Detecção de Fraudes em Transações de Cartões de Crédito Utilizando Aprendizado de Máquina

Andrea Monicque Silva, Francisco Marcelo Damasceno, Lucas Pereira Reis, Marcos Wenneton Araújo

[1]Núcleo de Computação
Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200 – Manaus – Amazonas

{amdss, fmmd, lpr, mwvda}.cid20@uea.edu.br

Resumo. *Fraudes em transações financeiras são uma preocupação constante em um mundo cada vez mais conectado e integrado digitalmente. À medida que as plataformas digitais evoluem elas também se tornam alvos de ataques cada vez mais sofisticados, exigindo conhecimento igualmente refinados para combater essas práticas. E para isso, a utilização de inteligência artificial tem se mostrado uma técnica bastante eficiente. Neste artigo, foram testadas e avaliadas de forma objetiva através de métricas como* F1-Score *e ROC-AUC algumas das principais técnicas de Aprendizado de Máquina disponíveis hoje na literatura, com objetivo desenvolver uma solução para auxiliar na detecção de fraudes em transações financeiras realizadas por meio de cartões de crédito.*

1. Introdução

Com o crescimento da tecnologia, tornou-se mais acessível a realização de compras por cartões de crédito e transações bancárias de forma digital, gerando assim um crescimento no número de operações deste tipo por pessoas físicas e jurídicas. Esse aumento da informatização traz inúmeros benefícios para o setor financeiro, como por exemplo, a redução de custos operacionais, bem como a realização de operações financeiras de forma mais ágil e simples para os usuários.

Entretanto, o aumento na informatização traz também o aumento de problemas relacionados à utilização dessas novas tecnologias como, por exemplo, fraudes de boletos, compras ou transferências não-autorizadas e aumento de fraudes e crimes cibernéticos. Segundo relatório de fraudes globais da empresa Kroll para os anos de 2016 e 2017 [Carvalho 2018] no Brasil, 89% das empresas do setor financeiro sofreram algum tipo de incidente cibernético (por exemplo, fraudes, ataques a seus sistemas informatizados ou roubos de informação) e 91% dessas empresas acreditam que a exposição à fraude aumentou nos anos em que a pesquisa foi realizada. Estes tipos de fraudes levam à perda de dinheiro tanto para indivíduos como para empresas, afetam a reputação de bancos ou comerciantes individuais causando muitas vezes uma circunstância incontrolável e perdas irreparáveis.

Com objetivo de inibir esse número crescente de transações fraudulentas e crimes virtuais, existem vários trabalhos desenvolvidos aplicando métodos de Inteligência Artificial e Aprendizado de Máquina para identificar padrões não comportamentais em atividades financeiras de usuários [Beltran 2019], [Shenvi et al. 2019], [Zamini and Montazer 2018].

Deste modo, este trabalho visa elaborar mecanismos que detectam operações suspeitas em cartões de crédito, aumentando a segurança e confiabilidade. Para isto, serão aplicadas abordagens utilizando algoritmos de Aprendizado de Máquina bem conhecidos na literatura, considerando os tipos de aprendizado supervisionado e não-supervisionado.

2. Objetivo

O objetivo geral deste trabalho consiste em desenvolver um modelo de *Machine Learning* capaz de realizar detecção de fraudes em transações de cartões de crédito através de estratégias de aprendizagem supervisionada e não-supervisionada. Para alcançar esta meta, faz-se necessário estabelecer objetivos específicos, descritos a seguir:

1. Consolidar uma base de dados representativa de transações bancárias de dados reais para treinamento e teste dos modelos de AM;
2. Explorar a utilização das arquiteturas de AM com exemplos da base de dados;
3. Treinar e testar arquiteturas de aprendizagem supervisionada e não-supervisionada para a tarefa em questão;
4. Avaliar comparativamente os resultados dos modelos através de métricas consolidadas na literatura para cada tipo de aprendizagem;
5. Comparar os melhores resultados obtidos para entender qual tipo de aprendizagem apresentou melhor desempenho para a base de dados.

3. Materiais e Métodos

Nesta seção, será feito um detalhamento sobre o conjunto de dados utilizado, o pré-processamento utilizado para preparar os dados para os modelos de predição e as técnicas de aprendizagem supervisionada e não-supervisionada para o problema em questão.

