числа. Това са: индикатор за принадлежност към клас, стойност 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(МГА), на англ *Principal Component Analysis (PCA)* (фиг.1). При това те са силно небалансирани – само 492 от предоставените транзакции са измамни, което се равнява на 0.172% от всички записи (фиг.2).

V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
38321	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	0.0
30018	-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	0.0
03198	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	0.0
10309	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	0.0
07193	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	0.0

Фигура 1: Част от атрибутите, описващи данните

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, например – SMOTE[3], ADASYN[4], SVM-SMOTE[7], LoRAS[2]*. Прилагането на различни алгоритми проверява хипотезата, че крайният резултат, при

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



Фигура 2: Разпределение на данните по класове

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 разлика 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;

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

- Seaborn[4]– за визуализация;
- 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:

- Нормализиране на стойностите на атрибута Amount;
- Премахване на атрибута Time;
- 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.

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 се тренира *denoising autoencoder*, като при трениране върху данните се добавя шум, който е извадка от нормално разпределение $\mathcal{N}(0, 0.5)$. Архитектурата на аутоенкодер мрежите е взаимствана от [9]. Трениране автоенкодер мрежи се използват за последващо трениране на следните два класификатора:

- Първият класификатор се тренира върху данни преминали през цяла автоенкодер

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

мрежа, т.е. входните данни първо преминават през вече тренирания автоенкодер и след това захранват класификатора. От своя страна, класификатора представлява невронна мрежа, чиято архитектура е подобна на тази в [9] – броят и размерите на силно свързаните слоеве е еднакъв, но на изхода от слоевете(с изключение на последния се прилага активационна функция от тип ReLu).

- 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 10, в сравнение с оригиналната дължина от 29. 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	Baseline	SMOTE	ADASYN	SVMSMOTE	LoRAS
Full autoencoder	.86/.79/.82	.15/.87/.26	.12/**.88**/.21	.79/.79/**.79**	**.8**/.77/.79
Encoder part only	**.84**/.73/**.78**	.11/**.88**/.2	.13/.88/.23	.65/.78/.71	.18/.76/.29

Оказва се, че използването на някои от методите за *oversampling* могат да имат отрицателно влияние върху тренирания модел – въпреки че възврата 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-добре се представя SVMSMOTE алгоритъма, като разликата с LoRAS е минимална в случая с първия класификатор, а липсата на алгоритъм за *oversampling* (представен в колона *Baseline)* е подход, който също се справя добре.

Тъй като резултатите от табл.1 показват, че, макар и с малко, по-добре се справя правият класификатор, т.е. този, който използва цяла автоенкдоер мрежа, 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. 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	.04/**.95**/.07	.03/**.95**/.07	.15/**.82**/.26	.02/**.92**/.04
0.3	.06/.93/.12	.06/.93/.11	.28/.82/.42	.03/.91/.05
0.4	.09/.9/.17	.09/.91/.17	.38/.81/.51	.04/.9/.08
0.5	.13/.9/.22	.13/.91/.22	.45/.81/.58	.05/.87/.09
0.6	.16/.89/.27	.16/.89/.27	.5/.8/.61	.06/.85/.12
0.7	.2/.89/.32	.2/.89/.33	.56/.79/.65	.06/.84/.14
0.8	.**29**/.87/**.43**	**.3**/.87/**.44**	**.6**/.78/**.68**	**.1**/.9/**..18**

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

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.02/**.97**/.03	.02/**.97**/.05	.1/**.86**/.17	.006/**.83**/.01
0.3	.03/.97/.05	.04/.92/.07	.13/.86/.23	.007/.82/.01
0.4	.04/.95/.08	.06/.92/.11	.2/.86/.33	.009/.82/.02
0.5	.06/.91/.11	.09/.9/.16	.2/.86/.33	.009/.82/.02
0.6	.09/.9/.17	.12/.89/.21	.35/.86/.38	.01/.82/.02
0.7	.15/.89/.25	.17/.88/.28	.33/.81/.47	.01/.82/.02
0.8	**.21**/.87/**.34**	**.24**/.87/**.38**	**.38**/.8/**.51**	**.01**/.82/**.02**

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

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.02/**.93**/.03	.02/**.94**/.03	.1/**.85**/.18	.02/**.97**/.03
0.3	.07/.89/.13	.07/.9/.13	.25/.83/.38	.1/.88/.18
0.4	.2/.84/.32	.18/.85/.3	.55/.82/.66	.25/.84/.38
0.5	.48/.83/.6	.45/.83/.58	.73/.82/.77	.75/.82/.78
0.6	.69/.83/.75	.69/.83/.75	.8/.8/**.8**	.82/.77/**.79**
0.7	.82/.8/.81	.83/.81/.81	.83/.74/.78	.84/.68/.75
0.8	**.86**/.77/**.81**	**.86**/.78/**.82**	**.85**/.67/.75	**.84**/.6/.7

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

Threshold	SMOTE	ADASYN	SVMSMOTE	LoRAS
0.2	.006/**.98**/.01	.005/..**98**/.01	.02/**.88**/.03	.006/..**98**/.01
0.3	.03/.89/.05	.02/.89/.04	.04/.84/.08	.03/.9/.06
0.4	.07/.87/.13	.07/.86/.13	.24/.83/.37	.1/.87/.19
0.5	.17/.84/.28	.19/.84/.30	.61/.82/.7	.19/.84/.31
0.6	.33/.83/.48	.35/.83/.62	.75/.79/.77	.39/.77/.52
0.7	.63/.8/.71	.61/.82/.73	.83/.78/**.8**	.82/.7/**.75**
0.8	**.8**/.79/**.78**	**.79**/.78/**.78**	**.85**/.72/.78	**.83**/.59/.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 SMOTE, 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