3.1. Conjunto de Dados

O *dataset* de transações de cartões de crédito foi adquirido em [Dal Pozzolo et al. 2015]. Esta base de dados contém um total de 284807 de amostras de transações na sua versão atual disponível no *Kaggle*, onde 284315 representam transações legítimas e apenas 492 das transações são rotuladas como fraudes. Com base nisso, pode-se então concluir que este *dataset* é altamente desbalanceado pois apenas 0.00172% dos dados são da classe de transações fraudulentas.

Todos os registros presentes no conjunto de dados são de transações reais de cartões de crédito efetuadas por usuários europeus durante o ano de 2013 [Dal Pozzolo et al. 2015]. Para a proteção das informações dos usuários envolvidos, foi utilizada a técnica de redução de dimensionalidade *Principal Component Analysis* (PCA) em algumas variáveis do *dataset*. Após este processamento, o *dataset* então resultou em 29 colunas intituladas de V1 a V28 representando os componentes principais dos dados, as quais foram disponibilizadas para o público em geral. Os únicos atributos não utilizados no cálculo do PCA foram os registro de tempo (*Time*), que apresenta a contagem de segundos desde o primeiro registro, e o valor da transação (*Amount*). Por fim, a coluna *Class* é a variável dependente contendo a classe a ser prevista, tendo o valor de 1 em caso de fraude ou 0 caso contrário.

3.2. Pré-Processamento de Dados

Em virtude das divergências relatadas anteriormente, para algumas das abordagens, foi necessária a aplicação de etapas de pré-processamento e balanceamento nos dados para evitar que os modelos não aprendam corretamente devido à falta de amostras correspondentes a uma classe.

Primeiramente, observou-se uma grande presença de *outliers* na variável *Amount*, ou seja, muitas transações apresentaram valores mais altos que a média de dados, como boa prática optou-se por normalizar os dados desta *feature* aplicando as técnicas de pré-processamento *MinMaxScaler* ou *RobustScaler*, dependendo da necessidade do modelo.

Ao decorrer da execução dos treinamentos, testou-se como os modelos se comportariam caso fossem removidas as colunas *Time* e *Amount*. Percebeu-se, então, que os modelos obtiveram um pequeno ganho em suas métricas de desempenho com a remoção de uma ou ambas as colunas.

Treinar o modelo no estado atual dos dados poderia causar um enviesamento dos dados, onde o modelo seria capaz de aprender exageradamente sobre transações legítimas mas teria problemas em detectar padrões de amostras fraudulentas. Desse modo, houve a necessidade de aplicar técnicas de *oversampling* ou *undersampling* para reduzir o viés atual e balancear os dados [Barandela et al. 2004].

Para os dados em questão, optou-se por utilizar os algoritmos *SMOTE* [Chawla et al. 2002] e *ADASYN* [He et al. 2008], o primeiro busca realizar técnicas de *over-sampling* na classe minoritária gerando novas amostras sintéticas com base nos vizinhos mais próximos, causando um efeito de clusterização em volta de cada amostra da classe minoritária. O algoritmo *ADASYN*, por sua vez, utiliza-se de distribuições ponderadas para diferentes amostras da classe minoritária de acordo com a sua dificuldade de aprendizagem, assim os dados sintéticos são gerados para amostras que são mais difíceis de serem aprendidas pelo modelo.

3.3. Abordagens e Modelos

Neste artigo serão abordadas técnicas de aprendizagem supervisionada e não-supervisionada para o desenvolvimento de modelos inteligentes. A aprendizagem supervisionada tem como base o treinamento de modelos a partir de bases de dados rotuladas com suas respectivas classes. Com vistas ao aprendizado de diferentes característica do problema, esses modelos, a cada iteração, sofrem ajustes para que possam seguir no caminho correto até sua taxa de erro atingir um nível aceitável [Hastie et al. 2009], podendo ainda ser dividido em problemas de classificação e regressão.

Como o problema proposto consiste em uma aprendizagem supervisionada do tipo classificação, realizou-se um levantamento de modelos elegíveis e optou-se por comparar quatro: KNN, *Naive Bayes*, *Random Forest* e Redes Neurais.

A proposta do KNN é encontrar a informação mais próxima daquela que você está tentando predizer, calculando a média dos valores vizinhos ou escolhendo a classe de resposta mais frequente entre eles. A estratégia de aprendizagem do KNN funciona como um tipo de memorização, algo como lembrar qual deveria ser a resposta quando a pergunta tem determinadas características. Um detalhe importante é que no uso do

KNN é imprescindível a definição de um valor de K adequado, sob pena de ocasionar um *overfitting* caso o valor de K seja muito pequeno [Mueller and Massaron 2016].

O *Naive Bayes* é um algoritmo classificador probabilístico baseado no Teorema de Bayes, que descreve a probabilidade de um evento ocorrer baseado em um conhecimento anterior que pode estar relacionado ao evento. A palavra *Naive* no nome decorre da característica do algoritmo de tratar cada *feature* como uma variável independente, o que pode ser considerado como uma suposição demasiadamente otimista (ou ingênua). Apesar disso, o *Naive Bayes* tem se mostrado bastante eficiente na resolução de uma grande variedade de problemas complexos, sendo inclusive muito utilizado em problemas de classificação na área diagnósticos médicos [Kharya et al. 2014].

As Redes Neurais Artificiais são modelos computacionais inspirados na capacidade de processamento do cérebro humano, constituídas de unidades de processamento simples denominadas neurônios artificiais, onde cada neurônio possui uma equação matemática associando entradas, pesos e funções de ativação. As RNAs do tipo *Multilayer Perceptron* apresentam uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, tendo como principal característica seu alto grau de conectividade entre os neurônios [Rojas 2013, Haykin 2010].

A *Random Forest* se enquadra nos algoritmos que utilizam o método de aprendizado *essemble*, que consiste na junção de múltiplos modelos para produzir melhores resultados. Assim, a *Random Forest* são uma combinação de árvores preditoras, como Árvores de Decisão, onde cada árvore é treinada com diferentes partes do conjunto de dados para reduzir a variação entre as classes e no fim utiliza-se do voto majoritário para realizar a predição [Breiman 2001].

Diferente da aprendizagem supervisionada, na aprendizagem não-supervisionada não temos uma rotulação dos dados, o algoritmo recebe os dados de entrada e se responsabilizam por treinar o modelo sem interferência humana. Os algoritmos buscam identificar similaridades, distribuições e assim criar grupos de dados semelhantes, podendo ser dividido em clusterização e associação [Barlow 1989]. Para o problema em questão, optou-se por utilizar dois algoritmos: *K-Means* e *Isolation Forest*.

O *K-means* é um dos métodos mais populares de *clustering*. A ideia básica desse algoritmo é: dado um *cluster* inicial e não-ótimo, realoca-se cada ponto para um centro mais próximo, atualizam-se os centros dos *clusters* calculando a média dos pontos membros, e repete-se então a realocação e atualização até serem satisfeitos os critérios de convergência, como por exemplo, um número predefinido de iterações [Lloyd 1957] [MacQueen 1967].

Por fim, o *Isolation Forest* é um método de *ensemble* que utiliza-se do poder de aprendizado das Árvores de Decisão. Este método tem como objetivo a detecção de anomalia nos dados através da busca por *outliers*, ou seja, a busca de registros nos dados que se diferem dos demais. A ideia principal do modelo é classificar como anomalias aqueles dados que encontram-se mais próximos da raiz da árvore de decisão [Liu et al. 2008].

3.4. Abordagem Supervisionada

3.4.1. *K-Nearest Neighbors* (KNN)

Devido às características dos dados presentes no *dataset*, principalmente no que se refere ao forte desbalanceamento da classe alvo, os primeiros testes apresentaram resultados bastante irregulares e métricas pouco significativas. No intuito de melhorar essas métricas e tentar evoluir o modelo, algumas ações tiveram que ser adotadas.

Primeiramente foi utilizado um algoritmo de balanceamento de dados, e com base na literatura e nas pesquisas realizadas, optou-se por testar dois métodos: *SMOTE* e *ADASYN*. Após a realização de alguns testes, o método *SMOTE* mostrou-se ligeiramente melhor em relação ao *ADASYN*.

A segunda ação foi realizar a normalização de algumas *features*. Para a *feature Amount*, foi utilizada a função *MinMaxScaler* da biblioteca *scikit-learn* que escala os dados de acordo com um range definido pelo usuário. No caso deste experimento, foi utilizada a variação padrão da biblioteca, que corresponde aos valores entre 0 e 1, proporcionando uma maior consistência nos dados em relação às outras *features*. A *feature Time*, por sua vez, teve seu valor convertido para horas através de uma operação matemática.

Com essas duas ações já foi possível verificar uma melhora bastante significativa nas métricas geradas ao confrontar o modelo com os dados de testes. Os resultados desse experimento podem ser visualizados na Tabela 1.

Tabela 1. Desempenho dos modelos treinados utilizando *KNN*.

Modelos	F1 Score	ROC-AUC Score
KNN *Unbalanced*	0.817	0.854
KNN *SMOTE*	0.660	0.888
KNN *ADASYN*	0.659	0.888

3.4.2. *Naive Bayes*

Assim como no experimento anterior com KNN, os primeiros testes com o modelo *Naive Bayes* utilizando o *dataset* original sem modificações, não obtiveram resultados significativos, o que já era esperado devido às suas características previamente mencionadas.

As técnicas utilizadas para melhorar o modelo, como normalização e balanceamento, seguiu o mesmo padrão utilizado no modelo KNN com relação às *features Amount* e *Time*. Desta vez, destaca-se o uso do modelo *ComplementNB* que é uma variante do algoritmo de *Naive Bayes* da biblioteca *scikit-learn* para realização do experimento, já que esta é particularmente recomendada para dados desbalanceados.

Analogamente aos testes realizados com KNN, foram obtidas métricas bem interessantes e promissoras com o *Naive Bayes*, estas podem ser contempladas com mais detalhes na Tabela 2.

Tabela 2. Desempenho dos modelos treinados utilizando *Naive Bayes*.

Modelos	F1 Score	ROC-AUC Score
Naive Bayes *Unbalanced*	0.666	0.803
Naive Bayes *SMOTE*	0.697	0.803
Naive Bayes *ADASYN*	0.153	0.914

3.4.3. *Random Forest*

Para o *Random Forest*, optou-se por utilizar a função *RobustScaler* para normalizar a coluna *Amount*, pois apresenta melhor desempenho para uma grande quantidade de *outilers*, conforme descrito na documentação [Buitinck et al. 2013].

Os dados foram divididos em treino e teste seguindo o padrão já efetuado, com o detalhe que a coluna *Time* foi removida do processo. Para o balanceamento de dados, utilizou-se os métodos *SMOTE* e *ADASYN*. Os modelos de *Random Forest* foram criados com seus parâmetros padrões, exceto pelo número de estimadores que, para esta atividade, optou-se pela utilização de 10 estimadores ao invés do seu padrão, para diminuir o tempo de treinamento dos modelos.

Dessa forma, três modelos foram treinados: um recebendo os dados desbalanceados; um com dados balanceados pelo algoritmo *SMOTE* e outro pelo algoritmo *ADASYN*. Todos os três modelos apresentaram dados promissores, conforme disposto na Tabela 3.

Tabela 3. Desempenho dos modelos treinados utilizando *Random Forest*.

Modelos	F1 Score	ROC-AUC Score
Random Forest *Unbalanced*	0.838	0.882
Random Forest *SMOTE*	0.849	0.926
Random Forest *ADASYN*	0.854	0.93

3.4.4. Redes Neurais

A preparação dos dados para as Redes Neurais seguiram o mesmo padrão efetuada para a *Random Forest* e a instância dos modelos foram criados com os parâmetros padrões da biblioteca.

Três modelos foram treinados utilizando o algoritmo *Multilayer Perceptron*, um com a base de dados desbalanceada e outros dois com os resultados do *SMOTE* e *ADASYN*. Observou-se uma divergência entre os resultados, mas ainda assim todos obtiveram desempenho promissor, conforme pode ser constatado na Tabela 4.

Tabela 4. Desempenho dos modelos treinados utilizando Redes Neurais Artificiais (RNA).

Modelos	F1 Score	ROC-AUC Score
RNA *Unbalanced*	0.856	0.915
RNA *SMOTE*	0.77	0.926
RNA *ADASYN*	0.70	0.933

3.5. Abordagem Não-Supervisionada

3.5.1. *K-Means*

Foram realizadas duas experimentações com o algortimo *K-Means*. Para os dois experimentos, os dados foram selecionados com a função da biblioteca pandas *sample* e normalizados com a função *normalize* da biblioteca *scikit-learn*. Em seguida, foi utilizado o método de validação cruzada *holdout*, particionando o conjunto de dados em 70% para treino e 30% para teste. É importante notar que, por se tratar de um algoritmo de abordagem não-supervisionada, foi necessária a retirada o atributo alvo *Class*.

O Experimento 1 foi realizado com a utilização de todas as variáveis e, apesar de se ter obtido uma acurácia de 99%, o resultado não foi satisfatório devido a obtenção de valores baixos nas métricas precisão, *recall* e *F1-score*. O Experimento 2, por sua vez, foi realizado com a remoção da variável *Time*. Os resultados obtidos nos dois experimentos encontram-se na Tabela 5.

Tabela 5. Desempenho dos modelos treinados utilizando K-Means.

Modelos	Accuracy	Precision	Recall	F1 Score
Experimento 1	0.995	0.0	0.0	0.0
Experimento 2	0.795	0.003	0.5	0.007

3.5.2. *Isolation Forest*

Para o modelo gerado pelo algoritmo de *Isolation Forest*, optou-se pela exclusão da coluna *Time* e normalização dos dados na coluna *Amount* utilizando o *MinMaxScaler*. Foram escolhidos os parâmetros padrões do algoritmo encontrados na biblioteca *scikit-learn*.

A geração deste modelo foi realizada da seguinte forma: primeiramente, foi feita a divisão dos dados de transações legítimas e fraudulentas. Mais adiante, no conjunto de transações legitimas, foram selecionados, de forma pseudoaleatória, 70% dos dados para o treinamento do modelo e 30% para teste. O conjunto de transações fraudulentas foi utilizado completamente na etapa de teste. Por fim, por se tratar de um modelo de aprendizagem não-supervisionada, foi removida também a coluna com o atributo alvo do conjunto de treinamento. As métricas obtidas com o modelo resultante podem ser visualizadas na Tabela 6.

Tabela 6. Desempenho do modelo treinado utilizando Isolation Forest.

Accuracy	Precision	Recall	ROC-AUC Score	F1 Score
0.963	0.117	0.827	0.896	0.205

4. Discussão

Na Tabela 7, tem-se uma visão geral das métricas de todos os modelos submetidos à avaliação. Os resultados estão ordenados de forma decrescente pelas métricas *F1 Score* e *ROC-AUC*. Com relação a *Accuracy*, observa-se que praticamente todos os modelos tiveram um índice satisfatório, fato que não ocorre com as métricas *Precision* e *Recall*. Esses números refletem a característica de desbalanceamento do *dataset*.

No caso dos modelos supervisionados, foram realizados testes tanto com dados desbalanceados como balanceados através de *SMOTE* e *ADASYN*, e esperava-se que os resultados com a base de dados balanceada fossem predominantemente melhores, mas como é possível observar, isso não foi um fato para todos os modelos, como no caso do KNN e Redes Neurais, por exemplo. Sendo que, para este último, alcançou-se um melhor desempenho considerando as métricas *F1-Score* e ROC-AUC como as principais para avaliação.

Tabela 7. Desempenho dos modelos nos testes ordenados por F1 Score e ROC-AUC.

	Accuracy	Precision	Recall	F1 Score	ROC-AUC Score
RNA	0.999555	0.882812	0.830882	0.856061	0.915353
RF ADASYN	0.999532	0.847826	0.860294	0.854015	0.930024
RF SMOTE	0.999520	0.846715	0.852941	0.849817	0.926348
RF	0.999532	0.928571	0.764706	0.838710	0.882306
KNN	0.999450	0.963303	0.709459	0.817121	0.854706
RNA SMOTE	0.999192	0.703030	0.852941	0.770764	0.926183
RNA ADASYN	0.998818	0.587065	0.867647	0.700297	0.933337
NB SMOTE	0.999087	0.818182	0.608108	0.697674	0.803937
NB	0.998947	0.737705	0.608108	0.666667	0.803866
KNN SMOTE	0.998619	0.575000	0.777027	0.660920	0.888015
KNN ADASYN	0.998607	0.572139	0.777027	0.659026	0.888009
Isolation Forest	0.963211	0.117021	0.827236	0.205038	0.895616
NB ADASYN	0.983896	0.084574	0.844595	0.153752	0.914366
K-Means	0.796098	0.004823	0.485549	0.0078	0.641139

Para este problema, a métrica do ROC-AUC recebe mais importância devido à natureza desbalanceada do *dataset*, e esta métrica não é enviesada pela classe majoritária ou minoritária [He and Ma 2013]. Olhando apenas para esta métrica, percebe-se que as RNAs com *ADASYN* apresentaram o melhor resultado, com uma diferença de 0.003 da *Random Forest*, também com *ADASYN*. Apesar disso, no F1 Score este último modelo apresentou desempenho consideravelmente inferior.

No contexto geral das métricas, os modelos baseados em *Random Forest* apresentaram melhores desempenhos, pois os três aparecem no topo da Tabela 7. Isto se deve ao fato de que, naturalmente, este modelo consegue lidar bem com conjuntos desbalanceados devido sua estratégia de *essemble* [Breiman 2001].

5. Considerações Finais

Foram realizados vários experimentos com vistas a encontrar o melhor modelo que pudesse ser utilizado para detectar fraude em transações bancárias. De todos os modelos testados, fazendo uso de duas abordagens de Aprendizado de Máquina, o *Random Forest* com o uso do *ADASYN* e com abordagem supervisionada, apresentou o melhor desempenho com 91% de AUC e 85% de *F1-Score*.

Para trabalhos futuros podem ser realizados experimentos fazendo combinação das duas abordagens, ou explorando mais o uso de técnicas de *Deep Learning*. Pode-se ainda ser feita a realização de testes utilizando outras técnicas de balanceamento de dados ou com uma base melhor e mais balanceada do que a utilizada neste experimento.

Conforme visto em trabalhos relacionados, encontrar o modelo ideal para resolução desse problema é um desafio, visto que existe ainda a possibilidade do uso de técnicas IA para a realização de fraudes mais verossímeis. Dentro deste cenário é importante estar sempre buscando uma nova técnica ou a combinação de várias outras para que a segurança seja inflacionada e a vulnerabilidade reduzida.

Referências

Barandela, R., Valdovinos, R. M., Sánchez, J. S., and Ferri, F. J. (2004). The imbalanced training sample problem: Under or over sampling? In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pages 806–814. Springer.

Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, 1(3):295–311.

Beltran, R. D. (2019). Detecção de fraudes bancárias utilizando métodos de clustering.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Carvalho, J. (2018). Índice da kroll aponta que 89% das empresas brasileiras já sofreram fraude cibernética. Disponível em `https://tinyurl.com/3ab3s45d`. Acesso em 24 de junho de 2021.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166. IEEE.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). Overview of supervised learning. In *The elements of statistical learning*, pages 9–41. Springer.

Haykin, S. (2010). *Neural networks and learning machines, 3/E*. Pearson Education India.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE.

He, H. and Ma, Y. (2013). Imbalanced learning: foundations, algorithms, and applications.

Kharya, S., Agrawal, S., and Soni, S. (2014). Naive bayes classifiers: a probabilistic detection model for breast cancer. *International Journal of Computer Applications*, 92(10):0975–8887.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, Washington, DC, United States. IEEE Computer Society.

Lloyd, S. P. (1957). Least squares quantization in pcm. Technical report, Bell Lab.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, California. University of California Press.

Mueller, J. P. and Massaron, L. (2016). *Machine Learning For Dummies*. For Dummies, 1st edition.

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Shenvi, P., Samant, N., Kumar, S., and Kulkarni, D. V. (2019). Credit card fraud detection using deep learning. In *2019 5th International Conference for Convergence in Technology*, Pune, India. 12CT.

Zamini, M. and Montazer, G. (2018). Credit card fraud detection using autoencoder based clustering. 2018(9):486–491.